Supplemental Materials: Assessment of Mining Extent and Expansion in Myanmar Based on Freely-Available Satellite Imagery

Katherine J. LaJeunesse Connette, Grant Connette, Asja Bernd, Paing Phyo, Kyaw Htet Aung, Ye Lin Tun, Zaw Min Thein, Ned Horning, Peter Leimgruber and Melissa Songer

```
# ##########################################################################
# Make a mask of potential mining areas to focus search efforts
# Based on a multivariate normal model for LC8 reflectance data from
known mines
#
# Inputs:
# * LC8 Imagery - TOA reflectance with bands labeled "im_B4.tif", etc.
#   contained within a folder labeled with the Path-Row (e.g., 132_49)
# * Polygon shapefile of known mine locations
#
# Output:
# * Raster layer giving relative likelihood of potential mining locations
#
##########################################################################

# USER INPUTS NEEDED
##########################################################################

# Load Libraries
library(raster)
library(rgdal)
library(mvtnorm)
library(jagsUI)

# Step 1: Where is Landsat Data Saved?
OutFolder <- "/Volumes/My
PassportAndrea/MappingProject/Data/LSAT/01_MiningImagery/Output"

# Step 2: Specify tiles to use for making the filter
TilesForFilter <-
c("132_49","132_48","133_48","133_47","133_46","134_47","134_46","134_45"
,"134_44")

# Step 3: Load a polygon shapefile of known mines
MINES <- readOGR(dsn="/Volumes/My
PassportAndrea/MappingProject/MiningProject/StatesRegions/Magway",
```

```
                  layer="KHA_Feb_26_2015")

##########################################################################
#####################################

# Reproject mine shapefile to UTMs
MINES <- spTransform(MINES,crs("+proj=utm +zone=47 +datum=WGS84 +units=m
+no_defs +ellps=WGS84 +towgs84=0,0,0"))

SavedRefl <- list()

# Loop over each landsat tile to be covered with the current filter
for (TILE in 1:length(TilesForFilter)){

  # Extract TOA reflectance for bands 4-7 from known mine areas
  B4 <-
extract(raster(file.path(OutFolder,TilesForFilter[TILE],"im_B4.tif")),MIN
ES)
  B4 <- as.numeric(unlist(B4))

  B5 <-
extract(raster(file.path(OutFolder,TilesForFilter[TILE],"im_B5.tif")),MIN
ES)
  B5 <- as.numeric(unlist(B5))

  B6 <-
extract(raster(file.path(OutFolder,TilesForFilter[TILE],"im_B6.tif")),MIN
ES)
  B6 <- as.numeric(unlist(B6))

  B7 <-
extract(raster(file.path(OutFolder,TilesForFilter[TILE],"im_B7.tif")),MIN
ES)
  B7 <- as.numeric(unlist(B7))

  # Calulate Indices
  NDVI <-(B5-B4)/(B5+B4)
  NBR <- (B5-B7)/(B5+B7)
  NDMI <- (B5-B6)/(B5+B6)
  SWIR <- (B6)/1000

  # Combine data for filter
  MINEREFL <- cbind(NDVI,NBR,NDMI,SWIR,B4/1000)
```

```
  # Filter out NA pixels due to cloud
  MINEREFL <- MINEREFL[which(!is.na(MINEREFL[,5])),]


  # Exclude Water b/c not MV-Normal with Bare Ground (Zhe's Water Test)
  WT <- ((NDVI<0.01)&(B4<1100))|((NDVI<0.1)&(NDVI>0)&(B4<500))
  WT <- WT[which(!is.na(WT))]
  MINEREFL <- MINEREFL[-WT,]  # Remove water pixels

  SavedRefl[[TILE]] <- MINEREFL
}

SavedRefl <- do.call("rbind", SavedRefl)

# Calculate the number of landsat bands used to build the filter
ndim=ncol(MINEREFL)

################################################################################
########
# Specify multivariate normal model for filter

sink("mix.txt")
cat("
model {

  # Specify priors
  mu[1:ndim] ~ dmnorm(mu0[],M1[,])
  tau[1:ndim,1:ndim] ~ dwish(M2[,],ndim)
  sigma[1:ndim,1:ndim] <- inverse(tau[1:ndim,1:ndim])

  # Specify likelihood
  for(i in 1:nind){
    Y[i,1:ndim] ~ dmnorm(mu[],tau[,])
  }
}
",fill = TRUE)
sink()

# Bundle data to analyze in JAGS
mu0 = apply(SavedRefl,2,mean)
M1 = diag(0.01,ndim,ndim)  # Variance-covariance matrix for mv-normal
priors for mus
```

```
M2 = diag(0.01,ndim,ndim)  # Prior for precision matrix
mod.data <-
list(Y=SavedRefl,nind=dim(SavedRefl)[1],M1=M1,M2=M2,mu0=mu0,ndim=ndim)

# Call JAGS from R (< 1 min)
system.time(OUT <- jags(mod.data, inits=NULL,
parameters.to.save=c("mu","sigma"), "mix.txt",
                        n.chains = 3, n.thin = 2, n.iter = 500, n.burnin
= 300))

print(OUT,3)   # Output
ModMean <- as.numeric(OUT$mean$mu)  # Estimated means for each band
ModVar <- OUT$mean$sigma   # Estimated variance-covariance matrix

##############################################################################
########
# Create function to calculate probability density for each point on the
landscape

MakeFilter <- function(x, mean, sigma, filename) {
  out <- raster(x)
  bs <- blockSize(out,minrows=1000)
  out <- writeStart(out, filename, overwrite=TRUE)
  for (i in 1:bs$n) {  # loop over raster blocks and calculate indices
    cat(paste0(i," of ", bs$n),"\n")
    B4_A <-
getValues(raster(file.path(OutFolder,TilesForFilter[TILE],"im_B4.tif")),r
ow=bs$row[i],nrows=bs$nrows[i])
    B5_A <-
getValues(raster(file.path(OutFolder,TilesForFilter[TILE],"im_B5.tif")),r
ow=bs$row[i],nrows=bs$nrows[i])
    B6_A <-
getValues(raster(file.path(OutFolder,TilesForFilter[TILE],"im_B6.tif")),r
ow=bs$row[i],nrows=bs$nrows[i])
    B7_A <-
getValues(raster(file.path(OutFolder,TilesForFilter[TILE],"im_B7.tif")),r
ow=bs$row[i],nrows=bs$nrows[i])

    NDVI_A <-(B5_A-B4_A)/(B5_A+B4_A)
    NBR_A <- (B5_A-B7_A)/(B5_A+B7_A)
    NDMI_A <- (B5_A-B6_A)/(B5_A+B6_A)
    SWIR_A <- (B6_A)/1000
```

```
    ALLREFL <- cbind(NDVI_A,NBR_A,NDMI_A,SWIR_A,B4_A/1000)

    Vals <- dmvnorm(ALLREFL, mean=mean, sigma=sigma)
    out <- writeValues(out, Vals, bs$row[i])
  }
  out <- writeStop(out)
  return(out)
}

for (TILE in 1:length(TilesForFilter)){
  # Load one band to serve as template raster for the potential mining
mask
  tmp <- raster(file.path(OutFolder,TilesForFilter[TILE],"im_B2.tif"))

  # Create final raster
  s <- MakeFilter(tmp, mean=ModMean, sigma=ModVar,

filename=file.path(OutFolder,TilesForFilter[TILE],paste0("Filter_",TilesF
orFilter[TILE],".tif")))
}
```

Scheme S1. R scripts for creating a raster of potential mining areas.

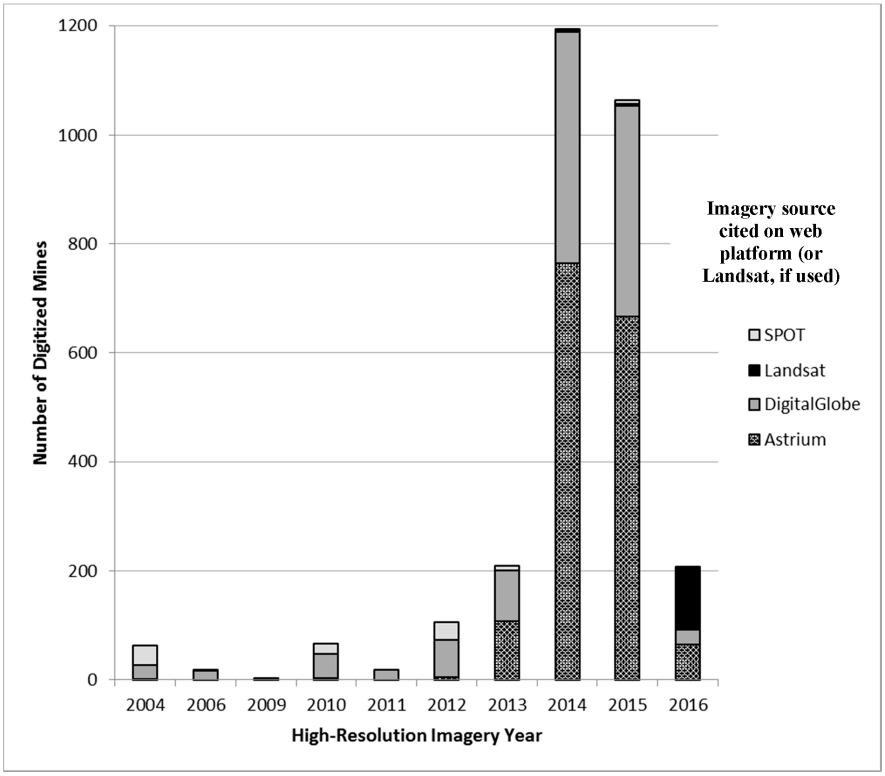


Figure S1. Distribution of years and satellite platform for high-resolution imagery used to digitize mining areas.

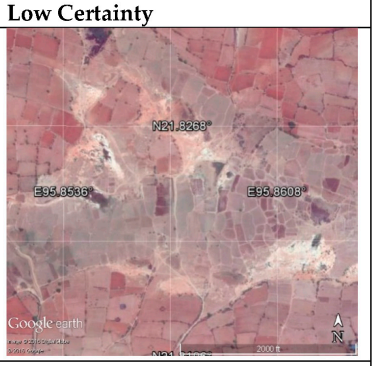
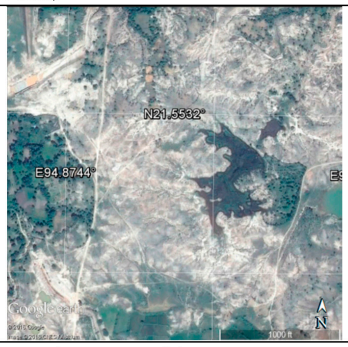
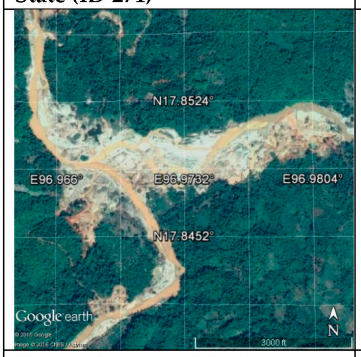
High certainty	Medium Certainty	Low Certainty
		
Taunggyi Township, Shan State (ID 2432)	**Kutkai Township, Shan State (ID 2681)**	**Tada-U Township, Mandalay Division (ID: 696)**
		
Hpakant Township, Kachin State (ID 271)	**Myaing Township, Magway Division (ID 537)**	**Ngazun Township, Mandalay Division (ID 703)**
		
Shwegyin Township, Bago Division (ID 2893)	**Lawksawk Township, Shan State (ID 2387)**	**Taungtha Township, Mandalay Division (ID 821)**

Figure S2. Examples of high, medium, and low certainty mining areas (Images exported from Google Earth Pro).

```
##############################################################################################
# Revised July 2016

# This script was written to calculate the location of new extractive mines using two Landsat ETM+
# and OLI images acquired on different dates and a polygon Shapefile outlining recent mines. The
# user needs to enter the location of the early Landsat images (these are the two-date image
# stacks from the land cover mapping work), the location of the directory containing all of the
# recent image sub-directories, and the output image directory. In addition to the image location
# information the user needs to enter the no-data value (probably 0) for both the early and
# recent images, the threshold for the difference in albedo between the two dates and the recent
# image albedo threshold. Three output classes are calculated: no-data (DN=255) for pixels outside
```

```
# of the recent mine locations, mining on the early date and recent date image (DN=0), and no mine
# on early date and mine on recent date (DN=1). A mine is classified as a change from no mine to
# mine if the difference between the recent and early albedo is greater than the diffAlbedoThreshold
# and the recent albedo is greater than the diffAlbedoThreshold value.
#
# To account for differences in overall albedo between the two images the average albedo for the early
# and recent date albedo images is calculated and the diffAlbedoThreshold and diffAlbedoThreshold
# values are normalized.

# It is important to note that these are global image adjustments and if there is haze in part of one
# image that could impact the results since haze will increase pixel albedo.
#
# The mines Shapefile is rasterized, after projecting it to the same projection as the early date image,
# using the input image extent and resolution. A pixel in counted as being in a polygon if the polygon
# covers the center of a raster cell.

# The output images are copied to the output directory and will be labeled with the image path/row.
# If a full-image difference is useful the user can uncomment the two lines under the comment
# Output diffImage.
#
#Set the variables below in the "SET VARIABLES HERE" section of the script.

#This script was written by Ned Horning [horning@amnh.org] and edited by Grant Connette [grmcco@gmail.com]

#This script is free software; you can redistribute it and/or modify it under the terms of the GNU General
# Public License as published by the Free Software Foundation either version 2 of the License, or ( at
# your option ) any later version.
#
################################################################################

#####
#Load libraries
require(raster)
require(rgdal)
library(stringr)
###################  SET VARIABLES HERE  ############################
# Data set name for the vector file containing digitized mine polygons. This must
include the ".shp" extension
miningLayerDsn <- 'J:/FinalDigitizedMines/The_whole_country_mine.shp'

# Directory with the composite images with the early date images (Landsat 2002)
compositeImageDir <- "J:/Data/Landsat/2002_2014stacks"

# Directory with directories which have most recent image bands (Landsat 2015)
```

```
recentImageDir <- "J:/Data/Landsat/LSAT_2015"

# Output directory
outputDir <- "J:/MiningOutputfinal2"

# satellite image nodata value for early image.
early_nd <- 0

# satellite image nodata value for recent image.
recent_nd <- 0

# Threshold for change in albedo
diffAlbedoThreshold <- 300.0
recentAlbedoThreshold <- 1150.0
################################################################################
##########
#rasterOptions(tmpdir=tmpDir)

# Functions for calculating sun corrected reflectance for Landsat TM and OLI
imagery
fun_SunReflOLI <- function(x) {x / cos(solarZenithRad) }

# Functions for albedo and difference in albedo
###albedoFun <- function(B1, B2, B3, B4, B5) {((0.356*B1) + (0.130*B2) +
(0.373*B3) + (0.085*B4) + (0.072*B5) -0.018) / 1.016}
#albedoFun <- function(B1, B2, B3, B4, B5) {((0.443*B1) + (0.317*B2) +
(0.240*B3))}
# Difuse visible albedo
albedoFun <- function(B1, B2, B3) {round(((0.556*B1) + (0.281*B2) + (0.163*B3) -
0.0014))}

albedoDiffFun <- function(earlyAlbedo,recentAlbedo){
  ifelse(  (((recentAlbedo-earlyAlbedo)>diffAlbedoThreshold) &
(recentAlbedo>recentAlbedoThreshold))==1, 2,
           # If change and bright, assign a 2
           (recentAlbedo>recentAlbedoThreshold))
  # If not, assign 0 if dark to dark or bright to dark, 1 if bright to bright
}

calcTOA_ReflFun <- function(imageBandName, index) {
    # Apply gain and offset for reflectance
    imageBand <- raster(imageBandName)
    gain(imageBand) <- reflectanceMult[index]
    offs(imageBand) <- reflectanceAdd[index]
```

```
    round(calc(imageBand, fun_SunReflOLI) * 10000)
    #calc(imageBand, fun_SunReflOLI) * 10000
}

## Start processing
startTime <- Sys.time()
cat("Start time", format(startTime),"\n")

# Read the vector file
cat("Reading the vector file\n")
# Parse out the layer name from the miningLayerDsn text string
miningLayerFilename <- unlist(strsplit(miningLayerDsn,
"/"))[length(unlist(strsplit(miningLayerDsn, "/")))]
miningLayerName <- unlist(strsplit(miningLayerFilename, "\\."))[1]
vec <- readOGR(miningLayerDsn, miningLayerName)

compositFiles <- list.files(compositeImageDir)

recentImageFileList <- list.files(recentImageDir)

for (i in 1:length(compositFiles)) {
    pathRow <- paste(substr(compositFiles[i], 1,3), substr(compositFiles[i],
5,6), sep="_")
    cat("Processing path/row:", pathRow, "\n")
    earlyImageName <- paste(compositeImageDir, compositFiles[i], sep="/")

    # Read the satellite image stack and set the no-data value
    earlyImage <- brick(earlyImageName)
    NAvalue(earlyImage) <- early_nd

    # Read band 2, 3, 4 from the recent Landsat 8 image directory
    pathRow <- paste("LC8", sub("_", "0", pathRow), sep="")
    recentDirectoryIndex <- grep(pathRow, recentImageFileList)
    recentDirectoryName <- recentImageFileList[recentDirectoryIndex]

    Landsat8Dir <- paste(recentImageDir, recentDirectoryName, sep="/")
    fileName <- list.files(path = Landsat8Dir, pattern="MTL.txt")
    bandList = c(2,3,4)
    # Open the metadata file
    conn=file(paste(Landsat8Dir, fileName, sep="/"),open="r")
    # Read the meatadata file and store lines in linn variable
    linn=readLines(conn, warn=FALSE)
    close(conn)
```

```
    # Get solar zenith angle
    for (i in 1:length(linn)){
        if(str_detect(linn[i], "SUN_ELEVATION")) {
            solarZenithDeg <- 90 - as.numeric(unlist(strsplit(linn[i], "="))[2])
            solarZenithRad <- solarZenithDeg * pi /180
        }
    }

    # Initialize arrays to hold reflectance gain and offsets for each band and
set array counters to 1
    reflectanceAdd <- integer()
    reflectanceAddCount <- 1
    reflectanceMult <- integer()
    reflectanceMultCount <- 1

    # Search for reflectance Add (offset) and Mult (gain) values in the metadata
file and insert values into array
    for (i in 1:length(linn)){
        reflectanceAddName <- paste("REFLECTANCE_ADD_BAND_",
bandList[reflectanceAddCount], " ", sep="")
        reflectanceMultName <- paste("REFLECTANCE_MULT_BAND_",
bandList[reflectanceMultCount], " ", sep="")
        if(str_detect(linn[i], reflectanceAddName)) {
            reflectanceAdd[reflectanceAddCount] <-
as.numeric(unlist(strsplit(linn[i], "="))[2])
            reflectanceAddCount <- reflectanceAddCount + 1
        }
        if(str_detect(linn[i], reflectanceMultName)) {
            reflectanceMult[reflectanceMultCount] <-
as.numeric(unlist(strsplit(linn[i], "="))[2])
            reflectanceMultCount <- reflectanceMultCount + 1
        }
    }

    band2 <- paste(Landsat8Dir, paste(recentDirectoryName, "_B2.TIF", sep=""),
sep="/")
    band3 <- paste(Landsat8Dir, paste(recentDirectoryName, "_B3.TIF", sep=""),
sep="/")
    band4 <- paste(Landsat8Dir, paste(recentDirectoryName, "_B4.TIF", sep=""),
sep="/")

    cat("Calculating TOA reflectance\n")
    band2TOA <- calcTOA_ReflFun(band2, 1)  # Second number is index in the
bandList vector
```

```
    band3TOA <- calcTOA_ReflFun(band3, 2)
    band4TOA <- calcTOA_ReflFun(band4, 3)

    recentImage <- stack(band2TOA, band3TOA, band4TOA)
    NAvalue(recentImage) <- recent_nd

    recentImageFilename <- paste(outputDir, "/PR", pathRow, "_recentImage.tif",
sep="")

    # Read in the cloud/shadow mask from fmask output
    cloudmaskpath <- paste(Landsat8Dir, paste(recentDirectoryName, "_MTLFmask",
sep=""), sep="/")
    cloudmask <- raster(cloudmaskpath)

    # This cropping approach assumes that the pixels line up between the early
and recent dates
    # and that the projections are the same
    cat("Cropping early and recent images so they match\n")
    commonExtent <- intersect(extent(earlyImage), extent(recentImage))
    earlyImage <- crop(earlyImage, commonExtent)
    recentImage <- crop(recentImage, commonExtent)
    cloudmask <- crop(cloudmask, commonExtent)
    # If the image pixels for the early and recent images do not line up exactly
or the projections are different
    # Then uncomment the following line and add comments to the three lines above
this comment block
    # recentImage <- projectRaster(recentImage, earlyImage, method="ngb")

    # Create the blank raster image that will be used to rasterize the Shapefile
    mineRaster <- raster(ext= extent(earlyImage), crs=crs(earlyImage) ,
nrows=nrow(earlyImage), ncols=ncol(earlyImage) , vals=NULL)

    cat("Rasterizing mining vector file\n")
    vec <- spTransform(vec, crs(earlyImage))
    mineRaster <- rasterize(vec, mineRaster, field=1)

    cat("Calculating change in albedo\n")
    # First calcualte early and recent albedo images
    earlyAlbedo <- overlay(earlyImage[[1]], earlyImage[[2]], earlyImage[[3]],
fun=albedoFun)
    recentAlbedo <- overlay(recentImage[[1]], recentImage[[2]], recentImage[[3]],
fun=albedoFun)
```

```
    # ######################## Normalize albedo
##################################
    # earlyAlbedoSamps <- sampleRegular(earlyAlbedo, 100000)
    # recentAlbedoSamps <- sampleRegular(recentAlbedo, 100000)
    #
    # albedoRegression <- lm(earlyAlbedoSamps ~ recentAlbedoSamps)
    #
    # print(albedoRegression)
    # cat("r squared =",summary(albedoRegression)$adj.r.squared, "\n")
    #
    # earlyAlbedo <- earlyAlbedo * albedoRegression[[1]][2] +
albedoRegression[[1]][1]
    #
##############################################################################



    # Here we will add cloud and shadow removal
    earlyAlbedo[which((cloudmask[]==2)|(cloudmask[]==4))] <- NA
    recentAlbedo[which((cloudmask[]==2)|(cloudmask[]==4))] <- NA

    # Make missing values NA in recent albedo image
    id_missing<-(recentAlbedo[]<=0)
    recentAlbedo[id_missing==1] <- NA

    # Mask out parts of image not included in early composite (primarily outside
country boundaries)
    recentAlbedo <- mask(recentAlbedo,earlyAlbedo,maskvalue=NA)

    # Calculate albedo difference image
    diffImage <- overlay(earlyAlbedo, recentAlbedo, fun=albedoDiffFun)

    #Replace "no data" pixels with 99.  This is so that we will have a value for
cloud holes that are within polygons.  - now added above
    diffImage[is.na(diffImage[])] <- 99

    # Output mining change image name
    cat("Applying mask and output\n")
    miningChangeFilename <- paste(outputDir, "/PR", pathRow, "_Change.tif",
sep="")
    diffImage <- mask(diffImage, mineRaster, filename=miningChangeFilename,
overwrite=TRUE, datatype='INT1U')

    # Remove all temporary files
    removeTmpFiles(h=0)
```

```
}

# Calculate processing time
timeDiff <- Sys.time() - startTime
cat("Processing time", format(timeDiff), "\n")
```

Scheme S2. R scripts for mining change analysis.

Mining Change for Medium- and Low-Certainty Mining Areas

Table S1. Mining area change in hectares for medium-certainty mining areas.

State/Region	New Bare Ground in A Mining Area	Existing Bare Ground in A Mining Area	Vegetation within Mining Area	Cloud Interference over A Mining Area	Total Number of Hectares	Percent of Bare Ground That Is New *
Ayeyarwady	65	11	17	4	97	-
Bago	236	51	293	61	641	82%
Chin	3	-	1	1	5	-
Kachin	899	522	585	287	2293	63%
Kayah	-	1	1	-	2	-
Kayin	11	-	18	-	29	-
Magway	219	272	101	-	592	45%
Mandalay	633	2742	200	852	4427	19%
Mon	190	18	82	12	302	91%
Naypyitaw	67	13	7	11	98	-
Rakhine	-	1	-	-	1	-
Sagaing	12,069	2043	559	923	15,594	86%
Shan	326	114	858	384	1682	74%
Tanintharyi	81	30	272	63	446	73%
Yangon	7	3	8	24	42	
Total	14,806	5821	3002	2622	26,251	72%

* For states with more than 100 ha of mining.

Table S2. Mining area change in hectares for low-certainty mining areas.

State/Region	New Bare Ground in A Mining Area	Existing Bare Ground in A Mining Area	Vegetation within Mining Area	Cloud Interference over A Mining Area	Total Number of Hectares	Percent of Bare Ground That Is New *
Ayeyarwady	-	4	1	2	7	-
Bago	98	49	120	48	315	67%
Chin	-	-	-	12	12	-
Kachin	287	295	421	179	1182	49%
Kayah	-	-	-	-	-	-
Kayin	-	-	-	-	-	-
Magway	-	-	-	-	-	-
Mandalay	478	2519	22	805	3824	16%
Mon	17	1	33	-	51	-
Naypyitaw	14	2	-	5	21	-
Rakhine	-	-	7	-	7	-
Sagaing	1954	914	481	328	3677	68%
Shan	542	43	951	255	1791	93%
Tanintharyi	201	45	313	27	586	82%
Yangon	-	-	-	-	5	-
Total	3591	3872	2349	1666	11,478	48%

* For states with more than 100 ha of mining.