Development of improved snow cover data using MODIS snow cover product

**MODIS**

MODIS stands for Moderate Resolution Imaging Spectroradiometer. It is a sensor lunched in 1999 through terra satellite and in 2002 onboard the Aqua satellite. This satellite has 36 spectral bands with spatial resolution ranges from 250m to 1000m. The revisit time (temporal resolution of this satellite is 1-2 days. Here is detail information about MODIS (<https://modis.gsfc.nasa.gov/about/> ).

**MODIS standard snow cover product**

MODIS standard snow cover product is produced by NSIDC on regular basis. This product is based of normalized difference snow index (NDSI) followed by some additional filtering like thermal masking. The snow product is available at 500 m and 5 km spatial resolution with daily, eight days, and monthly temporal scales. For 8-day maximum snow cover extent, the cell is mapped as snow if snow is observed in a cell on any day in the period. More information about MODIS snow cover extent can be obtained from <https://modis-snow-ice.gsfc.nasa.gov/uploads/C6_MODIS_Snow_User_Guide.pdf> .

**Improved maximum 8-day snow cover product**

Muhammad and Thapa (2020) improved MODIS 8-day snow cover over High Mountain Asia which reduces overestimation and underestimation by a huge margin. The product is referred to as MOYDGL06\* and is freely available to download from <https://rds.icimod.org/TemporalTIff?id=snow> . Underestimation of snow resulting from cloud cover is reduced applying seasonal filter followed by temporal and spatial filters whereas overestimation is reduced by combining aqua and terra snow cover products. Approximately 99.98% of the cloudy pixels equivalent to 3.66% of snow and 46% of overestimation in snow were removed in MOYDGL06\* product. The detailed methodology and dataset are described in Muhammad and Thapa (2020), <https://doi.org/10.5194/essd-12-345-2020> .

**Download MODIS snow cover product**

In this exercise, we download MODIS/Terra Snow Cover 8-Day L3 product and improve it implementing the algorithm described in Muhammad and Thapa (2020).

We use R/RStudio software to download and process the data. Several base packages are installed with the basic installation. But, these packages may not be sufficient when the user want to focus on specific application. In such case, user has to either develop their own function/packages or install external packages developed by R community. Lorenzo Busetto and Luigi Ranghetti have developed a package named “MODIStsp” which allows automation of time series of rasters derived from MODIS Land Products data including MODIS snow cover product.

**NOTE:** Note: The text in the box highlighted in white is the R code. Users / participants are encouraged to run these codes in R / R Studio.

Let’s install stable version of MODIStsp package from CRAN. Note that this package requires R verison >= 3.6.3.

install.packages("MODIStsp")

You might need several dependencies for this package (e.g. raster, rgdal). Use install.packages to install each of them. With the package and dependencies installed, we are ready to begin downloading.

Let’s clean the working environment and load the package that we need to download and process MODIS snow data.

rm(list=ls())  
library(MODIStsp)

A user who is interested in knowing the names of products and layers in the packages can retrieve that information by calling MODIStsp\_get\_prodlayers().

MODIStsp\_get\_prodnames()

In this exercise, we focus on the MODIS 8-day snow product. Let's take a closer look.

MODIStsp\_get\_prodlayers("M\*D10\_A2")

Setting the working directory before starting a specific task is often preferred. It is possible to do this manually using setwd, but for this exercise, we will use a function from the rstudioapi package. This function will automatically get the path to your script and set that as the working directory. Please note that this function does not work if your R script has not been saved. Let’s install and load rstudioapi package.

install.packages("rstudioapi")

library(rstudioapi)

setwd(dirname(getActiveDocumentContext()$path))

getwd()

Creating a new folder to store downloaded files in this directory and storing them in a variable is the next step.

dir.create('downloaded\_data')  
out\_path='./downloaded\_data'

Specify the time period you are interested in. Let's select a shorter duration so that the data download becomes faster.

#define start and end date  
s\_date='2022.01.01'   
e\_date='2022.02.20'

To crop the data, we now define an area of interest. The extents should be listed in the following order: xmin (minimum longitude), ymin (minimum latitude), xmax (maximum longitude), ymax (maximum latitude). The extent provided herein is of the Hunza River Subbasin in the Indus Basin, Pakistan as projected by WGSS84 (epsg=4326). However, it can be defined in any other standard projection system.

my\_extent=c(74,35.8,76,37.1) # x1,y1,x2,y2

The package downloads data from the NSIDC website. You need to open an account (register) at Earthdata login to access the NSIDC. Type in your userID and password in the following code.

my\_Earthdata\_Login\_userID="XXX"  
my\_Earthdata\_Login\_password="YYY"

The data is ready to be downloaded now. Run the following commands to download the data.

MODIStsp(gui = FALSE,   
 out\_folder = out\_path,   
 selprod = "Snow\_Cov\_8-Day\_500m (M\*D10\_A2)",  
 sensor = 'Both',  
 bandsel = c("MAX\_SNW"),  
 quality\_bandsel = NULL,  
 indexes\_bandsel = NULL,  
 user = my\_Earthdata\_Login\_userID ,  
 password = my\_Earthdata\_Login\_password,  
 start\_date = s\_date,   
 end\_date = e\_date,   
 spatmeth = "bbox",  
 out\_format = "GTiff",  
 out\_projsel = "User Defined",  
 output\_proj = 4326,  
 out\_res\_sel ="Native",  
 resampling ="near",  
 bbox = my\_extent,  
 verbose = TRUE,  
 compress = "LZW",  
 reprocess = TRUE,  
 delete\_hdf = TRUE,  
 ts\_format = FALSE  
)

The download will create a folder (./downloaded\_data/Snow\_Cov\_8-Day\_500m\_v6/MAX\_SNW) inside out\_path which will contain the output images.

**Temporal Filter**

Before implementing temporal filter, we reclassify the original class of snow product into three different classes (snow, no snow and cloud) for both aqua and terra sensors. We reclassify missing data (0), no decision (1), night (11), cloud (50) detector saturated sensor (254) and fill (255) classes to cloud. Additionally, lake ice (100) and snow (200) is reclassified as snow (200) whereas lake (37), ocean (39) and land (25) is reclassified as land (25).

Now, we implement temporal filter to replaces the cloudy pixels with non-cloudy pixels from the chronologically preceding and subsequent images. The basis for implementing temporal filter is snow cover remained constant under continuous cloudy conditions. In this exercise, we selected four images (two preceding and two subsequent 8 d composite images) at most for removing cloudy pixels. This is based on our earlier study (Muhammad and Thapa (2020). Each cloudy pixel was compared to its corresponding pixel in the following image. In case the pixel was snow or no snow, the cloudy pixel was replaced; otherwise, the previous image was evaluated according to same criteria. The process was repeated for up to two preceding and following images if the cloud persisted. If cloud cover appears continuously in all four images, we proceed from the temporal filter to the spatial filter to remove the remaining cloudy pixels.

To begin, let's clean the working environment as usual.

rm(list=ls())

We are now going to import the necessary packages. Please make sure that you have already installed these libraries and their dependencies.

library(raster)

library(rstudioapi)

The working directory can be set manually or automatically.

#setwd(dirname(getActiveDocumentContext()$path)) ;getwd()  
setwd('C:/amrit\_2022/pakistan\_snow\_training')

Set a variable by specifying the path of the files where the 8-day snow extent product was downloaded earlier.

data\_path='./downloaded\_data/Snow\_Cov\_8-Day\_500m\_v6/MAX\_SNW'

To implement temporal filter, we make a list of all tif files of aqua and terra separately and loop through each image of each sensor at once.

MOD\_list=list.files(data\_path,pattern='MOD10\_A2\_',full.names=T);MOD\_list

MYD\_list=list.files(data\_path,pattern='MYD10\_A2\_',full.names=T);MYD\_list

Let's compose a raster from the first tif file in the terra list and visualize it to understand how the data looks.

r1\_MOD10A2=raster(MOD\_list[1]);plot(r1\_MOD10A2,main=names(r1\_MOD10A2))

Let's analyze the classes present in the first Terra product.

freq(r1\_MOD10A2)

In a similar fashion, we create a raster from the first tif of the aqua product and calculate the frequency of classes.

r1\_MYD10A2=raster(MYD\_list[1]);plot(r1\_MYD10A2,main=names(r1\_MYD10A2))

freq(r1\_MYD10A2)

Check the length of the aqua and terra images downloaded in the previous step.

length(MOD\_list)

length(MYD\_list)

Let's create two new directories for output from the temporal filter, one for aqua and one for terra, inside the working/root directory.

dir.create("temporal\_filter\_MOD10A2")

dir.create("temporal\_filter\_MYD10A2")

We now define two variables to make a list of files to import and to create vector of names for the temporal filter.

for\_file\_name\_MOD10A2=list.files(data\_path,pattern='MOD10\_A2\_',full.names=F)  
for\_file\_name\_MYD10A2=list.files(data\_path,pattern='MYD10\_A2\_',full.names=F)  
output\_dir\_temporal=c("./temporal\_filter\_MOD10A2/","./temporal\_filter\_MYD10A2/")

We now loop through each image and reclassify it as snow, no snow, or cloud, then apply a temporal filter to reduce the cloud fraction in the image.

for (j in 1:length(output\_dir\_temporal)){   
 if(j==1){output\_dir\_temporal\_j=output\_dir\_temporal[j];new\_filename=for\_file\_name\_MOD10A2;r\_loop=MOD\_list }else{output\_dir\_temporal\_j=output\_dir\_temporal[2];new\_filename=for\_file\_name\_MYD10A2;r\_loop=MYD\_list}  
 print(output\_dir\_temporal\_j)  
 print(j)

Overall, we are using five images. The acting image is t2. First, reclassify the image according to the earlier instructions (snow, no snow, and clouds).

for (i in 3:(length(r\_loop)-2)){

# create raster from first five raster  
 t0=raster(r\_loop[i-2])  
 t1=raster(r\_loop[i-1])  
 t2=raster(r\_loop[i])  
 t3=raster(r\_loop[i+1])  
 t4=raster(r\_loop[i+2])  
   
 t0[t0==0 | t0==1 | t0==11 | t0==254 | t0==255]<-50 #cloud  
 t0[t0==100]<-200 #snow   
 t0[t0==37 | t0==39]<-25 #no snow / land  
   
 t1[t1==0 | t1==1 | t1==11 | t1==254 | t1==255]<-50 #cloud   
 t1[t1==100]<-200 #snow   
 t1[t1==37 | t1==39]<-25 #no snow / land  
   
 t2[t2==0 | t2==1 | t2==11 | t2==254 | t2==255]<-50 #cloud   
 t2[t2==100]<-200 #snow   
 t2[t2==37 | t2==39]<-25 #no snow / land  
   
 t3[t3==0 | t3==1 | t3==11 | t3==254 | t3==255]<-50 #cloud   
 t3[t3==100]<-200 #snow   
 t3[t3==37 | t3==39]<-25 #no snow / land  
   
 t4[t4==0 | t4==1 | t4==11 | t4==254 | t4==255]<-50 #cloud   
 t4[t4==100]<-200 #snow   
 t4[t4==37 | t4==39]<-25 #no snow / land

Cloudy pixels are identified in t2 and replaced with values from non-cloudy corresponding pixels in the preceding and following images.

We will first identify the cloudy pixels in t2, and the snowy pixels in t1 and t3, then we will replace the cloudy pixels in t2 by snowy pixels from t1 and t3. The assumption is that snow is unlikely to melt when it is cloudy.

idx\_SCS<- values(t2)==50 & values(t3)==200 & values(t1) == 200  
 values(t2)[idx\_SCS]<-200

This is the same as above, except the cloudy pixel will be replaced by the no snow pixel since the following pixels have no snow. The assumption is that snow does not melt within a single time step.

idx\_LCL<- values(t2)==50 & values(t3)==25 & values(t1) == 25  
 values(t2)[idx\_LCL]<-25

When preceding image has snow/cloud/no snow in the corresponding cloudy pixels of the acting image (t2), but the pixels in the following image is no snow, the following image gets priority.

idx\_SCL<- values(t2)==50 & values(t3)==25 & values(t1) == 200  
 values(t2)[idx\_SCL]<-25

idx\_LCS<- values(t2)==50 & values(t3)==200 & values(t1) == 25  
 values(t2)[idx\_LCS]<-200

idx\_CCL<- values(t2)==50 & values(t3)==25 & values(t1) == 50  
 values(t2)[idx\_CCL]<-25

When corresponding pixels in the activating and preceding images are cloudy, but the following image is snowy, we replace cloudy pixels in the acting image with snowy pixels in t3.

idx\_CCS<- values(t2)==50 & values(t3)==200 & values(t1) == 50  
 values(t2)[idx\_CCS]<-200

Similarly, when both the acting and following images have clouds, but the preceding image is snowless, we replace the cloudy pixels with the no snow pixels from the t1 image.

idx\_LCC<- values(t2)==50 & values(t3)==50 & values(t1) == 25  
 values(t2)[idx\_LCC]<-25

Cloudy pixel in t2, but also in t3 but snow in t1 results in snow.  
   
 idx\_SCC<- values(t2)==50 & values(t3)==50 & values(t1) == 200  
 values(t2)[idx\_SCC]<-200

We noticed that the use of two more images one from 2 steps backward (t0) and one from two following steps could reduce cloudy quite a bit. Let’s use 2nd following image from acting image to fill cloudy image.

idx\_CCC<- values(t2)==50 & values(t3)==50 & values(t1) == 50

values(t2)[idx\_CCC]<-values(t4)[idx\_CCC]

If the relevant pixels in all four images (acting, preceding, two consecutive following have cloud) we look back two step back in time (t0) to see if the corresponding pixel is not cloudy. If the corresponding pixels in all five images are cloudy, they are left as is and are eliminated by other filters.

idx\_CCCC<-values(t2)==50  
 values(t2)[idx\_CCCC]<-values(t0)[idx\_CCCC]  
   
Finally, we save the temporal output to the folder we established previously. The loop is repeated until the final image is processed.

writeRaster(t2,filename=paste0(output\_dir\_temporal\_j,"t\_5img\_",new\_filename[i]),format="GTiff",overwrite=T,datatype='INT1U')  
print(i)  
 }   
}

**Spatial Filter**

The output from temporal filter is used as input for the spatial filter, which is based on the majority algorithm. Based on the majority votes of the non-cloudy surrounding pixels, the cloudy pixel is reclassified as snow or no snow. A 3x3 moving window is implemented here based on available literature. If there is a tie between pixels of no snow and pixels of snow in the surroundings, the cloudy pixel is replaced by snow. This filter is implemented three times because running it only once may not guarantee the removal of all cloudy pixels. A pixel remains cloudy only if all the eight neighbouring pixels are also cloudy.

Let’s set the seed to create simulations or random objects that can be reproduced.

set.seed(123)

Now we import required library. Raster for raster data processing and rstudioapi for setting working directory automatically.

library(raster)

library(rstudioapi)

You can now set the working either manually or automatically with rstudioapi.

*#setwd(dirname(getActiveDocumentContext()$path)) ;getwd()*  
setwd('C:/amrit\_2022/pakistan\_snow\_training')

Create two separate folders for storing the output from the spatial filter. One for aqua and one for terra.

dir.create("spatial\_filter\_1\_MOD10A2")

dir.create("spatial\_filter\_1\_MYD10A2")

To make a loop and to store output files, let's make a vector of output directories.

output\_dir\_spatial\_1=c("./spatial\_filter\_1\_MOD10A2/","./spatial\_filter\_1\_MYD10A2/")

To import data, define a variable of output folders from the temporal filter.

data\_path=c('./temporal\_filter\_MOD10A2','./temporal\_filter\_MYD10A2/')

Prepare a list of tif files that will be imported. The spatial filter is applied to the output of the temporal filter.

for\_file\_name\_MOD10A2=list.files(data\_path[1],pattern='t\_5img\_MOD10\_A2\_',full.names=F)  
for\_file\_name\_MYD10A2=list.files(data\_path[2],pattern='t\_5img\_MYD10\_A2\_',full.names=F)

New variables with no full path are defined for exporting output rasters of spatial filters.

MOD\_list=list.files(data\_path[1],pattern='t\_5img\_MOD10\_A2\_',full.names=T)  
MYD\_list=list.files(data\_path[2],pattern='t\_5img\_MYD10\_A2\_',full.names=T)

We now loop through both products (aqua and terra) and each 8-day image and apply spatial filters. To do this, we use the focal function. The arguments to this function are w (matrix of weight, we use 3\*3), fun (statistics used, we use modal because majority pixels are being counted), NA (logical, TRUE in this case), pad (logical, TRUE in this case). When pad is TRUE, additional 'virtual' rows and columns are padded to x such that there are no edge effects. If a filter function needs to access the central cell, then this can be useful.

**for** (j **in** 1:length(output\_dir\_spatial\_1)){   
 **if**(j==1){output\_dir\_spatial\_1\_j=output\_dir\_spatial\_1[j];new\_filename=for\_file\_name\_MOD10A2;r\_loop=MOD\_list  
 }**else**{output\_dir\_spatial\_1\_j=output\_dir\_spatial\_1[2];new\_filename=for\_file\_name\_MYD10A2;r\_loop=MYD\_list}  
 print(output\_dir\_spatial\_1\_j)  
 print(j)  
   
 **for** (i **in** 1:length(r\_loop))  
 {  
 snow\_for\_spatial=raster(r\_loop[i])

We reclassify cloudy pixels as NA and use majority vote to fill.

snow\_for\_spatial[snow\_for\_spatial == 50]<-NA  
 s\_filter <- focal(snow\_for\_spatial, w=matrix(1,3,3), fun=modal, na.rm=TRUE, NAonly=TRUE, pad=TRUE)

Return the NA values to the cloud class.

s\_filter[is.na(s\_filter)]<-50

We export output raster using following command. The loop is repeated until the final image is processed.

writeRaster(s\_filter,filename=paste0(output\_dir\_spatial\_1\_j,"s\_",new\_filename[i]),format="GTiff",overwrite=T,datatype='INT1U')  
 print(i)  
 }  
}

Likewise, we apply the spatial filter a second time. Here, input is the result of the first spatial filter.

rm(list=ls())  
set.seed(123)   
  
library(raster)  
library(rstudioapi)  
*#setwd(dirname(getActiveDocumentContext()$path)) ;getwd()*  
setwd('C:/amrit\_2022/pakistan\_snow\_training')  
dir.create("spatial\_filter\_2\_MOD10A2")

dir.create("spatial\_filter\_2\_MYD10A2")

*#location for output*  
output\_dir\_spatial\_2=c("./spatial\_filter\_2\_MOD10A2/","./spatial\_filter\_2\_MYD10A2/")  
data\_path=c('./spatial\_filter\_1\_MOD10A2','./spatial\_filter\_1\_MYD10A2/')  
for\_file\_name\_MOD10A2=list.files(data\_path[1],pattern='s\_t\_5img\_MOD10\_A2\_',full.names=F)  
for\_file\_name\_MYD10A2=list.files(data\_path[2],pattern='s\_t\_5img\_MYD10\_A2\_',full.names=F)  
MOD\_list=list.files(data\_path[1],pattern='s\_t\_5img\_MOD10\_A2\_',full.names=T)  
MYD\_list=list.files(data\_path[2],pattern='s\_t\_5img\_MYD10\_A2\_',full.names=T)  
  
**for** (j **in** 1:length(output\_dir\_spatial\_2)){ **if**(j==1){output\_dir\_spatial\_2\_j=output\_dir\_spatial\_2[j];new\_filename=for\_file\_name\_MOD10A2;r\_loop=MOD\_list  
 }**else**{output\_dir\_spatial\_2\_j=output\_dir\_spatial\_2[2];new\_filename=for\_file\_name\_MYD10A2;r\_loop=MYD\_list}  
 print(output\_dir\_spatial\_2\_j)  
 print(j)  
   
 **for** (i **in** 1:length(r\_loop))  
 {  
 snow\_for\_spatial=raster(r\_loop[i])  
 snow\_for\_spatial[snow\_for\_spatial == 50]<-NA  
 s\_filter <- focal(snow\_for\_spatial, w=matrix(1,3,3), fun=modal, na.rm=TRUE, NAonly=TRUE, pad=TRUE)  
   
 s\_filter[is.na(s\_filter)]<-50  
 writeRaster(s\_filter,filename=paste0(output\_dir\_spatial\_2\_j,"s\_",new\_filename[i]),format="GTiff",overwrite=T,datatype='INT1U')  
 print(i)  
 }  
}

In order to make this filter more effective, we will use a spatial filter one more time.

rm(list=ls())  
set.seed(123)   
  
library(raster)  
library(rstudioapi)  
*#setwd(dirname(getActiveDocumentContext()$path)) ;getwd()*  
setwd('C:/amrit\_2022/pakistan\_snow\_training')  
  
dir.create("spatial\_filter\_3\_MOD10A2")

dir.create("spatial\_filter\_3\_MYD10A2")

*#location for output*  
output\_dir\_spatial\_3=c("./spatial\_filter\_3\_MOD10A2/","./spatial\_filter\_3\_MYD10A2/")  
data\_path=c('./spatial\_filter\_2\_MOD10A2','./spatial\_filter\_2\_MYD10A2/')  
for\_file\_name\_MOD10A2=list.files(data\_path[1],pattern='s\_s\_t\_5img\_MOD10\_A2\_',full.names=F)  
for\_file\_name\_MYD10A2=list.files(data\_path[2],pattern='s\_s\_t\_5img\_MYD10\_A2\_',full.names=F)  
MOD\_list=list.files(data\_path[1],pattern='s\_s\_t\_5img\_MOD10\_A2\_',full.names=T)  
MYD\_list=list.files(data\_path[2],pattern='s\_s\_t\_5img\_MYD10\_A2\_',full.names=T)  
  
**for** (j **in** 1:length(output\_dir\_spatial\_3)){   
 **if**(j==1){output\_dir\_spatial\_3\_j=output\_dir\_spatial\_3[j];new\_filename=for\_file\_name\_MOD10A2;r\_loop=MOD\_list  
 }**else**{output\_dir\_spatial\_3\_j=output\_dir\_spatial\_3[2];new\_filename=for\_file\_name\_MYD10A2;r\_loop=MYD\_list}  
 print(output\_dir\_spatial\_3\_j)  
 print(j)  
   
 **for** (i **in** 1:length(r\_loop))  
 {  
 snow\_for\_spatial=raster(r\_loop[i])  
 snow\_for\_spatial[snow\_for\_spatial == 50]<-NA  
 s\_filter <- focal(snow\_for\_spatial, w=matrix(1,3,3), fun=modal, na.rm=TRUE, NAonly=TRUE, pad=TRUE)  
   
 s\_filter[is.na(s\_filter)]<-50  
 writeRaster(s\_filter,filename=paste0(output\_dir\_spatial\_3\_j,"s\_",new\_filename[i]),format="GTiff",overwrite=T,datatype='INT1U')  
 print(i)  
 }  
}

Now that the product improvement using spatial filters has been completed, we can go on to combining the improved aqua product and the terra product to remove remaining cloud and to reduce some of the overestimation.

**Combine improved Aqua and Terra**

Finally, Terra and Aqua are merged in such a way that the snow appears in final product only if snow is found in both products. This filter also reduces unlikely snow found in lower altitudes. Through this process, we will be able to improve the snow product, mainly removing the heavy overestimation in images captured from an off-nadir position and edge-pixels reproduction due to MODIS's wide field of view.

We start by cleaning the working environment, loading the required libraries and setting the working environment.

rm(list=ls())  
library(raster)

library(rstudioapi)

#setwd(dirname(getActiveDocumentContext()$path)) ;getwd()  
setwd('C:/amrit\_2022/pakistan\_snow\_training')

Specify a directory in which to store output files.

dir.create("final\_combined\_product\_after\_5t\_3s")

#location for output  
output\_dir\_combined=c("./final\_combined\_product\_after\_5t\_3s/")

To import data, create a vector of the final output directory from the spatial filter.

data\_path=c('./spatial\_filter\_3\_MOD10A2','./spatial\_filter\_3\_MYD10A2/')

List tif files from the final output folder of the spatial filter. These are the input for this filter.

input\_terra\_after\_3s=list.files(data\_path[1],pattern='s\_s\_s\_t\_5img\_MOD10\_A2\_',full.names=T)  
input\_aqua\_after\_3s=list.files(data\_path[2],pattern='s\_s\_s\_t\_5img\_MYD10\_A2\_',full.names=T)

To define the output filenames and loop through each image, let's extract doy from aqua and terra. Please note that this code does not match the date in aqua and terra. To use this filter, users must ensure that the date of aqua and terra are identical (exactly the same number of aqua and terra images with the same date).

library(stringr)

doy\_terra=as.numeric(paste0(str\_sub(input\_terra\_after\_3s,-12,-9),str\_sub(input\_terra\_after\_3s,-7,-5)));head(doy\_terra)

doy\_aqua=as.numeric(paste0(str\_sub(input\_aqua\_after\_3s,-12,-9),str\_sub(input\_aqua\_after\_3s,-7,-5)));head(doy\_aqua)

Generate a vector of new filenames for exporting output files.

new\_filename=paste0("AQUA\_TERRA\_8day\_combined\_5t\_3s\_",doy\_aqua,".tif");head(new\_filename)

Each image of aqua and terra is looped through and the product is combined. As explained, snow is considered if it appears in both sensors.

for(i in 1:length(doy\_aqua))  
{   
 aqua=raster(input\_terra\_after\_3s[i])  
 terra=raster(input\_aqua\_after\_3s[i])

The final output will be stored in a raster. Let's create a raster to store the final output.

combined\_aqua\_terra=terra;combined\_aqua\_terra[]=NA

Create an index for snow in both aqua and terra, and extract snow to an empty output file. A 200 value is associated with snow.

idx\_S\_S<- values(aqua)==200 & values(terra)==200 #;table(idx\_S\_S)  
 values(combined\_aqua\_terra)[idx\_S\_S]<-200

Create an index for snow in aqua and no snow in terra. This value will be represented as -200 on the output product, meaning the snow was removed by the filter.

# snow, no snow/cloud combination to -200  
 idx\_S\_aqua\_C\_NS\_terra<- values(aqua)==200 & ( values(terra)==50 |values(terra)==25 ) #;table(idx\_S\_aqua\_C\_NS\_terra)  
 values(combined\_aqua\_terra)[idx\_S\_aqua\_C\_NS\_terra]<-(-200)

We now create an index for snow in Terra and no snow in Aqua and store the value as -200 in the output raster.

# snow, no snow/cloud combination to -200  
 idx\_S\_terra\_C\_NS\_aqua<- values(terra)==200 & ( values(aqua)==50 |values(aqua)==25 ) #;table(idx\_S\_terra\_C\_NS\_aqua)  
 values(combined\_aqua\_terra)[idx\_S\_terra\_C\_NS\_aqua]<-(-200)

Finally, we identify pixels with clouds both in aqua and terra (if any) and store them as the same 50 value in the final output.

# cloud cloud combination result to cloud  
 idx\_CC<- values(aqua)==50 & values(terra)==50 #;table(idx\_CC)  
 values(combined\_aqua\_terra)[idx\_CC]<-50

The remaining class combinations are stored as no snow by 25.

#rest is no snow  
 combined\_aqua\_terra[is.na(combined\_aqua\_terra)]=25

Now the final product is exported using following command. The loop is repeated until the final image is processed.  
 writeRaster(combined\_aqua\_terra,filename=paste0(output\_dir\_combined,new\_filename[i]),format="GTiff",overwrite=T,datatype='INT2S')  
 print(i)  
}

The development of the improved 8-day snow cover product has been completed. The use of this product can benefit climate change studies as well as glacio-hydrological studies. We developed improved snow products for our area of interest during this hands-on exercise. This product incorporates filters devised by Muhammad and Thapa (2020).

**Calculating snow cover area and making visualization**

We'll load the improved snow cover data from the previous exercise, plot it, and calculate snow cover statistics in this exercise.

As usual, let’s clean the working environment, load required libraries and set working environment.

rm(list=ls())  
library(raster)

library(rstudioapi)

#setwd(dirname(getActiveDocumentContext()$path)) ;getwd()  
setwd('C:/amrit\_2022/pakistan\_snow\_training')

Define a new variable with location path where we have stored final improved data from previous exercise.

data\_path="./final\_combined\_product\_after\_5t\_3s/"

Make list of all tif files from the data directory.

tif\_list=list.files(data\_path,pattern='.tif$',full.names=T);tif\_list

MODIS product are named with date in yydoy format. Let’s convert this date format to plain (yy-mm-dd) format.

library(stringr)

modis\_doy=as.numeric(str\_sub(tif\_list,-7,-5))  
modis\_year=as.numeric(str\_sub(tif\_list,-11,-8))  
calender\_date=(as.Date(modis\_doy, origin = paste0(modis\_year,"-01-01")))-1

Now, we tend to produce the stack of all tif files and create default plot. This plot does not look attractive but give sense of how data looks. We will use ggplot2 package to create attractive plots at the end of this exercise

rts\_snow=stack(tif\_list)  
plot(rts\_snow)

We want to derive the fraction of snow cover area for each image in the stack. This can be done by using freq command in R. Additionally, let’s count other classes in the stack as well.

snow\_stat=freq(rts\_snow,value=200,merge=T)  
no\_snow\_stat=freq(rts\_snow,value=25,merge=T)  
cloud\_stat=freq(rts\_snow,value=50,merge=T)  
snow\_to\_nosnow\_cloud\_stat=freq(rts\_snow,value=-200,merge=T)

As we are interested in deriving snow cover as a fraction of total basin area, let’s count all pixels in the image.

total\_pixel\_count=sum(!is.na(as.matrix(rts\_snow[[1]])))

Finally, we make a new dataframe and store date and snow cover percentage.

df\_export=data.frame(Date=calender\_date,Snow\_percent=snow\_stat/total\_pixel\_count\*100,  
 Nosnow\_percent=no\_snow\_stat/total\_pixel\_count\*100,  
 Cloud\_percent=cloud\_stat/total\_pixel\_count\*100,  
 Snow2NosnowCloud\_percent=snow\_to\_nosnow\_cloud\_stat/total\_pixel\_count\*100)

Get detail of dataframe we just created.

print(df\_export)

Let’s export the final dataframe as csv file.

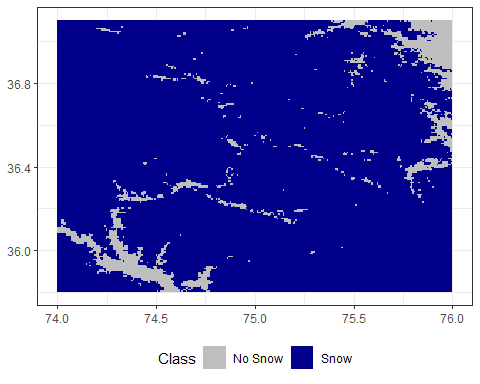
write.csv(df\_export,'snow\_statistics.csv',row.names=F)

Let’s make a publishable plot before ending this exercise. We will import ggplot2 library to make the plot. Make sure that you have installed ggplot2 library.

# load required packages  
library(ggplot2)

# get first image from snow stack  
r\_i=rts\_snow[[1]]  
  
# reclassify raster to snow and no snow  
r\_i[r\_i==200]=1  
r\_i[r\_i!=1]=0  
  
# quick check to see class in r\_i, (this should have 0 and 1)  
freq(r\_i)

# make a nice plot  
plot\_combined\_snow=gplot(r\_i) +   
 geom\_tile(aes(fill = as.factor(value))) +  
 theme\_bw()+  
 ylab(NULL)+xlab(NULL)+  
 theme(legend.position = 'bottom')+  
 scale\_fill\_manual(name='Class',breaks = c(0,1),  
 values = c("grey", "dark blue"),labels=c('No Snow','Snow'))  
  
# visualize plot  
print(plot\_combined\_snow)



# Export image in root directory  
jpeg(file="sample\_snow\_plot.jpg",width=10, height=10, units="cm", res=500)  
print(plot\_combined\_snow)  
dev.off()

**Disclaimer**

Users are free to distribute and change this code, but they do so at their own risk. The views and interpretations in this publication are those of the authors and are not necessarily attributable to the ICIMOD.

**Major References:**

Busetto, L., & Ranghetti, L. (2016). MODIStsp: An R package for automatic preprocessing of MODIS Land Products time series. *Computers & geosciences*, *97*, 40-48.

Hijmans, R. J., Van Etten, J., Cheng, J., Mattiuzzi, M., Sumner, M., Greenberg, J. A., ... & Hijmans, M. R. J. (2015). Package ‘raster’. *R package*, *734*.

Lamigueiro, O. P., Hijmans, R., & Lamigueiro, M. O. P. (2022). Package ‘rasterVis’.

Muhammad, S., & Thapa, A. (2020). An improved Terra–Aqua MODIS snow cover and Randolph Glacier Inventory 6.0 combined product (MOYDGL06\*) for high-mountain Asia between 2002 and 2018. *Earth System Science Data*, *12*(1), 345-356.

R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL: <https://www.R-project.org/>.

Riggs, G. A., Hall, D. K., & Román, M. O. (2015). MODIS snow products collection 6 user guide. *National Snow and Ice Data Center: Boulder, CO, USA*, 66.

RStudio Team (2019). RStudio: Integrated Development for R. RStudio, Inc., Boston, MA URL: <http://www.rstudio.com/>

Wickham, H. (2011). ggplot2. *Wiley interdisciplinary reviews: computational statistics*, *3*(2), 180-185.