**GLACIER TERMINUS ESTIMATION- PIPELINE**

**Introduction**

The Glacier Terminus Estimation project computes the terminus location of glaciers over the years using Landsat images and examines if the glacier is retreating or advancing through time. This document describes the process pipeline to estimate the glacier terminus. The project mainly is mainly coded using Python and R. The programming aspects of the pipeline is performed using Python language, and the Statistical computations and plotting is performed using R. The project also uses the MySQL database to store glacier data related to the project.

**Problem definition and Goal**

The aim of the project is to find the terminus location of a glacier using the DEM and the Landsat images. The glacier flow line is identified for a glacier using the DEM. The intensities at each point in the flow line are extracted and an intensity matrix is created which is a matrix of year against distance where each row provides the intensity values of different points in the flow line of a particular year. This intensity profile matrix is used to identify the terminus and the results are plotted for visualization.

**Solution**

The solution to the above problem definition is achieved using multiple steps. The project pipeline starts with accepting the input settings for the computation. It is then followed by a series of actions to compute and estimate the terminus for a glacier given as input.

Given a glacier name the first step is to query the glims database for the glacier bounds and the initial flowline location. Then using the bounds, the Digital Elevation Model(DEM) and the Landsat images in bands – B2, B3, B4, B5, B6 are downloaded from the Google Earth Engine. Once the landsat images are downloaded the ndsi band images for the glacier is created using the B2 and B5 bands. The formula to compute ndsi is given as

NDSI = (B2 – B5) / (B2 + B5)

Along with the ndsi images, RGB images are created using B3, B4, B5. The method to create the rgb image is by choosing B5 as the red band, B4 as the green band and B3 as the blue band.

The next step in the pipeline is to classify the landsat images based on the clarity and the cloud cover. Only the images classified as good are retained. Then the pipeline proceeds to compute the flowline for the glacier using gradient descent algorithm using the DEM. If the flowline computed does not reach the boundary of the image, then A-star algorithm is used to complete the flowline till it reaches one of the image boundaries. This flowline is then smoothed and additional set of lines parallel to the flowline is computed as per user input.

Using the flowline from the DEM, intensity profiles are extracted along the flowline from the Landsat images to create an intensity profile matrix. The intensity profile matrix is spatially smoothed and the first derivative of the intensity profile matrix is used find the terminus path. Then these results are plotted. Finally the terminus locations for each image is calculated from the above results and plotted on landsat image along with flowline. The series of landsat images with the terminus location is compiled as a GIF image to view the transition in glacier terminus location.

**Database Description and Usage**

**DB usernmae – root. DB password – ncsu.**

Steps to open MySQL database:

- Open terminal and type the following command:

mysql -u root -p

- When prompted for password use the password mentioned above.

- type the following to load the glacier database:

use glaciers;

- To view the list of tables:

show tables;

- To view the structure of a table:

desc <table name>

example – desc glaciers;

- To query values from the table:

select <column names> from <table name>;

example – select \* from glaciers;

\* denotes all the columns from the table.

To give condtions:

Select glacier\_name from glaciers where area > 0.005;

The project pipeline takes the glacier name as input and estimates the terminus variation. The glacier information is maintained in MySQL database. The Glims database is used as the source. The Database consists of 4 tables.

1. Glaciers
2. Wgms Glaciers
3. Front Variation
4. CommonDB

Glims database consists data for over 199,000 glaciers. Of these 199,000 glaciers, the records provide information for six line types. Of these 6, the “glac\_bound” line type is the one category that this project focuses on. Therefore the final set of records reduces to around 122,000. The table contains the following columns.

glacier\_id , glacier\_name, wgms\_id, line\_type, left\_bottom\_x, left\_bottom\_y, right\_top\_x, right\_top\_y, location\_x, location\_y, location\_z, area .

Glaciers table consists of the glacier bounding box, the initial location of the glacier path and the area calculated using the bounding box. The wgms glacier table consists of the frontal characteristics, latitude and longitude of the glaciers. This is a subset of the glims database and consists of 2790 records. The columns present in wgms\_glaciers are:

glacier\_name, political\_unit, latitude, longitude, primary\_classification, form, frontal\_characteristics.

The front variation table consists of the ground record collection for the glaciers. The glacier names in this table is a subset of the wgms glaciers.

glacier\_name, front\_variation, front\_variation\_accuracy, year.

In order to link the glims records and the wgms records, we have a table called the CommonDB to cross reference values from one table to the other. The commonDB consists of the following attributes.

Glims\_name, glims\_id, wgms\_name and area.

The first part of the pipeline is to extract the information for the given glacier from the database. MySQLdb python package is used to interface the pipeline and the database.

**Python Packages**

The following python packages were used in the development of the pipeline.

os – Package to perform file and folder operations

mySQLdb – Package to integrate mySQL database with python

xlrd – Python Excel integration package

ee – Google earth engine API to communicate to the earth engine for downloading images.

zipfile – package to perform file zipping/unzipping operations

numpy – Numerical python package

scipy – Scientific python package

matplotlib – package for matlab plotting functions

PIL – Image package to perform image processing

urllib2 – package to download html content

gdal – Package to read geo images

rpy2 – Package to inegrate R and python

math – mathematical operations package

shutil – Shell utilities package

libtiff – Package to work with tiff images

images2gif – Package to create GIF using a series of images.

The pipeline consists of the following files:

main.py – The main pipeline code

querydb.py – Python code with all the functions interacting with MySQL database.

ee\_download.py – Python code with functions to download DEM and Landsat images from earth engine.

rgbplot.py – Python code to create the rgb and ndsi images.

method1.py – Python code to compute the flowline, smooth flowline, compute parallel lines and plot flowline.

connectedComponentAStar.py – Python code to complete the gradient descent flowline using A star algorithm.

method2.py – Python code to classify the landsat images.

method3.py – Python code to compute the intensity profiles from the landsat images.

method4.py – Python code to estimate the terminus and generate plots.

terminus\_est.py – Python code to compute the terminus path from the intensity profile derivatives.

TI.py – Python code to plot the terminus on the flowline on the landsat images.

gd\_linear.R – R file to compute the flowline using gradient descent algorithm.

IntensityProfile.R – R file to extract the intensity profiles from Landsat images.

terminus.R – R file to estimate terminus and plot the results.

plot\_terminus.R – R file to plot flowline with terminus on landsat image.

**Funtions**

This documentation discusses the functions from each of the python codes seperately.

1. querydb.py :

findBoundingBoxByName -

Input – name: The name of the glacier to find bounding box.

Operation – The function queries the DB using the name to fetch the bounding box of the given glacier.

Output – Returns a tuple of 4 values (right top x, right top y, left bottom x, left bottom y).

findBoundingBoxById -

Input – gid: The Glims id of the glacier to find bounding box.

Operation – The function queries the DB using the id to fetch the bounding box of the given glacier.

Output – Returns a tuple of 4 values (right top x, right top y, left bottom x, left bottom y).

findStartByName -

Input – name: The name of the glacier to find the initial coordinates.

Operation – The function queries the DB using the name to fetch the Initial location of the given glacier.

Output – Returns a tuple of 2 values (location x, location y).

findStartById -

Input – gid: The name of the glacier to find the initial coordinates.

Operation – The function queries the DB using the id to fetch the Initial location of the given glacier.

Output – Returns a tuple of 2 values (location x, location y).

findGroundMeasurement -

Input – glacier: The name of the glacier to find ground measurements.

Operation – The function fetches the ground measurements if present from the wgms DB for the given glacier.

Output – Returns a list of tuples with two values ( year, front\_variation).

[(year1, fv1), (year2, fv2), … , (year n, fv n)]

2. ee\_download.py:

This file requires the Earth engine package. All the functions in this file use the following two files – GoogleAccount.txt, GoogleKey.p12, to interact with the earth engine.

ee\_download\_DEM -

Input – path: Path to the home folder of the pipeline.

Glacier: Name of the glacier.

MaxLon: Upper bound of the longitude for the glacier geo image.

MinLon: Lower bound of the longitude for the glacier geo image.

MaxLat: Upper bound of the latitude for the glacier geo image.

MinLat: Upper bound of the latitude for the glacier geo image.

Operation – The function uses the bounds to query the earth engine server and downloads the DEM for the glacier. The 'srtm90\_v4' elevation model is used. If this image turns out to be empty then the 'USGS/GMTED2010' elevation model is used.

Output – It downloads the image to the Data folder in the pipeline directory. It also returns the name of the DEM downloaded (srtm90\_v4.elevation.tif or GMTED2010.be75.tif) .

ee\_download\_Allbands -

Input – path: Path to the home folder of the pipeline.

Glacier: Name of the glacier.

MaxLon: Upper bound of the longitude for the glacier geo image.

MinLon: Lower bound of the longitude for the glacier geo image.

MaxLat: Upper bound of the latitude for the glacier geo image.

MinLat: Upper bound of the latitude for the glacier geo image.

Operation – The function uses the bounds to query the earth engine server and downloads the Landsat images for the glacier. The bands doenloaded are B2, B3,

B4, B5, B6/B6\_VCID\_1.

Output – It downloads the images to the Data folder in the pipeline directory. It also returns the bounding box.

ee\_download function is similar to the ee\_downlaod\_Allbands but downloads only the band B6\_VCID\_1.

3. rgbplot.py:

downloadAndClassifyLandsat -

Input – GlacierName: the name of the glacier.

Path: path to the home folder of the pipeline project.

BoundingBox: bounding box of the glacier to download.

Operation – The function is used used to call other functions that perform download and classification. It first calls the ee\_download\_Allbands to download the landsat images and then calls the classify function to classify the landsat images. Finally it calls functions to create the NDSI and RGB images.

Output – It downloads, classifies and creates NDSI & RGB images. Does not return any values.

rgbPlot -

Input – image: the path to the folder that contains the B3, B4, B5 images.

Operation – The function creates the rgb image of the corresponding landsat by overlapping B5 as red band, B4 as green band and B3 as blue band.

Output – It saves the rgb image in the 'image' folder with an extension '.rgb.tif'.

ndsi -

Input – image: the path to the folder that contains the B2, B5 images.

Operation – The function creates the ndsi image of the corresponding landsat by using the formula

NDSI = (B2 – B5) / (B2 + B5)

Output – It saves the ndsi image in the 'image' folder with an extension '.ndsi.tif'.

4. Method1.py :

getDimensions -

Input – filename: The complete file name of the image to obtain the dimensions.

Operation – The function opens the specified image and computes the number of rows and columns in the image.

Output – Returns a tuple of 2 values with rows and columns (ncol, nrow).

beginningPoint -

Input - MaxLon: Upper bound of the longitude for the glacier geo image.

MinLon: Lower bound of the longitude for the glacier geo image.

MaxLat: Upper bound of the latitude for the glacier geo image.

MinLat: Upper bound of the latitude for the glacier geo image.

Lon – Longitude of the initial location.

Lat – Latitude of the initial location.

ncol – Number of coolumns in the image

nrow – Number of rows in the image.

margin - Margin of the image boundary to be excluded.

Operation – The function finds the initial point of the glacier using the Lat and Lon and rescaling them to row and coolumn pixel in the image.

Output – Returns a tuple of 2 values with the initial row and column pixel. (beg\_row, beg\_col).

findPath -

Input – DEMpath: the path to the DEM image.

start\_x: the column for the initial pixel of the flowline.

start\_y: the row for the initial pixel of the flowline.

flowline: the type of method to be used in the gradient descent algorithm. It can be either regular or median.

Operation – The function interfaces with R for the first part and passes the inputs to R function gd\_linear\_sample to compute rhe flowline using GD. If the flowline returned does not reach the boundary then it calls the A-star algoritm to complete the path.

Output – Returns a list GLpath that consists of the pixel coordinates of the points on the flowline.

[x1 y1,

x2 y2,

…

xN yN]

smoothPath -

Input – glacier\_path: The list of coordinates corresponding to the glacier path.

Operation – The function uses the list of coordinates from the findPath function and smooths the path with equally spaced points by passing the values to R function path\_smooth.

Output – Returns a list PathVector consisting of the coordinates of the smoothed flowline.

parallel\_path -

Input – pathVector: The list of coordinates corresponding to the smoothed glacier path.

DEMpath: the path to the DEM image.

numParallel: The number of parallel lines to be generated on either side of the smoothed path.

Operation – The function uses the list of coordinates from the smooth\_path function and produces the set of parallel lines by passing the values to the R function path\_parallel.

Output – Returns a matrix PathVectors consisting of a list of pathVector where each pathVector is a list of coordinates corresponding to the flowline or a line parallel to the flowline.

plot\_paths -

Input – pathVectors: A list of pathVector lists consisting of the folwline and its parallel paths.

Direc: The path to the directory in whoch the image needs tot be saved.

Glacier: Name of the glacier.

Landsatpath: Path of the Landsat image on which the flowline is to be plotted.

Numparallel: Number of parallel lines on each side of the flowline. Invert: Flag to indicate the negation of the the image values.

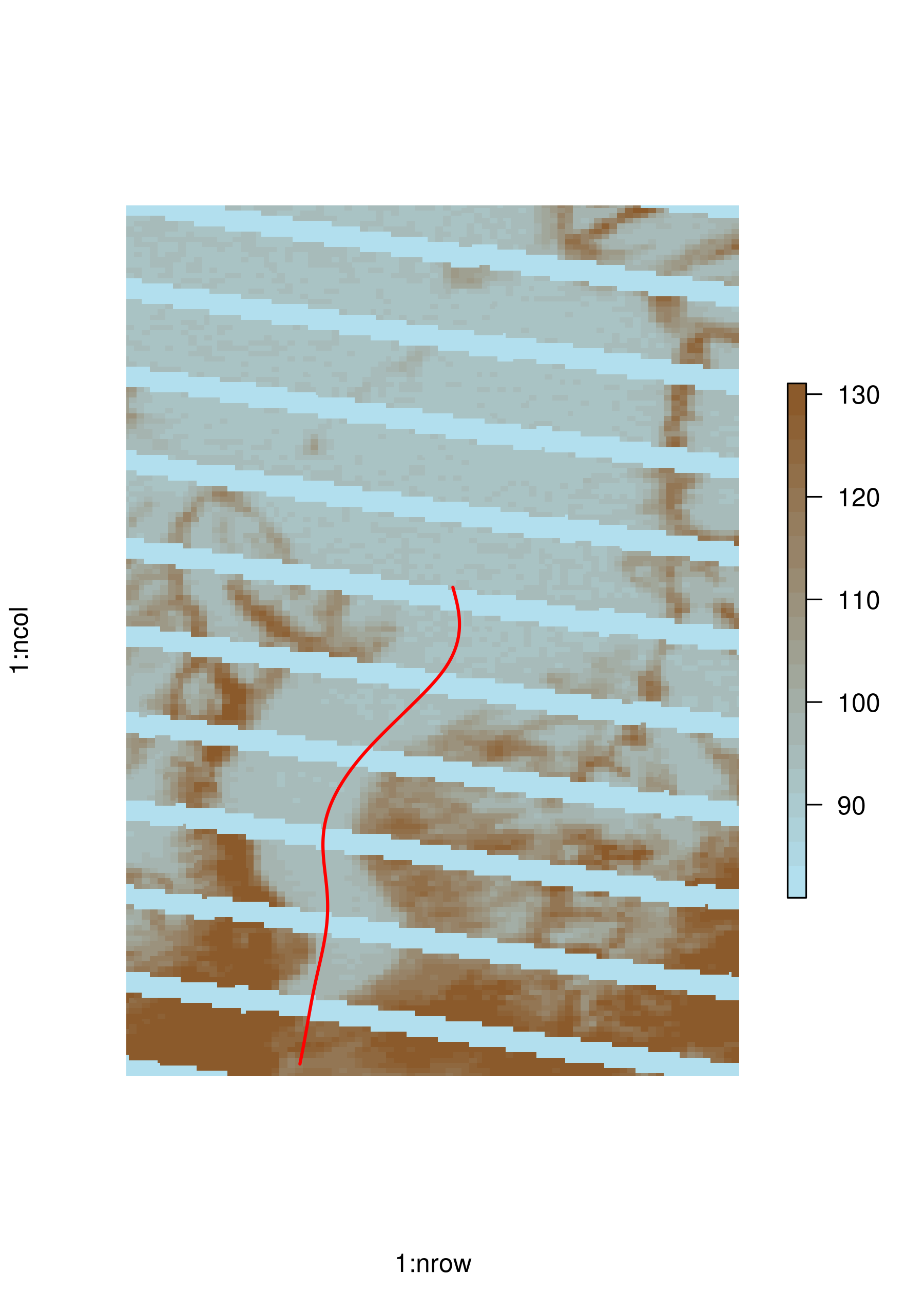
Input: Type of band given as input.

lower: lower limit for the range of values in the landsat image.

Upper: Upper limit for the range of values in the landsat image.

Operation – The function passes the values to the R function plot\_allpath that plots the flowline and the parallel lines if present on the specified Landsat image.

Output – The function saves the image with the flowline in the specified folder.



Example of the image plotted by the function for Rhonegletscher.

PlotPathsDEM -

Input – pathVectors: A list of pathVector lists consisting of the folwline and its parallel paths.

Direc: The path to the directory in whoch the image needs tot be saved.

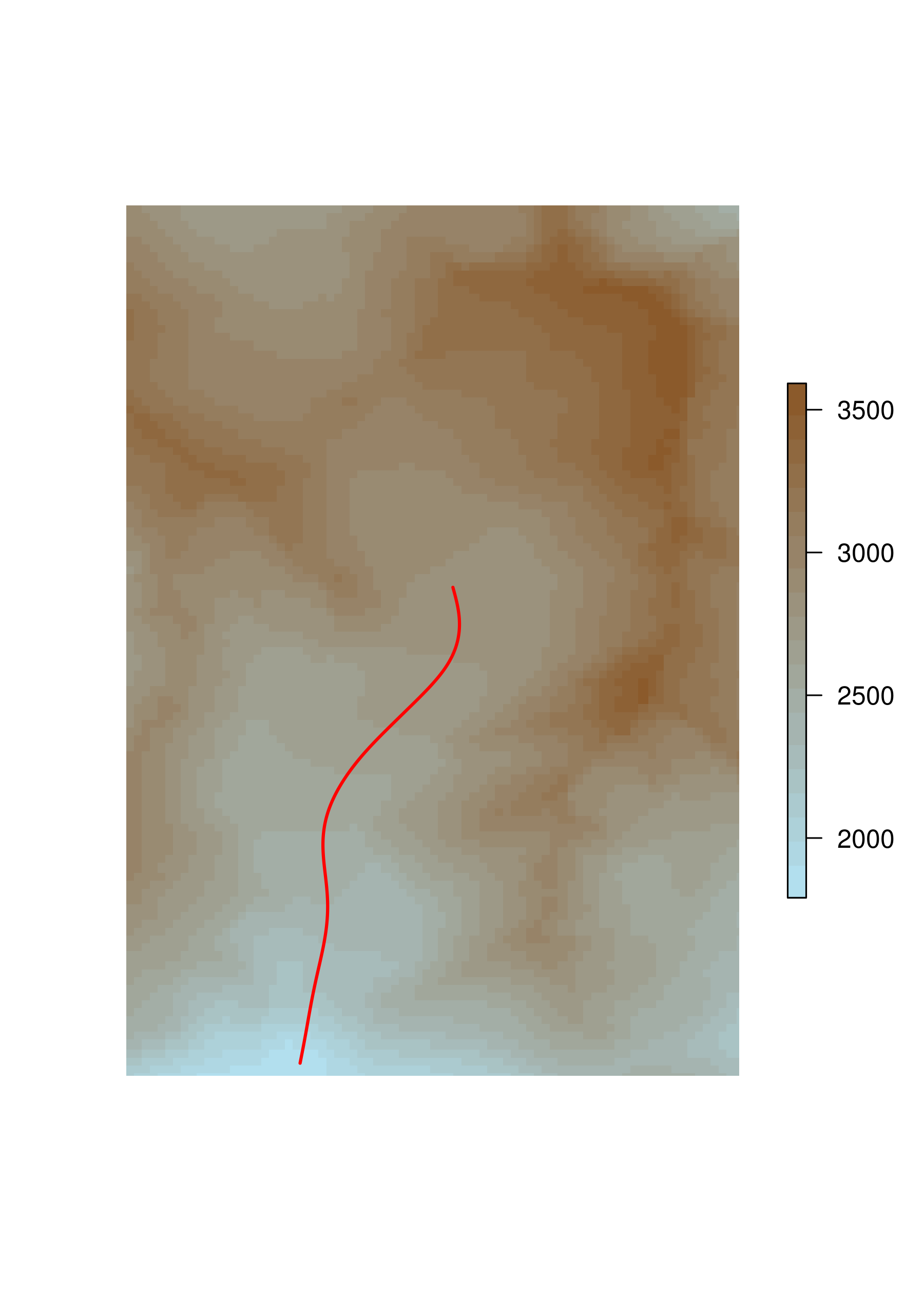
Glacier: Name of the glacier.

DEMpath: Path of the DEM image on which the flowline is to be plotted.

Numparallel: Number of parallel lines on each side of the flowline. Invert: Flag to indicate the negation of the the image values.

Operation – The function passes the values to the R function plot\_allpathDEM that plots the flowline and the parallel lines if present on the specified DEM image.

Output – The function saves the image with the flowline in the specified folder.



5. Method2.py:

classify -

Input – path: path to the folder that consists of the Landsat images to be classified.

p\_nonzero: Percent of non zero pixels to be present in the image.

b0,b1,b2: Parametric values used to classify the images as good and bad.

Operation – The function iterates through each image in the given folder and classifies as good or bad image. The good images are retained and the bad images are deleted from the folder.

Output – The function does not return any value. It deletes the bad images from the folder.

6. Method3.py:

isLeap -

Input – year: The year which is to be checked for leap year.

Operation – The function checks if the given year is leap year or not.

Output – Returns 1 if leap year and 0 if not.

arcLengthVector -

Input – pathVector: The smoothed path vector.

StepSize: the step size to be used to create the vector of equal arc lengths

resolution: resolution of the image used.

Operation – The function computes the arc length vector with points equally spaced.

Output – Returns a list of values, arcVector.

intensityProfiles -

Input – imagespath: the path to the folder consisting of the Landsat images.

Path: the flowline of the glacier.

Input: The type of band used as input.

Weights: the weighting scheme to be used to extract intensity profiles from parallel lines. It can be linear, central or constant.

Operation – The functtion passes the values to the R function IPBL that extracts the intensity profiles from the landsat image along the flowline using bilinear interpolation. It also keeps track of the list of images from which intensities are extracted.

Output – Returns the following

timeline: the list of years for which the intensities are extracted.

IPTimeSeries: a dictionary of values with the intensity profile list through the years. The year from timeline is used as the key.

LandsatFiles: a dictionary of values with the landsat image paths for which intensities are extracted. The year from timeline is the key.

7. Method4.py:

estimateTerminus -

Input – path: The path to the folder in which results are to be saved.

Glacier: The name of the glacier.

ArcVector: The list arcVector with equally spaced values.

Timeline: The list of years in the intensity profile matrix. IpTimeSeries: The dictionary of values with intensity profiles.

gm: Ground measurements for the glacier if present.

Invert: The flag to indicate if negation of landsat is reqiored or not.

DistPerYear: Distance per year that is allowed for the search space for terminus search algorithm.

Operation – The function converts the values to compatible R datatypes and passes them to the R function terminus to estimate the terminus and plot results.

Output – Returns a complex datastructure with a set of results like the predicted measurements, range of values in the landsat image, line fit, first derivative matrix.

8. TI.py:

terminusImages -

Input - pathVectors: The list with path vector lists for the flowline and its parallel lines.

LandsatFiles: The dictionary with the landsat image paths to be plotted.

GlacierName: Name of the glacier.

Terminus: the terminus data structure returned from the function estimateTerminus.

Timeline: The list of years in intensity profile matrix.

Direc: The path to the folder to save the images.

Invert: The flag to indicate if negation of landsat is reqiored or not.

Input: The type of band given as input.

Operation – The function calls the R function loc\_terminus to compute the terminus location for each image. This terminus location and the landast image is passed to R function mark\_terminus\_plot to plot on top of the image. The function also calls the method createGif to create a gif of the landsat images with the terminus.

Output – The images with the flowline and terminus location are saved in the specified directory.

CreateGIf -

Input – direc: The path to the folder in which the GIF is to be saved.

GlacierName: Name of the glacier.

Operation – The function creates a GIF using the set of images.

Output – The GIF is saved in the specified location.

**Instructions to run the Pipeline:**

1. The latest version of the pipeline is 14. To run the Pipeline the version 14 is to be used.

- Open terminal

- Browse to the directory with pipeline 14 using following command.

cd RA/

cd Pipeline14.0/

The home folder of pipeline consists of 3 folders

- Code : The folder with all the python code and R codes.

- Data: The folder with all the Landsat images and the DEM used for producing results.

- Results: The folder with all the results for a glacier stored seperately for each band.

To run the Pipeline change to Code directory using

cd Code/

Run the pipeline using

python main.py

2. The code to download all the Landsat images is rgbplot.py. To just download the landsat images of all bands, call the downloadAndClassify function with the glacier name and bounds.

3. The GUI can be used to view the results. The code for GUI is present in the file GUI.py. It is present in the Code folder of the Pipeline directory. To run the GUI use the following command after browsing to the Code directory of Pipeline v14.

python GUI.py