Source Code for the India Annual Winter Cropped Area, 2001-2016

August 2017

This document contains the source code used to construct the India Annual Winter Cropped Area, 2001-2016 data set.

Recommended citation for the Source Code:

Jain, M., P. Mondal, G. L. Galford, G. Fiske, and R. S. DeFries. 2017. Source Code for the India Annual Winter Cropped Area, 2001-2016. Palisades NY: NASA Socioeconomic Data and Applications Center (SEDAC). https://doi.org/10.7927/H47D2S3W. Accessed DAY MONTH YEAR.

Source Code:

```
library(raster)

library(rgdal)

library(snow)

library(parallel)

library(shapefiles)

library(maptools)

library(dplyr)


setwd('dir') # change this to the desired working directory

sa=brick('file.tif') # change file.tif to be the geotiff of the EVI time series


########### SPLINE SMOOTH TIME SERIES OF EVI FOR EACH PIXEL ##############

beginCluster(n=7) # this should be changed to use the desired number of cores for parallel processing
```

```
  spfuncl<-function(y){
    num=nlayers(y)
    spl60fun<-function(x){
       x[x==255]<-NA # in our data, 255 was set to represent NA values
    if (sum(is.na(x))>20)
      {spline<-rep(NA,num)
      return(spline)}
    else
      if (is.na(x[1]))
      {spline<-rep(NA,num)
      return(spline)}
      else
        if (is.na(x[num]))
        {spline<-rep(NA,num)
        return(spline)}
      else
      {x1<-c(1:length(x))
      y1<-x
      intx<-approx(x1,y1,xout=1:length(x1))$y
      spline<-smooth.spline(intx,df=60)$y
      return(spline)}}
    y2<-calc(y,spl60fun)
    return(y2)}
  sasp<-clusterR(sa,spfuncl)
endCluster()
```

```
############# CREATE A MASK TO MASK OUT ANY PIXELS WHERE > 20 PIXELS ARE NA IN THE TIME SERIES (NA MASK) ######

beginCluster(n=7) # this should be changed to use the desired number of cores for parallel processing

  mask255cl<-function(y){

    mask255<-function(x){

      namask<-ifelse(sum(is.na(x))>20,NA,1)

      return(namask)}

    y2<-calc(y,mask255)

    return(y2)}

  sasp255m<-clusterR(sasp,mask255cl)

  writeRaster(sasp255m,paste("sasp255m",name,'.tif',sep=''),format="GTiff",overwrite=TRUE)

endCluster()


############### SUBSET THE SPLINED TIME SERIES INTO THE SEASON OF INTEREST FOR EACH YEAR #############

saspwin=list()

for(i in c(0:15)){saspwin[[i+1]]=subset(sasp,(i*23+11):(i*23+22))} # these values should change to the values that correspond with the start and end of the growing season of interest

save(saspwin,file=paste('saspwin',name,'.Rdat',sep=''))

rm(sa) # these 3 steps are taken to remove old files and clear memory

rm(sasp)

gc()


################ CREATE A BINARY PEAK MASK WHERE 1 = PIXELS THAT HAVE A PEAK AND 0 = PIXELS THAT DO NOT HAVE A PEAK (PEAK MASK) ########

beginCluster(n=7) # this should be changed to use the desired number of cores for parallel processing

peakmaskcl<-function(y){

  peakmask<-function(x){
```

```
    dx<-diff(x)
    value<-rep(0,length(dx))
    adx<-ifelse(dx>0,1,0)
    for(i in 1:length(adx)){value[i]<-ifelse(adx[i] == 1 & adx[i+1]==0,i+1,0)}
    value2=rep(0,length(dx))
    adx2<-ifelse(dx<0,1,0)
    for(i in 1:length(adx2)){value2[i]<-ifelse(adx2[i] == 1 & adx2[i+1]==0,i+1,0)}
    sum1=sum(value,na.rm=TRUE)
    sum2=sum(value2,na.rm=TRUE)
    mask<-ifelse(sum1>0&sum2>0&sum2<sum1,1,NA)
    return(mask)}
  y2<-calc(y,peakmask)
  return(y2)}
saspwinpm=list()
for(i in c(1:length(saspwin))){saspwinpm[[i]]<-clusterR(saspwin[[i]],peakmaskcl)}
endCluster()
save(saspwinpm,file=paste('saspwinpm',name,'.Rdat',sep=''))


################ APPLY BOTH THE NA MASK AND THE PEAK MASK TO THE SPLINED SEASON DATASETS ###########
saspwinm=list()
for(i in c(1:length(saspwin))){saspwinm[[i]]<-mask(saspwin[[i]],saspwinpm[[i]])}
saspwinmm=list()
for(i in c(1:length(saspwinm))){saspwinmm[[i]]<-mask(saspwinm[[i]],sasp255m)}
save(saspwinmm,file=paste('saspwinmm',name,'.Rdat',sep=''))
```

```
################### CALCULATE THE MAXIMUM PEAK VALUES FOR ALL PIXELS THAT REMAIN UNMASKED  ##############################

beginCluster(n=7) # this should be changed to use the desired number of cores for parallel processing

peakvalcl<-function(y){
  peakval<-function(x){
    x<-as.vector(x)
    if (is.na(x[1]))
      return(NA)
    else
      if (x[1] == 0)
        return(0)
    else
      dx<-diff(as.vector(x))
    adx<-ifelse(dx>0,1,0)
    for(i in 1:length(adx)){
      if (adx[i] == 1 & adx[i+1]==0)
      {value<-x[i+1]
      return(value)}}}
y2<-calc(y,peakval)
return(y2)}


maxsaspwin=list()
for(i in c(1:length(saspwinmm))){
  maxsaspwin[[i]]<-clusterR(saspwinmm[[i]],peakvalcl)
  writeRaster(maxsaspwin[[i]],paste('maxsaspwin',name,i,sep='_'),format="GTiff",overwrite=TRUE)}
endCluster()
```

```
################## CALCULATE THE MINIMUM PEAK VALUES FOR ALL PIXELS THAT REMAIN UNMASKED ##################

beginCluster(n=7) # this should be changed to use the desired number of cores for parallel processing

minvalcl<-function(y){
  minval<-function(x){
    x<-as.vector(x)
    if(is.na(x[1]))
    {value<-NA
    return(value)}
    else
      if(x[1]==0)
      {value<-NA
      return(value)}
    else
    {value<-min(x)
    return(value)}}
  y2<-calc(y,minval)
  return(y2)}


minsaspwin=list()

for(i in c(1:length(saspwinmm))){
  minsaspwin[[i]]<-clusterR(saspwinmm[[i]],minvalcl)
  writeRaster(minsaspwin[[i]],paste('minsaspwin',name,i,sep='_'),format="GTiff",overwrite=TRUE)}

endCluster()


################### PULL IN SHAPEFILE OF REGIONS OVER WHICH TO DO SCALING ###################
```

```
shapefile=readOGR('.','shapefile') # change this to the name of the shapefile


################## FUNCTION THAT WILL SCALE ALL MAX EVI VALUES BETWEEN 0 AND 100 WITHIN EACH REGION ###############

clip<-function(raster,shape) {

   a1_crop<-crop(raster,shape)

   step1<-rasterize(shape,a1_crop)

   a1_crop*step1}


scalefun=function(y,z,a){

   for(i in 1:length(shapefile)){

      value<-(shapefile[shapefile$OBJECTID==i,]) # change this to match the list of features within the shapefile

      valueclipmax<-clip(y,value)

      vcmax90<-as.numeric(quantile(valueclipmax,c(.90))) # currently this scales the 90th percentile max EVI value to be 100 - this can be changed to 0.95, 0.98, etc

      valueclipmin<-clip(z,value)

      vcmin10<-as.numeric(quantile(valueclipmin,c(.10))) # currently this scales the 10th percentile min EVI value to be 0 - this can be changed to 0.05, 0.02, etc

      m<-100/(vcmax90-vcmin10)

      b<- -(vcmin10*m)

      scalefun=function(x){

         result=m*x+b

         return(result)}

      vscaled<-calc(valueclipmax,fun=scalefun)

      vscaled[vscaled<0]<-0

      vscaled[vscaled>100]<-100

      vscaled[is.na(vscaled)]<-0
```

```
        na255value<-clip(sasp255m,value)
        vscaled2<-mask(vscaled,na255value)
        vscaled2[is.na(vscaled2)]<-(-1)

writeRaster(vscaled2,paste(as.character(a),as.character(i),'.tif',sep=""),format="GTiff",overwrite=TRUE)
}}


for(i in c(1:length(maxsaspwin))){
    scalefun(maxsaspwin[[i]],minsaspwin[[i]],'scale')}
```