Uses of kernel statistics on volcanic vents

1 Introduction

Commonly in geology, physical processes create a partially random population of point features, such as sinkholes, volcanoes, or kettle lakes. The spatial distribution of these point features is governed by an underlying physical process, and the physical process can often be modeled to create a probability model of where points are more likely to be created. For instance, sinkholes are more likely to occur in locations where soluble carbonate rock is at or near the water table, so a map showing the probability of sinkhole development might reflect the distribution of shallow carbonate rocks. In contrast, volcanoes are formed from a magma source. If the location, depth, and productivity of the magma source is well known, models can be made to map the potential locations of volcanism on the Earth's surface, where a higher likelihood of eruption would be over the most productive locations of the magma source region.

Unfortunately for volcanologists, the distribution and characteristics of magma sources in the ground are generally far less well constrained than the distribution of volcanoes on the Earth's surface. Instead of creating a probabilistic spatial model of volcanic events using knowledge of the magma source, the existing distribution of volcanoes can be used. Using only the point locations of volcanoes, one might conclude that new volcanoes are likely to form in areas where more volcanoes already exist, where the spatial density of volcanoes is higher. This is an attempt to estimate the physical model of volcano production when knowledge of the underlying physical process is limited.

The spatial vent density in an area is the number of vents per square unit area. A simple estimation of vents/km^2 can be carried out by counting the number of volcanoes in each 1 km by 1 km cell of a rectangular grid laid over all the vents in a volcano cluster. Because vents in most clusters are spaced multiple km from each other, however, this would result in a grid of mostly 0 or 1 vents/km^2, which would appear similar to a map of vent locations drawn as points. Another disadvantage to this is that sharp lines delineate the change in vent density. On one side of a grid cell boundary, the vent density might be 1 vent/km2, while one meter away density might rapidly decrease 100% to 2 vents/km2. In nature, however, it is unlikely that a volcano is twice as likely to erupt in a one meter compared to its adjacent meter.

An alternative method to measure spatial vent density would be to assign a search radius to each point location on a map. Here the spatial vent density is the number of vents within the radius divided by the area of the circle defined by the search radius. This effectively smoothes vent density if the search radius is increased to several km since the range to which each vent influences the density model increases. While this model might be an improvement

over the grid method above, it still has the same problem of sharp changes in vent density at locations with 1 search radius's distance from a vent.

Because the goal of modeling vent density is to identify an underlying probabilistic model of vent creation, it is important to select a method that avoids these artificially sharp density changes. The method used should also be able to be validated using known distributions and point samples from these distributions. A third method of estimating spatial density avoids sharp density changes by substituting the search radius with a probability density function, centered on each vent location.

1.1 Kernel Density Estimation

The process of estimating the "true" point density distribution by assigning probability density functions (PDFs) to a population of randomly sampled points from this distribution is called Kernel Density Estimation (KDE). A kernel is an identical PDF assigned to points to describe their influence on their surrounding region. Essentially, If the kernel is the normal distribution centered on a point, the point has more influence at the point's location (at the maximum value of the normal distribution) and influence decreases away from the point following the value of the normal distribution. In KDE, density is modeled as influence, so that point density is highest at a point location and decreases away from the point, in effect smoothing the point.

Given a map catalog of volcanoes, a density value, λ, for each volcano can be calculated for any location on the map, using the associated kernel PDF. If the kernel is normally distributed (i.e. Gaussian), the kernel density of a volcano for location with cartesian coordinates (x,y) is calculated as

$$\lambda(x, y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[\frac{-(x - x_v)^2 - (y - y_v)^2}{2\sigma^2}\right] \tag{1}$$

where (x_v, y_v) is the coordinate pair of the volcano and σ is the PDF standard deviation, or kernel bandwidth. The larger the kernel bandwidth, the further the density of the point is spread across the map. The total volcano point density at this location can be found by summing the point densities of all volcanoes, i, in the catalog of N volcanoes:

$$\lambda(x, y) = \frac{1}{N\sigma\sqrt{2\pi}} \sum_{i=1}^{N} \exp\left[\frac{-(x - x_i)^2 - (y - y_i)^2}{2\sigma^2}\right]. \tag{2}$$

By calculating vent density for all locations in an area, a continuous, differentiable surface is produced, which is the modeled density function for the cluster. Note that in Equation 2 the entire formula is divided by the number of vents, N. Integrating λ over the x and y domains in Equation 2 then results in unity; given an eruption, there is a probability of 1 that the eruption will take place somewhere on the surface of the planet. This means that λ is treated as a 2D probability density function with the units m^{-2}. Vent density with the units vents m^{-2} can be found by multiplying the density curve by N. Formally, λ is a probabilistic density function and multiplying by the number of vents gives "intensity," though when discussing results in this paper, the phrase "vent density" will be used in conjunction with the units vents m^{-2}.

Kernel PDFs do not have to be axi-symmetric normal distributions. In this paper, a bivariate normal distribution will be used to model density, so kernels will be Gaussian ellipses instead of circles (circular distributions can be considered univariate as they are defined by one standard deviation or bandwidth). These ellipses will also be unconstrained in direction, so their major axes do not have to be aligned in a cardinal direction. This is done by multiplying the distance between locations and volcanoes, treated here a 1x2 distance matrix, by a covariance matrix that defines the standard deviation in the x and y directions and the covariance between these dimensions. The covariance matrix is given as

$$\begin{bmatrix} \sigma_x^2 & \sigma_x\sigma_y\rho \\ \sigma_x\sigma_y\rho & \sigma_y^2 \end{bmatrix} \tag{3}$$

where σ_x is the x-direction standard deviation, σ_y is the y-direction standard deviation, and ρ is the covariance of these two. A covariance of 0 results in a kernel ellipse with axes aligned with the cardinal directions, while higher covariance skews or rotates the ellipse away from the cardinal directions.

Kernel bandwidth, the amount that the kernel ellipse is smoothed in any given direction, is an important parameter in estimating the unknown density of point features. Again, the point of KDE is to estimate the underlying spatial probability of a point feature (e.g. a volcano) being created, using only the points that can already be observed. Selecting a bandwidth that is too small will overly concentrate volcano density around each existing volcanoes, underestimating the likelihood of a new volcano in between existing edifices. A bandwidth that is too large will overly smooth volcano density, overestimating the likelihood of new volcanic events far from the existing volcano population.

Several bandwidth selectors have been created for KDE and in this project the Summed Asymptotic Means Square Error (SAMSE) selector will be used Duong, 2003). This selector is chosen because it has been validated using point populations of known distributions (Duong, 2007) and it has been shown to be more stable than similar selectors (Duong and Hazelton, 2003). The SAMSE bandwidth selector also is preferred because it provides an unconstrained bandwidth model using the covariance matrix formula in Equation 3.

Calculating the estimated spatial density, $\hat{\lambda}$, of point features in a location, s, using the unconstrained, bivariate bandwidth is done by modifying Equation 2, so that

$$\hat{\lambda}(\mathbf{s}) = \frac{1}{2N\pi\sqrt{|\mathbf{H}|}} \sum_{i=1}^{N} \exp\left[-\frac{1}{2}\mathbf{b^T}\mathbf{b}\right] \tag{4}$$

where $\mathbf{H}$ is the 2x2 covariance matrix and $|\mathbf{H}|$ is its determinant, $\mathbf{b} = \mathbf{H}^{-0.5}\mathbf{d}$ and $\mathbf{b^T}$ is its transform. $\mathbf{d}$ is the 1x2 distance matrix, with the form

$$\begin{bmatrix} x_s - x_i \\ y_s - y_i \end{bmatrix} \tag{5}$$

where (x_s, y_s) are the cartesian coordinates of the location of interest and (x_i, y_i) are the coordinates of the ith volcano.

Volume density can also be modeled by including a list of weights for each volcanic vent, where weights are eruption volumes of the volcanoes. Weighting the density functions

corresponding to each volcanic vent requires expanding Equation 4 to include weights, ω, as follows.

$$\hat{\lambda}(\mathbf{x}, \mathbf{y}) = \frac{1}{2\pi\sqrt{|\mathbf{H}|}\sum\omega}\sum_{i=1}^{N}\left(\exp\left[-\frac{1}{2}\mathbf{b^T b}\right]\omega_i\right) \tag{6}$$

This function is normalized to unity by including the sum of all weights in the denominator instead of the number of point locations. In Equation 4, the number of locations is used, as all points are weighted equally (i.e. their weights are each equal to 1).

2 Methods

2.1 Data collection

Catalogs of 20 volcano clusters have been collected for this study, including 10 volcanic fields on Earth, 3 on Mars, and 7 on Venus. The volcanoes in all clusters are primarily described as monogenetic, though some clusters are formed nearby or on top of central-vent volcanoes (e.g. Mount Adams, WA and Arsia Mons, Mars).

Each cluster catalog includes the locations of volcanic vents that are presently observable in the area. Volcanic vents are generally located at the center of scoria cones, low shields, or tuff rings and are often depressions in the center of topographic rises. Scoria cones, low shields, and tuff rings in these fields are < 1 to a few km in diameter and are interperetted to be the products of monogenetic volcanic eruptions. One cluster, the San Rafael volcanic field, is heavily eroded to a depth of ∼1 km so no edifices remain. In this cluster, diabasic conduit diatremes are mapped and interpretted to be vent locations.

The locations of each volcanic vent are defined as 2D points on their planet's surface, even though volcanic vents are not precisely point features. A vent location point generally describes the center location of the vent. Points are not defined by the center of the volcano edifice, because they are used to identify the location where magma intersected and erupted out of the planet's surface, often as a dike. This point feature can then be used in the KDE method to model volcanic vent density.

2.1.1 Earth clusters

Ten distributed-style volcanic fields on earth have been mapped, producing vent catalogs that can be used to model vent density in the region. These clusters are located in four tectonic regimes: over subduction zones, on the Colorado Plateau, in the extensional Basin and Range region, and in the Anatolian Peninsula (Figure 1).

Subduction-related clusters   Distributed volcanic fields in the Cascades Arc have formed in association with the large central volcanoes, inluding Mount Adams [Barron et al., 2014] and Newberry Volcano [Bard et al., 2013]. This arc is the product of the subducting Juan de Fuca plate. In southern Japan, monogenetic volcano groups have also formed, due to subduction of the Filipino and Pacific plates. Two of these volcano groups, the Abu Monogenetic Volcano Group and the Izu-Tobu Volcano Group have been mapped in this region
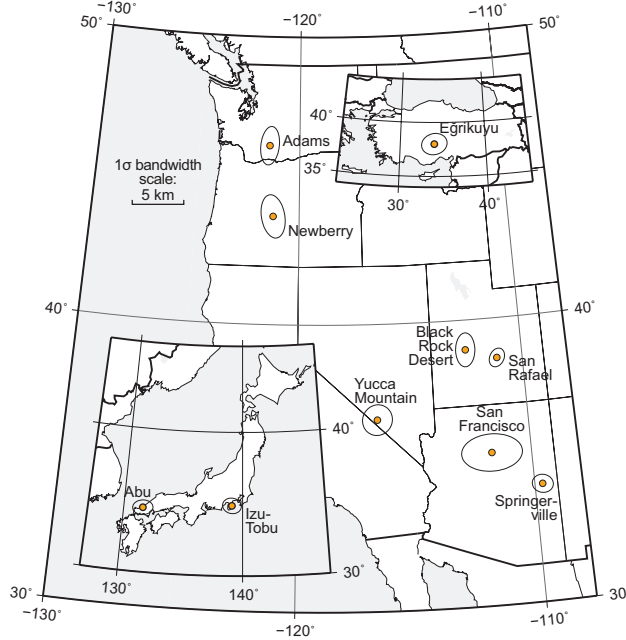
Figure 1: Locations of selected volcano clusters with their kernel bandwidth ellipses drawn over them. Ellipses are enlarged to show regional variation.

[Kiyosugi et al., 2010] and vent density has been modeled by Kiyosugi et al. [2012] Edifices in these catalogs are scoria cones, low shield volcanoes, and lava domes.

Colorado Plateau Basaltic volcanism has occurred as an annulus around the Colorado Plateau during the Cenozoic [Tanaka et al., 1986]. Most of this volcanism has been distributed in nature, forming isolated volcano clusters. Three of these clusters, the San Francisco Volcanic Field (SFVF), Black Rock Desert, and the San Rafael Volcanic Field, are included in this study.

Volcanoes in the SFVF, located in and around Flagstaff, AZ, have been mapped by Harburger [2014], including 583 vents identified as monogenetic. This field has been active for three magnetic epochs, enabling the field to be separated into three sub-fields which can be independently analyzed [Tanaka et al., 1986]. Black Rock Desert volcanoes in Utah were cataloged by Hintz [2008] and vent density has been modeled by Kiyosugi et al. [2012]. Kiyosugi et al. [2012] also mapped 126 conduits diatremes in the heavily eroded San Rafael Volcanic Field in central Utah and interpretted these to be previous vent locations when the field was active in the Pliocene.

Other tectonic settings Two other vent databases have been created for clusters in the Basin and Range, Nevada, and in Central Anatolia, Turkey. In the Basin and Range, Connor and Hill [1995] cataloged 39 scoria cones located in a cluster around Yucca Mountain. In Central Anatolia, Uslular et al. [2015] mapped 77 cinder cones, which form the Eğrikuyu volcano cluster.
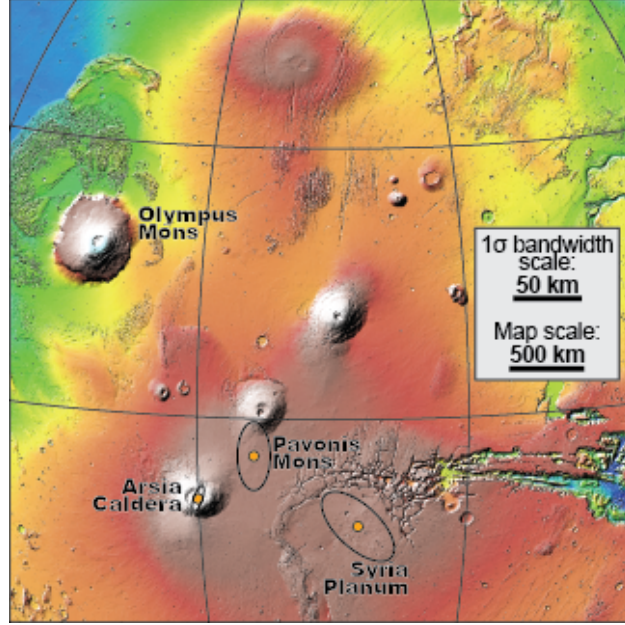
Figure 2: Locations of selected Martian volcano clusters in the Tharsis Volcanic Province. Kernel bandwidth ellipses are drawn at the 1σ contour over them. Ellipses are enlarged.

2.1.2 Mars clusters

Three distributed-style volcanic fields within the larger Tharsis Province are included in this study: Syria Planum, Southern Pavonis Mons, and Arsia Mons Caldera. The Tharsis Volcanic Province has been identified as a long lived center for volcanism since early space missions to Mars [Carr et al., 1977] and makes up about one-quarter of the total global surface. Tharsis is home to many of the largest central-vent style volcanoes in the Solar System, but also has over 1,000 small volcanic vents that are distributed in clusters around Tharsis.

Three clusters of vents have been identified in the southern Tharsis region that are separated from other vents in the database and are treated as isolated populations because of this. Each field, Syria, Pavonis, and Arsia, were potentially formed from one magma generation event, though they might have also been formed from multiple events that created overlapping volcano clusters.

Southern Pavonis Mons Pavonis Mons is a large shield volcano in the Tharsis Montes, a linear chain of three shield volcanoes within the Tharsis Province. To the north and south of each of these volcanoes, lavas have been emplaced on the shield flank, apparently after the main construction phases of each shield [Bleacher et al., 2007]. These lavas are known as flank aprons [Bleacher et al., 2007]. At Pavonis, a volcano cluster of 88 shield volcanoes (one to tens km in diameter) is present on top of the southern flank apron [Bleacher et al., 2009]. This cluster is aligned roughly N-S along the axis of the apron [Bleacher et al., 2009].
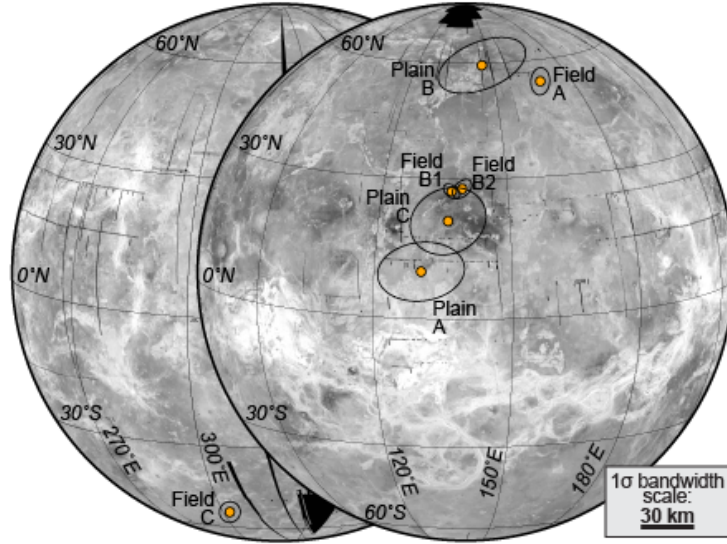
Figure 3: Locations of seven volcano clusters on Venus. Kernel bandwidth ellipses are drawn at the 1σ contour over each cluster and are enlarged.

Arsia Mons Caldera   Arsia Mons is the southernmost large shield volcano of the Tharsis Montes and has a 110 km diameter caldera at its summit. Within this caldera, 29 low shield volcanoes have been mapped (Chapter XXX, this dissertation). The caldera and shields within it are thought to be very young due to the absense of any impact craters larger than 1 km in the entire caldera [Crumpler et al., 1996].

Syria Planum   The Syria Planum region is a broad plateau with 263 low shield volcanoes [Richardson et al., 2013]. One volcano, Syria Mons, has lava flows that travel up to 1,000 km from its vent [Baptista et al., 2008]. The other low shields have been mapped by Richardson et al. [2013] and are one to tens of km in diameter with slopes of $< 6°$. This volcano cluster has been dated using crater retention rate models to have formed between 3.6-2.9 Ga spanning the Hesperian geologic period on Mars.

2.1.3   Venus clusters

Seven Venus clusters have been cataloged by Miller [2012], which fall into two distinct classes: four are "shield fields" which are composed of 10^3-10^4 volcanic vents and three are "shield plains" which host 10^4-10^5 volcanic vents. Both types of cluster are formed by low sloped (less than one to a few degrees) shields <10 km in diameter [Richardson et al., 2012]. These shields are radar bright or dark, enabling their mapping with left-look Magellan radar images, though their vents are often not visible, so vents are mapped as the centroid of the edifice [Miller, 2012]. Of the seven clusters cataloged, three are shield plains and four are shield fields [Miller, 2012].

   The large size difference between shield fields and shield plains has been observed since the return of Magellan radar images [Head et al., 1992]. Venusian shield fields are similar in size to the shield fields on Mars and have a similar number of vents (several hundreds) to

fields on Mars and Earth. Venusian plains, however, are regional in scale and include several thousand vents. The Venusian surface has very few impact craters, indicating that it has recently been resurfaced. The regional volcanic plains would have played an important role in this resurfacing, due to their large footprints [Guest et al., 1992].

2.2   Estimating vent density with the KDtools Python Library

A python library has been created to map the spatial density of geographic points, with specific functions to deal with geographic projections on Earth, Mars, and Venus. This library, called KDtools, contains seven functions which can be used to pre-process data, identify an optimal kernel bandwidth of the data, evaluate the local point density for locations on a map, and export these results to a raster file. Each function is given in an appendix below (Section 6). Two functions, explained below, are used to first identify an optimal kernel bandwidth and second evaluate the spatial density of points across a map grid.

The kernel bandwidth is determined with the SAMSE method in the **samse_bandwidth** function, by calling the R statistical language in which the SAMSE bandwidth selector has been programmed. The single input of this function is the list of locations of each volcanic vents projected in meter units. A 2×2, unconstrained covariance matrix is returned from this function.

Local vent density is calculated along a grid in the function **KD** with the covariance matrix defining the gaussian kernel ellipse. The smoothed density of each vent is calculated over the grid locations given their distance from the vent. This density is then added to the total density at each location in order to form the summation shown in Equation 4. After the density functions attributed to all vents have been calculated over the grid, all density values are normalized with the left half of Equation 4. The output of this function is a 2-dimensional array of density values corresponding to the grid surrounding vent locations. The sum of these density values approaches 1.0 as the grid size is increased around the volcano cluster. Volume density can also be modeled with function **KD** by providing weights to the function, following Equation 6.

3   Results

The locations, vent counts, and kernel bandwidths for each volcano cluster analyzed are reported in Table 1. Data and results for four clusters, Abu Monogenetic Volcano Group, Izu-Tobu Monogenetic Volcano Group, Black Rock Desert, and the Yucca Mountain Volcano Group are duplicated in Table 1 from Kiyosugi et al. [2012]. Vent density was modeled for the San Francisco Volcanic Field as a whole and as three separate sub-regions defined by magnetic epoch. These results are listed in Table 2.

Because each bandwidth covariance matrix describes a bivariate normal distribution (Equation 3), each bandwidth can be visualized as an ellipse drawn at a contour of the normal distribution. At the 1σ contour, the ellipse width in the x-direction is the square root of the top left number of the covariance matrix and the height is the square root of the bottom right number. Ellipses have been drawn for each cluster with the **ellipseGen** function in the KDtools library and are illustrated over each cluster location in Figure 1.

Table 1: Distributed-style Volcano Clusters

Cluster Name	Region, Country	Location Lat, Long	Vent Count	Bandwidth Matrix (km^2)	Data Source
Abu*	Chūgoku, Japan	34°30'N, 131°35'E	56	$\begin{bmatrix} 3.81 & 0.230 \\ 0.230 & 2.05 \end{bmatrix}$	Kiyosugi et al. [2012, 2010]
Adams	Washington, USA	46°10'N, 121°30'W	89	$\begin{bmatrix} 3.25 & 0.989 \\ 0.989 & 13.9 \end{bmatrix}$	Barron et al. [2014]
Blk. Rock Desert*	Utah, USA	39°N, 112°30'W	39	$\begin{bmatrix} 3.51 & 0.500 \\ 0.500 & 10.6 \end{bmatrix}$	Kiyosugi et al. [2012], Hintz [2008]
Eğrikuyu	Central Turkey	34°N, 38°E	77	$\begin{bmatrix} 10.8 & 1.00 \\ 1.00 & 7.57 \end{bmatrix}$	Uslular et al. [2015]
Izu-Tobu*	Chūbu, Japan	35°N, 139°20'E	126	$\begin{bmatrix} 3.29 & -0.200 \\ -0.200 & 2.27 \end{bmatrix}$	Kiyosugi et al. [2012]
Newberry	Oregon, USA	43°45'N, 121°15'W	327	$\begin{bmatrix} 5.57 & -2.43 \\ -2.43 & 17.3 \end{bmatrix}$	Bard et al. [2013]
San Francisco	Arizona, USA	35°20'N, 111°50'W	583	$\begin{bmatrix} 34.4 & -0.0396 \\ -0.0396 & 13.0 \end{bmatrix}$	Harburger [2014]
San Rafael	Utah, USA	38°35'N, 111°15'W	63	$\begin{bmatrix} 2.31 & 0.720 \\ 0.720 & 3.12 \end{bmatrix}$	Kiyosugi et al. [2012]
Springerville	Arizona, USA	34°15'N, 109°45'W	400	$\begin{bmatrix} 4.07 & -0.300 \\ -0.300 & 3.14 \end{bmatrix}$	Kiyosugi et al. [2012], Condit [2010]
Yucca Mountain*	Nevada, USA	36°40'N, 116°30'W	39	$\begin{bmatrix} 8.35 & 0.42 \\ 0.42 & 8.75 \end{bmatrix}$	Kiyosugi et al. [2012], Connor and Hill [1995]
Field-A	Atalanta, Venus	54°N, 168°E	344	$\begin{bmatrix} 115 & 7.74 \\ 7.74 & 238 \end{bmatrix}$	Miller [2012]
Field-B1	Vellamo, Venus	27°N, 137°E	135	$\begin{bmatrix} 88.7 & -35.4 \\ -35.4 & 71.4 \end{bmatrix}$	Miller [2012]
Field-B2	Vellamo, Venus	328°N, 139°E	169	$\begin{bmatrix} 125 & 51.7 \\ 51.7 & 116 \end{bmatrix}$	Miller [2012]
Field-C	Mylitta, Venus	52°S, 58°W	290	$\begin{bmatrix} 140 & 2.93 \\ 2.93 & 138 \end{bmatrix}$	Miller [2012]
Plain-A	Greenaway, Venus	11°N, 130°E	2919	$\begin{bmatrix} 2440 & 164 \\ 164 & 1120 \end{bmatrix}$	Miller [2012]
Plain-B	Atalanta, Venus	60°N, 150°E	10225	$\begin{bmatrix} 2470 & 709 \\ 709 & 1000 \end{bmatrix}$	Miller [2012]
Plain-C	Greenaway, Venus	20°N, 135°E	3460	$\begin{bmatrix} 1860 & 232 \\ 232 & 1460 \end{bmatrix}$	Miller [2012]
Arsia	Tharsis, Mars	9°S, 120°W	29	$\begin{bmatrix} 81.4 & 105 \\ 105 & 347 \end{bmatrix}$	Chapter X
Pavonis	Tharsis, Mars	4°S, 114°W	89	$\begin{bmatrix} 579 & -0.616 \\ -0.616 & 2520 \end{bmatrix}$	Bleacher et al. [2009]
Syria	Tharsis, Mars	14°S, 100°W	263	$\begin{bmatrix} 2810 & -1720 \\ -1720 & 2620 \end{bmatrix}$	Richardson et al. [2013]

* Cluster data (including bandwidth matrix) are reported in Kiyosugi et al. [2012], but vent locations are not included in this report.

Table 2: Sub-populations of the San Francisco Volcanic Field*

| Magnetic chronozone	Time Span Ma	Centroid Lat, Long	Vent Count	Bandwidth Matrix (km^2)
Brunhes	0.73 - Present	35°20'N, 111°30'W	239	$\begin{bmatrix} 21.6 & -5.66 \\ -5.66 & 11.7 \end{bmatrix}$
Matuyama	2.48 - 0.73	36°20'N, 112°W	209	$\begin{bmatrix} 15.0 & 1.38 \\ 1.38 & 25.4 \end{bmatrix}$
Pre-Matuyama	5-2.48	36°20'N, 112°15'W	135	$\begin{bmatrix} 13.3 & -3.10 \\ -3.10 & 12.5 \end{bmatrix}$
Entire Field	5 - Present	35°20'N, 111°50'W	583	$\begin{bmatrix} 34.4 & -0.0396 \\ -0.0396 & 13.0 \end{bmatrix}$

*Vent locations reported in Harburger [2014].
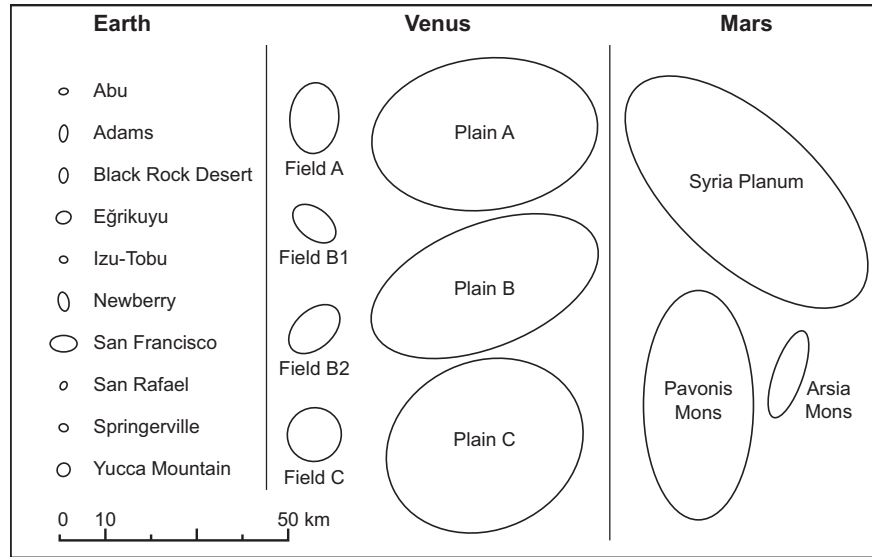
Figure 4: Kernel bandwidth ellipses for different clusters on Earth, Venus, and Mars. Each ellipse is drawn to scale, representing the 1σ contour of the Gaussian bandwidths. The tightest bandwidths correspond to Earth clusters, while the broadest bandwidths correspond to Venus shield plains and Martian clusters.

4   Discussion

The major finding of this study is that the spatial density of vents between planets is generally different by an order of magnitude. On Earth, six monogenetic volcano clusters are shown to all cluster on the order of 0.1 vents per km^2. On Mars, two clusters have a vent density of around 0.001 vents per km^2. One cluster, in the basaltic caldera of Arsia Mons, is more focused by a factor of five. In between these vent densities, Venus shield fields have densities of 0.01 vent per km^2. Venus shield fields are still more focused than the regional shield plains, which have an average density of 0.003 vents per km^2, similar to the Arsia Mons cluster. Essentially, volcanic vents in Earth fields have 10s km^2 of space between neighboring vents, on Venus, distributed volcanoes have 100s km^2 of individual space, and on Mars, two of three fields have 1000s km^2 of space between neighboring vents, while one is clustered on a "Venusian" scale.

4.1   Geologic implications of the kernel bandwidth

Each bandwidth ellipse mimics attributes of its corresponding vent cluster and likely reflects geologic properties which effect the volcano cluster. The bandwidth ellipse area at one standard deviation (1-sigma) reflects both the spatial extent of volcanoes in the cluster and the number of vents in the cluster. Larger volcano clusters correlate with larger bandwidths ellipses, while clusters with more volcanoes in the same amount of space have smaller bandwidths. Because bandwidth area is perfectly correlated with these two characteristics, it is better to refer to each characteristic directly. Similarly, bandwidth ellipse elongation, or the difference in the major and minor axis standard deviations, is a function of the anisotropy in vent production through the cluster, either because of a farther extent of volcanoes in one direction or because of more vents per km in one direction. The anisotropy of vent production in a cluster can be explored using bandwidth elongation as a proxy.

Elongation of the bandwidth of a volcano cluster can be explained by one or a combination of at least three geologic processes. First, the magma source region underlying the volcano cluster might be elongated, matching the distribution of volcanoes observed at the planet's surface. Second, the magma source region might have migrated with time or multiple magma source regions that are spatially adjacent might be overprinting pre-existing populations. Third, the hydrodynamic conductivity of the crust might itself be anisotropic, preferentially focusing magma in one direction or enabling magma to spread laterally away from a source region. While all three of these might apply to all volcanic fields in varying degrees of magnitude, I will discuss how the anisotropy seen in the bandwidth ellipse might be explained by one of these processes for example clusters.

4.1.1   Elongated Source Region

Cascadian volcanic fields are a good example of this.

4.1.2   Magma source migration or multiple overprinting sources

Both the Springerville and the San Francisco volcanic fields have migrated eastward over their respective histories.
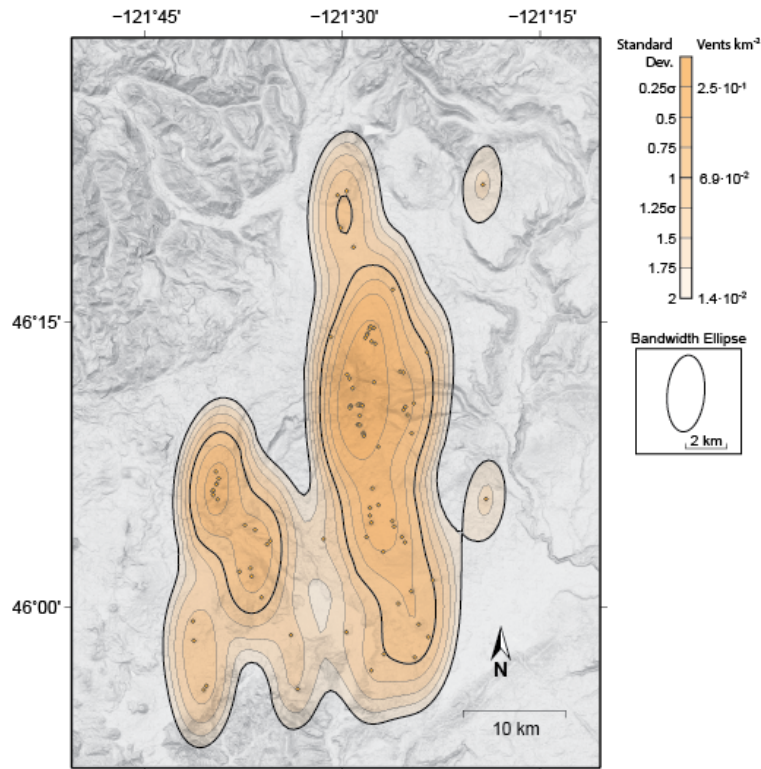
Figure 5: Volcanic vent density of monogenetic volcanoes near Mt. Adams (Washington, USA). If the magma source region is parallel to the subduction zone to the west, this might result in an elongated volcano distribution and kernel bandwidth ellipse.
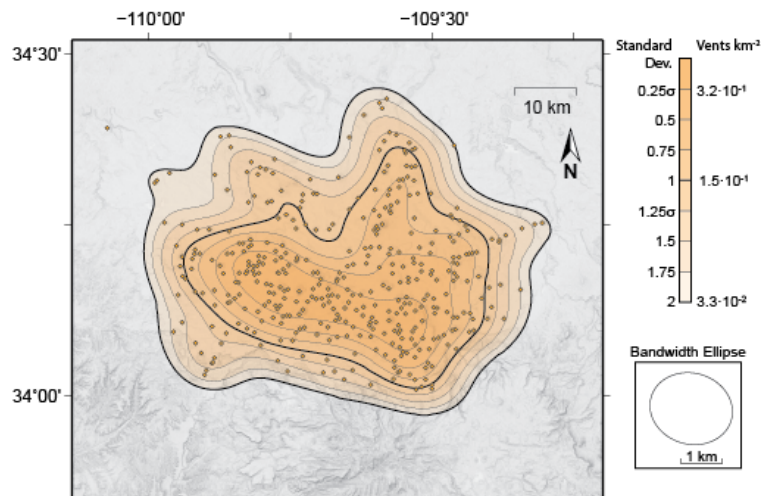

Figure 6: Volcanic vent density of the Springerville Monogenetic Volcanic Field (Arizona, USA). The migration of the magma source with respect to the crust over time might be the source of the E-W elongated kernel ellipse.
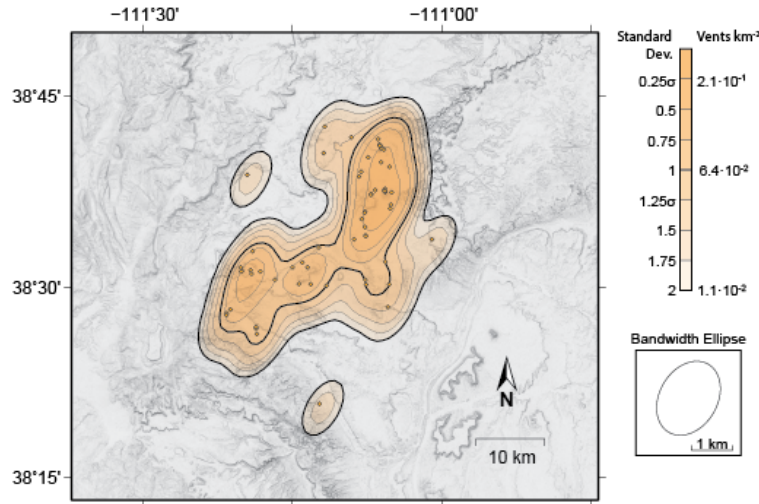
Figure 7: Volcanic conduit density of the San Rafael Volcanic Field (Utah, USA). Dikes are oriented north following pre-existing joint sets in the upper crust, possibly influencing the shape of the kernel ellipse.

4.1.3   Anisotropic crustal conductivity

San Rafael and Arsia Mons are perfect for this example. Maybe the cascadian volcanoes as well but I'm not too sure.

Bonnafede and Cenni (1997) modeled the crust under Mount Etna as a porous medium through which pressurized magma can freely flow. In this model, magma migrates based on a hydraulic head pressure and the hydraulic conductivity of the crust. If conductivity is low, magma will flow less, while if conductivity is high, magma will flow more easily through the medium. By modeling an anisotropic conductivity, where conductivity is less or more for magma traveling east or west than for magma traveling north or south, Bonnafede and Cenne (1997) were able to match the expected surface output of magma to the historical distribution of flank eruptions at Etna.

Hydraulic conductivity in the crust can be related to pre-existing fracture sets, as aligned fractures enable magma to propagate more easily in the direction parallel to the fractures. Conductivity can also be a product of dynamic stress, since dikes open in the direction of least compressive stress and propagate in directions normal to the least compressive stress.

4.2   Volume Flux Density

Volume can also be compared with this method. Spatial vent density and volume density compared across planets We can compare km3/km2-a between the caribou field and arsia. We can compare km3/km2 between arsia and egrikuyu

5 Conclusions

References

Ana R. Baptista, Nicolas Mangold, Véronique Ansan, David Baratoux, Philippe Lognonné, Eduardo I. Alves, David A. Williams, Jacob E. Bleacher, Philippe Masson, and Gerhard Neukum. A swarm of small shield volcanoes on Syria Planum, Mars. *Journal of Geophysical Research*, 113(E9):E09010, 2008. doi: 10.1029/2007JE002945. doi:10.1029/2007JE002945.

Joseph A Bard, David W Ramsey, Norman S MacLeod, David R Sherrod, Lawrence A Chitwood, and Robert A Jensen. Database for the geologic map of newberry volcano, deschutes, klamath, and lake counties, oregon. Technical report, US Geological Survey Data Series 771, 2013.

Andrew D Barron, David W Ramsey, and James G Smith. Database for the geologic map of upper eocene to holocene volcanic and related rocks in the cascade range, washington. Technical report, US Geological Survey, 2014.

Jacob E. Bleacher, R. Greeley, D. A. Williams, S. R. Cave, and G. Neukum. Trends in effusive style at the Tharsis Montes, Mars, and implication for the development of the Tharsis province. *Journal of Geophysical Research*, 112:E09005, 2007. doi: 10.1029/2006JE002873. doi:10.1029/2006JE002873.

Jacob E. Bleacher, Lori S. Glaze, Ronald Greeley, Ernst Hauber, Stephen M. Baloga, Susan E. H. Sakimoto, David A. Williams, and Timothy D. Glotch. Spatial and alignment analyses for a field of small volcanic vents south of Pavonis Mons and implications for the Tharsis province, Mars. *Journal of Volcanology and Geothermal Research*, 185(1–2): 96–102, 2009. doi: 10.1016/j.jvolgeores.2009.04.008.

M. H. Carr, R. Greeley, K. R. Blasius, J. E. Guest, and J. B. Murray. Some Martian volcanic features as viewed from the Viking orbiters. *Journal of Geophysical Research*, 82: 3985–4015, 1977.

Christopher D Condit. Dynamic digital map of the springerville volcanic field and the ddm-template: An example of an open-source tool to distribute maps, data, articles, and multi-media materials. *Geosphere*, 6(4):430–443, 2010.

Charles B Connor and Brittain E Hill. Three nonhomogeneous poisson models for the probability of basaltic volcanism: application to the yucca mountain region, nevada. *Journal of Geophysical Research: Solid Earth (1978–2012)*, 100(B6):10107–10125, 1995.

LS Crumpler, James W Head, and Jayne C Aubele. Calderas on mars: Characteristics, structure, and associated flank deformation. *Geological Society, London, Special Publications*, 110(1):307–348, 1996. doi: 10.1144/GSL.SP.1996.110.01.24.

John E Guest, Mark H Bulmer, Jayne Aubele, Kathi Beratan, Ronald Greeley, James W Head, Gregory Michaels, Catherine Weitz, and Charles Wiles. Small volcanic edifices

and volcanism in the plains of venus. *Journal of Geophysical Research: Planets*, 97(E10): 15949–15966, 1992.

Aleeza M Harburger. Probabilistic modeling of lava flows: A hazard assessment for the san francisco volcanic field, arizona. Master's thesis, University of South Florida, 2014.

James W Head, LS Crumpler, Jayne C Aubele, John E Guest, and R Stephen Saunders. Venus volcanism: Classification of volcanic features and structures, associations, and global distribution from magellan data. *Journal of Geophysical Research: Planets*, 97 (E8):13153–13197, 1992.

A.R. Hintz. Physical volcanology and hazard analysis of a young monogenetic volcanic field: Black Rock Desert, Utah. Master's thesis, University of South Florida, 2008.

K. Kiyosugi, C.B. Connor, P.H. Wetmore, B.P. Ferwerda, A.M. Germa, L.J. Connor, and A.R. Hintz. Relationship between dike and volcanic conduit distribution in a highly eroded monogenetic volcanic field: San Rafael, Utah, USA. *Geology*, 40(8):695–698, 2012. doi: 10.1130/G33074.1.

Koji Kiyosugi, CB Connor, D Zhao, LJ Connor, and K Tanaka. Relationships between volcano distribution, crustal structure, and p-wave tomography: an example from the abu monogenetic volcano group, sw japan. *Bulletin of volcanology*, 72(3):331–340, 2010.

Diana M Miller. Shield fields and shield plains on venus geologic characteristics and spatial relationships. Thesis (m.s.), State University of New York at Buffalo, 2012.

JA Richardson, DM Miller, JE Bleacher, C Connor, TK Gregg, LJ Connor, and LS Glaze. Comparison of monogenetic volcano clusters on earth, venus, and mars. In *AGU Fall Meeting Abstracts*, volume 1, page 04, 2012.

Jacob A Richardson, Jacob E Bleacher, and Lori S Glaze. The volcanic history of syria planum, mars. *Journal of Volcanology and Geothermal Research*, 252:1–13, 2013.

Kenneth L Tanaka, Eugene M Shoemaker, George E Ulrich, and Edward W Wolfe. Migration of volcanism in the san francisco volcanic field, arizona. *Geological Society of America Bulletin*, 97(2):129–141, 1986. doi: 10.1130/0016-7606(1986)97⟨129:MOVITS⟩2.0.CO;2.

G Uslular, G Gençalioğlu-Kuşcu, and A Arcasoy. Size-distribution of scoria cones within the eğrikuyu monogenetic field (central anatolia, turkey). *Journal of Volcanology and Geothermal Research*, 2015. doi: 10.1016/j.jvolgeores.2015.05.006.

6 Appendix: KDtools

Below are the functions as written in the kdtools python library. This library is available on github at `https://github.com/jarichardson/kdtools`.

6.1   contourBySigma

```
def contourBySigma(Z,
  sigmas=[0.25,0.5,0.75,1.0,1.25,1.5,1.75,2.0,2.25,2.5,2.75,3.0],
  gridspacings=[1,1]):
  '''
  Identifies Density contour levels given Sigma thresholds.
  Contours define areas within which lay X% of the total field density
  e.g.: Within the Sigma-2.0 contour lies 95.4% of total field density.
        The density value of each contour decreases with increasing
        sigma.
  Requires a numpy array of density values (Z), with any shape.
  Optionally, provide a list of requested sigma thresholds, and the
  grid size as a 2 item list, to normalize the density.

  Outputs a dictionary of {sigma-level: density value}. If sigma-levels
  are not found (off the grid if the grid is too small), they will not
  be included in the dictionary.
  '''
  #find cumulative density that is used to pass the given
  #sigma thresholds in "contours"
  cum_thresholds = 2*(norm.cdf(sigmas)-norm.cdf(0))

  integrate = 0.0
  curcontour = 0
  densitycontours = {}

  #sort and reverse Z
  Z = numpy.sort(Z,kind='quicksort',axis=None)[::-1]

  #assuming density units are m^-2, but spacing is not 1 cell m^-2
  #reduce cumulative threshold by grid resolution
  cum_thresholds *= 1.0/(gridspacings[0]*gridspacings[1])

  for i,d in enumerate(Z):
    integrate+=d
    #if the elapsed density surpasses the next contour
    if (integrate >= cum_thresholds[curcontour]):
      densitycontours[sigmas[curcontour]] = d
      curcontour += 1
      if (curcontour>=len(sigmas)):
        break

  return densitycontours
```

6.2 densityToRaster

```
def densityToRaster(griddata, ranges, spacings, outrastername, clon=-999,
   utmzone=-999, planet='earth', driver='GTiff', outproj="tm"):
   '''
   Outputs a 2-D array to a gdal-readable raster. Input expected to be
   in a transverse mercator projection.
   griddata: 2D data array
   outrastername: file name of raster output. If meter output is desired,
       it would be good practice to define clon or utm zone
   planet: 'earth','venus', or 'mars'. This is only needed to translate to
       latlong projections
   clon: center longitude of non-earth transverse mercator data
   utmzone: utm zone of earth data
   driver: gdal-readable raster short name [GTiff]
   outproj: 'tm' or 'll' for transverse mercator (no tranformation occurs)
       or latlong (gdalwarp will be implemented). [tm]

   ISSUES: If values are very low (normal for density grids), gdalwarp
       doesn't work, so it is suggested that output remain in meters.
       A workaround might be to supply log10 values of griddata.
   '''

   #print an extra line for good looks
   print ""

   #Check that the user's requested driver will actually work before
   #doing anything
   userdriver = gdal.GetDriverByName(driver)
   if userdriver==None:
     print '\nerror: Raster type "'+driver+'" not a valid gdal driver!'
     print '  No map created.'
     return None

   gdaldriver = gdal.GetDriverByName('GTiff')
   gdaldriver.Register()
   cols = numpy.shape(griddata)[1]
   rows = numpy.shape(griddata)[0]
   bands = 1

   griddata = griddata[::-1]

   dest_raster = gdaldriver.Create(outrastername, cols, rows, bands, \
     gdal.GDT_Float64 )
```

```python
#adfGeoTransform[0] /* top left x */
#adfGeoTransform[1] /* w-e pixel resolution */
#adfGeoTransform[2] /* rotation, 0 if image is "north up" */
#adfGeoTransform[3] /* top left y */
#adfGeoTransform[4] /* rotation, 0 if image is "north up" */
#adfGeoTransform[5] /* n-s pixel resolution */
geotrans = [ranges[0][0],spacings[0],0,ranges[1][1],0,(-1*spacings[1])]
dest_raster.SetGeoTransform(geotrans)

#set transverse mercator projection
if (utmzone>=1 and utmzone<=60):
  srs = osr.SpatialReference()
  srs.SetUTM( utmzone, 1 ) #1 means north, this could be problematic
  srs.SetWellKnownGeogCS( 'WGS84' );
  dest_raster.SetProjection( srs.ExportToWkt() )

elif (clon>=-360 and clon<=360):
  srs = osr.SpatialReference()

  if (planet == 'venus'):
    srs.ImportFromProj4( '+proj=tmerc +lat_0=0 +lon_0='+str(clon)+ \
      ' +k=0.9996 +x_0=0 +y_0=0 +a=6051800 +b=6051800 +units=m +no_defs' )
    srs.SetProjCS( "Venus 2000 Sphere, Custom Meridian" )
    srs.SetGeogCS( 'Venus 2000', 'D_Venus_2000', 'Venus_2000_IAU_IAG', \
      6051800.0, 0.0 )

  elif (planet == 'mars'):
    srs.ImportFromProj4( '+proj=tmerc +lat_0=0 +lon_0='+str(clon)+ \
      ' +k=0.9996 +x_0=0 +y_0=0 +a=3396190 +b=3396190 +units=m +no_defs' )
    srs.SetProjCS( "Mars 2000 Sphere, Custom Meridian" )
    srs.SetGeogCS( 'Mars 2000', 'D_Mars_2000', 'Mars_2000_IAU_IAG', \
      3396190.0, 169.89444722361179 )
  else:
    print '\nwarning: clon set but planet is not venus or mars.'
    print '  output raster will not have projection metadata'
    #return 0
  dest_raster.SetProjection( srs.ExportToWkt() )

dest_raster.GetRasterBand(1).WriteArray( griddata )
dest_raster = None

#warp to ll if necessary
if outproj=='ll':
  if planet=='earth':
    #catch invalid utmzone
```

```python
    if ((utmzone<1) or (utmzone>60)):
      print 'utm zone not valid (1-60). Cannot create latlong raster.'
      print 'utm raster saved at '+outrastername
      return 0

    #reproject the transverse mercator grid
    os.system('gdalwarp -r cubic -t_srs "+proj=longlat +datum=WGS84" '+ \
      outrastername+' tmpLL.tif')

  #if not earth, catch invalid center_lon
  elif ((clon<-360) or (clon>360)):
    print 'center longitude not valid (-360 to 360).', \
      ' Cannot create latlong raster.'
    print 'transverse mercator raster saved at '+outrastername
    return 0
  else:
    if planet=='mars':
      radius='3396190'
    elif planet=='venus':
      radius='6051800'
    else:
      print 'planet not either earth, venus, or mars.', \
        ' cannot create latlong raster.'
      print 'transverse mercator raster saved at '+outrastername
      return 0

    #reproject the transverse mercator grid
    os.system('gdalwarp -r cubic -t_srs "+proj=longlat +k=0.9996 '+ \
      '+x_0=0 +y_0=0 +a='+radius+' +b='+radius+' +no_defs" '+ \
      outrastername+' tmpLL.tif')

  #overwrite the transverse meter raster with the longlat raster file
  if driver=='GTiff':
    os.system('mv tmpLL.tif '+outrastername)
  else:
    os.system('gdal_translate -of '+driver+' tmpLL.tif '+outrastername)
    os.remove('tmpLL.tif')

#If output is in meters, but user wants a non-Tiff, change format here
elif driver!='GTiff':
  os.system('gdal_translate -of '+driver+' '+outrastername+' tmpM.tif')
  os.system('mv tmpM.tif '+outrastername)

if os.path.isfile('tmpM.aux.xml'):
  os.remove('tmpM.aux.xml')
```

```
    if os.path.isfile('tmpLL.aux.xml'):
      os.remove('tmpLL.aux.xml')
    if os.path.isfile(outrastername+'.aux.xml'):
      os.remove(outrastername+'.aux.xml')

  return 0
```

6.3   ellipseGen

```
def ellipseGen(bd,eps=False,epsfilename='bandwidth_ellipse.eps'):
  '''
  Identifies the major and minor axes directions and standard
  deviations of a Gaussian ellipse defined by a 2x2 covariance
  matrix. Precision is to the nearest degree.

  Prints out solution, and optionally uses GMT to draw the ellipse
  to epsfilename, if eps=True.

  Outputs major-axis direction, major-axis standard deviation, and
          minor-axis standard-deviation.
  '''
  detH = linalg.det(linalg.sqrtm(bd)) #determinate sqrt bandwidth
  invH = linalg.inv(linalg.sqrtm(bd)) #inverse sqrt bandwidth
  constant = 2.0*numpy.pi*detH

  radius = 20
  angles = numpy.arange(0,numpy.pi,(numpy.pi/180.0))
  D = numpy.zeros(len(angles))

  #simulate density in all directions
  for i,phi in enumerate(angles):
    dx = radius*numpy.cos(phi)
    dy = radius*numpy.sin(phi)
    dxdy = numpy.dot(invH,numpy.array([[dx],[dy]]))
    dist = numpy.dot(numpy.transpose(dxdy),dxdy)[0][0]
    D[i] = numpy.exp(-0.5*dist)/constant

  #Find azimuth of greatest, least density
  maxaz = angles[numpy.where(D==max(D))][0]
  minaz = angles[numpy.where(D==min(D))][0]

  #Calculate Density at vent location
  dxdy = numpy.dot(invH,numpy.array([[0],[0]]))
  dist = numpy.dot(numpy.transpose(dxdy),dxdy)[0][0]
  ventD = numpy.exp(-0.5*dist)/constant
```

```
    #Calculate standard deviations
    #For the major axis
    majsd = (10*(2**0.5))/(-1*numpy.log(max(D)/ventD))**0.5 #(radius=20 units)
    majdir = 90-numpy.degrees(maxaz) #Gives direction from North. East is +
    #For the minor axis
    minsd = (10*(2**0.5))/(-1*numpy.log(min(D)/ventD))**0.5
    mindir = 90-numpy.degrees(minaz)

    #Print out the results
    '''
    print '\nBandwidth Ellipse Information'
    print 'major axis:'
    print ('  degrees from north - %0.0f' % majdir)
    print ('  standard deviation - %0.4f' % majsd)
    print 'minor axis:'
    print ('  degrees from north - %0.0f' % mindir)
    print ('  standard deviation - %0.4f' % minsd)
    '''

    if eps==True:
      majaxis = 2*majsd
      minaxis = 2*minsd

      with open('ellipseGMT.xy','w') as f:
        f.write('0\t0\t%0.0f\t%0.4f\t%0.4f' % (majdir, majaxis, minaxis))
      os.system('psxy ellipseGMT.xy -SE -Wblack -JX6i ' + \
        ('-R%0.4f/%0.4f/%0.4f/%0.4f -Ba%0.4f -K >' % ((-1*majaxis),majaxis, \
        (-1*majaxis), majaxis, majsd)) + epsfilename)
      os.remove('ellipseGMT.xy')
      print ('\nPlotted ellipse at '+epsfilename)

    returnstats = [majdir,majsd,minsd]
    return returnstats
```

6.4 KD

```
def KD(bd,coors,ranges,spacings,weights=[]):
  '''
  Estimates point density using:
  bd       - a kernel bandwidth (2x2 covariance  matrix)
  coors    - 2xN list of coordinates for N points.
  ranges   - a 2x2 [[W,E],[S,N]] array
  spacings - a 1x2 [X-resolution,Y-resolution] array
  weights  - a 2xN list of wieghts for N points [None]
```

```
Outputs X,Y,D: Eastings, Northings, and Densities in a Meshgrid
format (i.e. X will be tiled, Y will be repeated)
'''

#If weights are given, test to see that they're valid
if weights != []:
  if numpy.shape(weights)[0] != numpy.shape(coors)[0]:
    print "error: weight array not same length as coordinate array!"
    print "  cannot create kernel density map."
    return None
#If weights are not given, make weights even across the board
else:
  weights = numpy.ones(len(coors))

weightaverage = numpy.sum(weights)/len(weights)

detH = linalg.det(linalg.sqrtm(bd)) #determinate sqrt bandwidth
invH = linalg.inv(linalg.sqrtm(bd)) #inverse sqrt bandwidth

#constant variable in gaussian pdf
constant = 2.0*numpy.pi*detH*len(coors) * weightaverage

#define map grid
x = numpy.arange(ranges[0][0],(ranges[0][1]+spacings[0]),spacings[0])
y = numpy.arange(ranges[1][0],(ranges[1][1]+spacings[1]),spacings[1])
X,Y = numpy.meshgrid(x,y)  #X and Y are now tiled to grid
D = numpy.zeros(numpy.shape(X)) #Density Grid
dist = numpy.zeros(numpy.shape(X)) #distance matrix grid

#Three for loop with enumerates... Nick Voss would be proud.
for w,v in enumerate(coors):
  for i,e in enumerate(x):
    for j,n in enumerate(y):
      dx = e-v[0]
      dy = n-v[1]
      dxdy = numpy.dot(invH,numpy.array([[dx],[dy]]))
      dist[j][i] = numpy.dot(numpy.transpose(dxdy),dxdy)[0][0]
  D += numpy.exp(-0.5 * dist) * weights[w]

D /= constant #normalize

return X,Y,D
```

6.5   main

```
def main():
  '''
  runs tests for kdtools functions using a synthetic dataset
  some tests are visual and require matplotlib
  '''
  import matplotlib.pyplot as plt
  from matplotlib.ticker import LogFormatter

  #create a random synthetic dataset of points
  data = numpy.random.uniform(31,35,[50,2])

  #or create a grid of synthetic points
  #data = numpy.zeros([100,2])
  #data_easts  = numpy.linspace(31,35,10)
  #data_norths = numpy.linspace(31,35,10)
  #E, N = numpy.meshgrid(data_easts,data_norths)
  #data[:,0] = E.reshape((100))
  #data[:,1] = N.reshape((100))

  data[:,1] *= 1.3 #stretch data in the N-S direction
  zone = 36   #utm zone on earth for this lat-long
  clon = 33.0 #center longitude of data on other planets

  #random synthetic weights
  weights = numpy.random.uniform(1,20,len(data))
  #weights = numpy.linspace(1,20,len(data)) #this puts the weights in order

  print "\nsynthetic dataset info"
  print "  %d points on an x,y grid" % len(data)
  print "  x min,mean,max - %0.3f, %0.3f, %0.3f" % (min(data[:,0]), \
    (sum(data[:,0])/len(data)),max(data[:,0]))
  print "  y min,mean,max - %0.3f, %0.3f, %0.3f" % (min(data[:,1]), \
    (sum(data[:,1])/len(data)),max(data[:,1]))

  #create grid matrix
  gridresolution = [10000,10000]


  #1. reproject(llcoors, planet='earth', utmzone=-999, clon=-999)
  data = reproject(data,planet='mars',clon=clon)
  print "\nReprojected synthetic dataset info"
  print "  %d points on an x,y grid" % len(data)
  print "  x min,mean,max - %0.3f, %0.3f, %0.3f" % (min(data[:,0]), \
```

```
    (sum(data[:,0])/len(data)),max(data[:,0]))
print "  y min,mean,max - %0.3f, %0.3f, %0.3f" % (min(data[:,1]), \
    (sum(data[:,1])/len(data)),max(data[:,1]))


#2. rangeBuffer(coords, B=0)
datarange = rangeBuffer(data,B=30)
print "\ndata range with 30% buffer:\n", datarange


#3. samse_bandwidth(coords)
bandwidth = samse_bandwidth(data)
if len(bandwidth) == 0:
  print "samse_bandwidth returned an error"
  return None
print "\nsamse bandwidth:\n", bandwidth



#4. ellipseGen(bd, eps=False, epsfilename='bandwidth_ellipse.eps')
ellipse_stats = ellipseGen(bandwidth, eps=True)
print "\nellipse stats:"
print "  ellipse major axis orientation (deg N) - ", ellipse_stats[0]
print "  ellipse major axis standard deviation  - ", ellipse_stats[1]
print "  ellipse minor axis standard deviation  - ", ellipse_stats[2]


#5. KD(bd, coors, ranges, spacings)
print "\nCalculating Density on grid..."
(X,Y,D) = KD(bandwidth, data, datarange, gridresolution, weights=weights)

integrateddensity = numpy.sum(D)*gridresolution[0]*gridresolution[1]
print ("  Total Density within grid - %0.3f%%" % (integrateddensity*100))
print ("  Maximum Density on grid   - %0.3e sq. unit area^-1" % \
  numpy.amax(D))



#6. contourBySigma(Z, sigmas, gridspacings)
contours = contourBySigma(D, sigmas=[0.5,1,2], \
  gridspacings=gridresolution)
print "\nDensity value contours at"
print "  0.5-sigma: %0.3e sq. unit area^-1" % contours[0.5]
print "    1-sigma: %0.3e" % contours[1]
print "    2-sigma: %0.3e" % contours[2]



#7. densityToRaster(griddata, ranges, spacings, outrastername, clon=-999,
#               utmzone=-999, planet='earth', driver='GTiff', outproj='tm')
gdalerr = densityToRaster(numpy.log10(D), datarange, gridresolution, \
```

```
    'synth.grd', clon=clon, planet='mars', outproj='ll', driver="GMT")
  if gdalerr != -1:
    print "\nOutput test raster to synth.tif"


  #8. Plot the results
  print "\nPlotting Density map with points"
  plt.clf()
  plt.subplot(1, 1, 1)
  plt.title('test density (points per square unit)')
  # set the limits of the plot to the limits of the data
  plt.axis([datarange[0][0], datarange[0][1], datarange[1][0], \
    datarange[1][1]])


  #Color plot of the density data from KD
  plt.pcolor(X, Y, D, cmap='YlOrRd', vmin=0, vmax=numpy.amax(D))
  #format the color bar labels to log scale
  formatter = LogFormatter(10, labelOnlyBase=False)
  plt.colorbar(format=formatter)


  #Contour plot from contourBySigma
  contourlevels = [contours[0.5],contours[1],contours[2]]
  CS = plt.contour(X,Y,D,contourlevels,colors='k')
  plt.clabel(CS, fontsize=9, inline=1, fmt=formatter)


  #Scatter Plot of synthetic dataset
  plt.scatter(data[:,0],data[:,1],c='k',s=(weights**2),marker='.')
  plt.show()
```

6.6   rangeBuffer

```
def rangeBuffer(coords,B=0):
  '''
  Creates a buffer of B% [default 0%, no buffer] around
  N-dimensional data. Input should be a numpy array.
  Output will be 2xN array, with min, max of each dimension in
  columns 1 and 2, respectively.

  ex:
  data          range output
  [[1,5],         [[0,2],
   [2,5],   =>     [4,9]]
   [0,4],
   [1,9]]
  '''
  extents = numpy.ones([numpy.shape(coords)[1],2])
```

```
  for dim in range(numpy.shape(coords)[1]):
    data = coords[:,dim]
    dataRange = data.max() - data.min()
    bufsize = dataRange*(B/100.0)
    extents[dim,0] = data.min() - bufsize
    extents[dim,1] = data.max() + bufsize

  return extents
```

6.7 reproject

```
def reproject(llcoors,planet="earth",utmzone=-999,clon=-999,inverse=False):
  '''
  Reprojects long-lat data into transverse mercator coordinates
  with units of meters. Optionally, set inverse=True to change
  transverse mercator to long-lat.

  Input should be numpy 2-col long, lat array.
  Output will be numpy 2-col easting, northing array.

  Planet options: 'earth', 'venus', or 'mars'
  Earth requires a valid UTM zone
  Venus and Mars require a valid center longitude of the dataset

  Earth Transverse Mercator fit to WGS84 datum
  Venus Transverse Mercator fit to Spheriod of radius 6051800 m
  Mars Transverse Mercator fit to Spheriod of radius 3396190 m
  '''

  if planet == "earth":
    if (utmzone<1) or (utmzone>60):
      print "error in reproject:", \
        " utm zone not set correctly (1<=utmzone<=60)"
      return 0
    TransMerc = pyproj.Proj('+proj=utm +datum=WGS84 +zone='+str(utmzone))
  elif planet == "venus":
    if (clon<-360) or (clon>360):
      print "error in reproject: ", \
        "center longitude not set correctly (-360<=clon<=360)"
      return 0
    TransMerc = pyproj.Proj('+proj=tmerc +lat_0=0 +lon_0='+str(clon)+ \
      ' +k=0.9996 +x_0=0 +y_0=0 +a=6051800 +b=6051800 +units=m +no_defs')
  elif planet == "mars":
    if (clon<-360) or (clon>360):
      print "error in reproject: ",
```

```
        "center longitude not set correctly (-360<=clon<=360)"
      return 0
    TransMerc = pyproj.Proj('+proj=tmerc +lat_0=0 +lon_0='+str(clon)+ \
      ' +k=0.9996 +x_0=0 +y_0=0 +a=3396190 +b=3396190 +units=m +no_defs')
  else:
    print "error in reproject: planet not earth, venus, or mars."
    return 0

  if inverse==True:
    reproj = TransMerc(llcoors[:,0],llcoors[:,1])
    mcoors = numpy.transpose(reproj)
  else:
    reproj = TransMerc(llcoors[:,0],llcoors[:,1])
    mcoors = numpy.transpose(reproj)

  return mcoors
```

6.8   samse_bandwidth

```
def samse_bandwidth(coords):
  '''
  Evaluates the SAMSE Kernel in R using a coordinate list (coords).
  Returns 2x2 bandwidth covariance matrix.
  Requires: R, KS library in R.
  '''

  bandwidthfile='tmpbdR.out'
  datafile    ='tmpcrs.out'
  #Writes the data to a file for R
  numpy.savetxt(datafile,coords)

  #Writes the batch file that R will use
  with open('samse_batch.r','w') as f:
    f.write('library(ks)\n')
    f.write('data<-read.table("'+datafile+'")\n')
    f.write('bd <- Hpi(x=data,nstage=2,pilot="samse",pre="sphere")\n')
    f.write('sink("'+bandwidthfile+'")\n')
    f.write('show(bd)\n')
    f.write('sink()')

  #command to run the batch file
  os.system('R CMD BATCH samse_batch.r')

  #check for output file doesn't exist, fail
  if not os.path.isfile(bandwidthfile):
```

```
    print "error: Output from R was not successful."
    print "    Is R and the KS library installed?"
    return []


#Extract the bandwidth matrix from the bandwidth txt file
bandwidth = numpy.loadtxt(bandwidthfile,skiprows=1,usecols=(1,2))

#remove all these temporary files
os.remove('samse_batch.r')
os.remove('samse_batch.r.Rout')
os.remove(bandwidthfile)
os.remove(datafile)

return bandwidth
```