Quick Contour with GEOmap

Jonathan M. Lees
University of North Carolina, Chapel Hill
Department of Geological Sciences
CB #3315, Mitchell Hall
Chapel Hill, NC 27599-3315
email: jonathan.lees@unc.edu
ph: (919) 962-0695

May 17, 2011

1 Setting up R

This set of routines is intended to show how to make a quick contour map using GEOmap, a package written in **R** . **R** and package GEOmap can be downloaded from the internet at:

`http://www.r-project.org/`.

You can install **R** on most computer systems, including Windows, MacOS-X and UNIX/LINUX. Once **R** is installed, the GEOmap package should by downloaded. This can be done via menu buttons or simply by starting **R** and informing **R** to install:

```
install.packages(``GEOmap'')
or
install.packages("GEOmap", contriburl = "http://streaming.stat.iastate.edu/CRAN/src/contrib"
```

If you do not specify, **R** will ask you to select a "mirror" site where the software will extracted. There are other commands you might use to manage packages, for example: 'update.packages', 'available.packages'....etc.

Once this is set, you must tell the current session you are going to use GEOmap,

```
> library(GEOmap)
```

2 Plotting LAT-LON maps with GEOmap

Suppose we have a set of LAT-LON pairs. We want to make a contour or an image of these data. First we have to make sure the coordinate system we

are working with is Cartesian. To get this we have to project the LAT-LON coordinates, which describe positions on the globe, onto a flat plane.

There are many ways to project from spherical coordinates to a plane. n GEOmap one can find the available projections by typing:

```
> projtype()
```

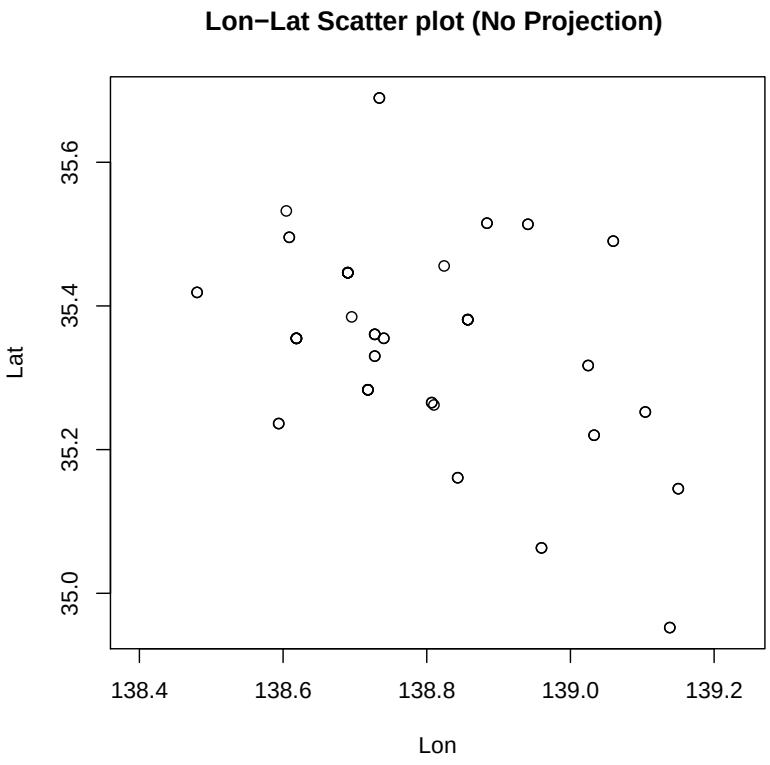
There are other packages in **R** that can project data. See maptools. In this vignette, however, I will illustrate GEOmap programs.

First I read in some data from a file. This is a station file from Japan. It has station name, Lat, Lon, and elevation. We want to contour the elevations.

```
> sta = scan(file = "newFUJIstation.LLZ", what = list(name = "",
     lat = 0, lon = 0, z = 0))
```

Next we plot the Lon-Lat pairs. This is wring, we know, becasue longitude and latitude are are not cartesian, they do not have equal units. Sometimes people plot these as is and it looks okay, but if you want to make calculations it will be wrong.

```
> plot(sta$lon, sta$lat, asp = 1, xlab = "Lon", ylab = "Lat")
> title("Lon-Lat Scatter plot (No Projection) ")
```



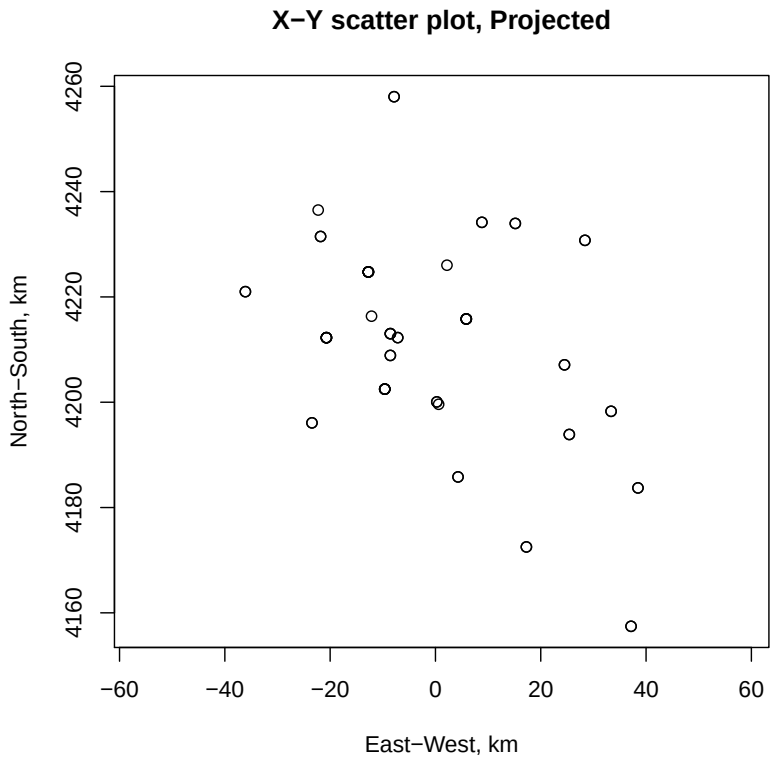
To remedy this we project the lat-lon pairs using a UTM projection. In GEOmap it is possible to set the origin to a specific Lat-Lon pair and project all the data from that point. All data will be in units of km relative to the origin. Since this is a projection, remember that far from the origin there may be distortion. Near the origin, however, the distortion should be small. The number 2 projection is spherical, if you need a projection that is more accurate, try an ellipsoidal projection.

The origin may be arbitrary. In this example we use the mean Lat-Lon. If you use this method, be sure the lons are all on the same hemisphere.

```
> CENLAT = mean(sta$lat)
> CENLON = mean(sta$lon)
> PROJ = setPROJ(type = 1, LAT0 = CENLAT, LON0 = CENLON)
> G = GLOB.XY(sta$lat, sta$lon, PROJ)
```

Now we are ready to plot the projected data as X-Y in km.

```
> plot(G$x, G$y, asp = 1, xlab = "East-West, km", ylab = "North-South, km")
> title("X-Y scatter plot, Projected")
```

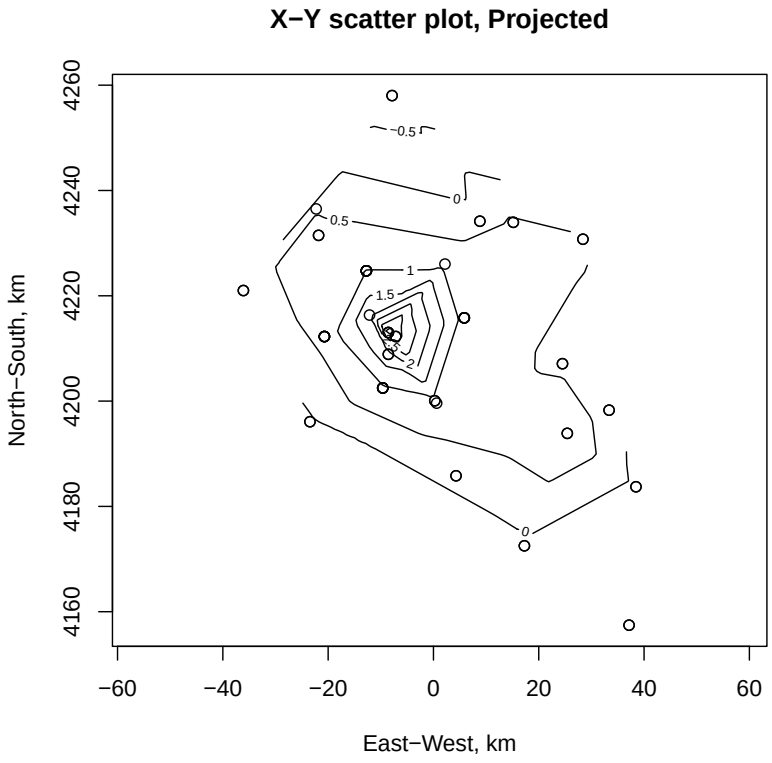


Nex we want to add contors to the plot. First we have to interpolate the data to a regular grid. There are several programs **R** for interpolation of 2D scatter data. We will use a very simple linear interpolation. We use 100 points, but this is probably overkill for this small data set.

```
> xo = seq(from = min(G$x), to = max(G$x), length = 100)
> yo = seq(from = min(G$y), to = max(G$y), length = 100)
> Z = interp(G$x, G$y, sta$z, xo, yo, duplicate = "mean")
```

Finally we plot the points and overlay the contours. The contour program is fairly versatile, check the help page.

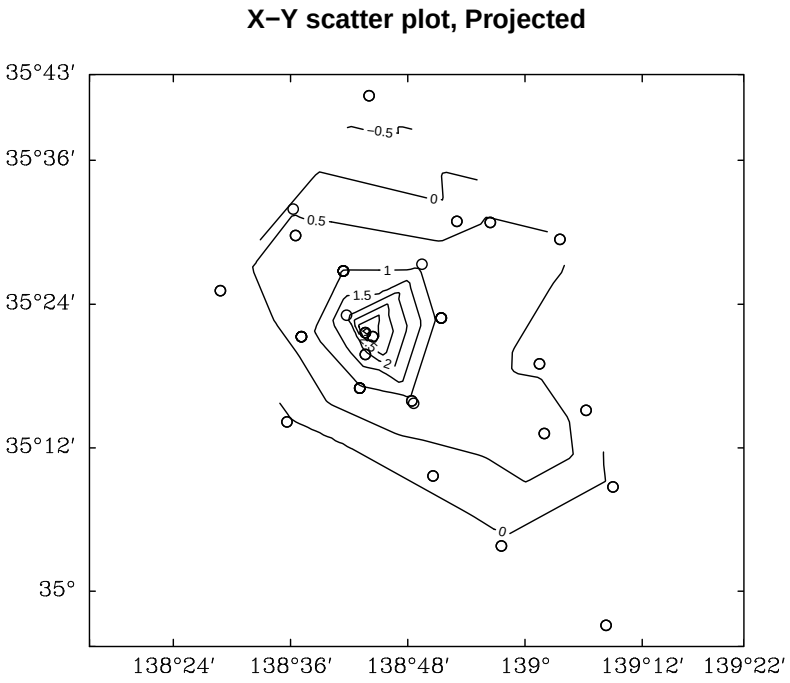
```
> plot(G$x, G$y, asp = 1, xlab = "East-West, km", ylab = "North-South, km")
> title("X-Y scatter plot, Projected")
> contour(Z, add = TRUE)
```



Note that the units and axes designation on this plot are in kilometers. This may or may not be desirable. One advantage is you do not need to add a scale for reference. You may, however, need to add a symbol signifying North-South. See function `NSarrow` to add this.

To add Lat-Lon ticks to an X-Y plot, first determine the outline of the plot using the `par('usr')` command. Get the Lat-Lon pairs for the perimeter and plot these. Here is an example:

```
> plot(G$x, G$y, asp = 1, ann = FALSE, axes = FALSE)
> title("X-Y scatter plot, Projected")
> contour(Z, add = TRUE)
> u = par("usr")
> LowerLeft = XY.GLOB(u[1], u[3], PROJ)
> UpperRight = XY.GLOB(u[2], u[4], PROJ)
> A = list(LON = range(c(LowerLeft$lon, UpperRight$lon)), LAT = range(c(LowerLeft$lat,
    UpperRight$lat)))
> PLAT = pretty(A$LAT)
> PLAT = c(min(A$LAT), PLAT[PLAT > min(A$LAT) & PLAT < max(A$LAT)],
    max(A$LAT))
> PLON = pretty(A$LON)
> PLON = c(min(A$LON), PLON[PLON > min(A$LON) & PLON < max(A$LON)],
    max(A$LON))
> addLLXY(PLAT, PLON, PROJ = PROJ, LABS = TRUE, PMAT = NULL, TICS = c(0.01,
    0.01), GRID = FALSE)
> box()
```



The units of this plot are kilometers, but the axes are labeled with Lat-Lon indications. Here you may want to add a bar showing the geographic scale. See function `zebra` for that purpose.

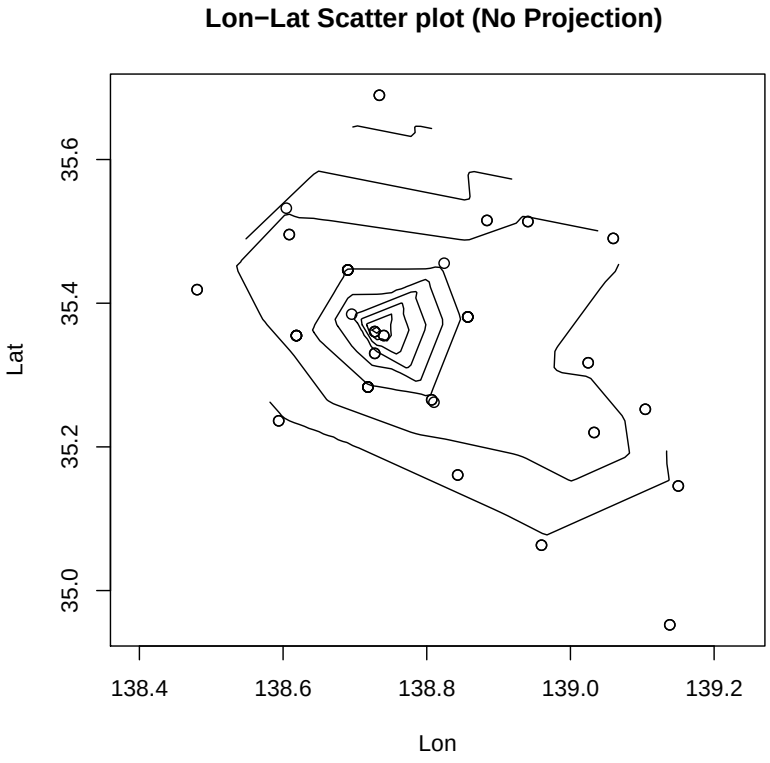
3 Convert Contour Lines Back to Lat-Lon

We may want to save the contours and convert them back to Lat-Lon.

```
> CLINE = contourLines(Z)
> length(CLINE)
> LLlines = list()
> for (i in 1:length(CLINE)) {
     v = XY.GLOB(CLINE[[i]]$x, CLINE[[i]]$y, PROJ)
     LLlines[[i]] = v
 }
```

Just to check, lets plot the original data as Lon-Lat and add the contours we just extracted. We know this is not correct, but we do it for fun.

```
> plot(sta$lon, sta$lat, asp = 1, xlab = "Lon", ylab = "Lat")
> title("Lon-Lat Scatter plot (No Projection) ")
> for (i in 1:length(LLlines)) {
     lines(LLlines[[i]]$lon, LLlines[[i]]$lat)
 }
```



4 Kamchatka

This example may be a little more illuminating since in this case the plot is in the far north and projections make a significant difference.

Read in depths to the top of the subducting Pacific Plate in the Kamchatka Subduction zone, based on earthquakes from global catalogues.

```
> KAM = scan(file = "KAM_slab_top", what = list(lat = 0, lon = 0,
     depth = 0))
```

In this case we will want to plot a map of the region along with the data. So we choose in advance the Lat-Lon bounds of the plot.

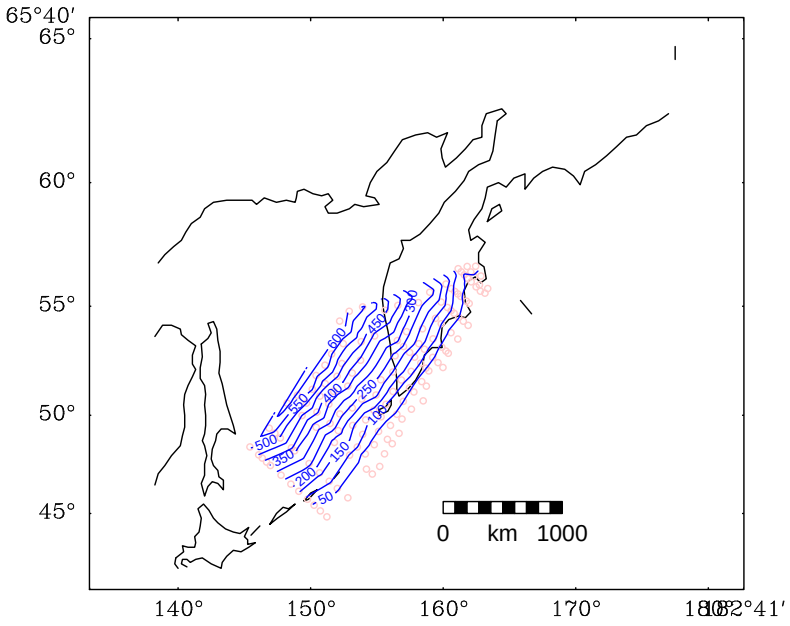
```
> data(worldmap)
> LAT = c(42, 65)
> LON = c(138, 178)
> CENLAT = mean(LAT)
> CENLON = mean(LON)
> PROJ = setPROJ(type = 1, LAT0 = CENLAT, LON0 = CENLON)
> B = GLOB.XY(LAT, LON, PROJ)
> G = GLOB.XY(KAM$lat, KAM$lon, PROJ)
> xo = seq(from = min(G$x), to = max(G$x), length = 200)
> yo = seq(from = min(G$y), to = max(G$y), length = 200)
> Z = interp(G$x, G$y, KAM$depth, xo, yo)
```

To get a scale on the plot we can calculate the zebra position.

```
> zeblat = 45
> zeblon = 160
> ZEBloc = GLOB.XY(zeblat, zeblon, PROJ)
```

Now we can make the plot. It is important to remember the argument `asp1=` to make sure the plot has the correct aspect ratio.

```
> plot(B$x, B$y, type = "n", ann = FALSE, axes = FALSE, asp = 1)
> u = par("usr")
> LowerLeft = XY.GLOB(u[1], u[3], PROJ)
> UpperRight = XY.GLOB(u[2], u[4], PROJ)
> A = list(LON = range(c(LowerLeft$lon, UpperRight$lon)), LAT = range(c(LowerLeft$lat,
    UpperRight$lat)))
> PLAT = pretty(A$LAT)
> PLAT = c(min(A$LAT), PLAT[PLAT > min(A$LAT) & PLAT < max(A$LAT)],
    max(A$LAT))
> PLON = pretty(A$LON)
> PLON = c(min(A$LON), PLON[PLON > min(A$LON) & PLON < max(A$LON)],
    max(A$LON))
> addLLXY(PLAT, PLON, PROJ = PROJ, LABS = TRUE, PMAT = NULL, TICS = c(0.1,
    0.1), GRID = FALSE)
> box()
> plotGEOmapXY(worldmap, LIM = c(LON[1], LAT[1], LON[2], LAT[2]),
    PROJ = PROJ, add = TRUE)
> points(G$x, G$y, col = rgb(1, 0.8, 0.8), cex = 0.6)
> contour(Z, add = TRUE, col = "blue")
> zebra(ZEBloc$x, ZEBloc$y, 1000, 100, 100, lab = "km")
```


65°40′
65°
60°
55°
50°
45°
140°
150°
160°
170°
0 km 1000