Digitize a JPEG figure using **R**

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) 962-1562

January 20, 2016

1 Introduction

This tutorial is for researchers who need to extract digital data from images. If yo need to digitize the information on a jpeg image you can use the codes described here to get the data.

For example, if there is a figure in an older journal that does not have a table of the corresponding data, you can get the data from the figure by digitizing the figure.

The trick is to find a function that converts the pixels to figure coordinates. This is done by estimating a (linear) function that scales the pixels in X and Y.

First read in the images, in this case they are stored in jpeg format. If they are stored in tiff format, you may need another R-package to perform the I/O. I used the unix program *convert* to convert the tiff file stored in the grab program on the MAC to a jpeg file.

The steps for digitizing a figure follow this procedure:

1. Convert figure to JPEG format
2. Start R
3. Load libraries: jpeg, GEOmap
4. Set coordinates of the axes (digitize these)
5. Digitize points and lines
6. Convert to figure coordinates

Start by reading in the list of files and setting up the libraries:

```
library(jpeg)
library(RPMG)
library(RFOC)
source("./digitize.R")
dir1 = './DATA'
LF =  list.files(path=dir1, pattern="jpg", full.names=TRUE)
```

2 Example 1: Data from Map

The first example is a plot of contours of the Phillipine Sea Plate as it subducts below Japan. The authors plotted several contours, although in this excersize we only need the new version highlighted in their paper.

Nakajima, J., Hirose, F., and Hasegawa, A., Seismotectonics beneath the Tokyo metropolitan area, Japan: Effect of slab-slab contact and overlap on seismicity, *J. Geophys. Res.*, 2009, 114.

```
g2 = grep("PHP", LF)
jpgMAP = LF[g2]
 myimage=readJPEG(jpgMAP  )
######  use this if you are doing an interactive plot
```

```
## X11()
###  plot( myimage )

 KALL = list()
    ATTS = attributes(myimage)
    DX = c(0,ATTS$dim[2])
    DY = c(0, ATTS$dim[1])
mfix = max(c(DX, DY))
plot(c(DX[1],DX[2])/mfix, c(DY[1],DY[2])/mfix , type='n', asp=1)
rasterImage(myimage, DX[1]/mfix,DY[1]/mfix, DX[2]/mfix,DY[2]/mfix )
```

```
JPNG(file="./FIGS/mapplot1.png", width=14, height=12)
par(mai=c(0,0,0,0))
mfix = max(c(DX, DY))
    plot(c(DX[1],DX[2])/mfix, c(DY[1],DY[2])/mfix , type='n', asp=1)
rasterImage(myimage, DX[1]/mfix,DY[1]/mfix, DX[2]/mfix,DY[2]/mfix )
###  rasterImage(myimage, 1.2, 1.27, 1.8, 1.7)

rect(0,0, DX[2]/mfix, DY[2]/mfix)
dev.off()
```

Since this is a tutorial, we do not have interaction. In that case I just assign the locations:

This loads the data already digitized

```
load("KALL.RDATA")
```

Next we plot the jpeg file and put the digitized coordinates on with arrows pointing to them so you can see:

```
JPNG(file="./FIGS/mapplot2.png", width=14, height=12)
 ## plot(myimage)

par(mai=c(0,0,0,0))
```

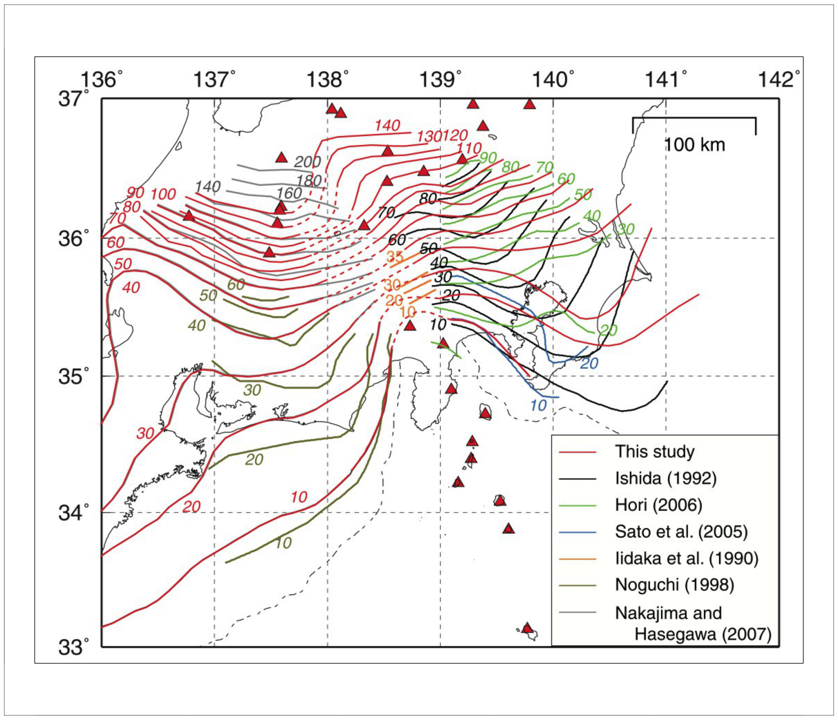

Figure 1: Plot of figure from the Nakajima paper. We will focus on digitizing the contours that are from “This Study”, i.e. the red lines.

```
 plot(c(DX[1],DX[2])/mfix, c(DY[1],DY[2])/mfix , type='n', asp=1)
rasterImage(myimage, DX[1]/mfix,DY[1]/mfix, DX[2]/mfix,DY[2]/mfix )
rect(0,0, DX[2]/mfix, DY[2]/mfix)
points(KALL[[1]]$x/mfix, KALL[[1]]$y/mfix , col='blue', pch=25, bg='red', fg='y
for(i in 1:length(L$x))
  {
fancyarrows( L$x[i]/mfix, L$y[i]/mfix , KALL[[1]]$x[i]/mfix ,    KALL[[1]]$y[
     thick =thick , headlength =  headlength,
     headthick =headthick, col ="gold" )
 alab = paste(i, ":", paste("(",KALL[[1]]$USERx[i], ",",
   KALL[[1]]$USERy[i], ")", sep=""), sep=""  )
  text(L$x[i], L$y[i], labels=alab, cex=2, col="purple",
       font=2, pos=tpos[i])
}
dev.off()
```

Next task is to digitize the points we are interested in saving:

```
  ###   normally we digitize the screen -
##       here we just load previous digits
load("KALL.RDATA")
if(FALSE)
  {
KALL = list()

KALL = digitize(KALL)

#####  next digitize the points
KALL = digitize(KALL)

####  you can repeat this many times
KALL = digitize(KALL)


###   set the USER coordinates for conversion
KALL[[1]]$USERx = c(136, 136, 136, 142 )
KALL[[1]]$USERy = c(33, 37,  37,  37)
}
PPL=list()
```

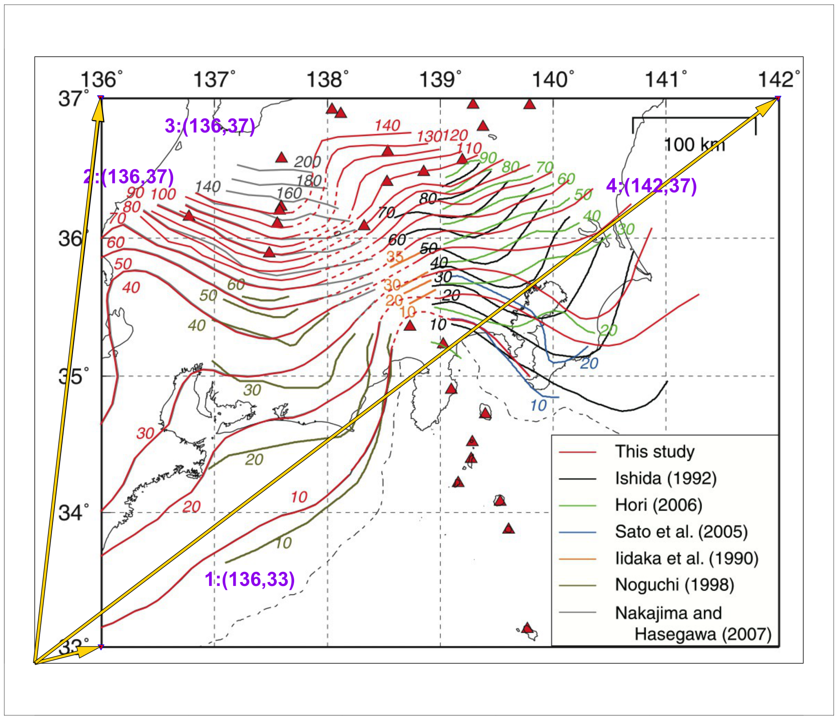

Figure 2: Plot of figure from the Nakajima paper.

```
PPL$x=c(487.904080403)
PPL$y=c(134.500202326)
```

We plot these and show where they are on the figure for clarity:

```
JPNG(file="./FIGS/mapplot3.png", width=14, height=12)
par(mai=c(0,0,0,0))
 plot(c(DX[1],DX[2])/mfix, c(DY[1],DY[2])/mfix , type='n', asp=1)
rasterImage(myimage, DX[1]/mfix,DY[1]/mfix, DX[2]/mfix,DY[2]/mfix )
rect(0,0, DX[2]/mfix, DY[2]/mfix)
for(i in 2:length(KALL))
{
points(KALL[[i]]$x/mfix,KALL[[i]]$y/mfix, col='blue', pch=25)
}
points( KALL[[2]]$x[5:8]/mfix, KALL[[2]]$y[5:8]/mfix, col='red', cex=3 )
arrows(PPL$x/mfix, PPL$y/mfix, KALL[[2]]$x[5:8]/mfix, KALL[[2]]$y[5:8]/mfix,
       length=0.1,col="purple",  lwd=2)
  RPMG::textrect(PPL$x/mfix, PPL$y/mfix, "Digitized points" ,
           textcol = "black", col = "white", border = "black",
     off = 0.06, brd = 0.06, pos = 4,  add=TRUE)
dev.off()
```

Next convert the points. This is the important part. we want to use the points we digitized on the axes to convert pixel coordinates to true data values. In this case we use the function RESCALE to perform the conversion, since we make the assumption that it is linear. There are situations where this assumption may not be true and in that case we may need to perform other calculations and stretching to get the correct transformation.

Finally, plot the points on a new figure with the new coordinates that have been converted to correct user coordinates.

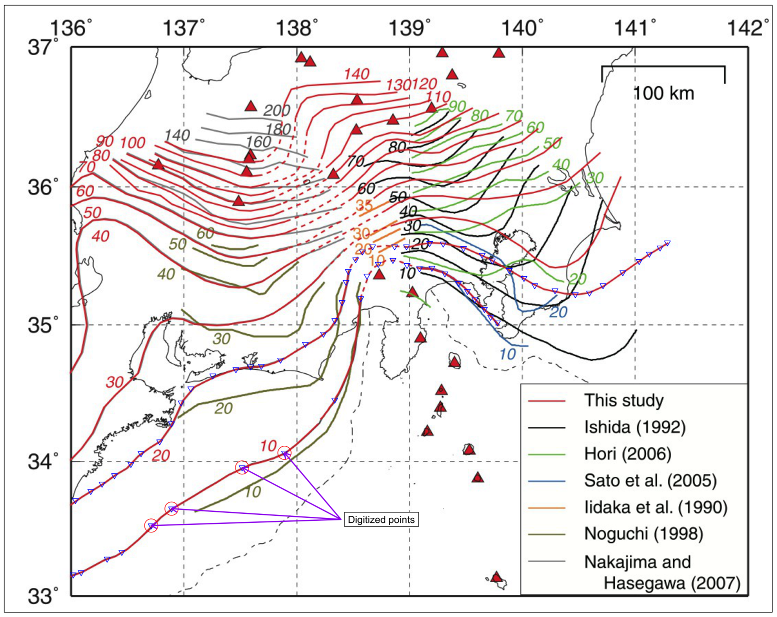

Figure 3: Plot of figure from the Nakajima paper. Some of the points that have been digitized are highlighted with arrows and red circles. The other digitized points are small blue triangles.

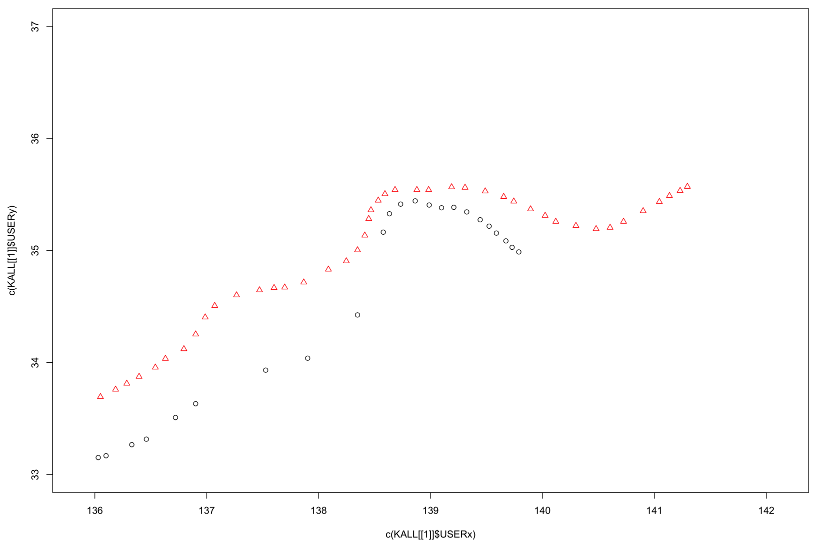

Figure 4: Plot of digitized points from Nakajima paper. These are just two of the contours shown in the figure. To plot them correctly and do more analysis, we will have to project them with geographic projection, e.g. UTM.

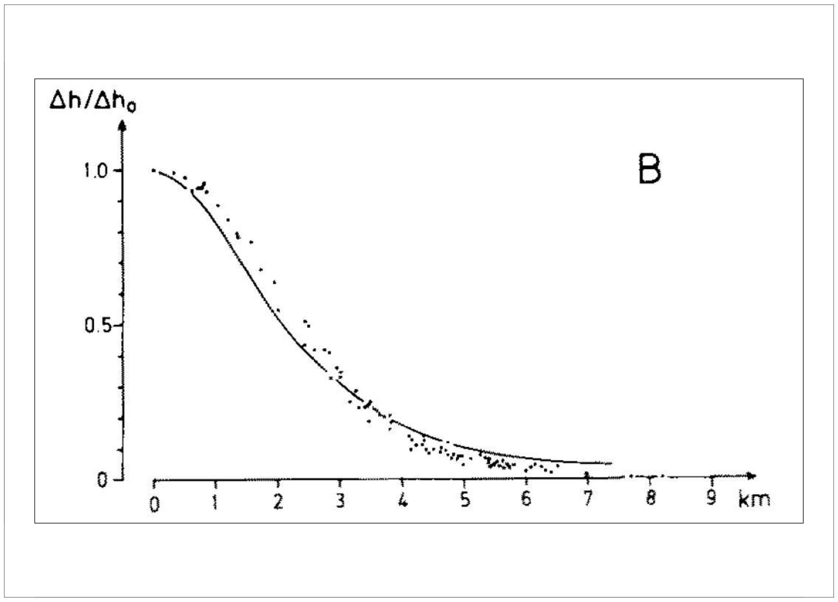

Figure 5: Plot of digitized points from Berrino paper. These points were published before digital data became very popular. We need the data to reproduce the modeling that is shown by the solid line in the figure.

3 Example 2: Data From a Graph

This case is an example of digitizing data from graph.

Data here is taken from this publication:

Berrino, G.; Corrado, G.; Luongo, G. and Toro, B. Ground deformation and gravity changes accompanying the 1982 Pozzuoli Uplift *Bulletin Volcanologique*, 1984, 47, 187-200

Plot this graph:

Do a few calculations to check the coordinates:

Plot the jpeg figure and the digitized points:

```
###   convert the points

PXY  =list(x=NULL, y=NULL)
for(i in 2:length(KALL))
{
```

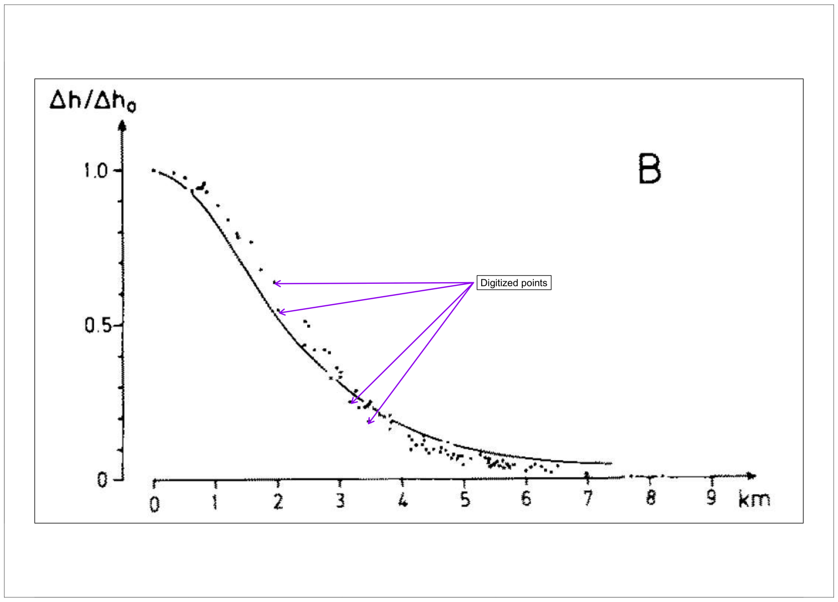

Figure 6: Plot of digitized points from Berrino paper.

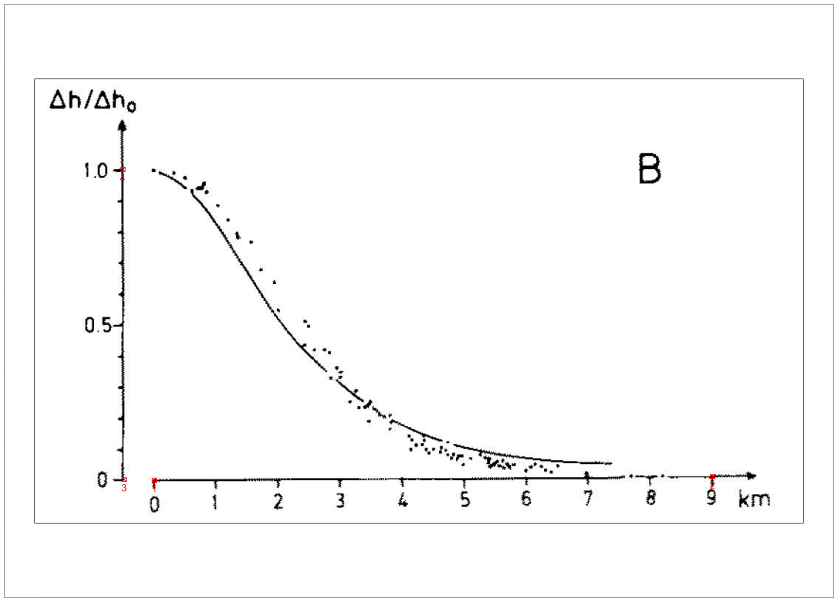

Figure 7: Plot of digitized points from Berrino paper.

```
####  points(KALL[[i]], col='purple', pch=6)

ddx = KALL[[i]]$x-KALL[[1]]$x[1]
ddy = KALL[[i]]$y-KALL[[1]]$y[1]
dd  = sqrt( (ddx)^2+(ddy)^2)


Kcosx =   (KX1[1]*ddx+KX1[2]*ddy)/(sqrt(KX1[1]^2+KX1[2]^2)*dd)

KDOTx = (KALL[[1]]$USERx[2]-KALL[[1]]$USERx[1])*Kcosx*dd/(KX1[1])

####print(KDOTx)

ddx = KALL[[i]]$x-KALL[[1]]$x[3]
ddy = KALL[[i]]$y-KALL[[1]]$y[3]
dd  = sqrt( (ddx)^2+(ddy)^2)


Kcosy =   (KY1[1]*ddx+KY1[2]*ddy)/(sqrt(KY1[1]^2+KY1[2]^2)*dd)

KDOTy = (KALL[[1]]$USERy[4]-KALL[[1]]$USERy[3])*Kcosy*dd/(KY1[2])

aXY = list(x=c(PXY$x, KDOTx), y=c(PXY$y, KDOTy))

PXY = aXY
}
##################
```

Finally the points were all digitized saved and stored for later use. Here we read in the final file and plot the data. These data are used for modeling Mogi sources on volcanoes.

```
JXY = scan(file="./DATA/PXY.BerrinoB.txt", skip=2, list(x=0, y=0))
JPNG(file="./FIGS/berrinoFIN.png", width=14, height=10)
plot(JXY, pch=6, col='purple', xlim=c(0,9), ylim=c(0, 1) )
dev.off()
```

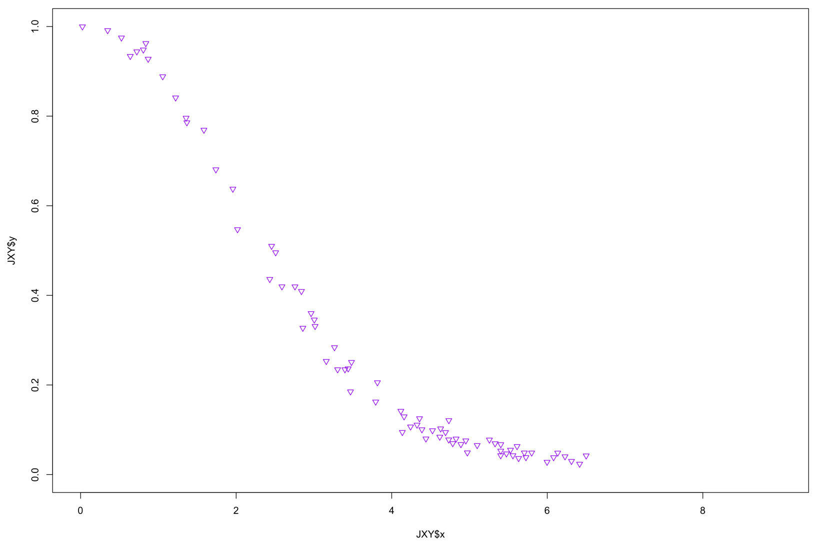

Figure 8: Final plot of digitized points from Berrino paper. These can be used for analysis. The Mogi model (depth and other magma chamber parameters) is derived by non-linear inversion of the data.

4 Appendix

This is a dump of the digitize function used above:

```
 print(digitize)
function (KALL)
{
    N = length(KALL)
    if (N < 1) {
        I = 1
        KALL[[I]] = list()
        print("Digitize points")
        p = locator(type = "p", col = "red", pch = 10)
        KALL[[I]] = p
        N = length(KALL)
        points(KALL[[I]], col = "blue", pch = 10)
        return(KALL)
    }
    for (i in 1:N) {
        if (length(KALL[[i]]$x) > 0) {
            points(KALL[[i]], col = "blue", pch = 10)
        }
    }
    p = locator(type = "p", col = "red", pch = 10)
    I = N + 1
    KALL[[I]] = list()
    KALL[[I]] = p
    return(KALL)
}
```