COMPUTING THE YIELD PER ACRE FOR A GIVEN FIELD USING GIS CAPABILITIES

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
Of Agriculture and Applied Science

By

Sadhana Badu

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

November 2016

Fargo, North Dakota

North Dakota State University

Graduate School

Title

COMPUTING THE YIELD PER ACRE FOR A GIVEN FIELD USING GIS CAPABILITIES

By

Sadhana Badu

The Supervisory Committee certifies that this disquisition complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Anne Denton
Advisor (typed)

Dr.Simone Ludwig

Dr. Xinhua Jia

Approved by Department Chair:

11/17/2016	Dr. Brian M. Slator
Date	Signature

ABSTRACT

Software tools and technologies, such as geographic information systems (GIS), are playing an important role in the field of agriculture. The capabilities of the open source GIS system GRASS are used to process LandSat imagery and calculate the Normalized Difference Vegetation Index (NDVI), as a measure of biomass, for a given agricultural field at times before and after early harvest activity. Areas for which the NDVI value decreases substantially between two images are assumed to have been harvested during that time. Knowledge of the harvested area, in conjunction with the delivery records from American Crystal Sugar, allows computing the yield per acre for a given field. The results are compared against the yield per acre for the complete field, which is available at the end of the year. The results correlate very well, demonstrating the value of the approach.

ACKNOWLEDGEMENTS

I would like to convey my earnestness to my primary advisor and mentor, Dr. Anne Denton for her enthusiastic and encouraging support during the pursuit of my research. This defense paper is as much as a product of her guidance as the work of my own research. I also express my sincere regards to Dr. Simon Ludwig from Computer Science Department and Dr. Xinhua Jia from Agricultural and Biosystems Engineering Department for being on my supervisory committee and providing their feedback. This material is based upon work supported by the National Science Foundation through grants PFI-1114363 and IIA-1355466. Also, we thank American Crystal Sugar Company for working with us on the analysis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Remote sensing refers to the activities of recording/observing /perceiving objects or events at remote places. More specifically, it is the technology that uses sensors to acquire information about the earth's surface and atmosphere. These sensors used are not in direct contact with the objects being observed, rather they can be on satellites or mounted on aircraft. The data is collected by detecting the energy that is reflected from Earth [3]. There are two types of remote sensors- passive and active. Passive sensors detect sunlight radiation reflected from the earth and thermal radiation in the visible and infrared of the electromagnetic spectrum. These sensors do not emit any radiation, rather receive natural light and thermal radiation from the earth's surface. Scanners such as LANDSAT are used by most passive sensors for imaging. There scanners have spectrometers which measure signals at several spectral bands simultaneously, resulting in multispectral images which allows numerous interpretations [2].

In contrast, active sensors emit artificial radiation to monitor the earth surface or atmospheric features. For example, a laser-beam remote sensing system projects a laser onto the surface of Earth and measures the time that it takes for the laser to reflect back to its sensor.

The output of a remote sensing system is usually an image representing the scene being observed. A further step of image analysis and interpretation is required in order to extract useful information from the image. This can be achieved through the availability of new tools such as Geographic Information Systems (GIS). GIS is defined as a powerful set of tools for collecting, storing, retrieving, transforming and displaying spatial data from the real world for a particular set of purposes [4]. GIS allows access to large amounts of information quickly and efficiently. Also, information can be visualized in new ways that help reveal relationships, patterns, and trends [5]. It has given a vast scope to the applicability of remote sensing-based analysis and nowadays, it is this GIS technology that is becoming an essential tool for combining various map and satellite information sources in models that simulate the interactions of complex natural systems. GIS together coupled with remote sensing, is revolutionizing planning and management in the field of agriculture. Agriculture has always been playing an important role in economies of both developed and undeveloped countries. GIS is playing an increasing role in agriculture production throughout the world by helping farmers increase production, reduce costs, and manage their land more efficiently. While natural inputs in farming cannot be controlled, they can be better understood and managed with GIS applications such as crop yield estimates, soil amendment analysis, and erosion identification and remediation. Remote sensing can provide data that help identify and monitor crops. When these data are organized in a Geographical Information System along with other types of data, they become an important tool that helps in making decisions about crops and agricultural strategies. The ability of GIS to analyze and visualize agricultural environments and work flows has proved to be very beneficial to those involved in the farming industry.

1.1. Related Work

Yang et al. (2004) [13] has integrated remotely sensed data with an ecosystem model to estimate crop yield in north china. Normally, the traditional productivity simulations based on crop models are site specific. First, the spatial crop model is developed to simulate regional crop productivity in this study by integrating Geographical Information System (GIS) with Environmental Policy Integrated Climate (EPIC) model. Wu Bingfng and Liu Chenglin worked on Crop Growth Monitor System with Coupling of AVHRR and VGT data [12].

This paper is structured as follows. Chapter 1 introduces Remote Sensing and GIS. Chapter 2 describes GRASS GIS Capabilities. Chapter 3 throws light on NDVI. Chapter 4 describes the pre-processing tasks involved before getting to the computation logic. Chapter 5 presents the implementation details of the algorithm with a snapshot of pseudo code. Chapter 6 presents and discusses the results obtained together with screenshots depicting the same. Chapter 7 discusses the validity checks of the results obtained and Chapter 8 holds references.

2. GRASS GIS (CAPABILITIES)

Geographic Resources Analysis Support System (commonly termed GRASS GIS) is a geographic information system software suite used for geospatial data management and analysis, image processing, producing graphics and maps, spatial and temporal modeling, and visualizing [6]. It is Free (Libre) Software/Open Source released under GNU General Public License (GPL) that runs on multiple operating systems, including OS X, Windows and Linux. Originally developed by the U.S. Army Construction Engineering Research Laboratories (USA-CERL, 1982-1995), a branch of the US Army Corp of Engineers, as a tool for land management and environmental planning by the military, GRASS has become a high quality cutting edge GIS with an almost unparalleled depth of offering directly within the main software package. GRASS has evolved into a powerful utility with a wide range of applications in many different areas of scientific research. It is currently used in academic and commercial settings around the world, as well as many governmental agencies including NASA, NOAA, USDA, DLR, CSIRO, the National Park Service, the U.S. Census Bureau, USGS, and many environmental consulting companies [7].

GRASS has the following advantages:

- It has a simple user interface making it an ideal platform for those learning GIS for the first time, but it is powerful enough for expert users. An X/Motif based program called XGRASS lends a point-and-click interface to GRASS, where interactive windows take the place of some command line options.
- It allows for personal customization of the program and more sophisticated functionality to be fully integrated within GRASS. Users who wish to write their own code can do so by examining existing source code, along with the GRASS Programmers Manual and documented GIS libraries [8].

- GRASS is a raster/vector GIS containing over 350 programs and tools to render maps and images on monitor and paper; manipulate raster, vector, and sites data; process multi spectral image data; and create, manage, and store spatial data. It can interface with commercial printers, plotters, digitizers, and databases to develop new data as well as manage existing data [7].

2.1. Versions of GRASS

GRASS GIS 6.x is the old stable version which introduced a new topological 2D/3D vector engine and support for vector network analysis. Attributes are managed in a SQL-based DBMS (PostgreSQL, mySQL, SQLite, ODBC), by default in DBF format. A new display manager has been implemented. The NVIZ visualization tool was enhanced to display 3D vector data and voxel volumes. Messages are partially translated with support for FreeType fonts, including multibyte Asian characters. New LOCATIONs can be auto-generated e.g. by EPSG code number using a location wizard. GRASS GIS is integrated with GDAL/OGR libraries to support an extensive range of raster and vector formats, including OGC-conformal Simple Features [9].

The latest stable release version (LTS) is GRASS GIS 7, available since 2015. It offers large data support, an improved topological 2D/3D vector engine and much improved vector network analysis. Attributes are managed by default in SQLite format. The display manager has been improved for usability. The NVIZ visualization tool was completely rewritten. Image processing has also been extended. A full temporal framework has been added.

2.2. GRASS Data Structure

GRASS data is stored in a directory referred to as GISDBASE, often called grassdata/. Within this directory, GRASS GIS data is organized by projects stored in subdirectories called LOCATIONs as depicted in Figure 1 [1]. Each LOCATION is defined by its coordinate system, map projection and geographical boundaries. The subdirectories and files defining a LOCATION are created automatically when GRASS is started for the first time with a new LOCATION.

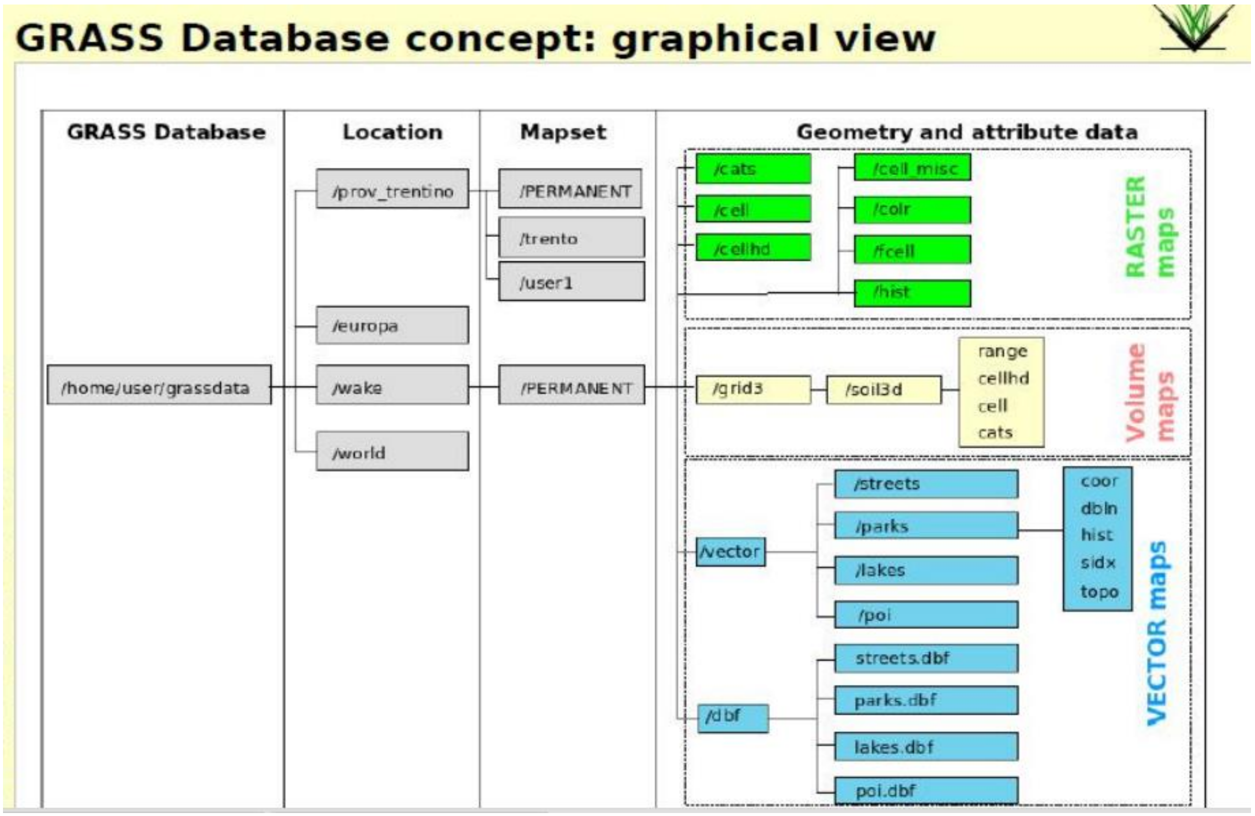


Figure 1: GRASS Data Structure

A LOCATION can have several MAPSETs (subdirectories of the LOCATION, Figure 1) that are used to subdivide the project into different topics, sub regions, or as workspaces for individual team members. Besides access to one's own MAPSET, each user can also read maps in other users' MAPSETs, but they can modify or remove only the maps in their own MAPSET. When creating a new LOCATION, GRASS automatically creates a special MAPSET called

PERMANENT designed to store the core data for the project, its default spatial extent in the DEFAULT_WIND file and coordinate system definitions. Only the owner of the PERMANENT MAPSET can add, modify or remove its data; however, these data can be accessed, analyzed, and copied by other users into their own MAPSETs. The PERMANENT MAPSET is therefore useful for providing other users working on the same project with baseline geospatial data such as elevation, roads or streams while keeping them write-protected.

2.3. Basic GRASS Commands

The command name indicates its use: first letter indicates data format or general functionality followed by a dot and a short word indicating the task that the command performs:

- d.* - display commands for graphical screen output: d.rast, d.vect, d.sites, d.mon
- g.* - general file management commands: g.list, g.copy
- i.* - image processing commands
- r.* - raster processing commands: r.slope.aspect, r.mapcalc
- v.* - vector processing commands: v.digit, v.to.rast
- s.* - site processing commands (point data): s.univar, s.surf.rst
- m.* - miscellaneous commands: m.in.e00
- p.* / ps.* - map creation ('print') commands

2.4. Raster Data Processing in GRASS GIS

A "raster map" is a data layer consisting of a gridded array of cells. It has a certain number of rows and columns, with a data point (or null value indicator) in each cell. These may exist as a 2D grid or as a 3D cube made up of many smaller cubes, i.e. a stack of 2D grids. The geographic boundaries of the raster map are described by the north, south, east, and west fields. These values describe the lines which bound the map at its edges [9].

2.5. Vector Data Processing

A "vector map" is a data layer consisting of several sparse features in geographic space. These might be data points (drill sites), lines (roads), polygons (park boundary), volumes (3D CAD structure), or some combination of all these. Typically, each feature in the map will be tied to a set of attribute layers stored in a database (road names, site ID, geologic type, etc.). As a rule, these can exist in 2D or 3D space and are independent of the GIS's computation region [9].

2.6. GRASS Commands Used in This Paper

Table 1. GRASS Raster Commands Used

r.in.gdal	Imports raster data into a GRASS raster map using GDAL library.
r.out.gdal	Exports GRASS raster maps into GDAL supported formats.
r.patch	Creates a composite raster map layer by using known category values from one (or more) map layer(s) to fill in areas of "no data" in another map layer.
r.to.vect	Converts a raster map into a vector map.
r.mapcalc	Raster map calculator.

Table 2. GRASS Vector Commands Used

v.import	Imports vector data into a GRASS vector map using OGR library and reprojects on the fly.
v.db.addcolumn	Adds one or more columns to the attribute table connected to a given vector map.
v.db.select	Prints vector map attributes.
v.db.update	Updates a column in the attribute table connected to a vector map.
v.extract	Selects vector features from an existing vector map and creates a new vector map containing only the selected features.
v.select	Selects features from vector map (A) by features from other vector map (B).
v.to.db	Populates attribute values from vector features.
v.to.rast	Converts (rasterize) a vector map into a raster map.

3. NORMALIZED DIFFERENCE VEGETATION INDEX (NDVI)

In an effort to monitor major fluctuations in vegetation and understand how they affect the environment, many years ago, Earth scientists began using satellite remote sensors to measure and map the density of green vegetation over the Earth. By carefully measuring the wavelengths and intensity of visible and near-infrared light reflected by the land surface back up into space, scientists use an algorithm called a "Vegetation Index" to quantify the concentrations of green leaf vegetation around the globe [10].

The Normalized Difference Vegetation Index (NDVI) is a numerical indicator that uses the visible and near-infrared bands of the electromagnetic spectrum, and is adopted to analyze remote sensing measurements and assess whether the target being observed contains live green vegetation or not. NDVI has found a wide application in vegetative studies as it has been used to estimate crop yields, pasture performance, and rangeland carrying capacities among others. It is often directly related to other ground parameters such as percent of ground cover, photosynthetic activity of the plant, surface water, leaf area index and the amount of biomass. NDVI was first used in 1973 by Rouse et al. from the Remote Sensing Centre of Texas A&M University [11].

To determine the density of green on a patch of land, researchers must observe the distinct colors (wavelengths) of visible and near-infrared sunlight reflected by the plants. As can be seen through a prism, many different wavelengths make up the spectrum of sunlight. When sunlight strikes objects, certain wavelengths of this spectrum are absorbed and other wavelengths are reflected. The pigment in plant leaves, chlorophyll, strongly absorbs visible light (from 0.4 to 0.7 µm) for use in photosynthesis. The cell structure of the leaves, on the other hand, strongly reflects near-infrared light (from 0.7 to 1.1 µm). The more leaves a plant has, the more these wavelengths of light are affected, respectively.

NDVI is calculated from the visible and near-infrared light reflected by vegetation. Healthy vegetation (left) absorbs most of the visible light that hits it, and reflects a large portion of the near-infrared light. Unhealthy or sparse vegetation (right) reflects more visible light and less near-infrared light. Nearly all satellite Vegetation Indices employ this difference formula to quantify the density of plant growth on the Earth — near-infrared radiation minus visible radiation divided by near-infrared radiation plus visible radiation. The result of this formula is called the Normalized Difference Vegetation Index (NDVI). Written mathematically, the formula is:

$$\boldsymbol{NDVI = \frac{NIR - VIS}{NIR + VIS}} \tag{1}$$

where, VIS and NIR stand for the spectral reflectance measurements acquired in the visible (red) and near-infrared regions, respectively.

These spectral reflectances are themselves ratios of the reflected over the incoming radiation in each spectral band individually, hence they take on values between 0.0 and 1.0. By design, the NDVI itself thus varies between -1.0 and +1.0. It should be noted that NDVI is functionally, but not linearly, equivalent to the simple infrared/red ratio (NIR/VIS). The advantage of NDVI over a simple infrared/red ratio is therefore generally limited to any possible linearity of its functional relationship with vegetation properties (e.g. biomass). The simple ratio (unlike NDVI) is always positive, which may have practical advantages, but it also has a mathematically infinite range (0 to infinity), which can be a practical disadvantage as compared to NDVI. Also in this regard, note that the VIS term in the numerator of NDVI only scales the result, thereby creating negative values. NDVI is functionally and linearly equivalent to the ratio NIR / (NIR+VIS), which ranges from 0 to 1 and is thus never negative nor limitless in range [5]. But the most important concept in the understanding of the NDVI algebraic formula is that, despite its name, it is a transformation of a spectral ratio (NIR/VIS), and it has no functional relationship to a spectral difference (NIR-VIS).

It can be seen from its mathematical definition that the NDVI of an area containing a dense vegetation canopy will tend to be positive values (say 0.3 to 0.8) while clouds and snow fields will be characterized by negative values of this index. Other targets on Earth visible from space include:

1. free standing water (e.g., oceans, seas, lakes and rivers) which have a rather low reflectance in both spectral bands (at least away from shores) and thus result in very low positive or even slightly negative NDVI values,

2. soils which generally exhibit a near-infrared spectral reflectance somewhat larger than the red, and thus tend to also generate rather small positive NDVI values (say 0.1 to 0.2).

4. PRE-PROCESSING

Even before the script to calculate the area harvested is run to help compute the yield per acre, there are some pre-processing steps to be executed. These pre-processing steps include:

1. Gathering the Field Boundaries.
2. Importing the Field Boundaries into GRASS as vector maps.
3. Converting the imported vector Field Boundary maps to their corresponding rasters.
4. Collecting LandSat imagery for our area of interest.
5. Importing the collected LandSat imagery to GRASS as raster maps.
6. Calculating NDVI for the area of interest using the imported LandSat imagery.

4.1. Gathering the Field Boundaries

The Field Boundaries are provided in Shape File format. This is a common geo-spatial vector data format for any GIS. This format essentially describes points, lines and polygon vector features representing geographical resources like rivers, lakes etc. The name ‘Shapefile’ format is not just a single file, rather has three main mandatory files associated with it without which the shape (.shp) file is incomplete for distribution. It constitutes:

- .shp file which is the shape format representing the feature geometry itself.
- .shx file which is the shape index format representing a positional index of the feature geometry to allow seeking backwards and forwards.
- .dbf file which represents the columnar attributes of each shape in attribute format.
- .sbn and .sbx represents the spatial index of the features.

4.2. Import Vector Field Boundaries into GRASS

This can be done using GRASS GIS gui or from the Command Prompt. When importing from command prompt the below command is used:

v.in.ogr dsn==/home/badu/Documents/ACSC_2013_Field_Boundary.shp l**ayer**=ACSC_2013_Field_Boundary **output**=ACSC_2013_Field_Boundary_complete -o **snap**=1e-8

This imports the Field Boundary shape files into a GRASS vector map using the OGR library.

dsn	Specifies the OGR datasource name which in case is our directory containing the Field Boundary shape maps.
Layer	Specifies the OGR layer name.
Output	Specifies the name of the output vector map.

4.3. Convert the Imported Vector Map to Raster

This pre-processing step involves converting the imported vector map to raster map.

v.to.rast in=ACSC_2013_Field_Boundary_Selected **out**=ACSC_2013_Field_Boundary_Selected_test_rast **use**=attr **attrcolumn**=cat

in	Specifies the name of input vector map.
out	Specifies name of output raster map.
use	Source of raster values. The ‘attr’ value for this specifies to read values from the attribute table.
attrcolumn	Specifies the name of the column for the ‘attr’ parameter.

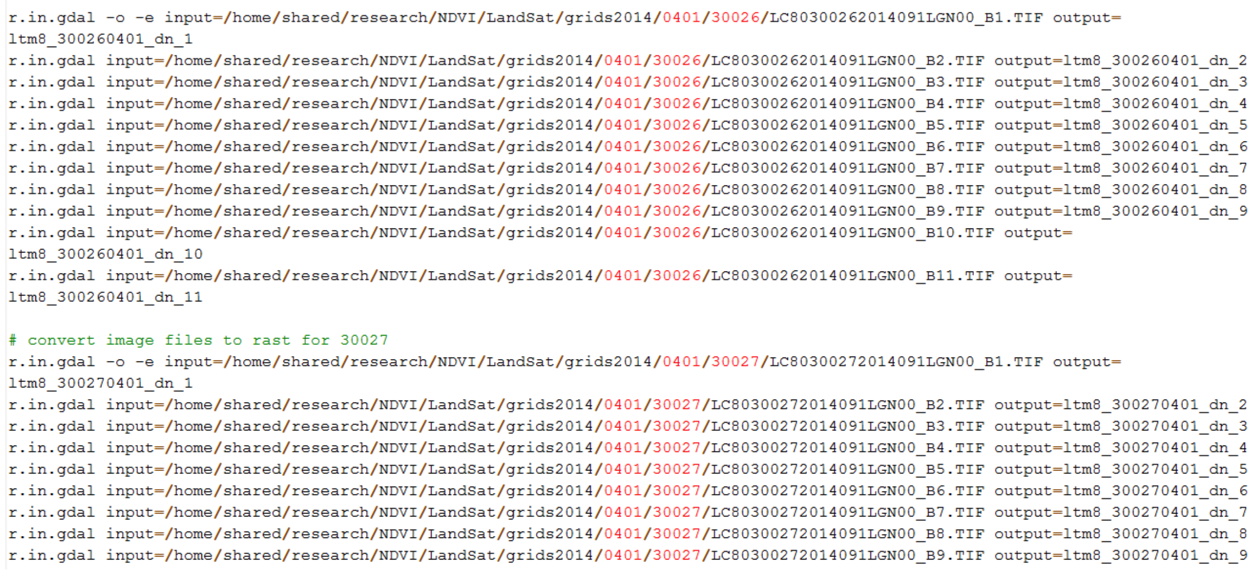
4.4. Collecting LandSat Imagery

Landsat images are available for downloading from http://glovis.usgs.gov/. ltm5 images were used most of the time initially, but it stopped transmitting images since March 2012. Images from ltm7 are available for downloading every 16 days, but they miss data in parts of the region due to technical failure. Currently latest ltm8 images are available and these are used for processing:

1. Click on the general region of Red River Valley (border between ND and MN) on the left top corner.
2. Select the two images LC80290282013289LGN00, LC80300272013264LGN00.
3. Add all the required images.
4. Click "Send to Cart".
5. Username and Password will be asked to proceed further (registration needed if you do not have an account already).
6. Wait for the confirmation email to download the image files (downloading Landsat images requires BDA application which must be installed, info would be available in the email).

4.5. Calculate NDVI

The various steps involved in the calculation of NDVI are put together in a shell script. Firstly, the collected Landsat images are imported into GRASS GIS as raster maps using the **r.in.gdal** command as shown in the figure below:



```
r.in.gdal -o -e input=/home/shared/research/NDVI/LandSat/grids2014/0401/30026/LC80300262014091LGN00_B1.TIF output=
ltm8_300260401_dn_1
r.in.gdal input=/home/shared/research/NDVI/LandSat/grids2014/0401/30026/LC80300262014091LGN00_B2.TIF output=ltm8_300260401_dn_2
r.in.gdal input=/home/shared/research/NDVI/LandSat/grids2014/0401/30026/LC80300262014091LGN00_B3.TIF output=ltm8_300260401_dn_3
r.in.gdal input=/home/shared/research/NDVI/LandSat/grids2014/0401/30026/LC80300262014091LGN00_B4.TIF output=ltm8_300260401_dn_4
r.in.gdal input=/home/shared/research/NDVI/LandSat/grids2014/0401/30026/LC80300262014091LGN00_B5.TIF output=ltm8_300260401_dn_5
r.in.gdal input=/home/shared/research/NDVI/LandSat/grids2014/0401/30026/LC80300262014091LGN00_B6.TIF output=ltm8_300260401_dn_6
r.in.gdal input=/home/shared/research/NDVI/LandSat/grids2014/0401/30026/LC80300262014091LGN00_B7.TIF output=ltm8_300260401_dn_7
r.in.gdal input=/home/shared/research/NDVI/LandSat/grids2014/0401/30026/LC80300262014091LGN00_B8.TIF output=ltm8_300260401_dn_8
r.in.gdal input=/home/shared/research/NDVI/LandSat/grids2014/0401/30026/LC80300262014091LGN00_B9.TIF output=ltm8_300260401_dn_9
r.in.gdal input=/home/shared/research/NDVI/LandSat/grids2014/0401/30026/LC80300262014091LGN00_B10.TIF output=
ltm8_300260401_dn_10
r.in.gdal input=/home/shared/research/NDVI/LandSat/grids2014/0401/30026/LC80300262014091LGN00_B11.TIF output=
ltm8_300260401_dn_11

# convert image files to rast for 30027
r.in.gdal -o -e input=/home/shared/research/NDVI/LandSat/grids2014/0401/30027/LC80300272014091LGN00_B1.TIF output=
ltm8_300270401_dn_1
r.in.gdal input=/home/shared/research/NDVI/LandSat/grids2014/0401/30027/LC80300272014091LGN00_B2.TIF output=ltm8_300270401_dn_2
r.in.gdal input=/home/shared/research/NDVI/LandSat/grids2014/0401/30027/LC80300272014091LGN00_B3.TIF output=ltm8_300270401_dn_3
r.in.gdal input=/home/shared/research/NDVI/LandSat/grids2014/0401/30027/LC80300272014091LGN00_B4.TIF output=ltm8_300270401_dn_4
r.in.gdal input=/home/shared/research/NDVI/LandSat/grids2014/0401/30027/LC80300272014091LGN00_B5.TIF output=ltm8_300270401_dn_5
r.in.gdal input=/home/shared/research/NDVI/LandSat/grids2014/0401/30027/LC80300272014091LGN00_B6.TIF output=ltm8_300270401_dn_6
r.in.gdal input=/home/shared/research/NDVI/LandSat/grids2014/0401/30027/LC80300272014091LGN00_B7.TIF output=ltm8_300270401_dn_7
r.in.gdal input=/home/shared/research/NDVI/LandSat/grids2014/0401/30027/LC80300272014091LGN00_B8.TIF output=ltm8_300270401_dn_8
r.in.gdal input=/home/shared/research/NDVI/LandSat/grids2014/0401/30027/LC80300272014091LGN00_B9.TIF output=ltm8_300270401_dn_9
```

Figure 2: Import commands to import LandSat imagery to GRASS

Then, the rasters for both the image areas spanning the Red River valley region are converted to reflectance using **i.landsat.toar** which transforms the calibrated digital number (DN) of Landsat images to the top-of-atmosphere radiance/reflectance and temperature (band 6) for Landsat. This command is executed with the below parameters:

r	This flag indicates to Output at-sensor radiance instead of reflectance for all bands.
input-prefix	Specifies the base name of input raster bands.
output-prefix	Specifies the prefixes for output raster maps.
metfile	Specifies the name of the Landsat metadata file (MTL.txt)
sensor	Specifies the spacecraft sensor.

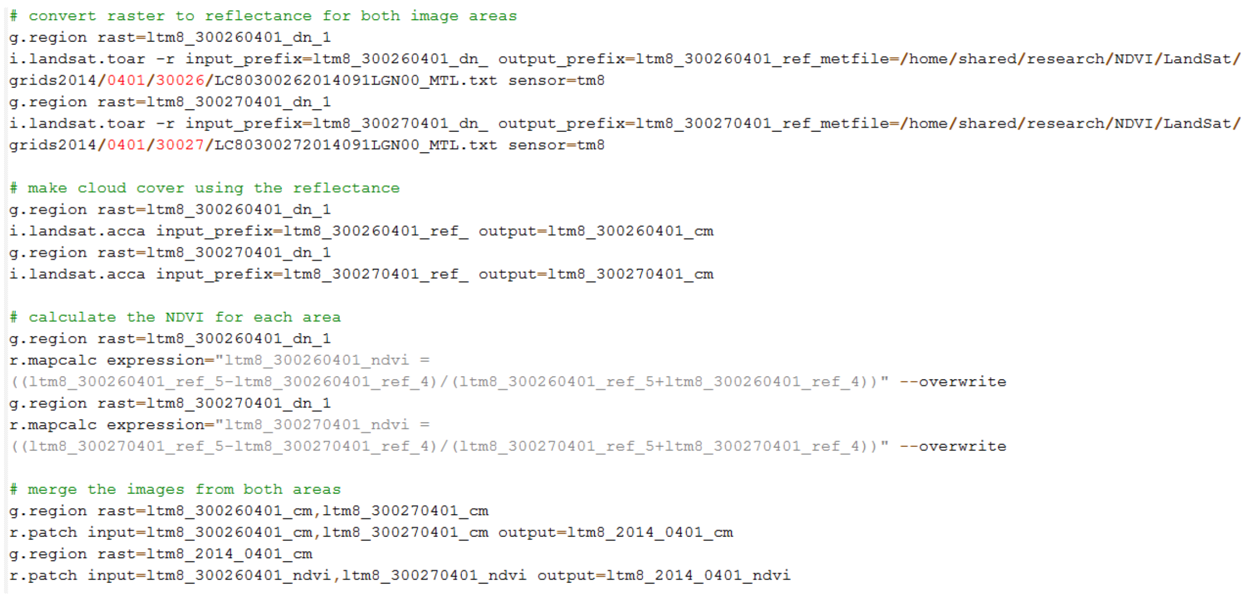
Once the radiance is calculated for all bands for raster maps, Automatic Cloud Cover Assessment (ACCA) is done using **i.landsat.acca**. It makes use of Landsat band numbers 2, 3, 4, 5, 6 which have already been processed into reflectance and band-6 temperature in the previous step.

Now the NDVI is computed using the **r.mapcalc** command which performs arithmetic on raster map layers. Arithmetic expressions involving existing raster map layers, integer/floating point constants and functions can create new raster map layers. This command takes the form “result = expression” where result is the name of the new raster map layer created holding the result of calculation, while expression is any legal arithmetic expression involving existing raster map layers.

In this case, the expression is the NDVI calculation formula while the result is the name of the resultant raster map layer. Reflectance bands 4 and 5 represent the Visible Infrared and Near Infrared regions respectively and they are embedded as below in the NDVI equation (1):

$$\mathbf{ltm8\_300260401\_ndvi} = \frac{(\mathbf{ltm8\_300260401\_ref\_5} - \mathbf{ltm8\_300260401\_ref\_4})}{(\mathbf{ltm8\_300260401\_ref\_5} + \mathbf{ltm8\_300260401\_ref\_4})} \qquad (2)$$

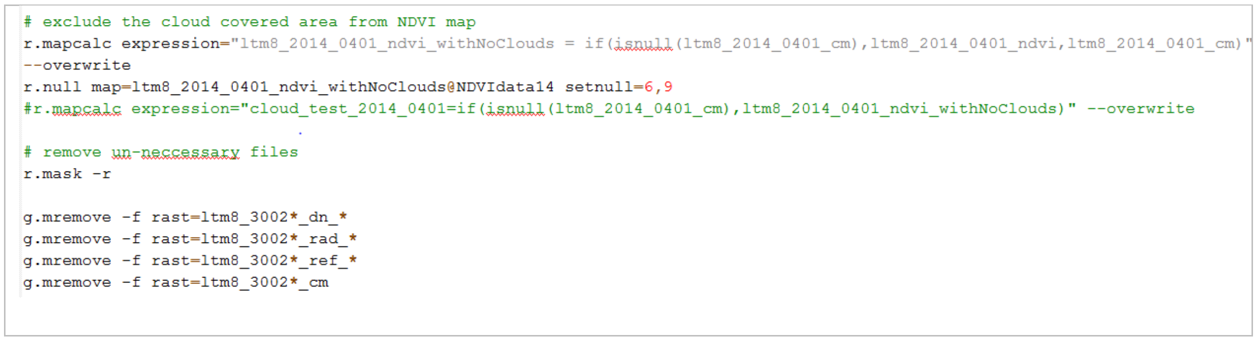
Next the Automatic Cloud Cover Assessment(ACCA) resultant raster maps and the previously computed NDVI maps for both images of the area are merged using **r.patch** which results in a new raster map with the size and resolution of current region. This is done by assigning known data values from input raster maps to the cells in the region. With this, all the calls which do not have any data or have NULL data or 0 data are filled in with the data from the first input map. Once this is done, the remaining no-data cells are filled in by the next input map. The result of this would be two composite raster layer maps formed form adjacent raster map layers – one for Cloud Cover Assessment and the other for NDVI. The following figure handles all of this:



```
# convert raster to reflectance for both image areas
g.region rast=ltm8_300260401_dn_1
i.landsat.toar -r input_prefix=ltm8_300260401_dn_ output_prefix=ltm8_300260401_ref_metfile=/home/shared/research/NDVI/LandSat/
grids2014/0401/30026/LC80300262014091LGN00_MTL.txt sensor=tm8
g.region rast=ltm8_300270401_dn_1
i.landsat.toar -r input_prefix=ltm8_300270401_dn_ output_prefix=ltm8_300270401_ref_metfile=/home/shared/research/NDVI/LandSat/
grids2014/0401/30027/LC80300272014091LGN00_MTL.txt sensor=tm8

# make cloud cover using the reflectance
g.region rast=ltm8_300260401_dn_1
i.landsat.acca input_prefix=ltm8_300260401_ref_ output=ltm8_300260401_cm
g.region rast=ltm8_300270401_dn_1
i.landsat.acca input_prefix=ltm8_300270401_ref_ output=ltm8_300270401_cm

# calculate the NDVI for each area
g.region rast=ltm8_300260401_dn_1
r.mapcalc expression="ltm8_300260401_ndvi =
((ltm8_300260401_ref_5-ltm8_300260401_ref_4)/(ltm8_300260401_ref_5+ltm8_300260401_ref_4))" --overwrite
g.region rast=ltm8_300270401_dn_1
r.mapcalc expression="ltm8_300270401_ndvi =
((ltm8_300270401_ref_5-ltm8_300270401_ref_4)/(ltm8_300270401_ref_5+ltm8_300270401_ref_4))" --overwrite

# merge the images from both areas
g.region rast=ltm8_300260401_cm,ltm8_300270401_cm
r.patch input=ltm8_300260401_cm,ltm8_300270401_cm output=ltm8_2014_0401_cm
g.region rast=ltm8_2014_0401_cm
r.patch input=ltm8_300260401_ndvi,ltm8_300270401_ndvi output=ltm8_2014_0401_ndvi
```

Figure 3: NDVI Calculation commands

Now, **r.mapcalc** is used with these two composite raster layer maps to eliminate any Cloud covered area and as a result get an NDVI raster map with no clouds. There is also some cleanup task included at the end of the script where in **g.mremove** command is used to remove any unused Dn/radiance/reflectance/cloud cover/Ndvi maps. Below if the portion of script that does this:



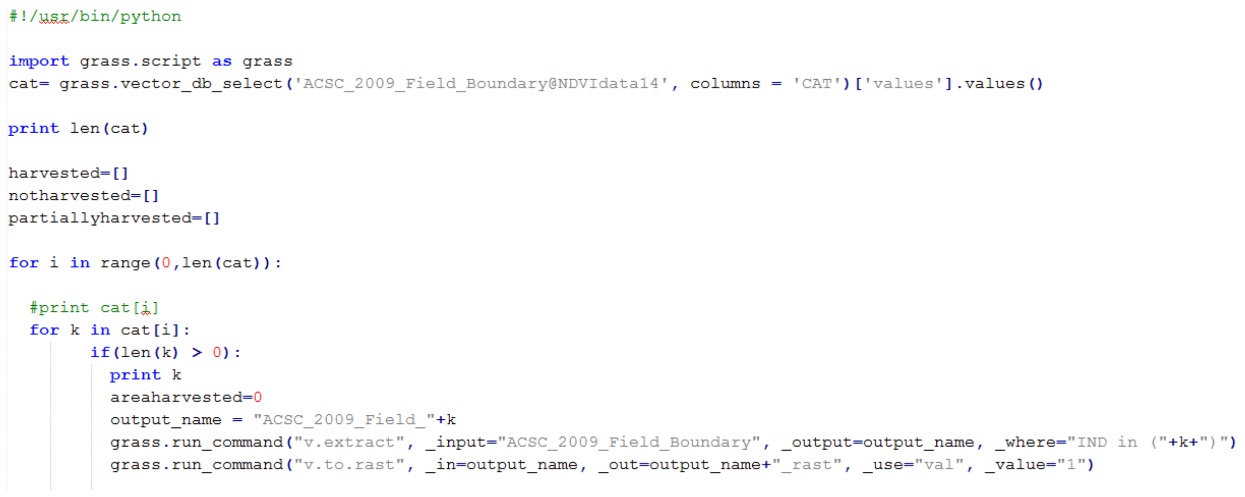
```
# exclude the cloud covered area from NDVI map
r.mapcalc expression="ltm8_2014_0401_ndvi_withNoClouds = if(isnull(ltm8_2014_0401_cm),ltm8_2014_0401_ndvi,ltm8_2014_0401_cm)"
--overwrite
r.null map=ltm8_2014_0401_ndvi_withNoClouds@NDVIdata14 setnull=6,9
#r.mapcalc expression="cloud_test_2014_0401=if(isnull(ltm8_2014_0401_cm),ltm8_2014_0401_ndvi_withNoClouds)" --overwrite

# remove un-neccessary files
r.mask -r

g.mremove -f rast=ltm8_3002*_dn_*
g.mremove -f rast=ltm8_3002*_rad_*
g.mremove -f rast=ltm8_3002*_ref_*
g.mremove -f rast=ltm8_3002*_cm
```

Figure 4: Handling Cloud cover and clean up

This process is repeated for all the months of year that are of area of interest. In this case these are the harvest period months. At this point, we would have raster maps with NDVI values for all the months of that year.

5. IMPLEMENTATION

Consider an early harvest period (September) and pick up the corresponding pre-harvest and post-harvest NDVI raster map images (say September and October). The script runs in iterations. For each iteration, we extract one field from the field boundaries vector map of that year and convert it into raster. Below is the part of script that handles this.



```
#!/usr/bin/python

import grass.script as grass
cat= grass.vector_db_select('ACSC_2009_Field_Boundary@NDVIdata14', columns = 'CAT')['values'].values()

print len(cat)

harvested=[]
notharvested=[]
partiallyharvested=[]

for i in range(0,len(cat)):

  #print cat[i]
  for k in cat[i]:
        if(len(k) > 0):
          print k
          areaharvested=0
          output_name = "ACSC_2009_Field_"+k
          grass.run_command("v.extract", _input="ACSC_2009_Field_Boundary", _output=output_name, _where="IND in ("+k+")")
          grass.run_command("v.to.rast", _in=output_name, _out=output_name+"_rast", _use="val", _value="1")
```

Figure 5: Portion of script that extracts a single field

v.extract command is used to select vector objects from an existing vector map and create a new map containing only the selected objects. Below are the parameters supplied for the extraction:

_input	Specifies the input vector map which in this case is the field boundary map for the year.
_output	Specifies the name of the newly created output vector map with selected features. In this case, it would be a unique name per field.

_where	Specifies the WHERE conditions of the SQL statement. Here using the WHERE condition the IND (index) field of the Field boundary vector data table.

The generated output vector map is converted to its corresponding raster map using **v.to.rast** command. Crop the pre-harvest NDVI image (September) as per the above converted field raster map. This is done using the **r.mapcalc** command.

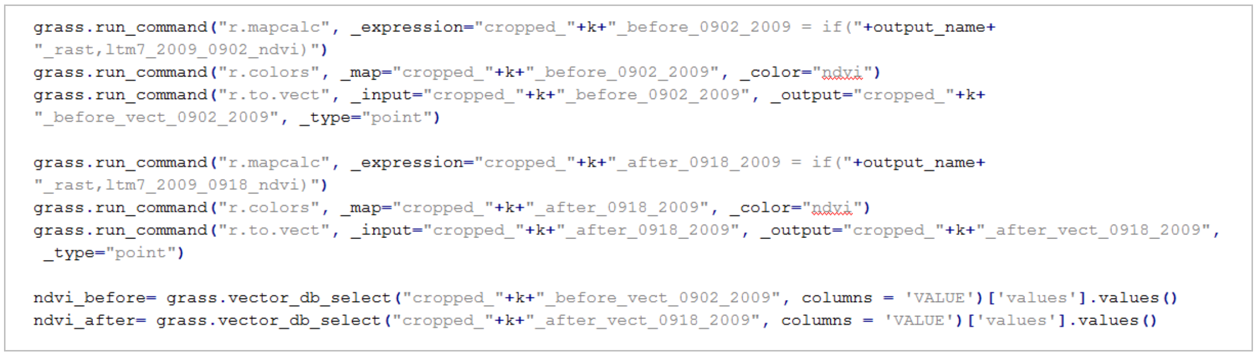
Now we will be having a new map with NDVI values (pre-harvest) for all the pixels within that field. Normalized Difference Vegetation Index colors are added to this newly created raster map by using the **r.colors** command which creates a color table for the specified map. This command takes two parameters:

map	Specifies the name of the raster map.
color	Specifies the color table that is to be created.

Before proceeding to processing the post-harvest image, the current raster map is converted back to vector format using **r.to.vect** which scans the input raster map layer, extracts point features and coverts data to GRASS vector format. The resultant vector map contains point features with NDVI values for the field.

Crop the post-harvest NDVI image (October) as per the above converted field raster map. This is a similar process as done with the pre-harvest image.

Now we will be having a new map with NDVI values (post-harvest) for the pixels within that field. The resultant of above process is two vector maps of a field representing the pre-harvest and post-harvest NDVI values. These values per pixel are read into two different arrays namely ndvi_before and ndvi_after in the script. **grass.vector_db_select** is used to accomplish this. This is all handled by the below portion of the script:



```
grass.run_command("r.mapcalc", _expression="cropped_"+k+"_before_0902_2009 = if("+output_name+
"_rast,ltm7_2009_0902_ndvi)")
grass.run_command("r.colors", _map="cropped_"+k+"_before_0902_2009", _color="ndvi")
grass.run_command("r.to.vect", _input="cropped_"+k+"_before_0902_2009", _output="cropped_"+k+
"_before_vect_0902_2009", _type="point")

grass.run_command("r.mapcalc", _expression="cropped_"+k+"_after_0918_2009 = if("+output_name+
"_rast,ltm7_2009_0918_ndvi)")
grass.run_command("r.colors", _map="cropped_"+k+"_after_0918_2009", _color="ndvi")
grass.run_command("r.to.vect", _input="cropped_"+k+"_after_0918_2009", _output="cropped_"+k+"_after_vect_0918_2009",
 _type="point")

ndvi_before= grass.vector_db_select("cropped_"+k+"_before_vect_0902_2009", columns = 'VALUE')['values'].values()
ndvi_after= grass.vector_db_select("cropped_"+k+"_after_vect_0918_2009", columns = 'VALUE')['values'].values()
```

Figure 6: Portion of script that crops the raster field map and add Ndvi color table

5.1. Core Logic

Checking Pre-Conditions-

- For every pixel in the field, we check if there exists corresponding NDVI values in the pre-harvest and post-harvest arrays
- We then check if the NDVI value of a pixel in the post-harvest array is less than the corresponding NDVI value of the pixel in pre-harvest array (Reduction in NDVI values can be attributed to area harvested)
- There is an additional check to see if the NDVI value of the pixel in pre-harvest array is greater than 0.1 (this is done to ensure that we do not consider areas which were left barren prior to the harvest)

Once the pre-conditions are satisfied, we go with the below logic:

If the pre-harvest NDVI value of the pixel is greater than or equal to (>=) 0.3 and the post-harvest NDVI of the same pixel is less than (<) 0.15, then the pixel area is considered to be harvested. (Since the maps are 30 m by 30 m resolution, area of pixel is considered to be 900 m^2)

If the pre-harvest NDVI value of the pixel is greater than or equal to (>=) 0.4 and the post-harvest NDVI of the same pixel is less than (<) 0.45, then the pixel area is considered to be partially harvested and the partially harvested area is computed by:

$$\mathbf{Partial\_area\_harvested} = \frac{(\mathbf{0.45\ - Post\_harvest\_NDVI\ value\ of\ pixel})}{\mathbf{0.45}} \tag{3}$$

If the pre-harvest NDVI value of the pixel is greater than or equal to (>=) 0.65 and the post-harvest NDVI of the same pixel is less than (<) 0.6, then the pixel area is considered to be partially harvested and the partially harvested area is computed by:

$$\mathbf{Partial\_area\_harvested} = \frac{(\mathbf{0.65 - Post\_harvest\ NDVI\ value\ of\ pixel})}{\mathbf{0.65}} \tag{4}$$

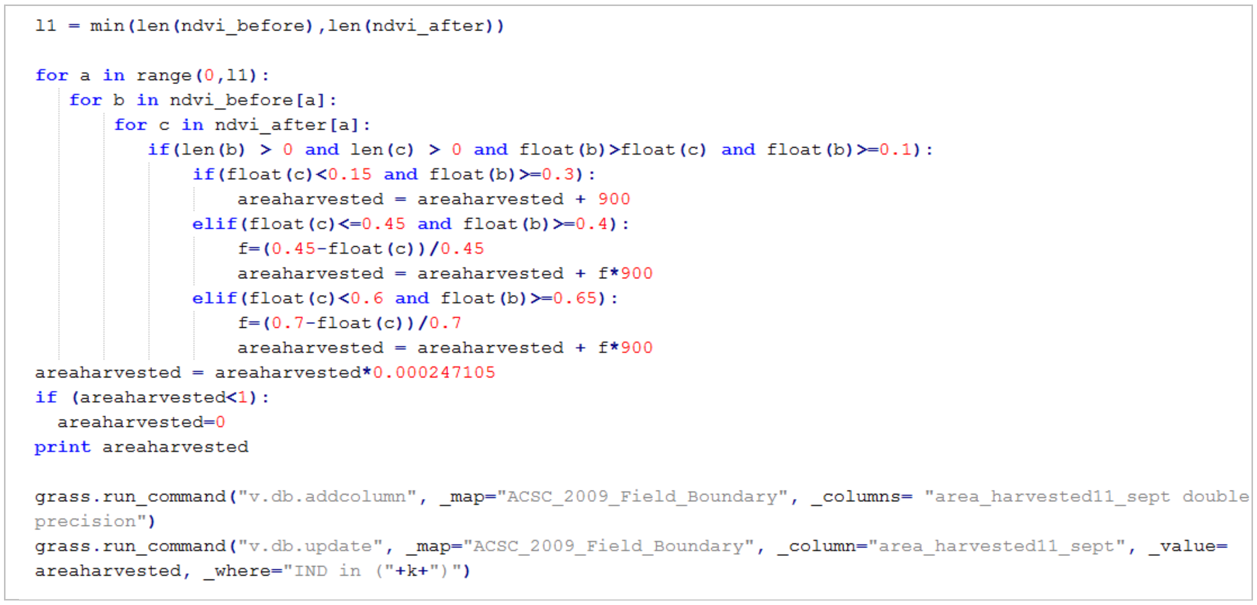
The area_harvested or partial_area_harvested for each pixel are summed up in order to get the total_area_harvested for a particular field and this value is added to the column area_harvested11 of the Field Boundary table ACSC_Field_Boundary_2013 using the commands **v.db.addcolumn** and **v.db.update** commands. **v.db.addcolumn** adds new columns to an existing vector map and is supplied with two parameters:

map	Specifies the vector map to which the new column is to be added.

columns	Specifies the name and type of the new column where the type depends on the backend database.

v.db.update assigns a new value to a column in the attribute table connected to a given map and is supplied with four parameters:

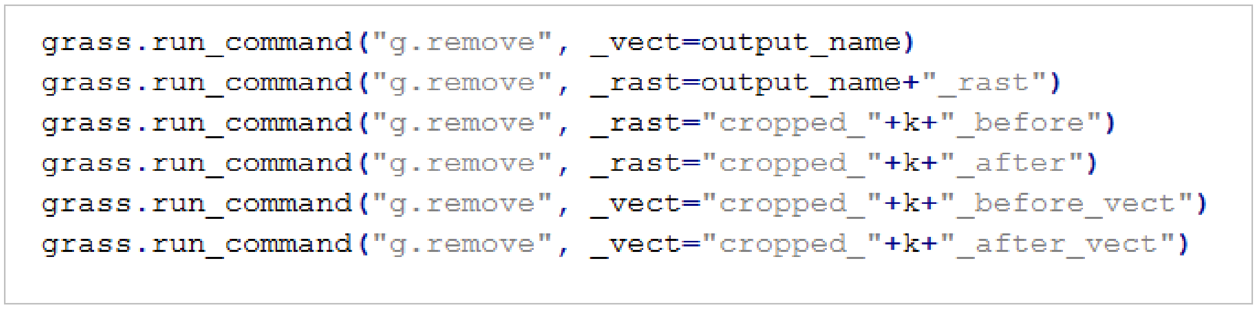
map	Specifies the name of the vector map.
column	Specifies the name of the attribute column to update.
value	Specifies the literal value to update the column with.
where	Specifies the WHERE conditions of the SQL statement.



```
l1 = min(len(ndvi_before),len(ndvi_after))

for a in range(0,l1):
   for b in ndvi_before[a]:
      for c in ndvi_after[a]:
         if(len(b) > 0 and len(c) > 0 and float(b)>float(c) and float(b)>=0.1):
            if(float(c)<0.15 and float(b)>=0.3):
               areaharvested = areaharvested + 900
            elif(float(c)<=0.45 and float(b)>=0.4):
               f=(0.45-float(c))/0.45
               areaharvested = areaharvested + f*900
            elif(float(c)<0.6 and float(b)>=0.65):
               f=(0.7-float(c))/0.7
               areaharvested = areaharvested + f*900
areaharvested = areaharvested*0.000247105
if (areaharvested<1):
  areaharvested=0
print areaharvested

grass.run_command("v.db.addcolumn", _map="ACSC_2009_Field_Boundary", _columns= "area_harvested11_sept double
precision")
grass.run_command("v.db.update", _map="ACSC_2009_Field_Boundary", _column="area_harvested11_sept", _value=
areaharvested, _where="IND in ("+k+")")
```

Figure 7: Handles calculation of area harvested

Following this, clean up commands are run to delete the intermediates maps that are created during the process of calculating the area harvested for each field.



```
grass.run_command("g.remove", _vect=output_name)
grass.run_command("g.remove", _rast=output_name+"_rast")
grass.run_command("g.remove", _rast="cropped_"+k+"_before")
grass.run_command("g.remove", _rast="cropped_"+k+"_after")
grass.run_command("g.remove", _vect="cropped_"+k+"_before_vect")
grass.run_command("g.remove", _vect="cropped_"+k+"_after_vect")
```

Figure 8: Cleaning up intermediate files

5.2. Post Running the Script

After all the iterations of the algorithm are run, we would then have the area harvested calculated and updated in the vector map table for each field in that year. We also have the corresponding delivery records for that year from which is an excel sheet contains details on the RAWTON from a field and the corresponding harvest times. With this we can compute the yield per acre for a given field.

6. RESULTS

Table 1 reports the results obtained for the computed Yield per acre for sample fields in 2013 while Table 2 represents the same for 2011. Each of the sugar beet field falls into one of the three distinct categories – 'Completely harvested'. 'Partially harvested' and 'Not harvested'.

A 'Completely harvested' field is the one which as a whole is harvested in a given harvested cycle which in the current case is the September harvest period.

A 'Partially harvested' is the one which is not as a whole harvested in a given harvest cycle and would be considered for harvest in the following harvest cycle.

A 'Not harvested' field would be the one which as a whole is not harvested and would be considered for harvest in the next harvest cycle.

<u>Description of the columns in the tables below:</u>

Field Type: Specifies which of the three types (Completely, Partially, Not Harvested) the corresponding sugar beet field is categorized into based of the area harvested,

Field Id: Uniquely identifies a sugar beet field.

Area Harvested: Represents the area harvested in the field calculated by the script for the considered early harvest period.

Total Rawton: This is a field fetched from the delivery records given by American Crystal sugar specifying how much rawton was obtained from the specific field based of harvesting during that period.

Computed Yield per Acre: This is the yield per acre calculated post running the script based of the calculated area harvested and the rawton recorded for the field in the delivery records.

Original Yield per Acre: This is the original yield per acre recorded for the field.

Table 3. Harvest Cycle based of LandSat imagery from Sep 5 - Oct 7, 2013

Field Type	**Field Id**	**Area Harvested (Sept 5 – Oct 7)**	**Total Rawton**	**Computed Yield per acre**	**Original Yield per acre**
Completely harvested	2851	96	2379	24.8	23.2
Partially harvested	1555	16	402	25	23.8
Partially harvested	3611	99	2831	28.6	27.7
Not harvested	2825	0	0	0	0
Completely harvested	3914	100	2362	23.7	23.5
Not harvested	2803	0	0	0	0
Completely harvested	1800	154	3673	23.8	23.2
Not harvested	3900	0	0	0	0
Partially harvested	2181	98	2343	23.9	25
Partially harvested	1135	105	2024	19.2	18.5
Completely harvested	931	95	2137	22.5	21.8
Not harvested	1606	0	0	0	0
Not harvested	3787	0	0	0	0

Table 4. Harvest Cycle based of LandSat imagery from Sept 8 – Sep 24, 2011

Field Type	**Field Id**	**Area Harvested**	**Total Rawton**	**Computed Yield per acre**	**Original Yield per acre**
Completely harvested	555	45	586	13	12.6
Partially harvested	1208	36	594	16.6	13.7
Partially harvested	1701	23	473	20.8	22.6
Not harvested	404	0	0	0	0
Partially harvested	1017	42	792	19	19
Not harvested	1137	0	0	0	0
Partially harvested	1142	36	797	21.8	19
Not harvested	1740	0	0	0	0
Partially harvested	1199	27	673	24.6	20.1
Not harvested	1254	0	0	0	0
Not harvested	961	0	0	0	0

Figure 9 depicts a simple linear regression model with two dimensional sample points with one independent variable and one dependent variable and find a linear function that, as accurately as possible, predicts the dependent variable values as a function of the independent variables. In this figure, the ‘Actual Yield per Acre’ on the y-axis is considered to be the dependent variable which could be computed as a function of the ‘Computed Yield per acre’ on the x-axis.

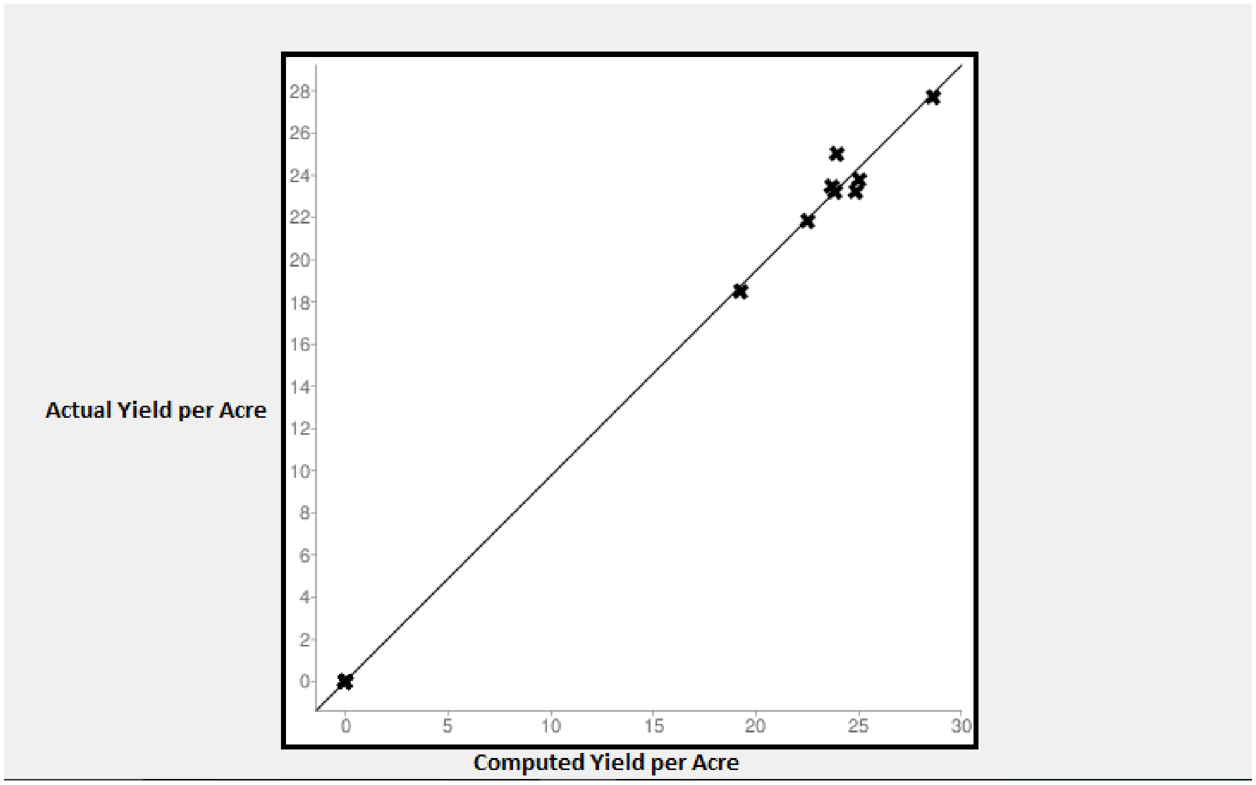


Figure 9: Linear regression 2013 (Sept – Oct)

Sample Size: 13

Mean x: 14.75

Mean y: 14.36

Intercept (a): 0.0088

Slope (b): 0.97

Regression line equation: $y = 0.0088 + 0.97 * x$

Figure 10 is a scatter plot having points that show relationship between the 'Computed Yield per acre' on the x-axis and the corresponding 'Actual yield per acre' on the y-axis. This relationship between these two variables is the correlation which in the current scenarios is:

Correlation coefficient (r): 0.99

If the data points make a straight line going from the origin out to high x- and y-values, then the variables are said to have a positive correlation. If the line goes from a high-value on the y-axis down to a high-value on the x-axis, the variables have a negative correlation.

The Correlation coefficient computed here represents a high positive correlation.

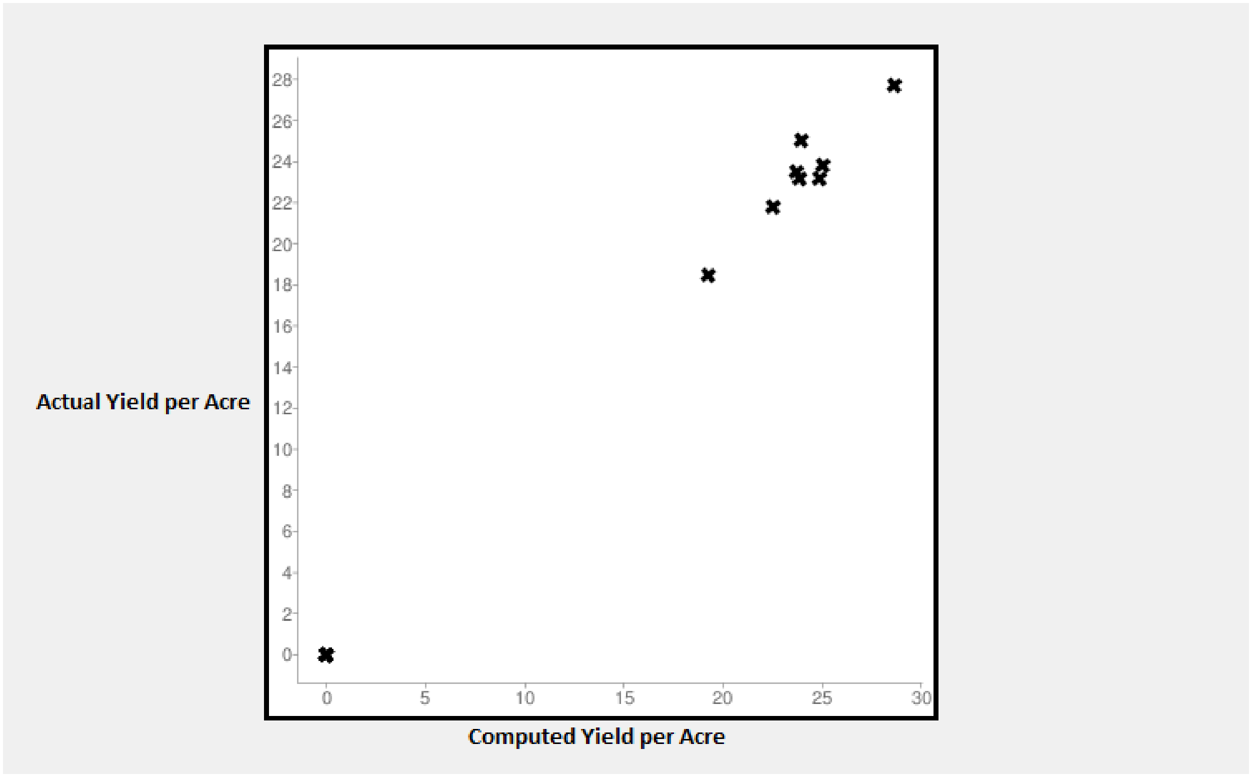


Figure 10 : Scatter plot for 2013 (Sept - Oct)

Figure 11 depicts a simple linear regression model with two-dimensional sample points with one independent variable which is the ‘Computed Yield per acre’ on the x-axis, and one dependent variable - the ‘Actual Yield per Acre’ on the y-axis.

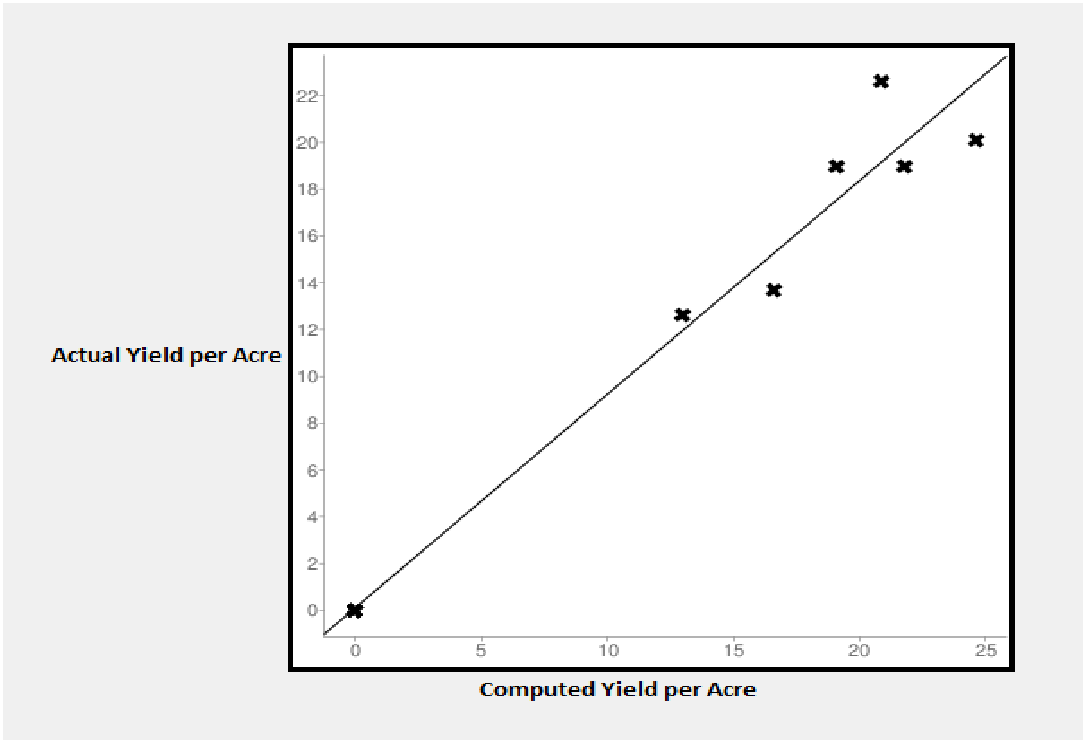


Figure 11: Linear regression 2011 (Sept)

Sample Size: 11

Mean x: 10.54

Mean y: 9.72

Intercept (a): 0.10

Slope (b): 0.91

Regression line equation: $y = 0.10 + 0.91 * x$

Figure 12 is a scatter plot having points that show relationship between the 'Computed Yield per acre' on the x-axis and the corresponding 'Actual yield per acre' on the y-axis for the year 2011. This relationship between these two variables is the correlation which in the current scenarios is:

Correlation coefficient (r): 0.98

The high Correlation coefficient indicate that the two sets of data are strongly linked together.

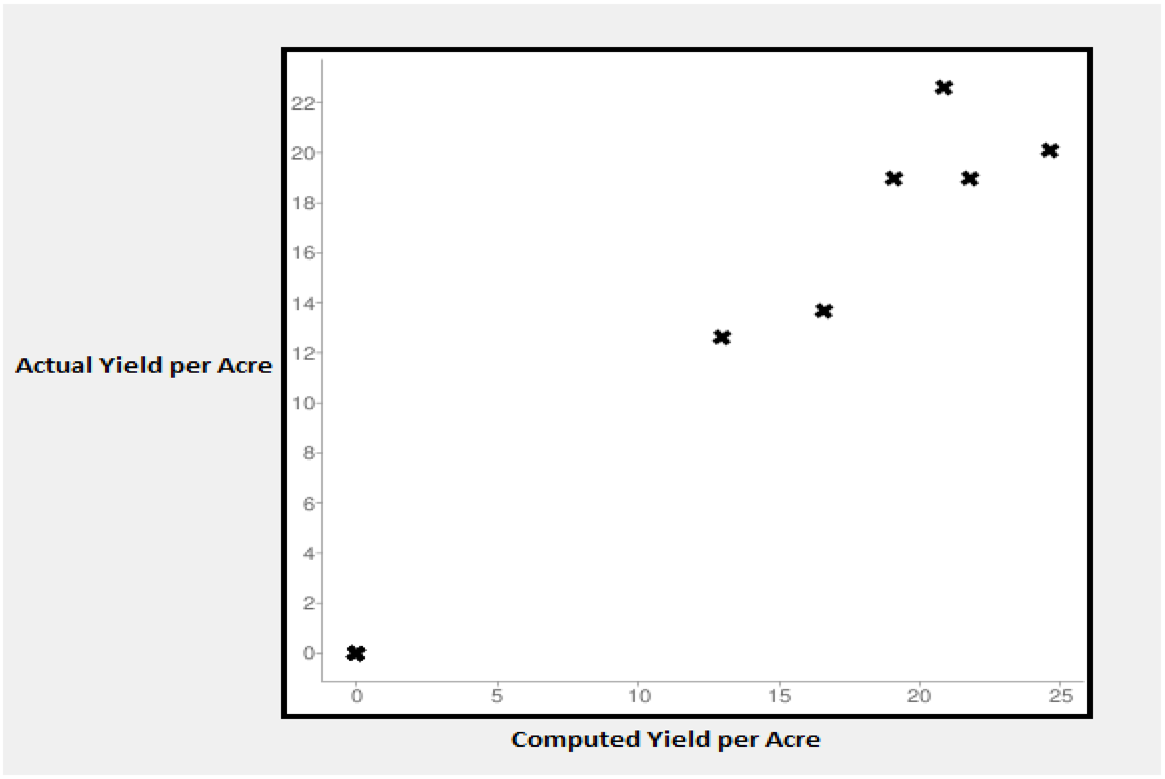


Figure 12: Scatter plot 2011 (Sept)

Following are the pre-harvest and post-harvest raster maps that can be viewed in GRASS for the three different categories the fields might fall into – Completely harvested, Partially harvested and Not harvested.

Figure 13 depicts a completely harvested field with the corresponding NDVI values pre and post-harvest.

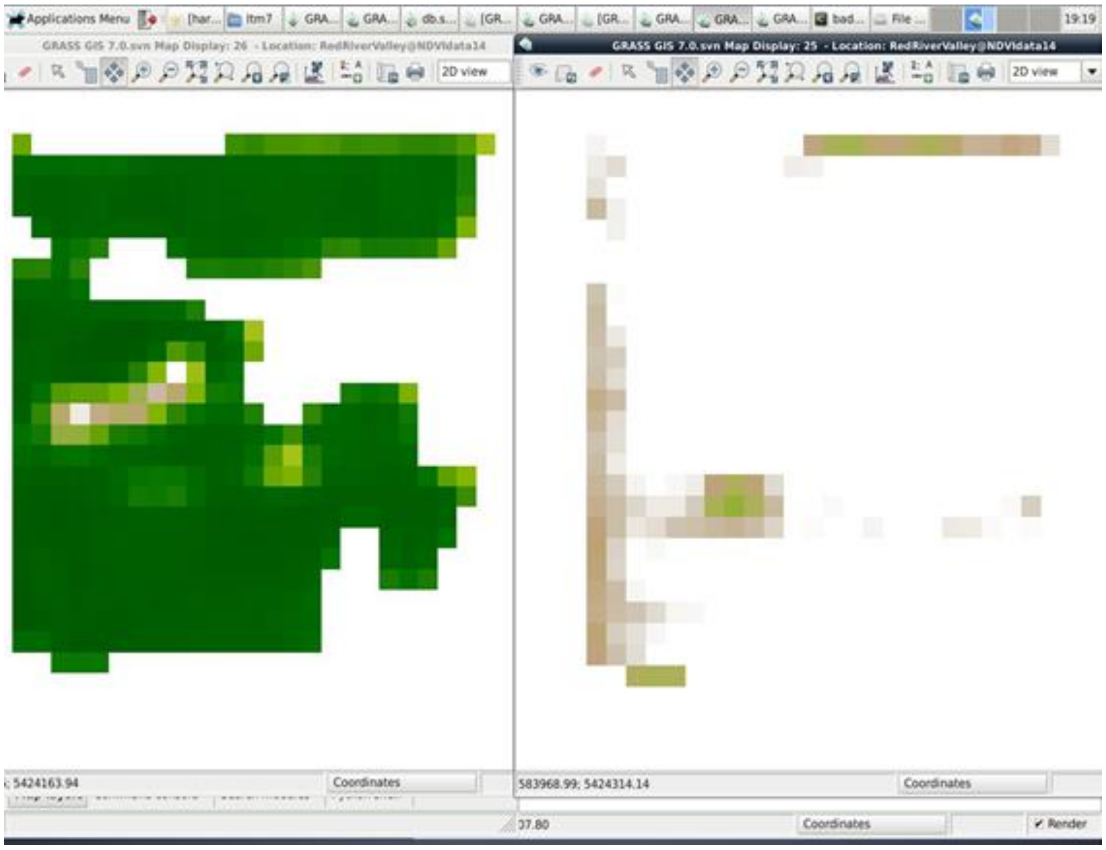

GRASS GIS 7.0.svn Map Display: 26 - Location: RedRiverValley@NDVIdata14
GRASS GIS 7.0.svn Map Display: 25 - Location: RedRiverValley@NDVIdata14
2D view
Coordinates
583968.99; 5424314.14
Render

Figure 13: NDVI Raster maps of completely harvested field

Completely harvested field: 2851

Area harvested: 96

Total RAWTON (from delivery records): 2379

Computed Yield per acre: 2379/96 = 24.8

Original recorded Yield per acre: 23.2

Figure 14 depicts a partially harvested field with the corresponding NDVI values pre and post-harvest.

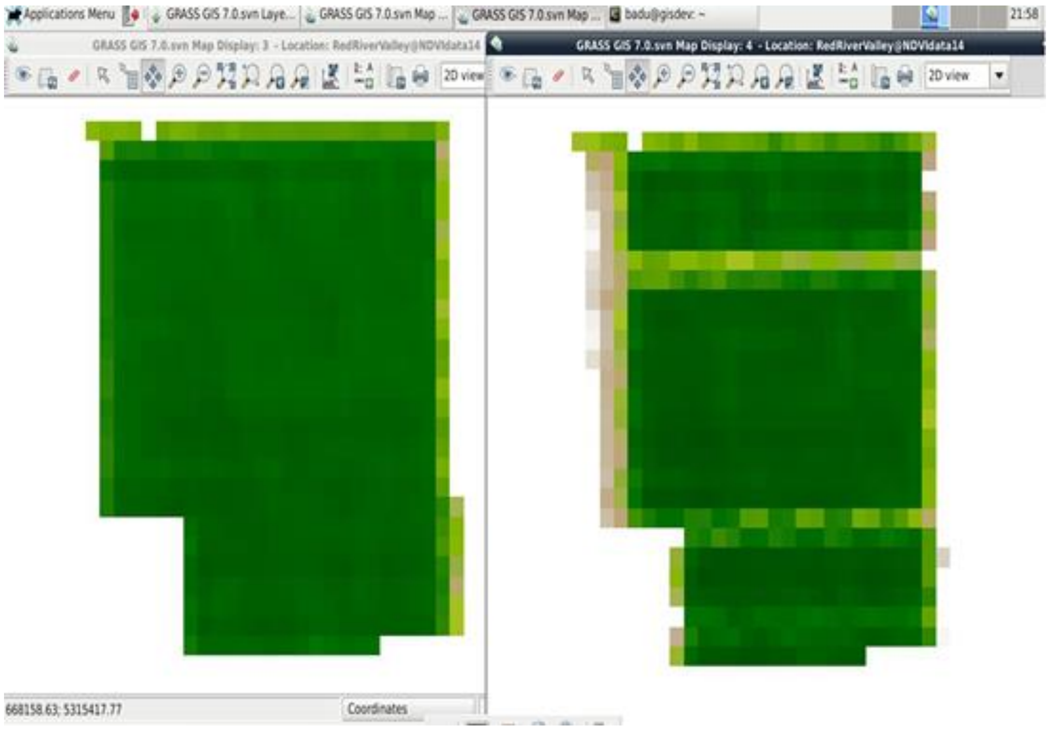

Applications Menu
GRASS GIS 7.0.svn Laye...
GRASS GIS 7.0.svn Map ...
GRASS GIS 7.0.svn Map ...
badu@gisdev: ~
21:58
GRASS GIS 7.0.svn Map Display: 3 - Location: RedRiverValley@NDVIdata14
GRASS GIS 7.0.svn Map Display: 4 - Location: RedRiverValley@NDVIdata14
2D view
2D view
668158.63; 5315417.77
Coordinates

Figure 14: NDVI Raster maps of partially harvested field

Partially Harvested field: 1555

Area harvested: 16

Total RAWTON (from delivery records): 402

Computed Yield per acre: 402/16 = 25

Original recorded Yield per acre: 23.8

Figure 15 depicts a field that has not been harvested during the period considered.

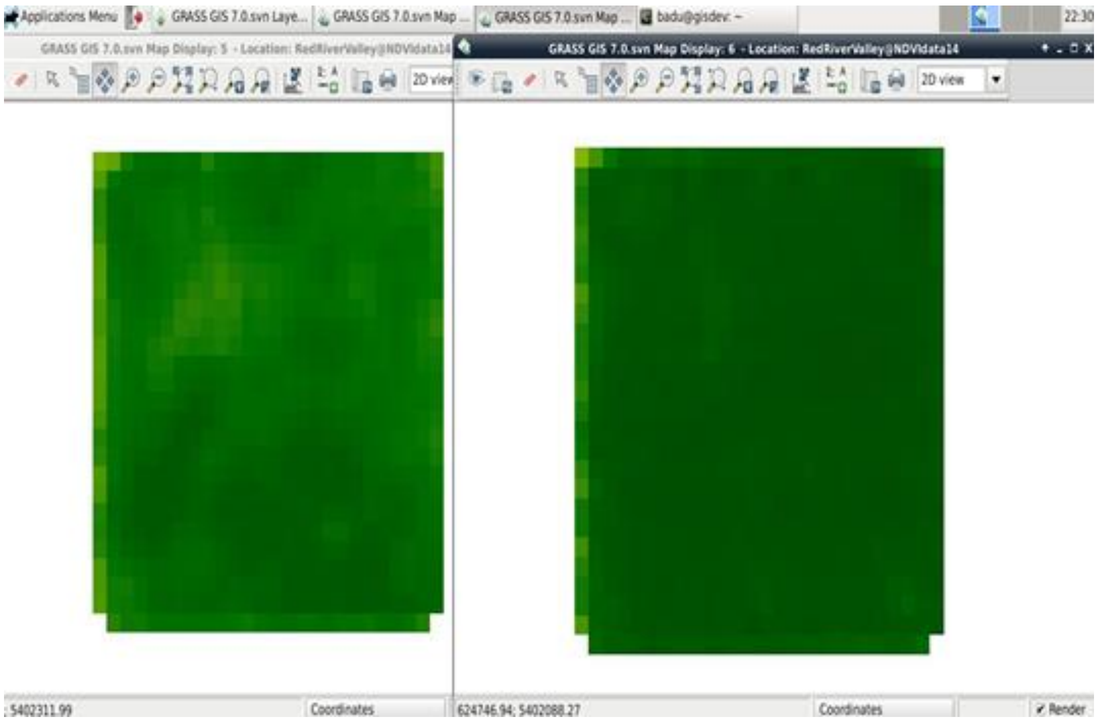

Applications Menu
GRASS GIS 7.0.svn Laye...
GRASS GIS 7.0.svn Map ...
GRASS GIS 7.0.svn Map ...
badu@gisdev: ~
22:30
GRASS GIS 7.0.svn Map Display: 5 - Location: RedRiverValley@NDVIdata14
GRASS GIS 7.0.svn Map Display: 6 - Location: RedRiverValley@NDVIdata14
2D view
2D view
: 5402311.99
Coordinates
624746.94; 5402088.27
Coordinates
Render

Figure 15: NDVI Raster maps of not harvested field

Not harvested field: 2825

Area harvested: 0

Total RAWTON (from delivery records): 0 (no delivery records for the first harvest period)

These pre-and post-harvest raster maps depict that there has only been increase in the NDVI values of pixels through the harvest cycle considered, which clearly indicates that this field hasn't been harvested in that cycle.

7. CONCLUSION & FUTURE WORK

The approach used in this paper helps to compute the yield per acre for a given sugar beet field by calculating the area harvested during early harvest together with information from delivery records provided by American Crystal Sugar seems promising. This process has been repeated for all the fields across all years from 2010 to 2013. Comparing the computed yield per acre to the actual value, one could see that they are close enough. GRASS GIS capabilities have been leveraged successfully in accomplishing the results. Below are some of the basic validity checks for the script being used, observations and few other factors that could be considered for future work:

7.1. Validity Check

The script being used uses 30mx30m (900 square meters) resolution per pixel when calculating the area harvested. A simple validity check that can be done here is to sum up all the pixel area together and see if that equals to the actual area covered by the field just to make sure the same area is taken into consideration during the computations.

7.2. Observations

- For the fields, which haven't been harvested during the period considered, the algorithm gives the area harvested as 0.0xxx which is a very low value that can be ignored. However, the program can be revisited to see if there is any minor change that can be done to have the results more accurate and the area harvested value can be brought to zero for those fields that are for sure not harvested during the considered harvest cycle.

- Not every computed value for the Yield per acre of the field matches exactly with the actual Yield per acre, there are some differences. This could be due to the account that we are only considering NDVI as a factor for calculating the Yield per acre.

7.3. Other Factors to be Considered

- Currently, only the first harvest period (September-October) is considered in the computations since the corresponding Landsat imagery for the second/third harvest period (post October/November) are cloudy. Even though GRASS GIS has capabilities to remove the cloud cover.
- Wilting of plants can contribute to lower NDVI values. This has not been taken account in the computation.

8. REFERENCES

[1] M. Neteler and H. Mitasova, Open Source GIS: a GRASS GIS approach (3rd ed.), New York: Springer, 2008.

[2] J, Albertz, Introduction to remote sensing. Basics of the interpretation of aerial and satellite images., Darmstadt, 2007.

[3] D. S. C. Liew, "PRINCIPLES OF REMOTE SENSING," Centre for Remote Imaging, Sensing and Processing, National University of Singapore, 2001. [Online]. Available: http://www.crisp.nus.edu.sg/~research/tutorial/rsmain.htm. [Accessed 2016].

[4] S.B, Goswani; S, Matin; and A. Saxena. G. Bairagi, "A Review: The application of Remote Sensing, GIS and GPS in Precise Agriculture," *International Journal of Advanced Technology & Engineering Research,* vol. 2, no. 1, pp. 2250-3536, 2012.

[5] A. L. Coleman and J. M. Galbraith, "Using GIS as an Agricultural Land-Use Planning Tool," December 2000. Available: https://scholar.lib.vt.edu/ejournals/vaes/00-2.pdf. [Accessed 2016].

[6] J. Westervelt, "GRASS roots. In Proceedings of the FOSS/GRASS User Conference, Bangkok, Thailand.," 12 September 2004. [Online]. Available: https://grass.osgeo.org/uploads/grass/history_docs/westervelt2004_GRASS_roots.pdf. [Accessed 2016].

[7] L. Martin, "GRASS GIS," 04 08 2010. [Online]. Available: https://grass.osgeo.org/home/. [Accessed 2016].

[8] B. Byars, M. Neteler, S. Clamons and S. Cherry, "Geographic Resources Analysis and Support System," [Online]. Available: https://grass.osgeo.org/gdp/tutorial/grass42factsheet.pdf. [Accessed 2016].

[9] G. D. Team, "GRASS General Overview," 24 Febraury 2015. [Online]. Available: https://grass.osgeo.org/documentation/general-overview/. [Accessed 2016].

[10] J. Weier and D. Herring, "Measuring Vegetation (NDVI & EVI)," 30 August 2000. [Online]. Available: http://earthobservatory.nasa.gov/Features/MeasuringVegetation/ measuring_vegetation_2.php. [Accessed 2016].

[11] M. Roderick, R. C. G. Smith and G. Ludwick, "Calibrating long term AVHRR-derived NDVI imagery. Remote Sensing of Environment 58," pp. 1-12, 1996. [Online]. Available: https://www.researchgate.net/publication/223924937_Calibrating_long-term_AVHRR-derived_NDVI_imagery. [Accessed 2016].

[12] P. Yang, T. G.X, Y. Zha and R. Shibasaki, "Integrating remotely sensed data with an ecosystem model to estimate crop yield in North China.," in *The International Archives of the Photogrammetry, Remote Sensing ans Spatial Information Sciences*, Istanbul, pp. 150-155, 2004.

[13] Al. Yang et, "Hybrid-maize-a maize simulation model that combines two crop modeling approaches Field Crops Res," 2004, pp. 131-154. [Online]. Available: http://digitalcommons.unl.edu/cgi/viewcontent.cgi?article=1137&context =agronomyfacpub. [Accessed 2016].