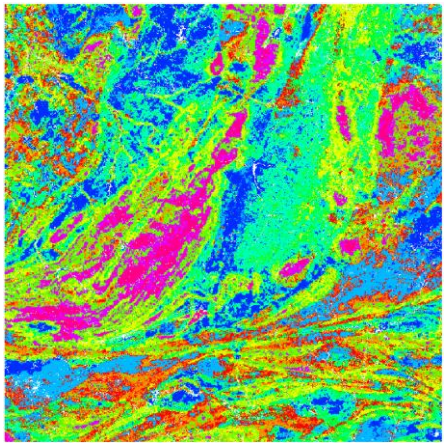
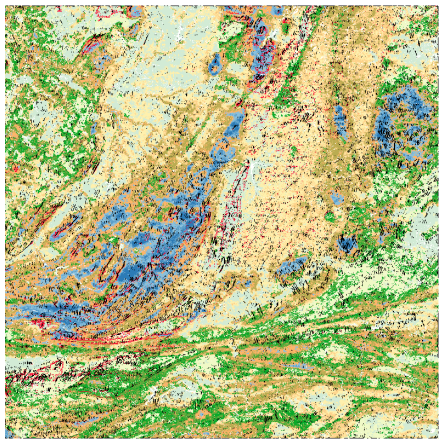
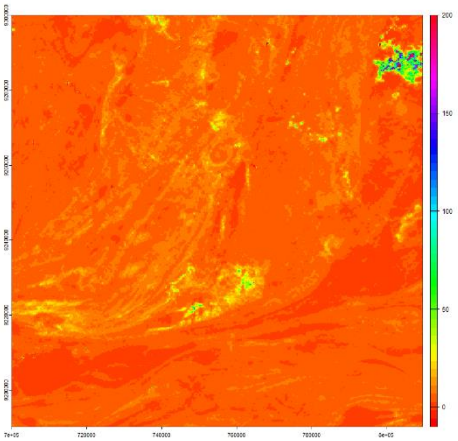
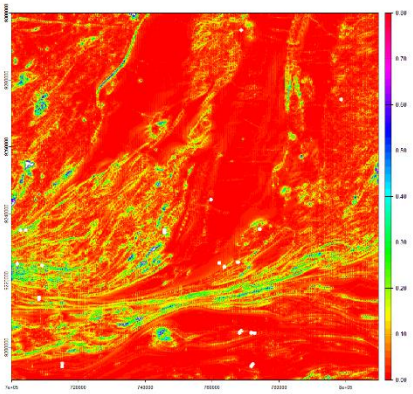
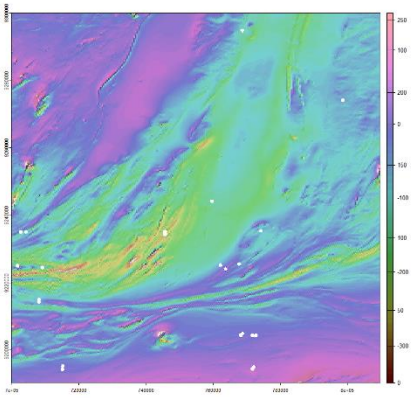
Mineral Exploration Targeting

PART 2 – Additional Data Extraction and Validation

André Luiz Costa, M.Sc, P.Geo., FAIG

Visit https://gdatasystems.com

PART 2

Part 2 is very important since we are going to prepare the layers that will be used in Part 3 based on the stack of raster generated in the last Part.

We will combine, evaluate, compute and adjust the raster layers using several layers through mathematical and statistical procedures. As a result, new raster layers will be created, and they will be the foundation for the target ranking evaluation. Little coding was used in Part 1, on the contrary, in this part we will use R intensively.

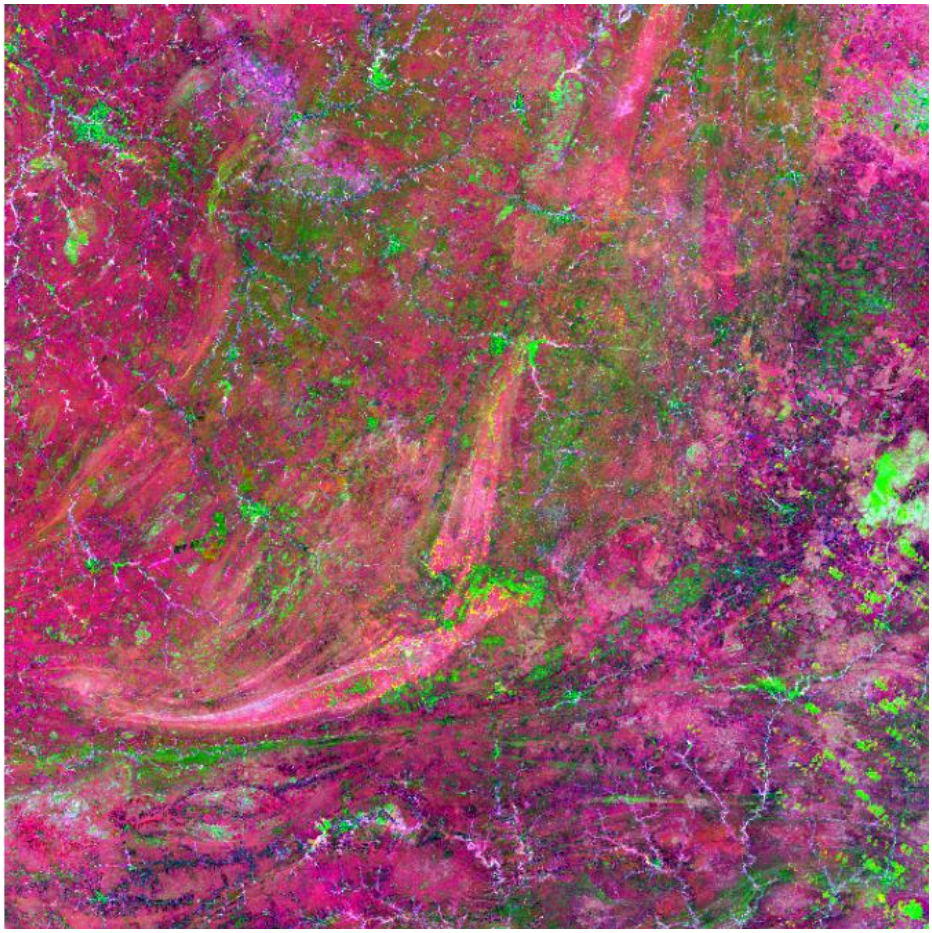
Extracting and validating new layers for the targeting process

Geology

The first reference layer we are going to create is the “geologic map” based on the response from sentinel2 images and radiometric ternary image.

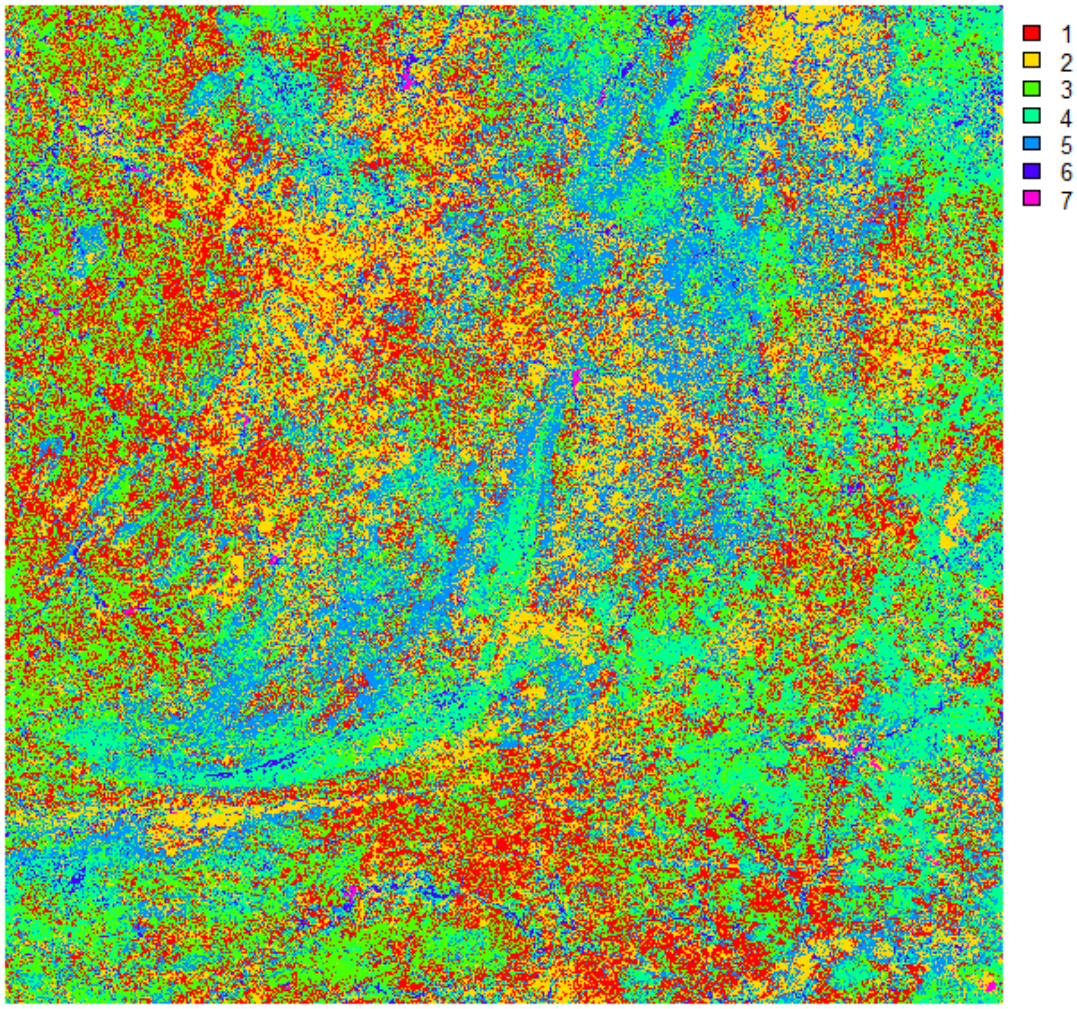
First we will create a filtering mask to remove the areas covered by water and heavy vegetation. In our case the area is little covered by these two factors.

```
library(terra)
#Working directory where the images and data are located
wd<-'C:/Users/User/Desktop/R algo/WORKING_DIR/'
setwd(wd)
master<-rast('prep_mosaic.tif')
#       Creating the indexes composite Image
#Calculating index to separate water, vegetation and exposed soil/rock
nir<-master[[7]]
red<-master[[3]]
savi<-(nir-red)/(nir+red+0.5)*1.5
green<-master[[2]]
swir<-master[[8]]
mndwi<-(green-swir)/(green+swir)
swir1<-master[[8]]
swir2<-master[[9]]
ndti<-(swir1-swir2)/(swir1+swir2)
rgbi<-rast(list(ndti, mndwi ,savi))
plotRGB(rgbi,stretch='lin')
```



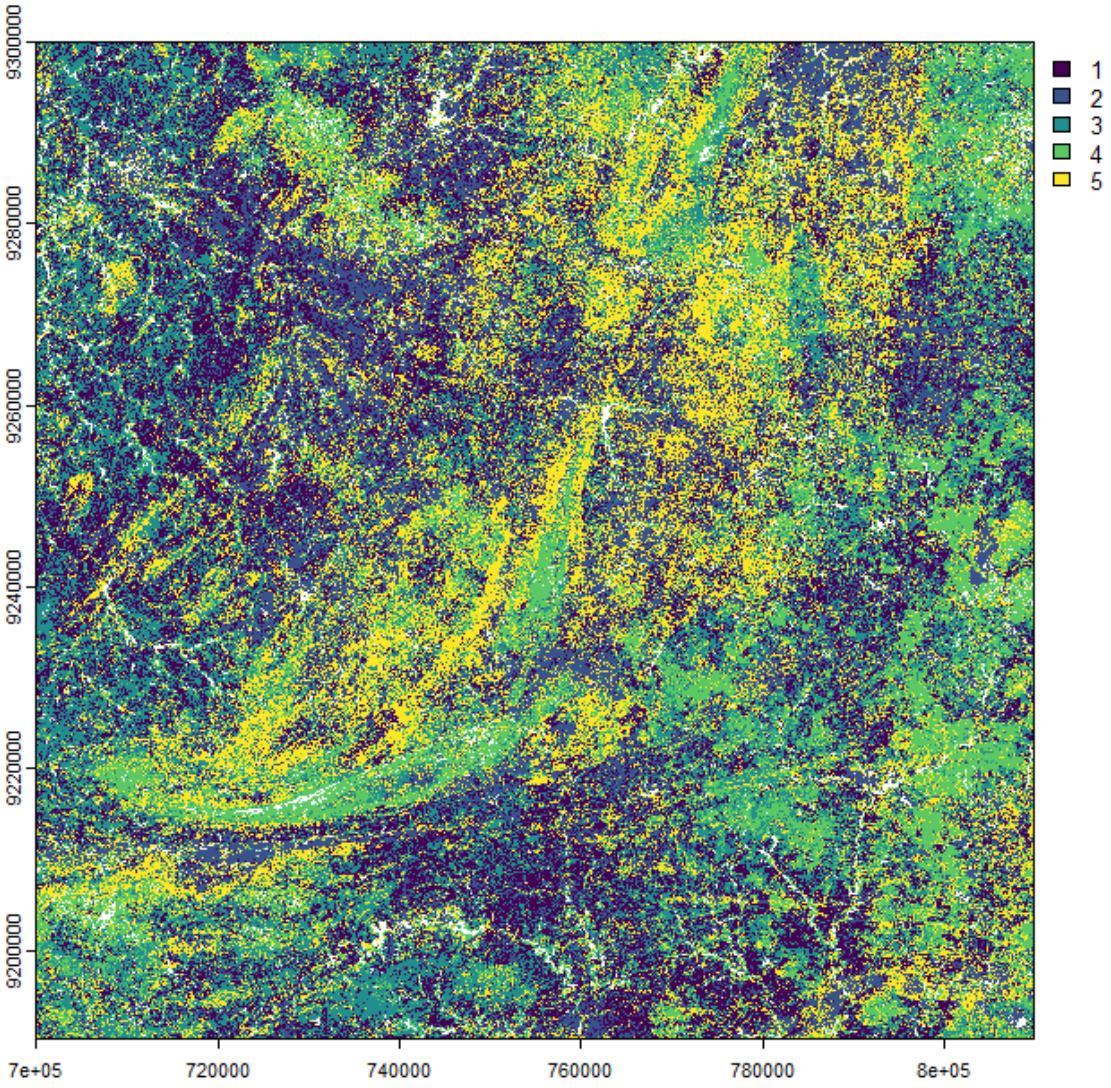
Using K-means clustering Classification based on indexes Composite scene above

```
library(cluster)
img<-c(rgbi[[1]],rgbi[[2]],rgbi[[3]])
ar <- values(img)
values <- which(!is.na(ar))
ar <- na.omit(ar)
Result <- clara(ar,7,samples=500,metric="manhattan",pamLike=T)# 7 minutes
kmraster <- rast (rgbi[[1]])
values(kmraster) <- Result$clustering
plot(kmraster, legend=T, axes=F, col = rainbow(7))
```



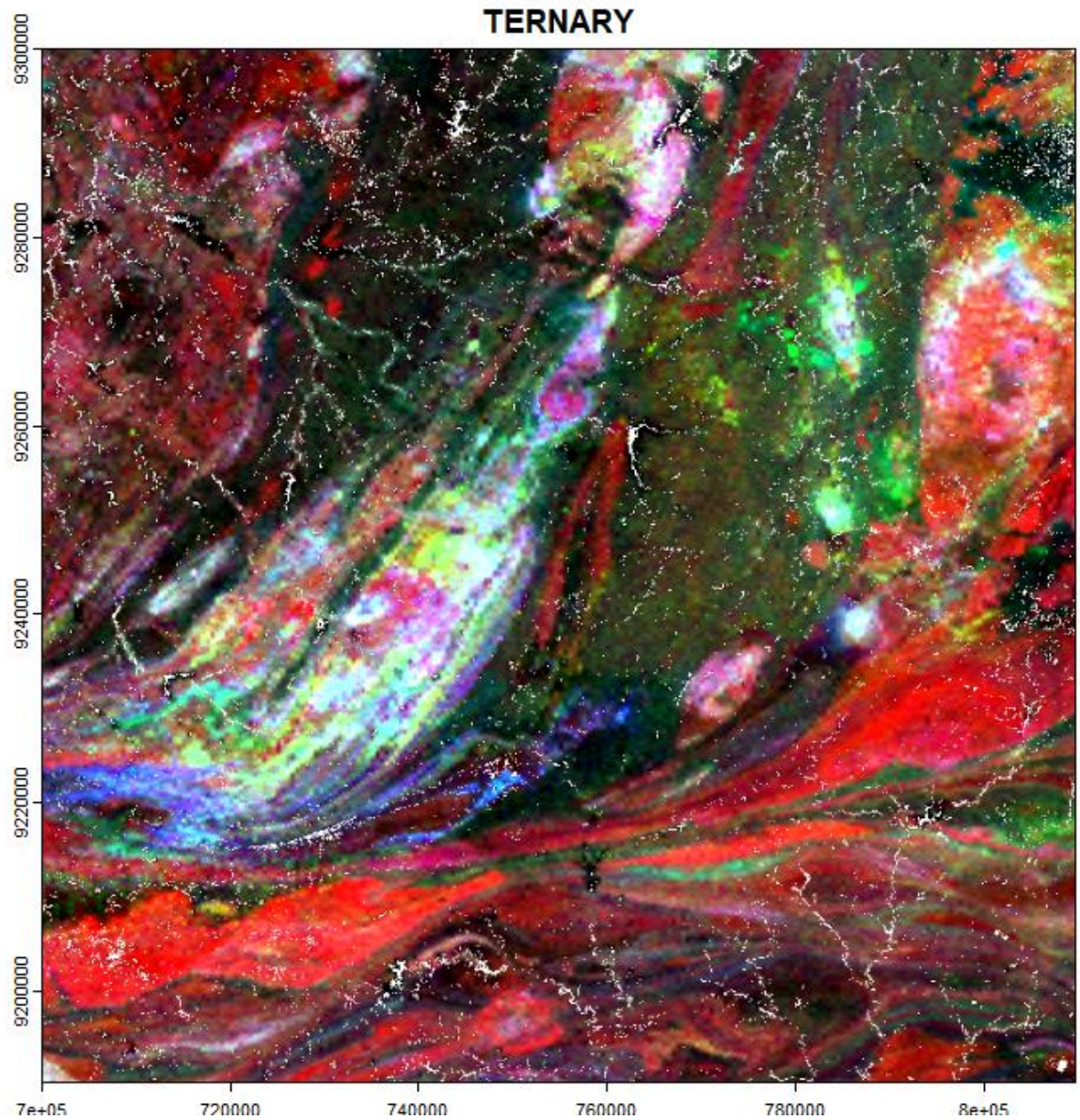
Creating a mask removing classes 6 and 7.

```
mask1<-kmraster
#removing classes 6 and 7
mask1[mask1 == 6 | mask1 == 7] <- NA
plot(mask1, col = rainbow(5))
```



We will generate now the Ternary image based on radiometric data and will also run a Principal Component Analysis

```
ter<- c(master[[11]],master[[12]],master[[13]])
m_ter <- mask(ter, mask1)
plotRGB(m_ter,stretch='lin',axes=TRUE,main='TERNARY',mar=c(1,1,2,1))
```

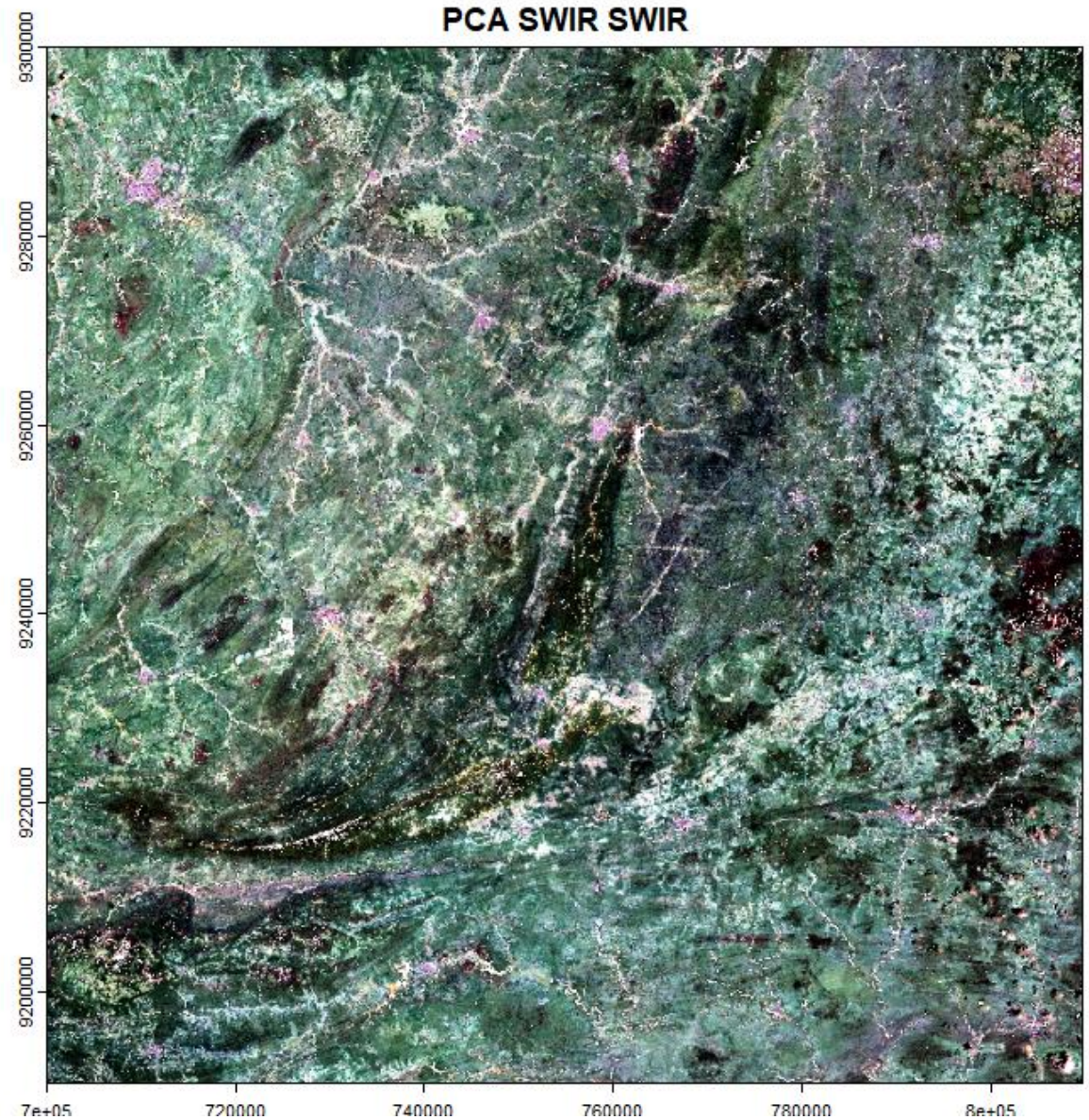


PCA composite

```
rlist<-c(master[[1]],master[[2]],master[[3]],master[[4]],master[[5]],
         master[[6]],master[[7]],master[[8]],master[[9]])
rlist2<-c("b2","b3","b4","b5","b6","b7", "b8a","b11","b12")
sentinel<-rlist
names(sentinel)<-rlist2
set.seed(1)
samples_ <- spatSample(sentinel, 3000000,method="random")# ~10% 2 minutes
pca <- prcomp(samples_, scale = TRUE)
rm(samples_)
pca
pcis<-predict(sentinel,pca,filename='pcis.tif',overwrite=TRUE)
```

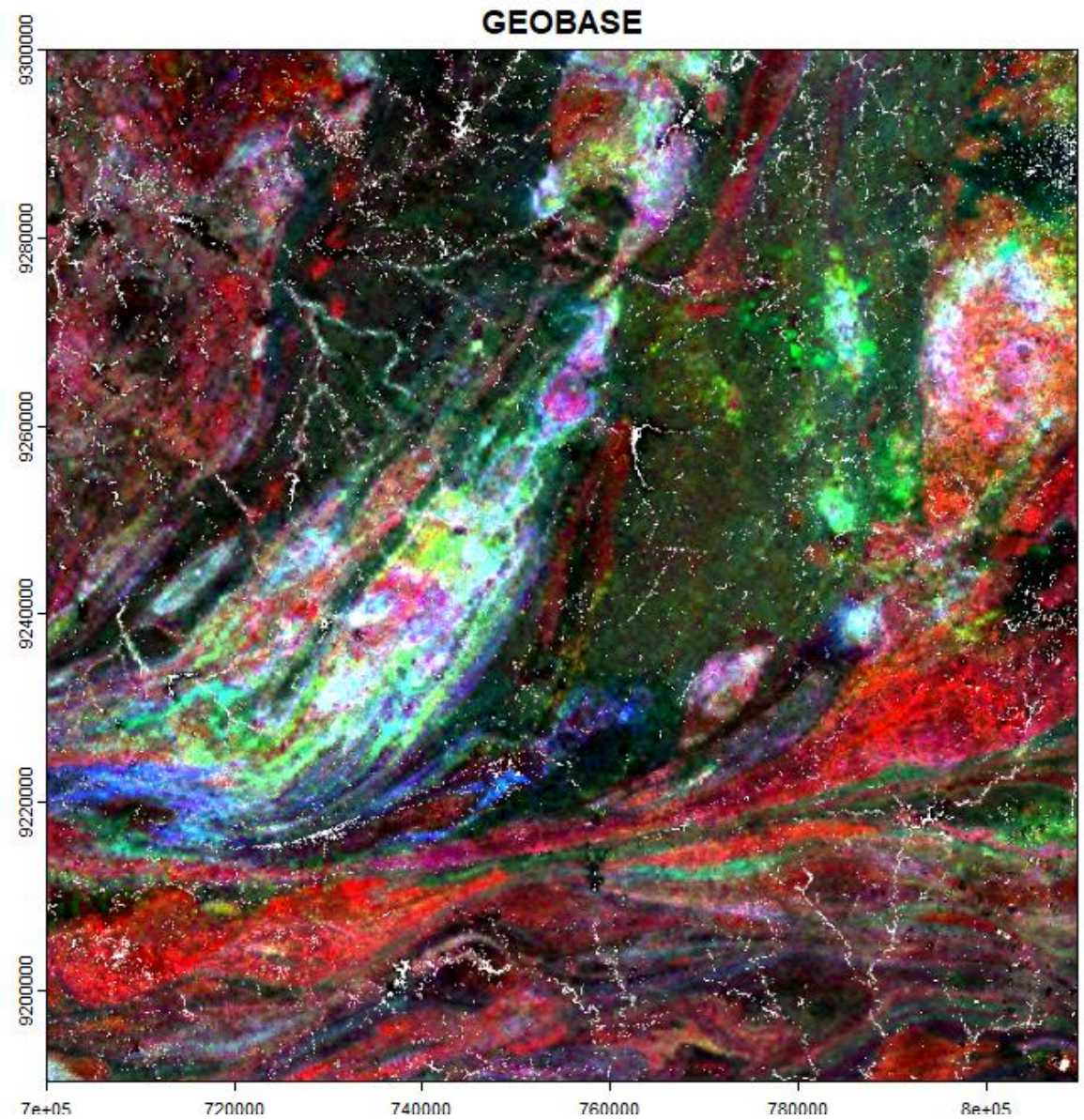
Leveling the First Component with the SWIR bands 11 and 12

```
pca1_swir_swir<-c(pcis[[1]]*400+5000,master[[8]],master[[9]])
pca1_swir_swir<- mask(pca1_swir_swir, mask1)
plotRGB(pca1_swir_swir,stretch='lin',axes=TRUE,main='PCA SWIR SWIR',mar=c(1,1,2,1))
```



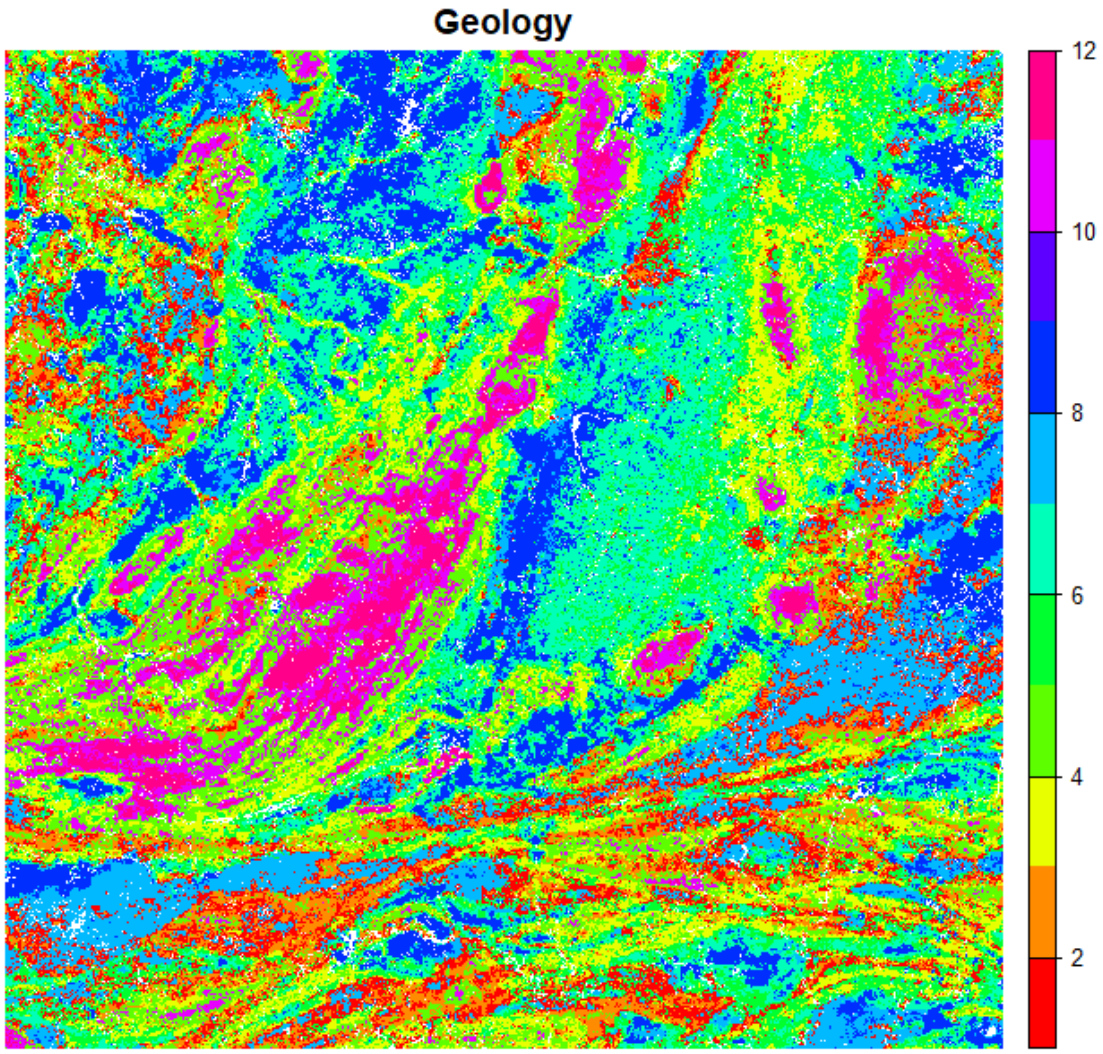
Creating the geobase scene and leveling to the range 0 to 1

```
geobase<-((pcaALL/100)*m_ter)/10000+0.02
plotRGB(geobase,stretch='lin',axes=TRUE,main='GEOBASE',mar=c(1,1,2,1))
```



Now running an unsupervised K-means Classification to create the Geology layer.

```
img<-c(geobase[[1]],geobase[[2]],geobase[[3]])
ar <- values(img)
values <- ar
#replacing  NA from vector by 0
ar[is.na(ar)] = 0
Result <- clara(ar,12,samples=500,metric="manhattan",pamLike=T)# 12 minutes
kmraster12 <- rast (geobase[[1]])
values(kmraster12) <- Result$clustering
#Here we mask the result and extract one of class resulting in 11 classes
kmraster11 <- mask(kmraster12, mask1)
plot(kmraster11, legend=T,axes=F,main='Geology',col = rainbow(11))
```




```
writeRaster(kmraster11,'geology.tif', overwrite=TRUE)
```

Validation

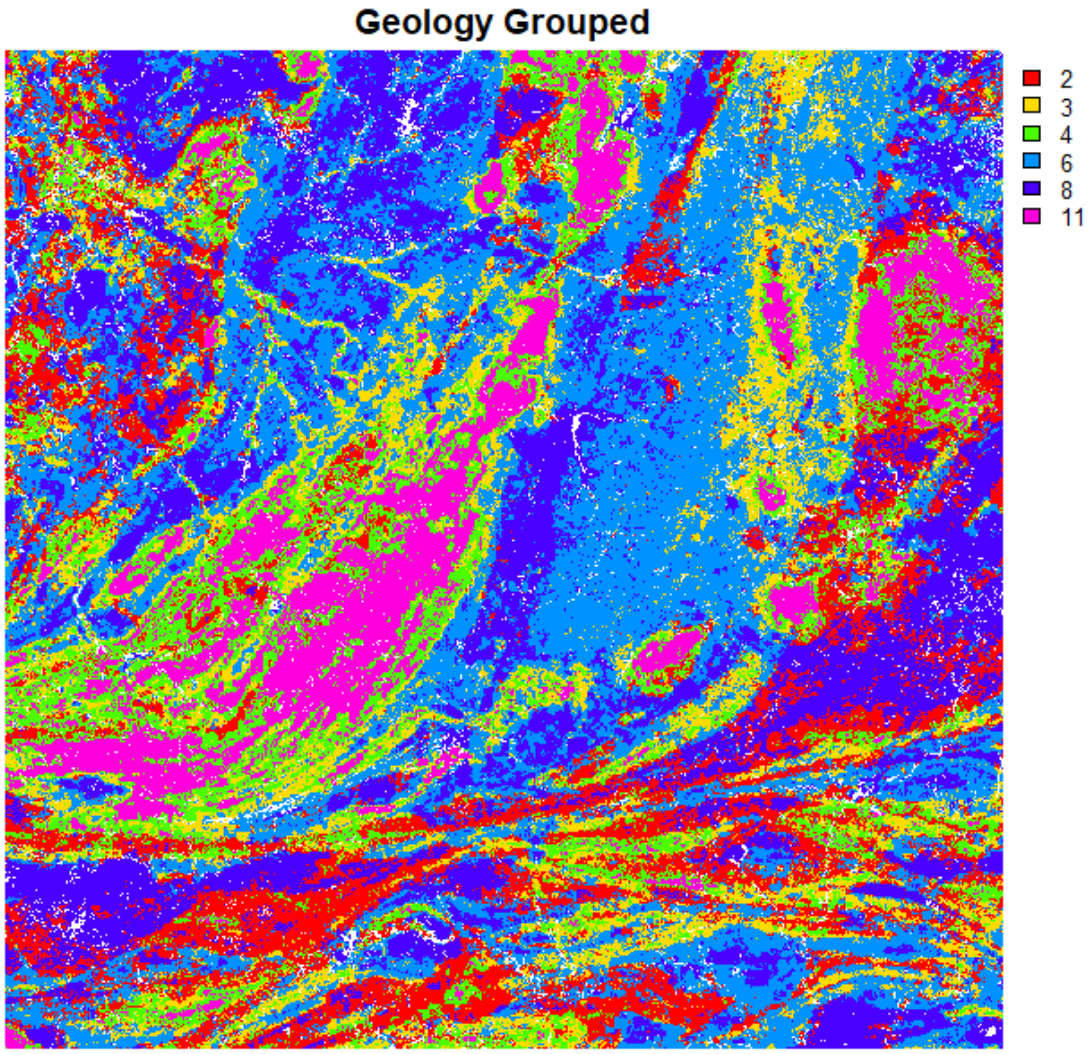
The result achieved has a strong correlation with the Sentinel2 bands and Ternary image. The PCA resulting in a first component that highlights the distinct spectral responses over the full set of bands.

Overall, the final clustering process using K-means resulted in 11 distinct classes. A few of these classes can be grouped for a better presentation but the original classes can hold some important correlation later in the process.

Create a grouped classification image can be achieved using:

```
geogroup<-kmraster11
geogroup[geogroup == 10] <- 11
geogroup[geogroup == 12] <- 11
geogroup[geogroup == 7] <- 8
```

```
geogroup[geogroup == 5] <- 6
geogroup[geogroup == 1] <- 2
plot(geogroup, legend=T,axes=F,main='Geology Grouped',col=rainbow(7))
```



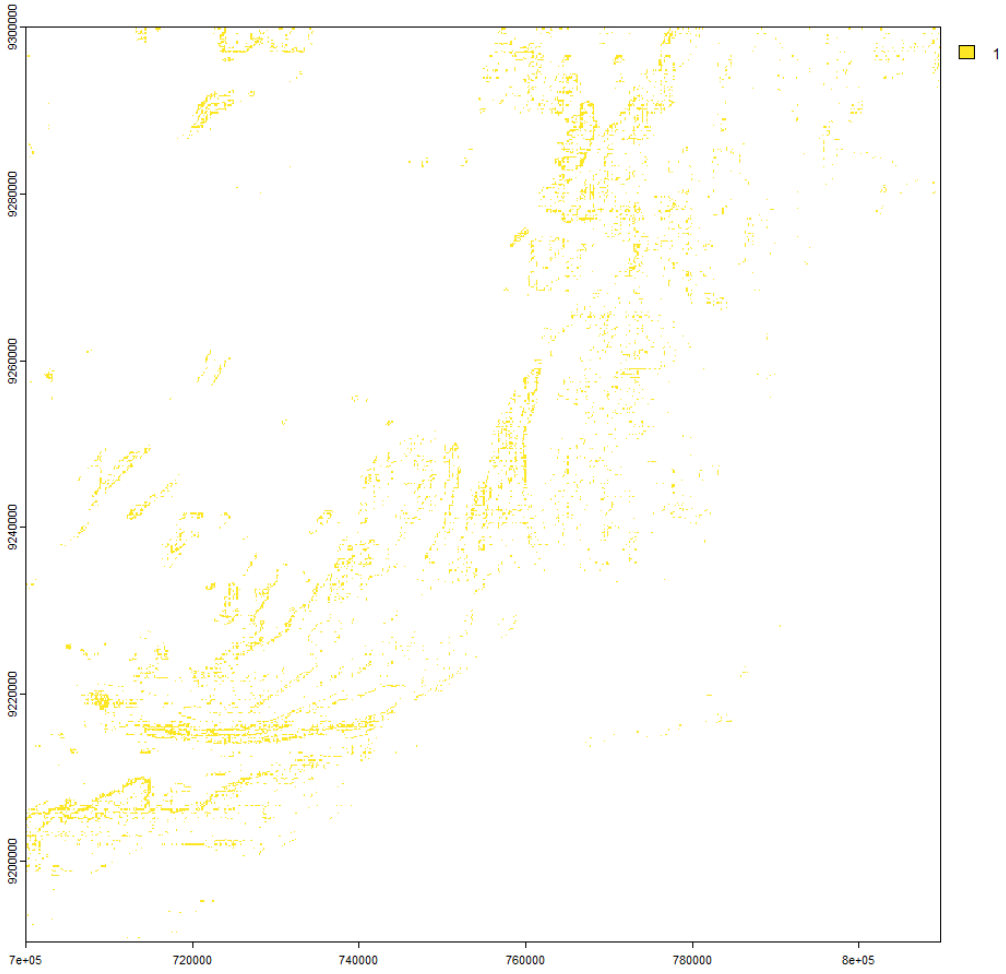

```
writeRaster(geogroup,'geology_grp.tif', overwrite=TRUE)
```

DEM lineaments and slopes

Extracting slopes and crests.

```
library(terra)
#Defining working directory
wd<-'C:/Users/User/Desktop/R algo/WORKING_DIR'
setwd(wd)
master<-rast('prep_mosaic.tif')
dem<-master[[10]]
rm(master)
gradiente<-terrain(dem,v='slope',unit='radians')
aspecto<-terrain(dem,v='aspect',unit='radians')
relevo<-shade(gradiente,aspecto,angle=45,direction=270,normalize= TRUE)
relevo2<-shade(gradiente,aspecto,angle=45,direction=90,normalize= TRUE)
```

```
relevo3<-shade(gradiente,aspecto,angle=45,direction=0,normalize= TRUE)
relevo4<-shade(gradiente,aspecto,angle=45,direction=180,normalize= TRUE)
crests<-relevo+relevo2+relevo3+relevo4
crests[crests > 600] <- NA
crests[crests >0] <- 1
plot(crests)
```




```
writeRaster(crests,'crests.tif', overwrite=TRUE)
```

The following code will extract the lineaments. It has **3 functions** at the beginning that were extracted from **wmtool**, a library that is not available anymore. I include here the function that we are going to use, they were created by Junji Sugiama and Kayoko Kobayashi.

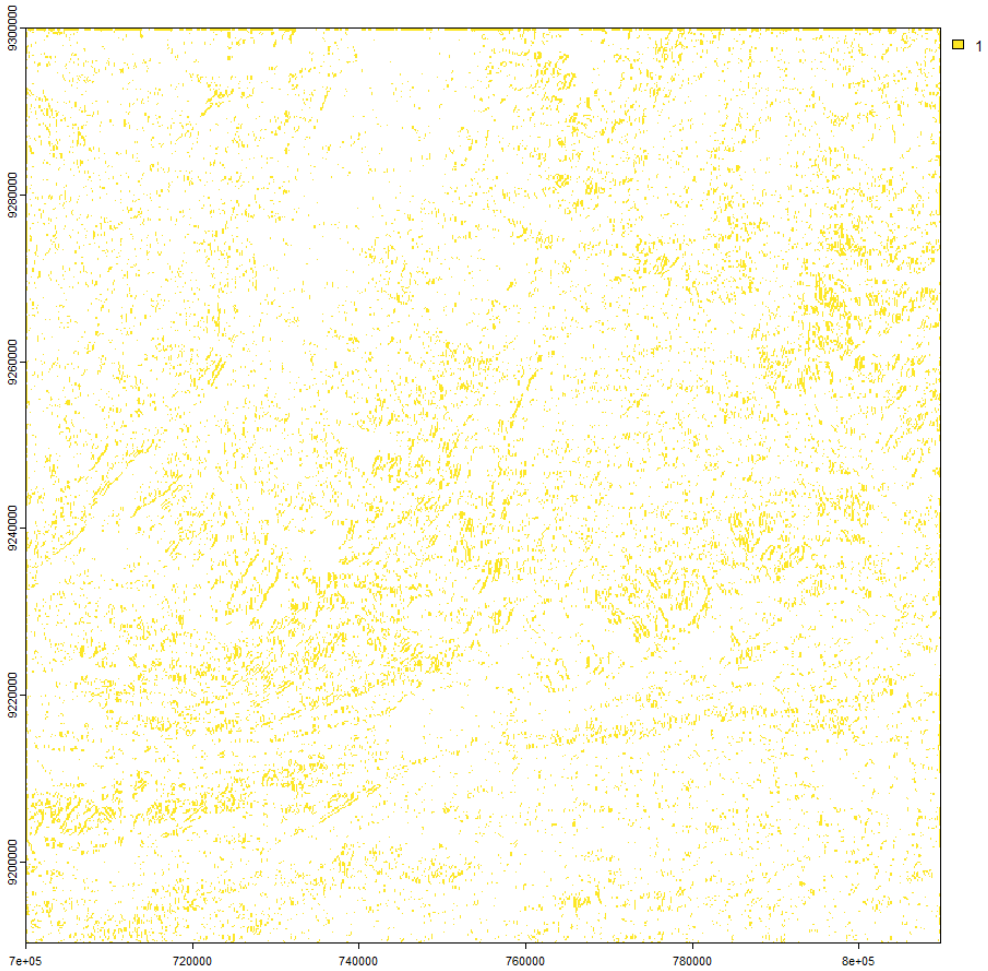
```
rgb2gray <- function(x, coefs=c(0.30, 0.59, 0.11)) {
          if (is.null(dim(x))) stop("image must have rgb type")
          if (length(dim(x))<3) stop("image must have rgb type")
          if (max(x) <= 1) {
                    x <- x*(2^(attr(x,"bits.per.sample"))-1)
          }
          imgdata <- round((coefs[1] * x[,,1] + coefs[2] * x[,,2] + coefs[3] * x[,,3]))
          attr(imgdata, "bits.per.sample")<-attr(x,"bits.per.sample")
          attr(imgdata, "samples.per.pixel") <- 1
          return(imgdata)
}

noise.filter <- function(x, n=3, method="median") {
if (length(dim(x))>2){warning("data must be grayscale image")
} else{
          if(max(x)<=1){
          x <- x*(2^attr(x,"bits.per.sample")-1)
   }
          px <- nrow(x)
          py <- ncol(x)
          img <- x
```

```
          f <- n %/% 2
          x_edge_plus <- matrix(NA, px+n-1, py+n-1)
          x_edge_plus[(f+1):(f+px),(f+1):(f+py)] <- x
          rasX <- matrix(NA,length(img),(n*n))
          if (method == "median" | method == "mean") {
          for(i in 1:n){
                    for(j in 1:n){
                              rasX[,(n*(j-1)+i)] <- array(x_edge_plus[(i:(px+i-1)),(j:(py+j-1))])
                    }
                    }
                    if(method == "median"){
                              o.img <- apply(rasX,1,median,na.rm=T)
                              dim(o.img)<- c(nrow(img),ncol(img))
                    } else if(method == "mean"){
                              o.img <- apply(rasX,1,mean,na.rm=T)
                              dim(o.img)<- c(nrow(img),ncol(img))
                    }
          } else if (method == "gaussian"){
                    if(n==3){
                              gcf<- c(1,2,1,2,4,2,1,2,1)/16
                    } else if(n==5){
                              gcf<- c(1,4,6,4,1,4,16,24,16,4,6,24,36,24,6,4,16,24,16,4,1,4,6,4,1)/256
                    } else {warning("only 3 or 5 for n")
                    }
                    rasX <- matrix(NA,length(img),(n*n))
                    for(i in 1:n){
                    for(j in 1:n){
                              rasX[,(n*(j-1)+i)] <- gcf[n*(j-1)+i]*array(x_edge_plus[(i:(px+i-1)),(j:(py+j-1))])
                    }
                    }
          o.img <- apply(rasX,1,sum,na.rm=T)
          dim(o.img)<- c(nrow(img),ncol(img))
                    if(n==3){
                              o.img[c(1,nrow(o.img)),c(1,ncol(o.img))] <- o.img[c(1,nrow(o.img)),c(1,ncol(o.img))]*16/9
                              o.img[c(1,nrow(o.img)),2:(ncol(o.img)-1)] <- o.img[c(1,nrow(o.img)),2:(ncol(o.img)-1)]*16/12
                              o.img[2:(nrow(o.img)-1),c(1,ncol(o.img))] <- o.img[2:(nrow(o.img)-1),c(1,ncol(o.img))]*16/12
                    }else if(n==5){
                              o.img[c(1,nrow(o.img)),c(1,ncol(o.img))] <- o.img[c(1,nrow(o.img)),c(1,ncol(o.img))]*256/121
                              o.img[c(1,nrow(o.img)),c(2,ncol(o.img)-1)] <- o.img[c(1,nrow(o.img)),c(2,ncol(o.img)-
1)]*256/165
                              o.img[c(2,nrow(o.img)-1),c(1,ncol(o.img))] <- o.img[c(2,nrow(o.img)-
1),c(1,ncol(o.img))]*256/165
                              o.img[c(1,nrow(o.img)),3:(ncol(o.img)-2)] <- o.img[c(1,nrow(o.img)),3:(ncol(o.img)-2)]*256/176
                              o.img[3:(nrow(o.img)-2),c(1,col(o.img))] <- o.img[3:(nrow(o.img)-2),c(1,col(o.img))]*256/176
                              o.img[c(2,(nrow(o.img)-1)),c(2,(ncol(o.img)-1))] <- o.img[c(2,(nrow(o.img)-
1)),c(2,(ncol(o.img)-1))]*256/189
                              o.img[c(2,nrow(o.img)-1),3:(ncol(o.img)-2)] <- o.img[c(2,nrow(o.img)-1),3:(ncol(o.img)-
2)]*256/240
                              o.img[3:(nrow(o.img)-2),c(2,ncol(o.img)-1)] <- o.img[3:(nrow(o.img)-2),c(2,ncol(o.img)-
1)]*256/240
                    }
          } else {
                    print(NULL)
          }
          attr(o.img, "bits.per.sample")<-attr(x,"bits.per.sample")
          attr(o.img, "samples.per.pixel")<-attr(x,"samples.per.pixel")
          out <- round(o.img)
}}

edge.detect <- function(x, thresh1=1, thresh2=15, noise="gaussian", noise.s=3, method="Canny") {
if (length(dim(x))>2){warning("data must be grayscale image")
} else {
img <- x
img.n <- noise.filter(img,noise.s, method= noise)
img.ed <- img[2:(nrow(img)-1),2:(ncol(img)-1)]
rasSX <- matrix(NA,length(img.ed),9)
rasSY <- matrix(NA,length(img.ed),9)
sx <- c(-1,-2,-1,0,0,0,1,2,1)/8
sy <- c(-1,0,1,-2,0,2,-1,0,1)/8
for(i in 1:3){
          for(j in 1:3){
                    rasSX[,(3*(j-1)+i)] <- sx[3*(j-1)+i]*array(img.n[(i:(nrow(img.n)+i-3)),(j:(ncol(img.n)+j-3))])
                    rasSY[,(3*(j-1)+i)] <- sy[3*(j-1)+i]*array(img.n[(i:(nrow(img.n)+i-3)),(j:(ncol(img.n)+j-3))])
                    }
          }
img.sx <- apply(rasSX,1,sum)
img.sy <- apply(rasSY,1,sum)
dim(img.sx) <- dim(img.sy) <- dim(img.ed)
img.sxsy <- (img.sx^2+img.sy^2)^0.5                              #magnitude of gradient
if(method=="Sobel"){
          out <- img.sxsy
} else if(method=="Canny"){
          img.th <- atan(img.sy/img.sx)
          img.th0 <- img.th                                                         #angles of gradient (0,45,90,135)
          img.th0[abs(img.th0) >= pi*3/8] <- 90
          img.th0[abs(img.th0) <= pi/8] <- 0
          img.th0[img.th0 > pi/8 & img.th0 < pi*3/8] <- 45
          img.th0[img.th0 > -pi*3/8 & img.th0 < -pi/8] <- 135
          #non-maximum suppression
          img.sxsy.bl <- matrix(0,nrow(img),ncol(img))
          img.sxsy.bl[2:(nrow(img)-1),2:(ncol(img)-1)] <- img.sxsy
          rasL <- matrix(NA,length(img.sxsy),9)
          for(i in 1:3){
          for(j in 1:3){
                    rasL[,(3*(j-1)+i)] <- array(img.sxsy.bl[(i:(nrow(img.n)+i-3)),(j:(ncol(img.n)+j-3))])
```

```
                }
        }
        rasLj <- matrix(0,length(img.sxsy),4)
        rasLj[which(rasL[,5] > rasL[,4] & rasL[,5] > rasL[,6]),3] <- 1
        rasLj[which(rasL[,5] > rasL[,7] & rasL[,5] > rasL[,3]),4] <- 1
        rasLj[which(rasL[,5] > rasL[,2] & rasL[,5] > rasL[,8]),1] <- 1
        rasLj[which(rasL[,5] > rasL[,1] & rasL[,5] > rasL[,9]),2] <- 1
        img.tl <- array(img.sxsy)
        img.tl[which(img.th0==0)] <- img.sxsy[which(img.th0==0)]*rasLj[which(img.th0==0),1]
        img.tl[which(img.th0==45)] <- img.sxsy[which(img.th0==45)]*rasLj[which(img.th0==45),2]
        img.tl[which(img.th0==90)] <- img.sxsy[which(img.th0==90)]*rasLj[which(img.th0==90),3]
        img.tl[which(img.th0==135)] <- img.sxsy[which(img.th0==135)]*rasLj[which(img.th0==135),4]
        dim(img.tl) <- dim(img.sxsy)
        #hysteresis threshold
        mxth <- sort(img.tl[which(img.tl!=0)])[round(length(img.tl[which(img.tl!=0)])*thresh2/100)]
        mnth <- sort(img.tl[which(img.tl!=0)])[round(length(img.tl[which(img.tl!=0)])*thresh1/100)]
        img.bn0 <- img.bn1 <- matrix(0,nrow(img.tl)+2,ncol(img.tl)+2)
        img.bn0[2:(nrow(img.bn0)-1),2:(ncol(img.bn0)-1)] <- img.bn1[2:(nrow(img.bn0)-1),2:(ncol(img.bn0)-1)] <- img.tl
        img.bn0[which(img.bn0 <= mnth)] <- 0
        img.bn0[which(img.bn0 > mnth)] <- 1
        img.lb <- matrix(0,nrow(img.tl)+2,ncol(img.tl)+2)
        ren8 <- rbind(c(-1,-1),c(-1,0),c(-1,1),c(0,-1)) #8-connected area
        k <- 0
        for(i in 2:(nrow(img.bn0)-1)){
        for(j in 2:(ncol(img.bn0)-1)){
                z8 <- t(c(i,j)+t(ren8))
                if(img.bn0[i,j]==0){
                        }else if(img.bn0[i,j]==1 && sum(img.bn0[z8]==0)==4){
                                k <- k+1
                                img.lb[i,j] <- k
                        }else {
                                zl <- z8[which (img.lb[z8] != 0),,drop=F]
                                z.lb <- img.lb[zl]
                                z.lb <- sort(z.lb[!duplicated(z.lb)])
                                img.lb[i,j] <- z.lb[1]
                                if(length(z.lb)>1){
                                        for(l in 2:length(z.lb)){
                                        img.lb[which(img.lb==z.lb[l])] <- z.lb[1]
                                                }
                                        }

                        }
                }
                }
        img.bn <- img.lb
        lb <- sort(img.lb[!duplicated(array(img.lb))])
        lb <- lb[-1]
        for(i in lb){
                if(length(which(img.bn1[which(img.lb==i)]>mxth))==0){
                        img.bn[which(img.lb==i)] <- 0
                }else {
                        img.bn[which(img.lb==i)] <- 1
        }
        }
        img.bn <- img.bn[2:(nrow(img.bn0)-1),2:(ncol(img.bn0)-1)]
        out <- img.bn
}else {
        print(NULL)
}
}}
```

Run the following code to extract the lineaments. The processing is intensive, and it can take up to 30 minutes to complete.

```
resul<-focal(aspecto,w=21, fun=sum, na.rm=TRUE)
h<-as.integer(values(resul))
hh<-array(h,c(dim(resul)[1],dim(resul)[2],3))
hh[is.na(hh)]<-0
hi<-rgb2gray(hh)
edg<-edge.detect(hi,noise="median", noise.s=3, method='Sobel')
r<-rast(nrows=dim(edg)[1],ncols=dim(edg)[2],crs=crs(resul),ext=ext(resul))
a<-as.vector(edg)
final<-setValues(r,a)
final[final<60]<-NA
final[final>=60]<-1
plot(final)
```

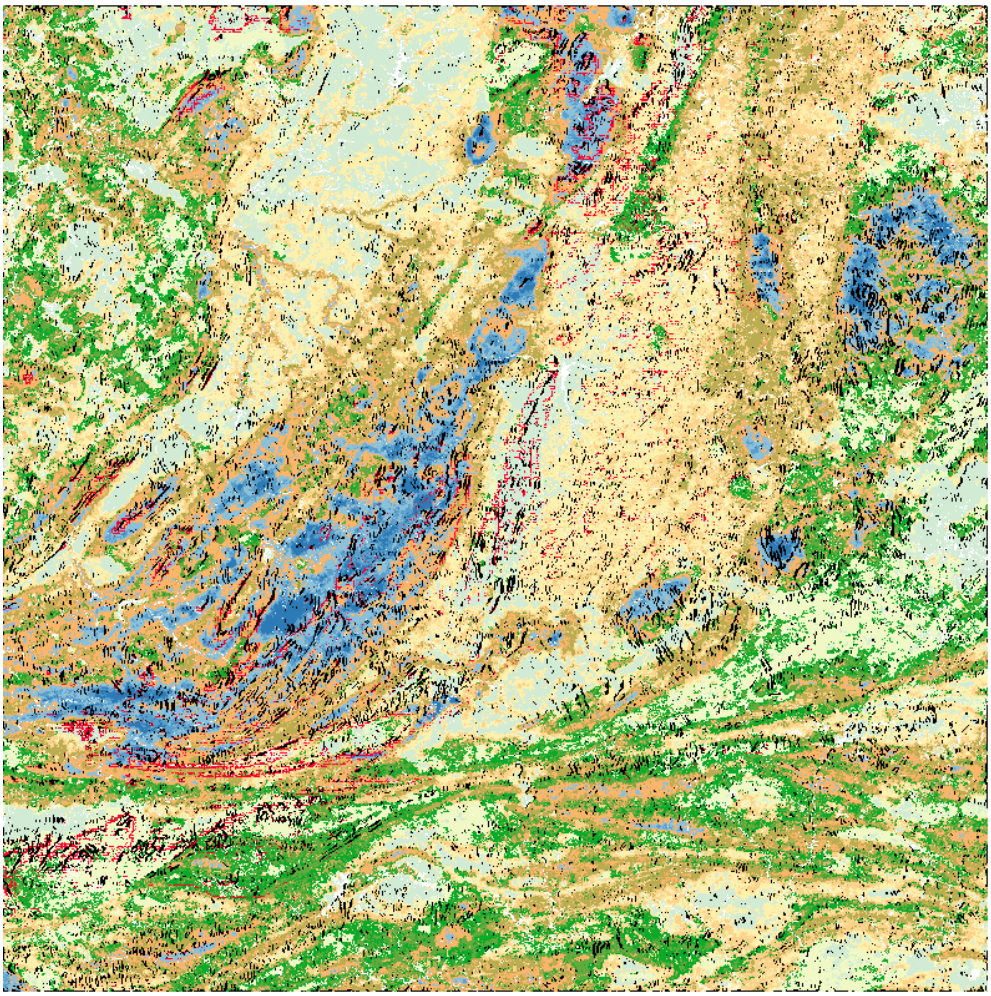



```
writeRaster(final,'lines.tif', overwrite=TRUE)
```

Validation

Automated linear features extraction, also known as edge-detection, has some limitations but they give us a reasonable indication of general structural directions of a region and can assist in delineating geostructural domains. More elaborated linear extraction can be achieved using robust algorithms, but they tend to be “heavy” and slow. It will depend on the goals of your project.

See below the integration of interpreted geology and linear features.



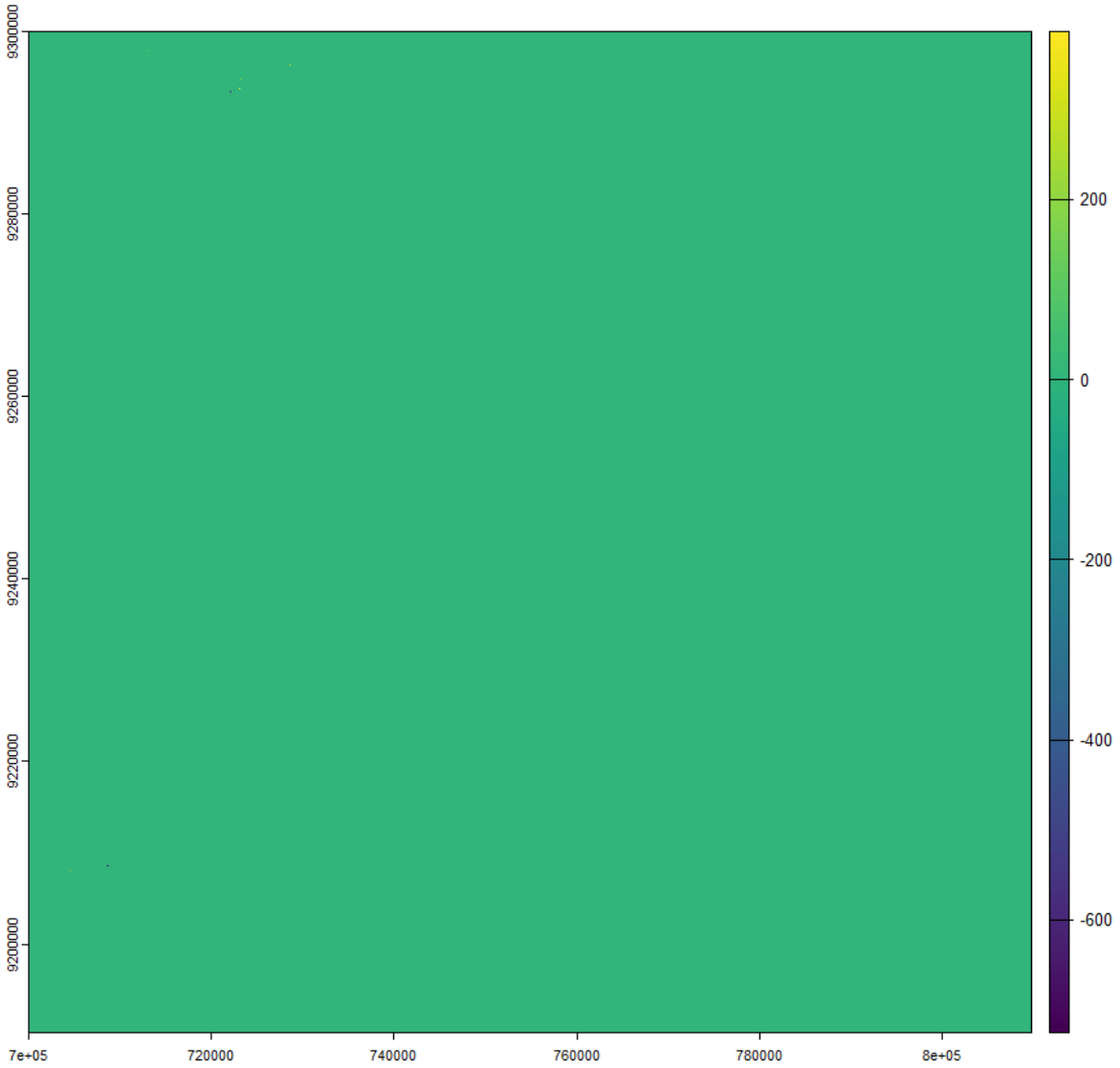
Gammaspectrometric Ratios

Airborne gammaspectrometry data is one of the most relevant in targeting because they represent a direct response to the geological framework of the area. We have already the individual K, U and Th layers plus the total count layer already prepared in the last part and now we are going to create 4 more ratio layers:

- eU/eTh
- eU/Kperc
- eTh/Kperc
- Kperc × eU/eTh (F factor)

eU/eTh

```
library(terra)
#Defining working directory
wd<-'C:/Users/User/Desktop/R algo/WORKING_DIR'
setwd(wd)
master<-rast('prep_mosaic.tif')
Kperc<-master[[11]]
eU<-master[[12]]
eTh<-master[[13]]
rtUTh<-eU/eTh
plot(rtUTh)
```

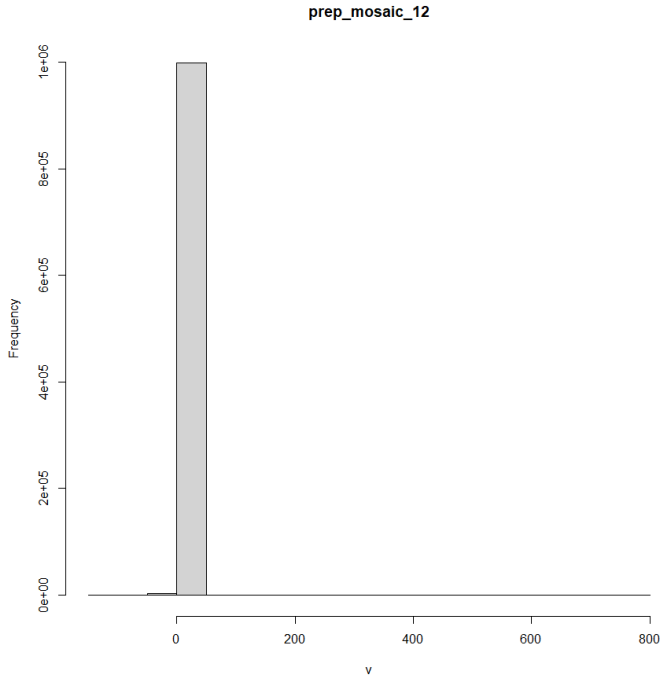


Well... not much to see here, right? Statistics are your friend, and we will enhance the display of this layer. Trust me the result shown above is right, but we cannot see it.
First let's examine the data distribution

```
rtUTh
class       : SpatRaster
dimensions  : 5490, 5490, 1  (nrow, ncol, nlyr)
resolution  : 20, 20  (x, y)
extent      : 699960, 809760, 9190240, 9300040  (xmin, xmax, ymin, ymax)
coord. ref. : WGS 84 / UTM zone 24S (EPSG:32724)
source(s)   : memory
varname     : prep_mosaic
name        : prep_mosaic_12
min value   :    -134326.973
max value   :       5961.378
```

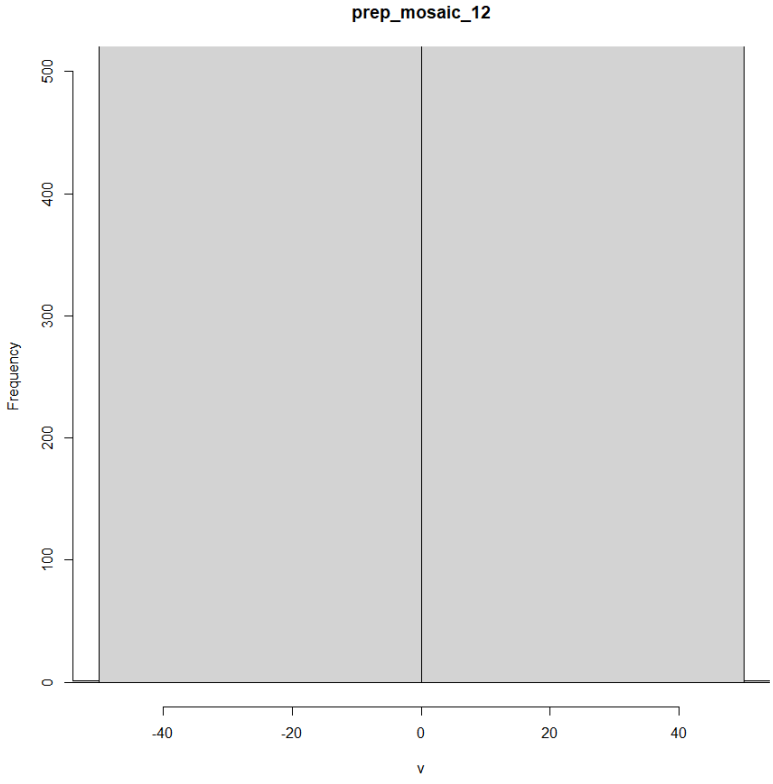
The data distribution ranges from -134327 to 5962 which is a very large range. We will plot an histogram from a selection of point values.

```
hist(rtUTh)
```



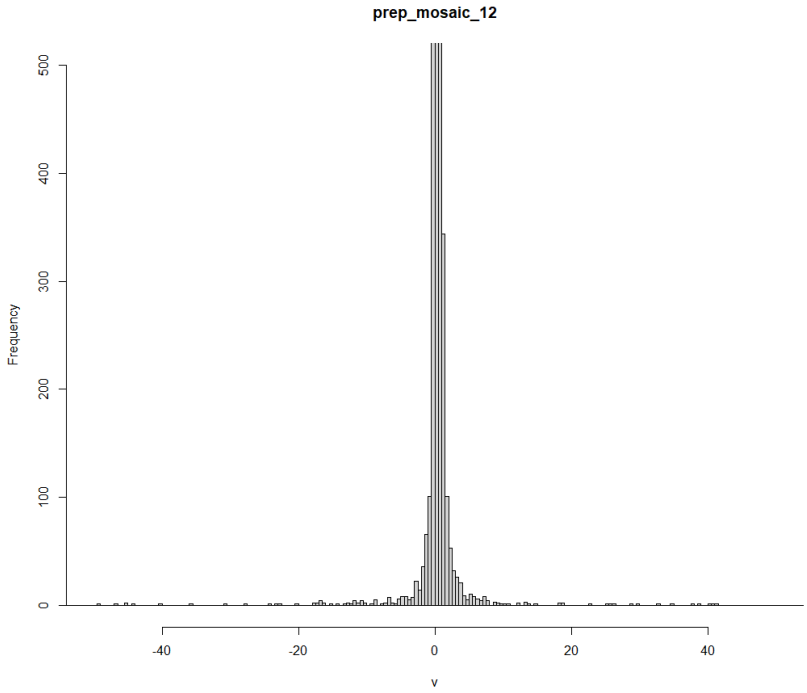
Now we can see that almost the entire result ranges from -50 to 50. Reducing the visualization ranging from 0 to 500 in Y and from -50 to 50 in X we can observe this.

```
hist(rtUTh,ylim=c(0,500),xlim=c(-50,50))
```



Increasing the number of histograms columns class to be displayed.

```
hist(rtUTh,ylim=c(0,500),xlim=c(-50,50),n=3000)
```

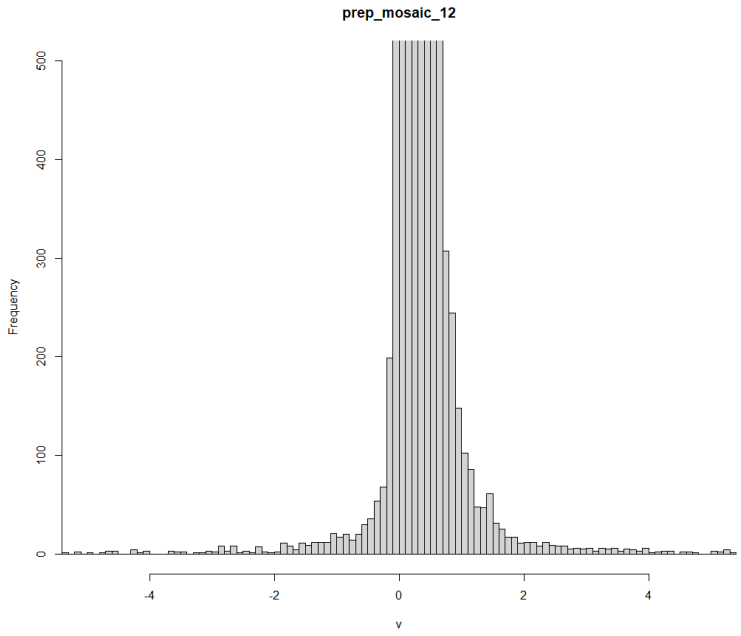


With Gauss' blessing we found where our data is hanging out.

```
hist(rtUTh,ylim=c(0,500),xlim=c(-50,50),n=3000)
```

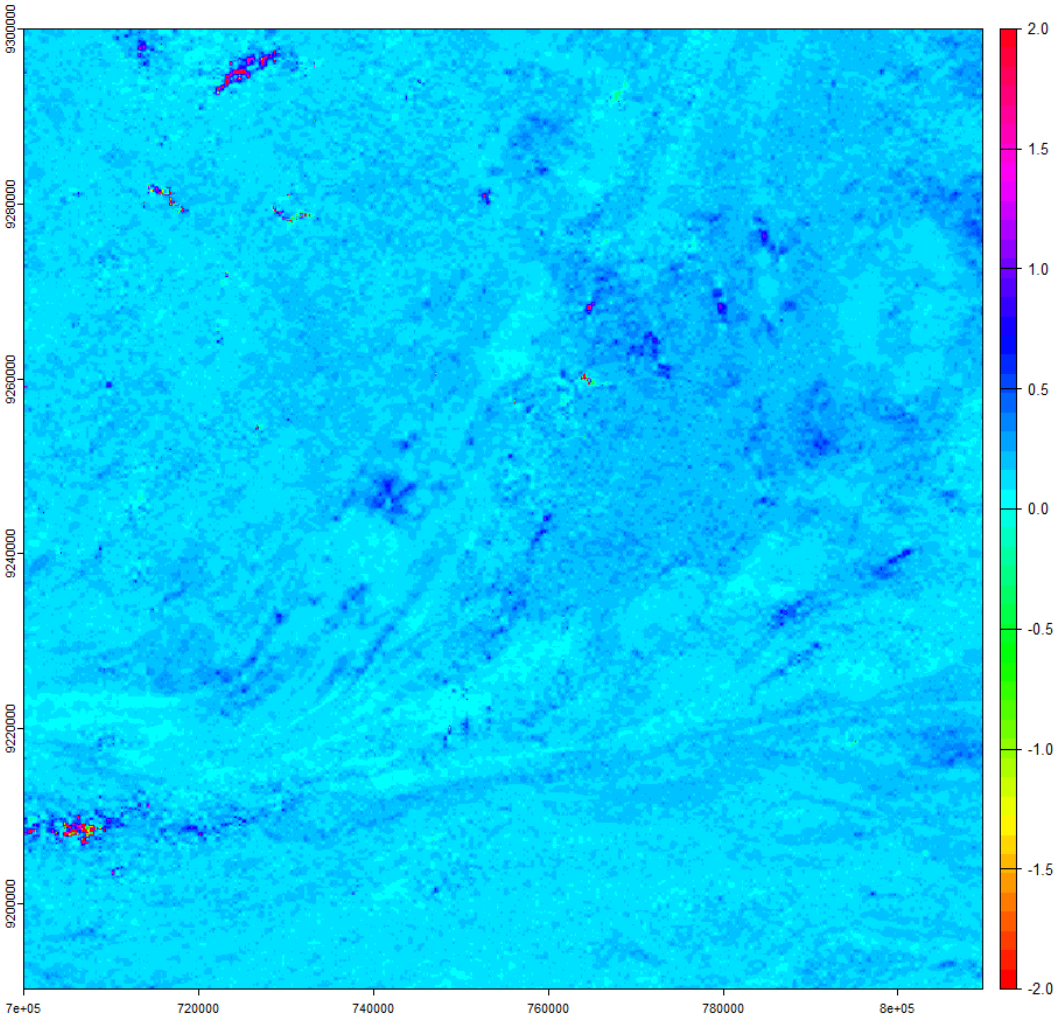
Zoom in this Normal distribution to get the limits to trim our data:

```
hist(rtUTh,ylim=c(0,500),xlim=c(-5,5),n=10000)
```



Now we will trim the data assigning to -2 everything below it and to 2 everything above it.

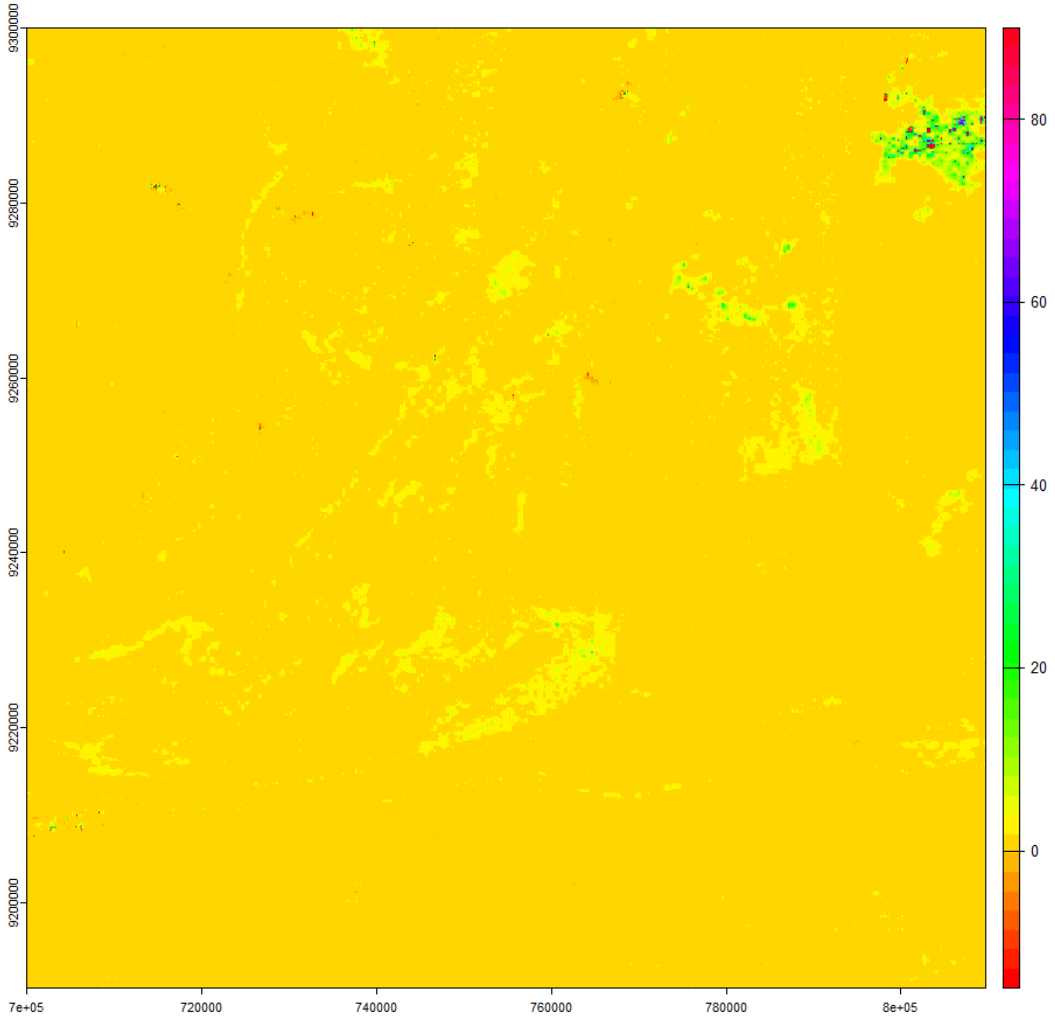
```
rtUTh[rtUTh > 2] <- 2
rtUTh[rtUTh < -2] <- -2
plot(rtUTh,col=rainbow(50))
```




```
writeRaster(rtUTh,'rtUTh.tif', overwrite=TRUE)
```

eU/Kperc

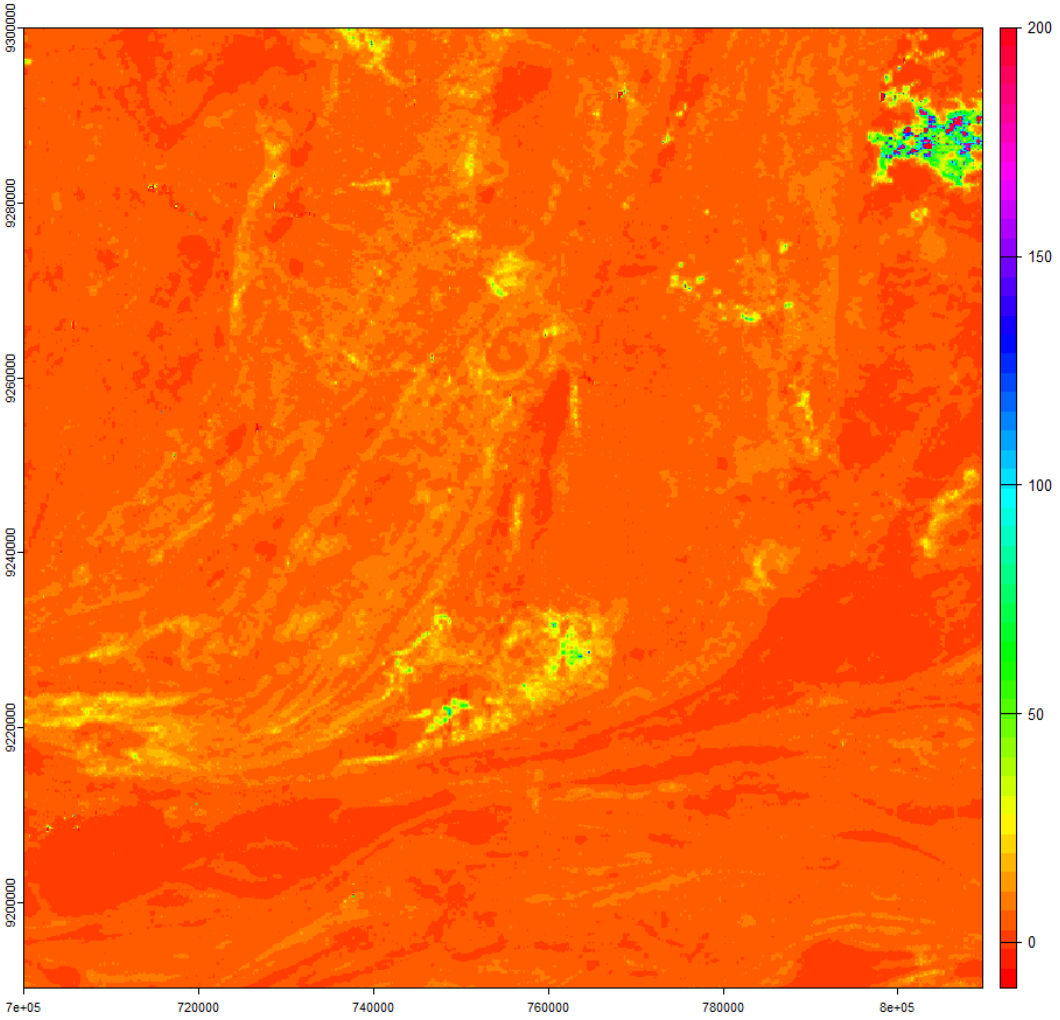
```
rtUK<-eU/Kperc
rtUK [rtUK  > 90] <- 90
rtUK [rtUK < -15] <- -15
plot(rtUK,col=rainbow(50))
```




```
writeRaster(rtUK,'rtUK.tif', overwrite=TRUE)
```

eTh/Kperc

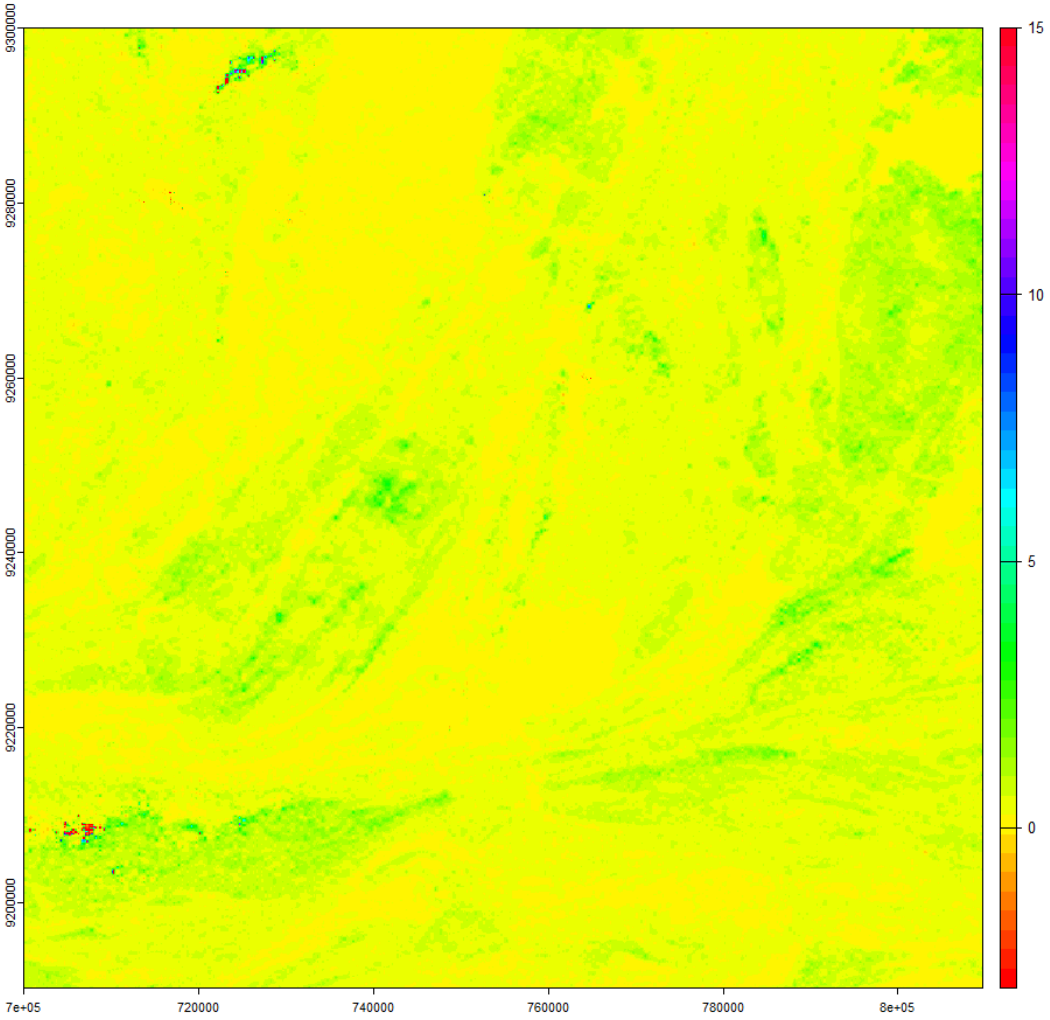
```
rtThK<-eTh/Kperc
rtThK [rtThK  > 200] <- 200
rtThK [rtThK < -10] <- -10
plot(rtThK,col=rainbow(50))
```




```
writeRaster(rtThK,'rtThK.tif', overwrite=TRUE)
```

K × eU/eTh (F factor)

```
ff<-Kperc*eU/eTh
ff [ff  > 15] <- 15
ff [ff < -3] <- -3
plot(ff,col=rainbow(50))
```




```
writeRaster(ff, 'ff.tif', overwrite=TRUE)
```

Validation

The ratio layers above achieve a good separation within the study area and may have a good correlation with what we will search for ahead.

Below is an example from Australia of how it is the gammaspectrometric response of different lithologies.

Rock type	*Rock*			*Soil*		
	K %	*U* ppm	*Th* ppm	*K* %	*U* ppm	*Th* ppm
Intrusives						
granitoids	0.3–4.5 (2.4)	0.4–7.8 (3.3)	2.3–45 (16)	0.4–3.9 (2.1)	0.5–7.8 (2.7)	2–37 (13)
gneissic rock	2.4–3.8 (2.4)	2.1–3.6 (2.5)	18–55 (15)	0.7–1.9 (1.3)	1.6–3.8 (2.2)	6–19 (12)
pegmatite	2.6–5.5 (3.7)	0.3–1 (0.7)	0.3–9.6 (2)			
aplites	0.6–4 (2.4)	1–8 (3.3)	3–20 (7)			
quartz–feldspar porphyry	1–5 (2.9)	1.3–2.9 (1.7)	6–14 (13)			
intermediate intrusives	0.7–5.6 (2.7)	0.1–1.2 (0.8)	0.8–6.1 (2.4)	0.7–3.4 (1.6)	1.5–2.3 (1.9)	2.9–8.4 (5.6)
mafic intrusives	0.1–0.8 (0.4)	0.0–1.1 (0.3)	0.0–3.1 (1.2)			
Extrusives						
felsic volcanics	2.0–4.4 (3.7)	1.4–13 (2.4)	13–28 (17)	1.8–3.2 (2.4)	1.3–2.4 (2.1)	10–18 (13)
intermediate volcanics	1.8–4.1 (2.7)	0.9–5.6 (2.3)	1.5–15. (9)	1.0–2.7 (1.9)	1.2–3.6 (2.1)	4–17 (10)
low-K andesites	0.7–0.9 (0.8)	1.0–2.5 (1.6)	3–8 (5)	0.8–1.5 (1.1)	1.2–1.5 (1.3)	4–6 (5)
mafic volcanics	0.3–1.3 (0.9)	0.3–1.3 (0.7)	2.0–5.0 (3.0)	0.2–1.4 (0.7)	0.6–2.5 (1.6)	3.3–13 (7.9)
ultramafic volcanics	0.2–0.9 (0.4)	0.3–0.9 (0.6)	0.0–4.0 (1.2)	0.6	2.0	6.
Sedimentary rocks						
Archaean shales	0.4–1.6 (0.9)	0.3–1.3 (0.9)	1–5 (2.7)	0.8	1.2	3
other shales	0.1–4.0 (2.6)	1.6–3.8 (2.6)	10–55 (19)	0.7–3.0 (1.5)	1.2–5 (2.3)	6–19 (13)
arenites	0.0–5.5 (1.8)	0.7–5.1 (2.3)	4–22 (12)	0.1–2.4 (1.3)	1.2–4.4 (2.1)	7–18 (11)
carbonates	0.0–0.5 (0.2)	0.4–2.9 (1.6)	0–2.9 (1.4)			

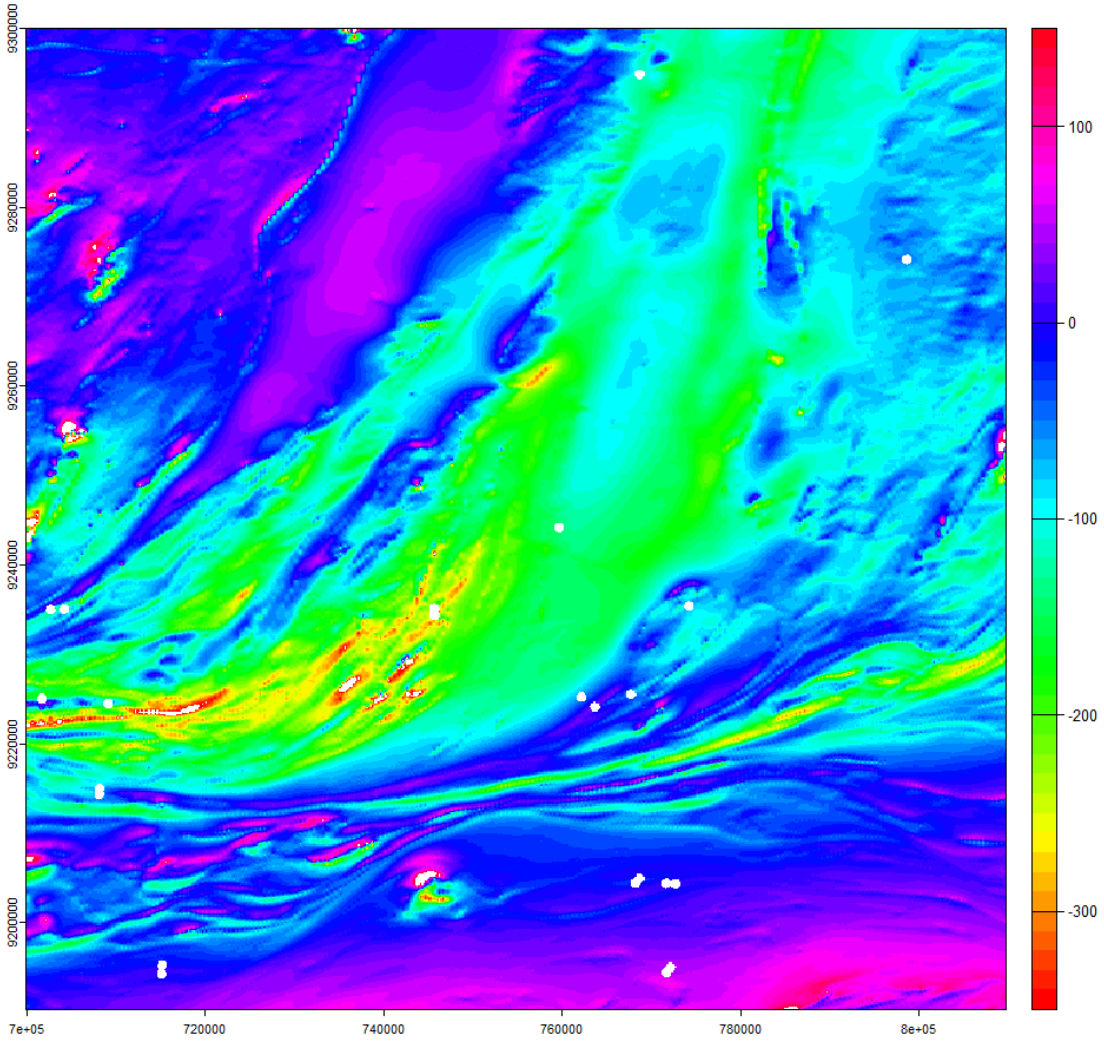
From: B.L. Dickson & K.M. Scott 1997 AGSO journal of Australian geology & geophysics 17:187-199 17:187-199

Magnetometry

Extracting layers from magnetometric surveys can be achieved using the following simple procedure.

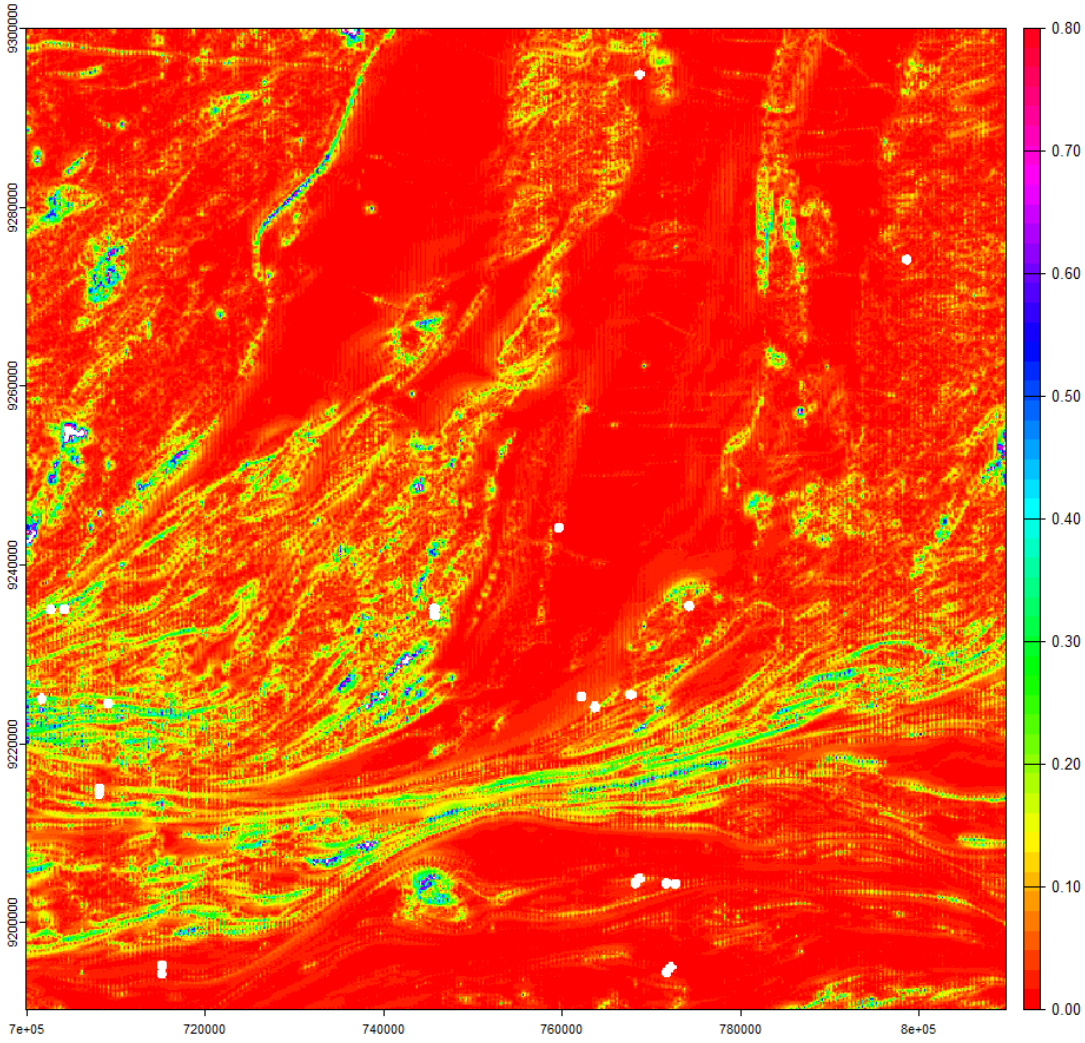
Total Magnetic Field

```
library(terra)
#Working directory where the images and data are located
wd<-'C:/Users/User/Desktop/R algo/WORKING_DIR/'
setwd(wd)
master<-rast('prep_mosaic.tif')
tmf<-master[[15]]
plot(tmf,col=rainbow(50),range=c(-350,150))
```



Analytical Signal

```
asa<-terrain(tmf,v='slope',unit='radians')
plot(asa,col=rainbow(50),range=c(0,0.8))
```

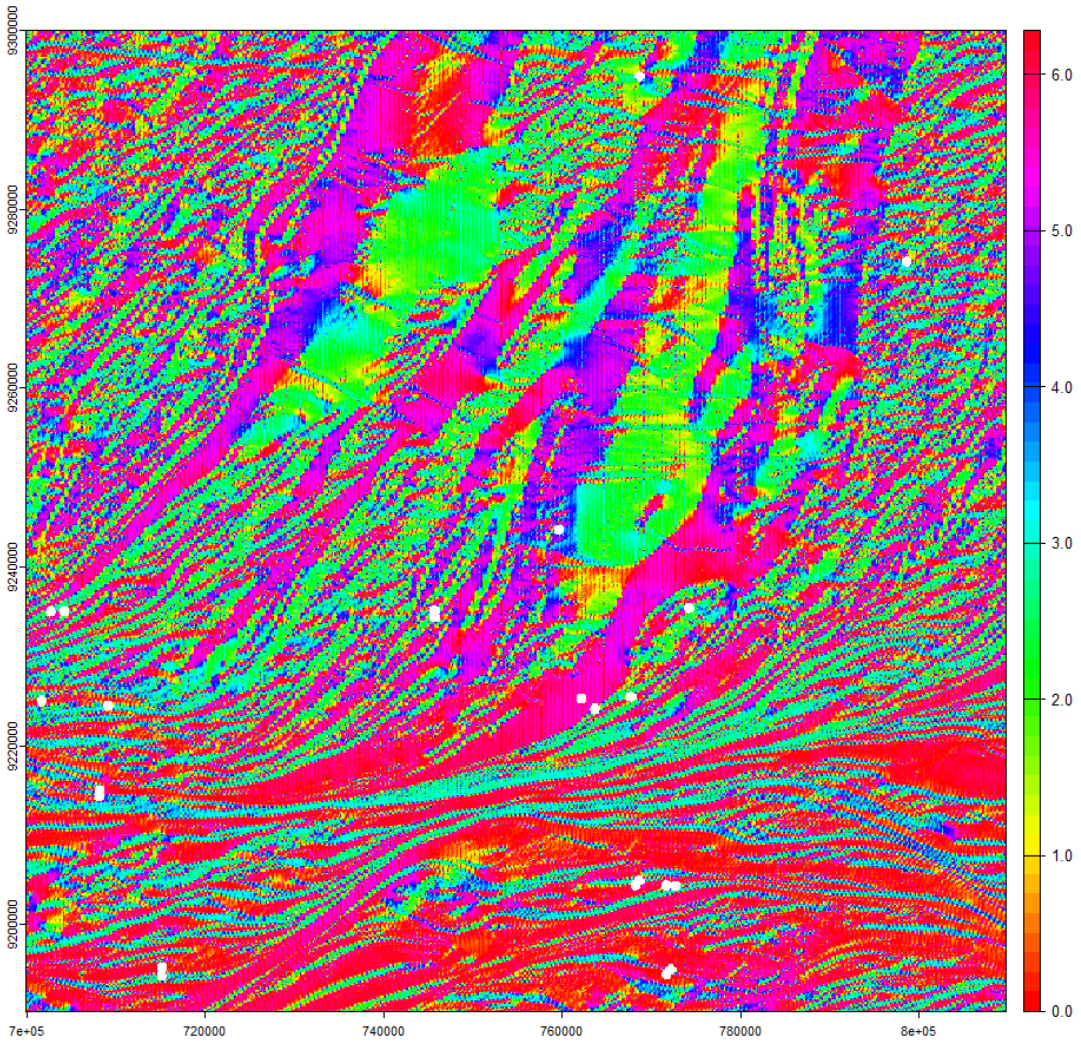



```
writeRaster(asa,'asa.tif', overwrite=TRUE)
```

Aspect (Pseudo First Derivative)

The aspect is an alternative way to get the same structures highlighted by a real first derivative of the total field processing.

```
dv1<-terrain(tmf,v='aspect',unit='radians')
plot(dv1,col=rainbow(50))
```

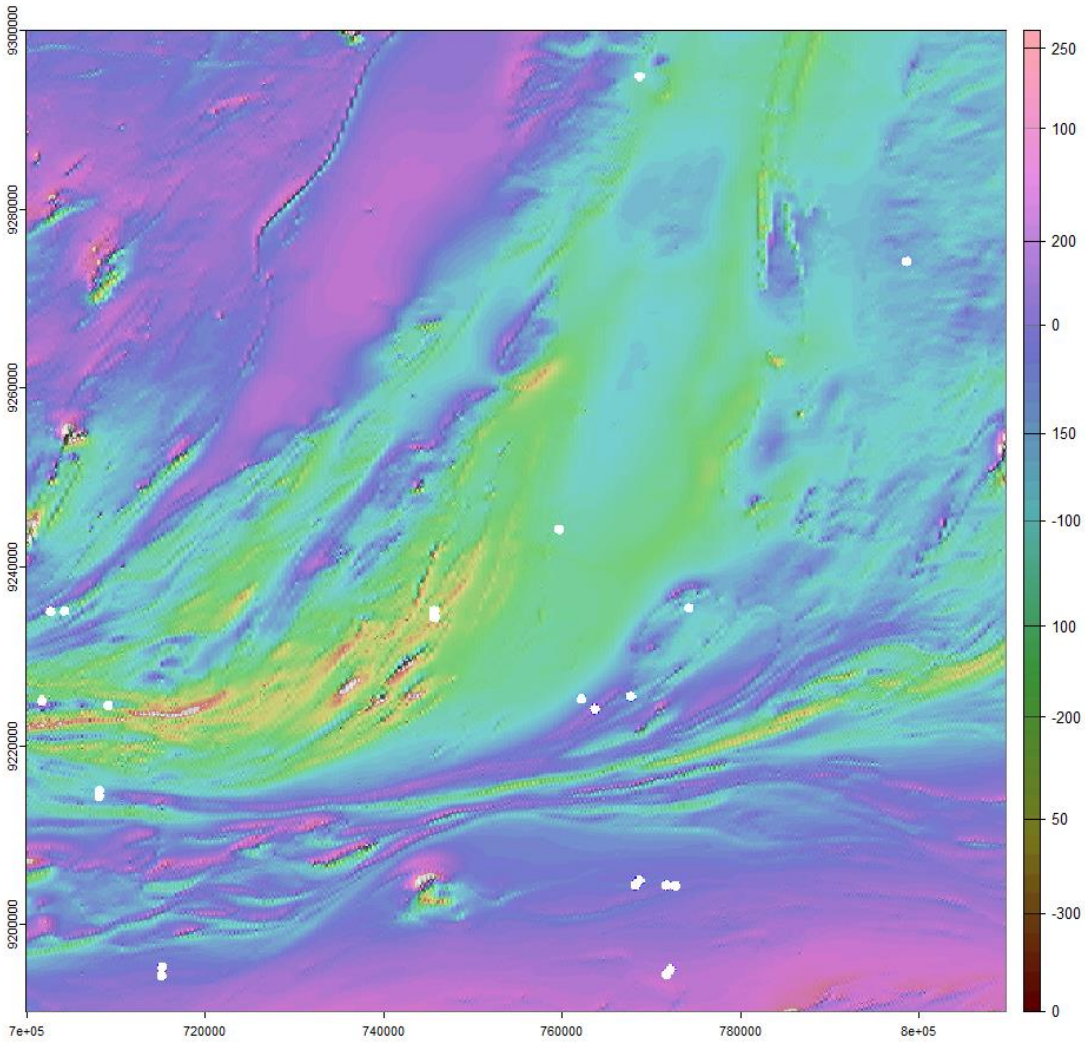



```
writeRaster(dv1,'dv1.tif', overwrite=TRUE)
```

Validation

Layers generated here are important to understand the potential association of mineralized zones and deeper structures.

```
relevo<-shade(asa,dv1,angle=45,direction=270,normalize= TRUE)
plot(tmf,col=rainbow(50),range=c(-350,150))
plot(add=T,relevo,col=grey(0:100/100),zlim=c(0,250),alpha=0.65)
```



Gravity Data

The gravity data for this exercise was created in Part 1.

Validation

The 3 gravity products are modeled from the earth topography and the disturbances calculated from the Bouguer values. They may be useful in some distinct kinds of mineralization correlated to topographic signature.

Creating the reference master stack

We conclude this Part creating a stacked raster with some of the layers already created from Part 1 and the ones created here.

```
 library(terra)
#Working directory where the images and data are located
wd<-'C:/Users/User/Desktop/R algo/WORKING_DIR/'
setwd(wd)
master<-rast('prep_mosaic.tif')
dem<-master[[10]]
k<-master[[11]]
u<-master[[12]]
th<-master[[13]]
tc<-master[[14]]
mtf<-master[[15]]
asa<-rast('asa.tif')
dv1<-rast('dv1.tif')
bg<-master[[16]]
gdist<-master[[17]]
gdist2dv<-master[[18]]
geo1<-rast('geology.tif')
geo2<-rast('geology_grp.tif')
crests<-rast('crests.tif')
lines<-rast('lines.tif')
lines<-resample(lines,crests)
rtUTh<-rast('rtUTh.tif')
rtUK<-rast('rtUK.tif')
rtThK<-rast('rtThK.tif')
ff<-rast('ff.tif')
rm(master)
targ_stack<-rast(list(geo1,geo2,dem,crests,lines,k,u,th,tc,rtUTh,rtUK,rtThK,
                      ff,mtf,asa,dv1,bg,gdist,gdist2dv))
names(targ_stack)<-c('geology','geology2','DEM','Crests','Lines','Kperc',
                     'eU','eTh','tot_count','U_Th','U_K','Th_K','Ffactor',
                     'Tot_Field','ASA','pseud_1dv','Bouguer',
                     'GDisturb','g2dvDistrurb')
writeRaster(targ_stack,'targ_stack.tif', overwrite=TRUE)
```

The **19 bands** raster file generated above is available at https://gdatasystems.com/targeting/

In Part 3 we will move into Mineral Exploration Targeting topic using this file.