In[1]:= startTime = AbsoluteTime[];

Projecting a plane across a hillshade map, given the plane orientation and the location of one point along the plane

Plane-Projected-Across-Hillshade-Map.nb

This code is copyright © 2004-2017 by Vincent S. Cronin.

Modified from earlier drafts on 24 November 2017

---

**Input**

*Remember to change the names of the output files*

- **Input the location of a point on the plane**

  In the blue boxes below, input the latitude (`pointLat`), longitude (`pointLong`), and depth (`pointDepthKm`) below sea level in km of one point along the plane.

  
  In[2]:= pointLat = 39.4112;

  
  In[3]:= pointLong = -120.164;

  
  In[4]:= pointDepthKm = 4.03877;

- **Input the plunge and trend of the dip vector for the plane to be projected.**

  In the blue boxes below, input the dip trend (`planeDipTrend`) and dip angle (`planeDipPlunge`) of the plane, noting that the plane dip trend (trend of the dip vector) is an azimuth that is 90° clockwise from the right-hand-rule strike.

---

This code was written by Vince Cronin
In[5]:= planeDipTrend = 121.8288;

In[6]:= planeDipPlunge = 88.3864;

- **Input the DEM file that is used to make the hillshade map.**

1. Enter the following as an input line (in the blue box, below):

   ```mathematica
   mydata = Import[];
   ```

2. Put your cursor between the square brackets in the `Import[]` statement and click to establish the insertion point we will need in the next step.

3. Go to the **Insert** menu, select **File Path**, navigate to the correct input data file and choose it, and the correct file path will be inserted at the cursor. In this example, the Excel file is located on the desktop of Vince’s computer, and so Mathematica will insert the path 

   "\Users\vincecronin\Desktop\thinDVFZCrop.dat"

   between the brackets. The path will be different for every different file, and on every different computer.

   ```mathematica
   mydata = Import["\Users\vincecronin\Desktop\thinDVFZCrop.dat"];```

   You can make sure it is an input line by clicking on that line’s bracket on the far right edge of this window, going to the **Format** menu, selecting the **Style** submenu, and then choosing **Input**

   ```mathematica
   In[7]:= mydata = Import["\Users\vincecronin\Desktop\thinDVFZCrop.dat"];```

   This won’t work unless you follow directions and add the appropriate input line in the blue box above this line. Your input line should look something like the example:

   ```mathematica
   mydata = Import["\Users\vincecronin\Desktop\thinDVFZCrop.dat"];```

- **Input selected parameters related to the map area**

   The **zoneMeridian** is the longitude at the center of the UTM zone that contains the epicentral region. If the epicenter is near a zone boundary, base the UTM grid for this analysis in the zone that is to the west. The zone meridian for the Tahoe studies is -123

   ```mathematica
   In[8]:= zoneMeridian = -123;
   ```

   Input the geographic coordinates of the corners of the map area. Input the corners as a quadrangle.
The `widthFactor` is a multiplier that allows for the adjustment of the half-width of the zone between parallel planes that generate the trace of the mean nodal plane on the maps generated by this code, in meters. A value of 1.5 for the `widthFactor` sets the distance between the planes at 1.5 times the cell size of the DEM.

```
In[13]:= widthFactor = 1.5;
```

The `topoBase` value forces one point of the DEM to have a particular value, which in turn allows the user to adjust the tonal range of the hillshade map.

```
In[14]:= topoBase = 0;
```

The constant `minusOne` is used as input for the pointEvaluator module, in the place for the null value.

```
In[15]:= minusOne = -1;
```

The `meanMultipler` increases the width of the zone between parallel planes around the mean nodal plane, and is used to simplify interpretation of the seismo-lineament. Care must be taken so that the meanMultipler value does not become so large that the ground-surface trace of the resulting planes extends beyond the boundaries of the seismo-lineament.

```
In[16]:= meanMultipler = 1;
```

### Some user-defined functions

```
In[17]:= makeVector[plunge_, trend_] := {Cos[plunge Degree] Sin[trend Degree],
          Cos[plunge Degree] Cos[trend Degree], -Sin[plunge Degree]};

In[18]:= vectorNorm[x_] := Sqrt[x.x];

In[19]:= unitVector[x_] :=
          {x[[1]] / vectorNorm[x], x[[2]] / vectorNorm[x], x[[3]] / vectorNorm[x]};

In[20]:= vectorAngle[a_, b_] := ArcCos[a.b / (vectorNorm[a] vectorNorm[b])];
```

The lat/long to UTM conversion is after Snyder (1982), and assumes use of the North American Datum of 1927 (NAD27). Another excellent resource for converting from lat/long to UTM and back, using a variety of datums (e.g., NAD27, WGS84 and so on) is provided by Steven Dutch at http://www.uwgb.edu/dutchs/UsefulData/UTM-Formulas.htm, and by an online calculator at http://www.rcn.montana.edu/resources/tools/coordinates.aspx
In[21]:= convertToUTM[inLat_, inLong_, centMerid_] := Module[
  {c1, c2, c3, c4, c5, v1, v2, v3, v4, v5, v6, utmX, utmY},
  c1 = 6378206.4;
  c2 = 0.00676866;
  c3 = 0;
  c4 = centMerid;
  c5 = 0.9996;
  v1 = c2 / (1 - c2);
  v2 = c1 / Sqrt[1 - (c2 * (Sin[inLat Degree]^2))];
  v3 = Tan[inLat Degree]^2;
  v4 = v1 * (Cos[inLat Degree]^2);
  v5 = (Cos[inLat Degree]) * (inLong - c4) / 180;
  v6 = (11132.0894 * inLat) - (16076.94 * Sin[2 * inLat Degree]) +
    (17.21 * Sin[4 * inLat Degree]) - (0.02 * Sin[6 * inLat Degree]);
  utmX = (c5 * v2 * (v5 + ((((1 - v3 + v4) * v5^2) / 6) +
      ((5 - (18 * v3) + (v3^2) + (72 * v4) - (58 * v1)) * v5^2) / 120)) * 500000;
  utmY = (c5 * v6 - 0 + (v2 * Tan[inLat Degree]) *
    (v5^2 / 2) + (((5 - v3 + (9 * v4) + (4 * (v4^2)) * v5^2)) / 24) +
    (((61 - (58 * v3) + (v3^2) + (600 * v4) - (330 * v1)) * v5^2) / 720)));
  {utmX, utmY};
}

The following module differentiates between points that are within "width" meters from the plane and those that are further away. Given unit vector N that is normal to the plane that passes through the origin of the coordinate system, the distance from an arbitrary point (whose position vector is P) to that plane is given by IN-PI.

In[22]:= pointEvaluator[xCoord_, yCoord_, zCoord_, width_, fltUNrml_, nulData_] := Module[{locVect, distToFlt, result},
  locVect = {xCoord, yCoord, zCoord};
  distToFlt = Abs[Dot[fltUNrml, locVect]];
  result = If[(distToFlt <= width), 10., nulData];
  result];

The module findHalfWidth is used to determine the half-width of the uncertainty envelope given the lengths of the semi-axes of the triaxial uncertainty ellipsoid around the hypocenter, the azimuth of the greatest horizontal semi-axis, and the dip azimuth and dip angle of a nodal plane. The uncertainty envelope is bounded by parallel planes that are tangent to the uncertainty ellipsoid.

Input Parameters
• dAz is the trend or azimuth of the dip vector of the nodal plane, measured in degrees clockwise from north. All input azimuths are measured relative to the geographic coordinate system. Hence, an azimuth of 90° corresponds to due east.
• dAng is the plunge of the dip vector of the nodal plane, measured in degrees down from horizontal. Hence, a dip angle of 90° corresponds to a vertical vector pointing down.
• hEr1 is the length of the a semi-axis of the 95% CI uncertainty ellipsoid surrounding the mean hypocenter, measured in kilometers. Hence, a value of 1 for hEr1 means that the surface of the 95% CI uncertainty ellipse is 1 km from the mean hypocenter measured horizontally in the direction hEr1Az.
• hEr1Az is the trend of the e1h uncertainty axis (i.e., of the a semi-axis of the 95% CI uncertainty ellipsoid surrounding the mean hypocenter), measured in degrees clockwise from north. All input azimuths are measured relative to the geographic coordinate system. Hence, an azimuth of 90° corresponds to due east.
• hEr2 is the length of the b semi-axis of the 95% CI uncertainty ellipsoid surrounding the mean hypocenter, measured in kilometers. Hence, a value of 1 for hEr2 means that the surface of the 95% CI uncertainty ellipse is 1 km from the mean hypocenter measured horizontally 90° from the direction hEr1Az.
• vEz is the length of the c semi-axis of the 95% CI uncertainty ellipsoid surrounding the mean hypocenter, measured in kilometers. Hence, a value of 1 for vEz means that the surface of the 95% CI uncertainty ellipse is 1 km from the mean hypocenter measured vertically.

Local Variables
• var1 is the angle, in degrees, through which the geographic coordinate system (in which north is {0,1,0}) and
east is \{(1,0)\} is transformed by rotation around the common z axis to the coordinate system that is fixed to the axes of the triaxial uncertainty ellipsoid around the hypocenter (in which the x axis coincides with the a ellipsoidal axis and with hEr1, the y axis coincides with the b ellipsoidal axis and with hEr2, and the z axis is vertical and coincides with the c ellipsoidal axis and with vEr).

- **var2** is the unit vector parallel to the dip vector, in the geographic coordinate system in which north is \(\{0,1,0\}\) and east is \(\{1,0,0\}\).
- **var3** is the unit vector parallel to the right - hand - rule strike, in the geographic coordinate system in which north is \(\{0,1,0\}\) and east is \(\{1,0,0\}\).
- **var4** is the 3x3 rotation matrix in which a positive angle results in an anticlockwise rotation around the positive z axis.
- **var5** is the unit vector that is normal to the nodal plane in the geographic coordinate system in which north is \(\{0,1,0\}\) and east is \(\{1,0,0\}\).
- **var6** is the unit vector that is normal to the nodal plane in the ellipsoid-centered coordinate system that is rotated \(\text{var1}\) degrees around the vertical (z) axis.
- **var7–9** are intermediate values that are ultimately used to compute the value of the Lagrangian multiplier lambda.
- **var10** is the Lagrangian multiplier lambda
- **var11** is the set of components of the location vector to one of the tangent points, expressed in the ellipsoid-centered coordinate system that is rotated \(\text{var1}\) degrees around the vertical (z) axis.
- **var12** is the distance, in km, between the a nodal plane through the hypocenter and a parallel nodal plane through one of the tangent points to the triaxial uncertainty ellipsoid around the hypocenter.

**Output**

- **result** is the distance, in km, between a nodal plane through the hypocenter and a parallel nodal plane through one of the tangent points to the triaxial uncertainty ellipsoid around the hypocenter.

```math
\text{In}[23]:= \text{findHalfWidth}[\text{dipAz}_-, \text{dipAng}_-, \text{hEr1}_-, \text{hEr1Az}_-, \text{hEr2}_-, \text{vEr}_-] := \\
\text{Module}[\{\text{var1}, \text{var2}, \text{var3}, \text{var4}, \text{var5}, \text{var6}, \text{var7}, \text{var8}, \\
\text{var9}, \text{var10}, \text{var11}, \text{var12}, \text{result}\}, \text{var1} = \text{hEr1Az} - 90; \\
\text{var2} = \{\text{Cos}[\text{dipAng} \text{Degree}] \text{Sin}[\text{dipAz} \text{Degree}], \\
\text{Cos}[\text{dipAng} \text{Degree}] \text{Cos}[\text{dipAz} \text{Degree}], -\text{Sin}[\text{dipAng} \text{Degree}]\}; \\
\text{var3} = \{\text{Sin}[\{\text{dipAz} - 90\} \text{Degree}], \text{Cos}[\{\text{dipAz} - 90\} \text{Degree}], 0\}; \\
\text{var4} = \{\{\text{Cos}[\text{var1} \text{Degree}], -\text{Sin}[\text{var1} \text{Degree}], 0\}, \\
\{\text{Sin}[\text{var1} \text{Degree}], \text{Cos}[\text{var1} \text{Degree}], 0\}, \{0, 0, 1\}\}; \\
\text{var5} = \text{Cross}[\text{var2}, \text{var3}]; \\
\text{var6} = \text{var4}.\text{var5}; \\
\text{var7} = (1 / \text{hEr1}^2) (\text{var6}[\{1\}] (\text{hEr1}^2 / 2))^2; \\
\text{var8} = (1 / \text{hEr2}^2) (\text{var6}[\{2\}] (\text{hEr2}^2 / 2))^2; \\
\text{var9} = (1 / \text{vEr}^2) (\text{var6}[\{3\}] (\text{vEr}^2 / 2))^2; \\
\text{var10} = \sqrt{(1 / (\text{var7} + \text{var8} + \text{var9}))}; \\
\text{var11} = \{\text{var10} * \text{var6}[\{1\}] * (\text{hEr1}^2 / 2), \\
\text{var10} * \text{var6}[\{2\}] * (\text{hEr2}^2 / 2), \text{var10} * \text{var6}[\{3\}] * (\text{vEr}^2 / 2)\}; \\
\text{var12} = \text{var6}.\text{var11}; \\
\text{result} = \text{var12};
```

**Computation**

- **Read and interpret the DEM header information**

```math
\text{In}[24]:= \text{headerData} = \text{Table}[\text{mydata}[\{i, j\}], \{i, 6\}, \{j, 2\}];
```

*This code was written by Vince Cronin*
In[25]:= ncols = headerData[[1, 2]]
Out[25]= 938

In[26]:= nrows = headerData[[2, 2]]
Out[26]= 1164

In[27]:= xllcorner = headerData[[3, 2]]
Out[27]= 732 209.

In[28]:= yllcorner = headerData[[4, 2]]
Out[28]= 4.35602 × 10^6

In[29]:= cellsize = headerData[[5, 2]]
Out[29]= 28.1312

In[30]:= nodataValue = headerData[[6, 2]]
Out[30]= -9999

- **Read and interpret the point location data**

The UTM coordinates that specify the horizontal position of the earthquake focus (i.e., the epicenter location) are given in meters. The variable focalDepth gives the depth of the earthquake focus, in meters.

In[31]:= pointDepth = pointDepthKm * (-1000)
Out[31]= -4038.77

The reported vertical and horizontal uncertainties in the location of the earthquake focus are given in kilometers. The variables horizError and vertError convert these to meters.

In[32]:= eh1 = 0.01;
In[33]:= eh2 = 0.01;
In[34]:= eh1Az = 0.01;
In[35]:= ez = 0.01;

- **Convert the input coordinates of the earthquake focus to the same coordinate system as the DEM data**

In[36]:= utmCoordinates = convertToUTM[[pointLat, pointLong, zoneMeridian]];

The "focus" is the (x, y, z) coordinates in meters of the reported earthquake focus, in a coordinate system in which the origin is the lower-left (southwest) corner of the DEM at sea level.

In[37]:= pointMean = {utmCoordinates[[1]] - xllcorner, (utmCoordinates[[2]] - yllcorner), pointDepth};

In[38]:= rawElevDataM = Table[mydata[[i + 6, j]] - pointMean[[3]], {i, nrows}, {j, ncols}];

*This code was written by Vince Cronin*
```
In[39]:= demImageFileM = Table[rawElevDataM[[i, j]], {i, nrows}, {j, ncols}];

- Adjustment for input geographic coordinates based on true north to the relevant UTM grid north

In[40]:= midLat = ((swLat - nwLat) / 2) + swLat;
In[41]:= midLong = ((seLong - swLong) / 2) + swLong;

The gridNorthAdjustment is the angle between true north and grid north at/near the center of the map area. If grid north is found by a clockwise rotation from true north, the sign of the gridNorthAdjustment is negative; otherwise, it is positive. The gridNorthAdjustment is approximately equal to (zoneMeridian – (longitude at center of map area)) * \[\sin(\text{latitude at center of map area})\].

In[42]:= gridNorthAdjustment = (zoneMeridian - (midLong)) * Sin[midLat Degree]
Out[42]= -1.82769

In[43]:= planeDipTr = If((planeDipTrend + gridNorthAdjustment) < 0),
   (360 + planeDipTrend + gridNorthAdjustment),
   If((planeDipTrend + gridNorthAdjustment) > 360),
   (planeDipTrend + gridNorthAdjustment - 360),
   (planeDipTrend + gridNorthAdjustment));
In[44]:= rawElevDataInput = Table[mydata[[i + 6, j]], {i, nrows}, {j, ncols}];
In[45]:= elev = Table[demImageFileM[[i, j]], {i, nrows, 1, -1}, {j, ncols}];
In[46]:= planeDipTrRad = planeDipTr * (\[Pi] / 180);
In[47]:= lightDirection = 
   If[planeDipTrRad > (3 \[Pi] / 2), (2 \[Pi] - planeDipTrRad, (\[Pi] / 2) - planeDipTrRad];

- Make the hillshade image

In[48]:= hillshadeMap1 = ReliefPlot[elev, ColorFunction -> "GrayTones", LightingAngle -> {lightDirection, \[Pi] / 12}];

- Find the intersection of the plane with the hillshade map

Convert the fault dip vector from trend and plunge to a unit location vector

In[49]:= planeDipVect = unitVector[makeVector[planeDipPlunge, planeDipTr]]; Find the strike vector defined using the right-hand rule

In[50]:= planeStrike = If[(planeDipTr < 90), (planeDipTr + 270), (planeDipTr - 90)];
In[51]:= planeStrikeVect = {Sin[planeStrike Degree], Cos[planeStrike Degree], 0}; Find the unit vector that is normal to the fault plane and is directed upwards

In[52]:= planeNormalVect = unitVector[Cross[planeDipVect, planeStrikeVect]]; In[53]:= zoneHalfWidthMThick = widthFactor * cellsize * Sin[planeDipPlunge Degree];
```

This code was written by Vince Cronin
In[54]:= N[42.1801]

Out[54]= 42.1801

In[55]:= uncertSwath = Table[
  pointEvaluator[
    (cellsize*(j - 1) - pointMean[[1]])
    ,
    (cellsize*(nrows - 1) - pointMean[[2]]),
    demImageFileM[[i, j]],
    zoneHalfWidthMThin, planeNormalVect, minusOne], {i, nrows}, {j, ncols}];

In[56]:= elevData = Table[uncertSwath[[i, j]], {i, nrows, 1, -1}, {j, ncols}];

In[57]:= traceImgFileMThin = ListContourPlot[elevData, ContourShading -> False,
  AspectRatio -> Automatic, Contours -> {topoBase}];

Output

In[58]:= outFile1 = Show[hillshadeMap1, traceImgFileMThin];

In[59]:= Show[]

Out[59]=

- Export data and image files

The file created in the next line is JPEG image that contains the graphic showing the hillshade image developed from the DEM, curves marking the boundaries of the uncertainty regions, and a circle with 1 km diameter centered on the epicenter.

This code was written by Vince Cronin
IMPORTANT NOTE: The user must supply a name for the output file that is different from existing file names, or else the existing files may be over-written.

```
In[60]:= Export["/Users/vincecronin/Desktop/PointMeanPlane-1.jpg", outFile1];
```

- How long did this program take to run to this point, in minutes?

```
In[61]:= minutesForProcessing1 = (AbsoluteTime[] - startTime) / 60
Out[61]= 0.73844442
```

References


This code was written by Vince Cronin