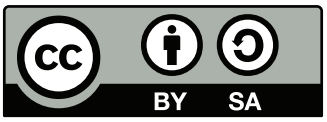
The Remote Sensing and GIS Software Library.

Pete Bunting and Daniel Clewley

Documentation and Examples of using RSGISLib.

March 31, 2013

Aberystwyth University



Acknowledgements

Authors

Peter Bunting

Dr Pete Bunting joined the Institute of Geography and Earth Sciences (IGES), Aberystwyth University, in September 2004 for his Ph.D. where upon completion in the summer of 2007 he received a lectureship in remote sensing and GIS. Prior to joining the department, Peter received a BEng(Hons) in software engineering from the department of Computer Science at Aberystwyth University. Pete also spent a year working for Landcare Research in New Zealand before rejoining IGES in 2012 as a senior lecturer in remote sensing.

Contact Details

EMail: pfb@aber.ac.uk

Senior Lecturer in Remote Sensing
Institute of Geography and Earth Sciences
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
United Kingdom

Daniel Clewley

Dr Dan Clewley joined IGES in 2006 undertaking an MSc in Remote Sensing and GIS, following his MSc Dan undertook a Ph.D. entitled Retrieval of Forest Biomass and Structure from Radar Data using Backscatter Modelling and Inversion under the supervision of Prof. Lucas and Dr. Bunting. Prior to joining the department Dan completed his BSc(Hons) in Physics within Aberystwyth University. Dan is currently a post-doc researcher at the University of Southern California.

Contact Details

Email: clewley@usc.edu

Postdoctoral Research Associate
Microwave Systems, Sensors, and Imaging Lab (MiXIL)
Ming Hsieh Department of Electrical Engineering
The University of Southern California
Los Angeles
USA

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The remote sensing and GIS software library (RSGISLib) was developed at Aberystwyth University by Pete Bunting and Daniel Clewley. Development started in April 2008 and has been actively maintained and added to ever since. For more information see `http://www.rsgislib.org`.

1.2 Using RSGISLib

RSGISLib has a command line user interface where the main commands you will be using are:

rsgisexe - the main command to execute scripts

rsgislibxmllist - a command to list all the available commands within the library (there are over 300!!)

rsgislibcmdxml.py - a command to allow script templates to be populated with file paths and names.

rsgislibvarsxml.py - a command to input variable values into a template script.

1.3 The RSGISLib XML Interface

1.3.1 XML Basics

RSGISLib is parameterised through the use of an XML script. XML stands for Extensible Markup Language.

Extensible - XML is extensible. It lets you define your own tags, the order in which they occur, and how they should be processed or displayed. Another way to think about extensibility is to consider that XML allows all of us to extend our notion of what a document is: it can be a file that lives on a file server, or it can be a transient piece of data that flows between two computer systems.

Markup - The most recognizable feature of XML is its tags, or elements (to be more accurate).

Language - XML is a language that's very similar to HTML. It's much more flexible than HTML because it allows you to create your own custom tags. However, it's important to realize that XML is not just a language. XML is a meta-language: a language that allows us to create or define other languages. For example, with XML we can create other languages, such as RSS, MathML (a mathematical markup language).

```
<parent_element>
    <some_information>
    </some_information>
    <some_information name="some data" value="some other data" />
</parent_element>
```

XML is made up of opening and closing elements, where the hierarchy of the elements provides meaning and structure to the information stored. Therefore, every element has an opening and closing element. This can be defined in two ways; firstly with two tags, where the opening tag is just enclosed with angled brackets ($< tag >$) and the closing tag contains a backslash and angled brackets $< /tag >$. Using this method further tags for data can be stored between the two tags, providing structure as shown above. The second method uses just a single

tag with an ending backslash ($< tag/ >$). This second method is used when no data or further tags are to be defined below current element.

```
<element></element>

<element/>
```

1.3.2 Escape Characters

As with all computing languages there are certain characters which have specific meanings and therefore an escape character needs to be used if these characters are required within the input.

& - &

′ - '

” - "

< - <

> - >

= - =

```
<element attribute="&apos;hello&apos;"/>
    <element>
        1 is &lt; 100
   </element>
<element attribute="&quot;world&quot;"/>
```

1.3.3 Commenting

To add comments to XML code and temporally comment out parts of your XML script you need to use the XML commenting syntax as show below.

```
<!-- Some useful comment -->
<parent_element>
    <some_information>
    </some_information>
    <!-- This is some really useful information in this comment -->
```

```
    <some_information name="some data" value="some other data" />
</parent_element>
```

All parts of the document between the opening and closing comment tags will be ignored by the parser.

1.3.4 RSGISLib XML

For parameterisation of the rsgisexe application you will need to create an XML file in the correct format, which the RSGISLib executable understands, while adhering to the rules of XML outlined above. The basis for the RSGISLib XML is to provide a list of commands. Therefore, the XML has the following structure:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
    Description:
        XML File for execution within RSGISLib
    Created by **ME** on Wed Nov 28 15:53:41 2012.
    Copyright (c) 2012 **Organisation**. All rights reserved.
-->

<rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">

  <!-- ENTER YOUR XML HERE -->
  <rsgis:command algor="name" option="algor_option" attr1="foo"
                                                    attr2="bar">
    <rsgis:data attribute="blob" />
  </rsgis:command>
  <rsgis:command algor="algor_name" option="algorithm_option"
                                                    attr="data"/>

</rsgis:commands>
```

Where all the input parameters are defined using element attributes and each algorithm and option have their own set of attributes to be specified. Within the XML file imported into rsgisexe multiple command elements can be specified and they will all be executed in the order specified in the XML file. Therefore, a sequence of events can be specified and executed without any further interaction.

1.4 Basic UNIX

When interacting with a UNIX terminal you will need to so using a UNIX command line console. The basic UNIX commands for navigating the file system are shown below.

```
# List the current directory
ls

# Enter the directory called 'directory_name'
cd directory_name

# Go down a directory
cd ..

# Display contents of a file
cat file.txt

# Display contents of a file and stop at end of each page
cat file.txt | more

# Display header information for an image
gdalinfo image.kea | more

# Rename file1.txt to file2.txt
mv file1.txt file2.txt

# Move file.txt to within directory output
mv file.txt output/file.txt

# Copy the file file2.txt to file2.txt
cp file1.txt file2.txt

# Copy a directory files1 to files2
cp -R ./files1 ./files2

# Delete a file
rm file.txt

# Delete a directory
```

```
rm -Rf ./files

# List commands previously executed
history

# Edit a text file
ne name_of_text_file.txt
```

1.4.1 Editing a text file

All the scripts you will use to interact with the machine are text files (even if they have the extension .xml or .sh) to edit them we recommend you use the 'ne' editor ('the nice editor'). This editor uses commonly used desktop keyboard shortcuts for saving files and quitting the application. Table 1.1 lists the common shortcuts while pressing escape will show a menu from which you can select the option you require.

Table 1.1: Keyboard shortcuts for the ne editor.

Function	Shortcut
Saving	ctrl-s
Quit	ctrl-q
Find	ctrl-f
Jump to Line	ctrl-j

1.5 Using the Batch Queue on HPC Wales

The high performance computer (HPC) facility 'HPC Wales' might be available to some of you. The system uses a batch priority queue to manage the jobs submitted by multiple users. The job of the queue manages the jobs, which are being submitted to the computer system to make sure that as much of the system as possible is being utilised. Therefore, if you submit a job which is small (i.e., does not use much runtime, memory or a large number of processes) it is likely to run much sooner than a job which requires a large amount of resources as it will take longer for the scheduler to find space on the system for the large job.

There are a number of command line tools associated with queue facility, including:

```
# Submit a job
bsub < jobfile.lsf

# List your jobs submitted
bjobs

# List all jobs in queue
bjobs -u all

# Info on previously run jobs
bhist -l <job id>

# Remove a job from the queue
bdel <job id>
```

To submit a job you need to create a job file script which provides the scheduler information such as run time, user account etc such that it can appropriate allocate the job. A template is shown below.

```
#!/bin/bash --login
#BSUB -J JOB_NAME
#BSUB -o JOB_CONSOLE_OUTPUT_FILE.out
#BSUB -e JOB_CONSOLE_ERROR_FILE.err
#BSUB -W HH:MM (RUN TIME)
#BSUB -P sam0004
#BSUB -n 1 (NUMBER OF PROCESSING CORES)
#BSUB -R "span[ptile=1]"

# ENTER THE COMMAND WHICH NEEDS TO EXECUTED
rsgisexe -x processSomeData.xml
```

Chapter 2

Installing RSGISLib

Chapter 3

Examples – Image Processing

3.1 Stacking Image Bands

To stack image bands within RSGISLib there are two commands are provided, the first attempts to stack all the images (with a specified file extension) within a directory while the second stacks a specified list of images but in both cases all the images need to intersect and have the same image resolution.

```
<!--
    Stacks all the image bands within a directory into a
    single image file
-->
<rsgis:command algor="stackbands" option="dir" dir="input_DIR"
               output="outputimage" ext="file_extension"
               format="GDAL Format" datatype="Byte | UInt16 |
               Int16 | UInt32 | Int32 | Float32 | Float64"  />
<!--
    Stacks all the image bands provided in list
    into a single image file
-->
<rsgis:command algor="stackbands" option="imgs" output="outputimage"
               format="GDAL Format" datatype="Byte | UInt16 | Int16 |
               UInt32 | Int32 | Float32 | Float64" skipValue="float" >
    <rsgis:image name="band(s) name" file="image1" />
    <rsgis:image name="band(s) name" file="image2" />
```

```
18      <rsgis:image name="band(s) name" file="image3" />
19      <rsgis:image name="band(s) name" file="image4" />
20  </rsgis:command>
```

3.2 Basic Landsat Imagery Pre-Processing

The first step when downloading optical imagery is to pre-process it such that it is geographically registered and spectrally it is a measure of reflectance.

You have been provided with four Landsat 7 ETM+ scenes covering parts of Wales. Your task is to:

1. Convert the measured spectral values to at sensor radiance.
2. Convert the measured at sensor radiance to top of atmosphere (TOA) reflectance.
3. Where possible pre-calculate image pyramids and statics for visualisation.
4. Stack the individual Landsat ETM+ bands to create a single image.
5. Re-project the scene to the Ordnance Survey National Grid from the Universal Transverse Mercator (UTM) it is provided in.
6. Calculate the Normalised Difference Vegetation Index (NDVI)

Ideally you would correct the data to surface reflectance but to do so requires the use of an atmospheric model. Atmospheric models require parameterisation so to simplify the process you will just correct the data to top of atmosphere reflectance.

Top of atmosphere (TOA) reflectance is the ratio of the incoming light, from the Sun (i.e., source), and the light reflected and measured at the sensor. This assumes there is no atmospheric interference or bi-directional reflectance (BRDF) effects within the scene. These are clearly false assumptions but it does provide a simple and fast method of correcting to a measured unit which is corrected for the variation in the solar irradiance, due to season and angle.

To correct the imagery to TOA the first step is to convert the 8 bit (0 – 255) digital number (DN) pixels values to floating point radiance values (i.e., the amount of energy measured at the sensor).

Side Question

What is radiance and what is its unit?[ab]

[a]Radiance is defined as "The power passing through a unit area in a unit solid angle about the normal to the area (per unit spectral interval)"

[b]watts per steradian per square nano metre $W \cdot sr^{-1} \cdot m^{-3}$ or $W \cdot sr^{-1} \cdot nm^{-1}$ for a given wavelength.

To convert DN's to radiance you need to use Equation 3.1 and the gains and offsets for each landsat 7 band, provided in Table 3.1. The gains and offsets are also available with the MTL text file associated within each Landsat scene downloaded from the USGS.

$$L = \left(\frac{LMAX - LMIN}{QCAL_{max} - QCAL_{min}} \right) (DN - QCAL_{min}) + LMIN \tag{3.1}$$

Table 3.1: Landsat ETM+ (7) gains and offsets for converting DN's to radiance

Band	LMin	LMax	QCal$_{min}$	QCal$_{max}$	Wavelength (nm)	Resolution (m)
1	-6.2	191.6	1	255	450–515	30
2	-6.4	196.5	1	255	525–605	30
3	-5.0	152.9	1	255	630–690	30
4	-5.1	157.4	1	255	750–900	30
5	-1.0	31.06	1	255	1550–1750	30
6 (1)	0.0	17.04	1	255	10400–12500	60
6 (2)	3.2	12.65	1	255	10400–12500	60
7	-0.35	10.8	1	255	2090–2350	30
8	-4.7	243.1	1	255	520–900	15

For this exercise you only need to process bands 1-5 and 7, which are the visible, NIR and SWIR bands with a resolution of 30 m. Bands 6 (1 and 2) are within the thermal part of the electrometric spectrum and have a resolution of 60 m. Finally, band 8 is a panchromatic band (measuring from green to NIR) with a spatial resolution of 15 m.

To convert the radiance image to TOA Equation 3.2 needs to be used with the coefficients shown in Table 3.2.

$$\rho = \pi \cdot L_\lambda \cdot d^2 \cdot ESUN_\lambda \cdot cos\theta_s \tag{3.2}$$

where, L_λ is the radiance measure at the sensor, d is the distance to the sun, $ESUN_\lambda$ is the solar irradiance for the wavelength of the image band and θ is the solar zenith.

Table 3.2: Solar irradiance for the Landsat ETM+ bands.

Band	Solar Irradiance ($W \cdot m^{-3}$)
1	1997
2	1812
3	1533
4	1039
5	230.8
7	84.9
8	1362

3.2.1 Convert to Radiance

To convert to the landsat imagery to radiance RSGISLib provides the following command specifically for Landsat processing.

```
<!--
    A command to calibrate image Landsat data from at sensor
    DNs to at sensor radiance (EQ: ((lmax-lmin)/(qcalmax-qcalmin))
    * (DNs - qcalmin) + lmin) Eq is from landsat manual.
-->
<rsgis:command algor="imagecalibration" option="landsatradcal"
               output="image_out.env" format="GDAL Format" >
    <rsgis:band name="string" image="image1" band="int"
                [sensorband="string" |
                lmin="float" lmax="float" qcalmin="float"
                qcalmax="float"]/>
    <rsgis:band name="string" image="image1" band="int"
                [sensorband="string" |
```

```
                lmin="float" lmax="float" qcalmin="float"
                qcalmax="float"]/>
    <rsgis:band name="string" image="image1" band="int"
                [sensorband="string" |
                lmin="float" lmax="float" qcalmin="float"
                qcalmax="float"]/>
</rsgis:command>
```

Within the example XML given above the | is symbolising 'or' so either a string (as shown below) needs to be specified for the sensor band or the values need to be given. Therefore to convert Landsat data to radiance create an XML file containing the following, making sure the file name is the same at the file you are processing, needs to be created.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
    Description:
        XML File for execution within RSGISLib
        to pre-process Landsat ETM+ imagery.
    Created by Pete on Wed Nov 28 18:59:38 2012.
    Copyright (c) 2012 Aber Uni. All rights reserved.
-->

<rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
    <rsgis:command algor="imagecalibration" option="landsatradcal"
                   output="L7_20323_20000607_rad.kea" format="KEA" >
        <rsgis:band name="b1" image="L71203023_02320000617_B10.TIF"
                    band="1" sensorband="LETM7_B1" />
        <rsgis:band name="b2" image="L71203023_02320000617_B20.TIF"
                    band="1" sensorband="LETM7_B2" />
        <rsgis:band name="b3" image="L71203023_02320000617_B30.TIF"
                    band="1" sensorband="LETM7_B3" />
        <rsgis:band name="b4" image="L71203023_02320000617_B40.TIF"
                    band="1" sensorband="LETM7_B4" />
        <rsgis:band name="b5" image="L71203023_02320000617_B50.TIF"
                    band="1" sensorband="LETM7_B5" />
        <rsgis:band name="b7" image="L72203023_02320000617_B70.TIF"
                    band="1" sensorband="LETM7_B7" />
    </rsgis:command>
    <rsgis:command algor="imageutils" option="popimgstats"
                   image="L7_20323_20000607_rad.kea" ignore="0"
```

```
                    pyramids="yes" />
</rsgis:commands>
```

After the radiance calibration command you will notice there is an image utilities command called popimgstats. This command will calculate the image band statistics including the image histogram, minimum, maximum, mean, median, mode and standard deviation of the pixel values for each image band. It will also generate image pyramids, which allow fast visualisation of the image within a pyramid aware image viewer.

Running the XML commands

To run this script from a UNIX terminal you should run the rsgisexe command from within the same directory as your data (this XML file needs to be saved there as well) and it is executed with the following command:

```
rsgisexe -x PreProcessLandsat.xml
```

3.2.2 Convert to TOA

RSGISLib has a command to convert the radiance Landsat image to top of atmosphere (TOA) reflectance. The XML command is shown below, note this also includes the XML for the radiance image as well.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
    Description:
        XML File for execution within RSGISLib
        to pre-process Landsat ETM+ imagery.
    Created by Pete on Wed Nov 28 18:59:38 2012.
    Copyright (c) 2012 Aber Uni. All rights reserved.
-->

<rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
    <rsgis:command algor="imagecalibration" option="landsatradcal"
                   output="L7_20323_20000607_rad.kea" format="KEA" >
        <rsgis:band name="b1" image="L71203023_02320000617_B10.TIF"
                    band="1" sensorband="LETM7_B1" />
```

```
        <rsgis:band name="b2" image="L71203023_02320000617_B20.TIF"
                    band="1" sensorband="LETM7_B2" />
        <rsgis:band name="b3" image="L71203023_02320000617_B30.TIF"
                    band="1" sensorband="LETM7_B3" />
        <rsgis:band name="b4" image="L71203023_02320000617_B40.TIF"
                    band="1" sensorband="LETM7_B4" />
        <rsgis:band name="b5" image="L71203023_02320000617_B50.TIF"
                    band="1" sensorband="LETM7_B5" />
        <rsgis:band name="b7" image="L72203023_02320000617_B70.TIF"
                    band="1" sensorband="LETM7_B7" />
    </rsgis:command>
    <rsgis:command algor="imageutils" option="popimgstats"
                   image="L7_20323_20000607_rad.kea" ignore="0"
                   pyramids="yes" />
    <rsgis:command algor="imagecalibration" option="topatmosrefl"
                   input="L7_20323_20000607_rad.kea"
                   output="L7_20323_20000607_toa.kea"
                   format="KEA" scaleFactor="1000"
                   day="17" month="06" year="2000"
                   elevation="57.2241705" datatype="UInt16" >
        <rsgis:band sensorband="LETM7_B1" />
        <rsgis:band sensorband="LETM7_B2" />
        <rsgis:band sensorband="LETM7_B3" />
        <rsgis:band sensorband="LETM7_B4" />
        <rsgis:band sensorband="LETM7_B5" />
        <rsgis:band sensorband="LETM7_B7" />
    </rsgis:command>
    <rsgis:command algor="imageutils" option="popimgstats"
                   image="L7_20323_20000607_toa.kea" ignore="0"
                   pyramids="yes" />
</rsgis:commands>
```

The options to note are the use of a scale factor of 1000 (so each reflectance value is multiplied by 1000), giving a range from 0 to 1000 and the use of unsigned 16 bit integers to stored the outputted data. Following processing compare the files sizes of the radiance and TOA images, what do you notice?

To compare the file sizes list the directory using the 'ls' command with the '-lh' switches, as shown below.

```
ls -lh
```

Side Question

Why is the unsigned integer 16 bit image so much smaller than the 32 bit floating point image?[a]
Is any information / precision lost converting to an unsigned integer rather than using a floating point value?[b]

[a] A 16 bit value take up half the space of a 32 bit value but integers also compress more efficiently than floating point values as whole values are repeated

[b] No, multiplying by 1000 means than each increment (i.e., value of 1) is equal to 0.1% reflectance and this is well below the noise threshold of the system.

Running the XML commands

To run your commands just add these commands to your radiance XML script and then resrun your script using the rsgisexe command. Both the radiance and the TOA images will be calculated

3.2.3 Re-projecting to OSGB

To reproject the image data to the Ordnance Survey National Grid we need to use the command gdalwarp. GDAL is a software library which allows spatial located images to read and written where a common interface is provided for all the supported image formats. RSGISLib uses GDAL to read and write images and supports all the image formats that GDAL supports. GDAL also includes a number of command line tools for common tasks such as converting between image formats (gdal_translate) and warping images (gdalwarp). For more information on GDAL visit the website `http://www.gdal.org`.

To use gdalwarp the input and output image coordinate systems and projections need to be specified. There are a number of ways in which this can be done but the preferred method is through the 'Well-Known Text' (WKT) format. WKT files for the input projection (UTM 30N WGS84; utm30wgs83.wkt) and output projection (OSGB 36; osgb36.wkt) have been provided.

To use gdalwarp the following options are required (Note, the slash is used to allow the command to be split across multiple lines withn the shell script). See the GDAL website for a description of these options - you should get used to consulting online resources to understand how to do what you need to do.

```
gdalwarp -s_srs ./utm30wgs83.wkt -t_srs ./osgb36.wkt -ot UInt16 \
         -wt float32 -srcnodata 0 -order 3 -r cubic -of KEA \
         L7_20323_20000607_toa.kea L7_20323_20000607_toa_osgb.kea
```

3.2.4 Calculating Statistics and Image Pyramids

To generate the image pyramids and image statistics for the warped image generated by gdalwarp there is a command gdalcalcastats, confusingly this is not part of of the GDAL project and has been independently developed but is built on GDAL, hence the name.

```
gdalcalcstats L7_20323_20000607_toa_osgb.kea -ignore 0
```

3.2.5 Calculate an NDVI

To calculate an NDVI the RSGISLib software will again be used. For this a new XML file needs to be generated, to generate a blank XML file the rsgisexe command with the -b option can be used, as shown below.

```
rsgisexe -b CalcLandsatNDVI.xml
```

Using the new XML file add the following XML to calculate the NDVI using the bandmaths tool.

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--
3     Description:
4         XML File for execution within RSGISLib
5         to calculate the NDVI for a landsat 7 scene
6     Created by Pete on Wed Nov 28 21:11:44 2012.
7     Copyright (c) 2012 Aber Uni. All rights reserved.
8 -->
9
```

```
<rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
    <rsgis:command algor="imagecalc" option="bandmaths"
                   output="L7_20323_20000607_osgb_NDVI.kea" format="KEA"
                   datatype="Float32" expression="(NIR-Red)/(NIR+Red)" >
        <rsgis:variable name="Red"
                        image="L7_20323_20000607_toa_osgb.kea"
                        band="4" />
        <rsgis:variable name="NIR"
                        image="L7_20323_20000607_toa_osgb.kea"
                        band="3" />
    </rsgis:command>
</rsgis:commands>
```

3.2.6 Expanding the Processing to Multiple Scenes

To undertake this process on multiple scenes you have been provided with a template script for RSGISLib which contains all the processing stages. Note that the command line processing command has be used to call the gdalwarp command from within the RSGISLib XML script.

```
<!-- A command to execute a command line utilities (e.g., mkdir) -->
<rsgis:command algor="commandline" option="execute" command="string" />
```

The the template is shown below and the key thing to notice is the lack of specified filenames and variables, instead variables start '$' have been used and will be replaced with the true values at a later stage.

```
<rsgis:command algor="imagecalibration" option="landsatradcal"
               output="$PATH/$FILENAME1_rad.kea" format="KEA" >
    <rsgis:band name="b1" image="$FILEPATH1"
                band="1" sensorband="LETM7_B1" />
    <rsgis:band name="b2" image="$FILEPATH2"
                band="1" sensorband="LETM7_B2" />
    <rsgis:band name="b3" image="$FILEPATH3"
                band="1" sensorband="LETM7_B3" />
    <rsgis:band name="b4" image="$FILEPATH4"
                band="1" sensorband="LETM7_B4" />
    <rsgis:band name="b5" image="$FILEPATH5"
                band="1" sensorband="LETM7_B5" />
    <rsgis:band name="b7" image="$FILEPATH6"
```

```
14                band="1" sensorband="LETM7_B7" />
15  </rsgis:command>
16  <rsgis:command algor="imageutils" option="popimgstats"
17                 image="$PATH/$FILENAME1_rad.kea" ignore="0"
18                 pyramids="yes" />
19  <rsgis:command algor="imagecalibration" option="topatmosrefl"
20                 input="$PATH/$FILENAME1_rad.kea"
21                 output="$PATH/$FILENAME1_toa.kea"
22                 format="KEA" scaleFactor="1000"
23                 day="$VAR1" month="$VAR2" year="$VAR3"
24                 elevation="$VAR4" datatype="UInt16" >
25      <rsgis:band sensorband="LETM7_B1" />
26      <rsgis:band sensorband="LETM7_B2" />
27      <rsgis:band sensorband="LETM7_B3" />
28      <rsgis:band sensorband="LETM7_B4" />
29      <rsgis:band sensorband="LETM7_B5" />
30      <rsgis:band sensorband="LETM7_B7" />
31  </rsgis:command>
32  <rsgis:command algor="imageutils" option="popimgstats"
33                 image="$PATH/$FILENAME1_toa.kea" ignore="0"
34                 pyramids="yes" />
35  <rsgis:command algor="commandline" option="execute"
36                 command="gdalwarp -s_srs ./utm30wgs83.wkt -t_srs
37                          ./osgb36.wkt -ot UInt16 -wt float32
38                          -srcnodata 0 -order 3 -r cubic -of KEA
39                          $PATH/$FILENAME1_toa.kea
40                          $PATH/$FILENAME1_toa_osgb.kea" />
41  <rsgis:command algor="imageutils" option="popimgstats"
42                 image="$PATH/$FILENAME1_toa_osgb.kea" ignore="0"
43                 pyramids="yes" />
44  <rsgis:command algor="imagecalc" option="bandmaths"
45                 output="$PATH/$FILENAME1_osgb_NDVI.kea" format="KEA"
46                 datatype="Float32" expression="(NIR-Red)/(NIR+Red)" >
47      <rsgis:variable name="Red"
48                      image="$PATH/$FILENAME1_toa_osgb.kea"
49                      band="4" />
50      <rsgis:variable name="NIR"
51                      image="$PATH/$FILENAME1_toa_osgb.kea"
52                      band="3" />
53  </rsgis:command>
54  <rsgis:command algor="imageutils" option="popimgstats"
```

```
55                image="$PATH/$FILENAME1_osgb_NDVI.kea" ignore="0"
56                pyramids="yes" />
```

To understand what the different variables have been used for see the list below:

$PATH - This is the path with in the file system where the output files will be written.

$FILENAME1 - This is the start of the file name which all output files will have.

$FILEPATH1 - Band 1 of the input Landsat scene.

$FILEPATH2 - Band 2 of the input Landsat scene.

$FILEPATH3 - Band 3 of the input Landsat scene.

$FILEPATH4 - Band 4 of the input Landsat scene.

$FILEPATH5 - Band 5 of the input Landsat scene.

$FILEPATH6 - Band 7 of the input Landsat scene.

$VAR1 - Day of capture

$VAR2 - Month of capture

$VAR3 - Year of capture

$VAR4 - Solar Elevation

To replace these variables with the real data values the rsgislibcmdxml.py and rsgislibvarsxml.py scripts needs to be used as shown below.

```
1 # Create an output directory for the results
2 mkdir L7_20323_20000617_Outputs
3 # Add the file names and paths to the script
4 rsgislibcmdxml.py -i LandsatProcessingTemplate.xml \
5                   -o LandsatProcessingTemplate_Filenames.xml \
6                   -p L7_20323_20000617_Outputs \
7                   -b L7_20323_20000617 \
8                   -f LE72030232000169EDC00/L71203023_02320000617_B10.TIF \
9                   -f LE72030232000169EDC00/L71203023_02320000617_B20.TIF \
```

```
                    -f LE72030232000169EDC00/L71203023_02320000617_B30.TIF \
                    -f LE72030232000169EDC00/L71203023_02320000617_B40.TIF \
                    -f LE72030232000169EDC00/L71203023_02320000617_B50.TIF \
                    -f LE72030232000169EDC00/L72203023_02320000617_B70.TIF
# Add the variable values to the script.
rsgislibvarsxml.py -i LandsatProcessingTemplate_Filenames.xml \
                    -o L7_20323_20000617_PreProcessing.xml \
                    -v 17 \
                    -v 6 \
                    -v 2000 \
                    -v 57.2241705
```

After you have expand the shell script for generating the scripts for the other images you should have something which looks like this.

```
mkdir L7_20323_20000617_Outputs
rsgislibcmdxml.py -i LandsatProcessingTemplate.xml \
                    -o LandsatProcessingTemplate_Filenames.xml \
                    -p L7_20323_20000617_Outputs \
                    -b L7_20323_20000617 \
                    -f LE72030232000169EDC00/L71203023_02320000617_B10.TIF \
                    -f LE72030232000169EDC00/L71203023_02320000617_B20.TIF \
                    -f LE72030232000169EDC00/L71203023_02320000617_B30.TIF \
                    -f LE72030232000169EDC00/L71203023_02320000617_B40.TIF \
                    -f LE72030232000169EDC00/L71203023_02320000617_B50.TIF \
                    -f LE72030232000169EDC00/L72203023_02320000617_B70.TIF
rsgislibvarsxml.py -i LandsatProcessingTemplate_Filenames.xml \
                    -o L7_20323_20000617_PreProcessing.xml \
                    -v 17 \
                    -v 6 \
                    -v 2000 \
                    -v 57.2241705

mkdir L7_20323_20020404_Outputs
rsgislibcmdxml.py -i LandsatProcessingTemplate.xml \
                    -o LandsatProcessingTemplate_Filenames.xml \
                    -p L7_20323_20020404_Outputs \
                    -b L7_20323_20020404 \
                    -f LE72030232002094EDC00/L71203023_02320020404_B10.TIF \
                    -f LE72030232002094EDC00/L71203023_02320020404_B20.TIF \
                    -f LE72030232002094EDC00/L71203023_02320020404_B30.TIF \
```

```
                    -f LE72030232002094EDC00/L71203023_02320020404_B40.TIF \
                    -f LE72030232002094EDC00/L71203023_02320020404_B50.TIF \
                    -f LE72030232002094EDC00/L72203023_02320020404_B70.TIF
rsgislibvarsxml.py -i LandsatProcessingTemplate_Filenames.xml \
                    -o L7_20323_20020404_PreProcessing.xml \
                    -v 4 \
                    -v 4 \
                    -v 2002 \
                    -v 39.9992339

mkdir L7_20323_20020911_Outputs
rsgislibcmdxml.py -i LandsatProcessingTemplate.xml \
                    -o LandsatProcessingTemplate_Filenames.xml \
                    -p L7_20323_20020911_Outputs \
                    -b L7_20323_20020911 \
                    -f LE72030232002254SGS00/L71203023_02320020911_B10.TIF \
                    -f LE72030232002254SGS00/L71203023_02320020911_B20.TIF \
                    -f LE72030232002254SGS00/L71203023_02320020911_B30.TIF \
                    -f LE72030232002254SGS00/L71203023_02320020911_B40.TIF \
                    -f LE72030232002254SGS00/L71203023_02320020911_B50.TIF \
                    -f LE72030232002254SGS00/L72203023_02320020911_B70.TIF
rsgislibvarsxml.py -i LandsatProcessingTemplate_Filenames.xml \
                    -o L7_20323_20020911_PreProcessing.xml \
                    -v 11 \
                    -v 9 \
                    -v 2002 \
                    -v 39.1746567

mkdir L7_20323_20030218_Outputs
rsgislibcmdxml.py -i LandsatProcessingTemplate.xml \
                    -o LandsatProcessingTemplate_Filenames.xml \
                    -p L7_20323_20030218_Outputs \
                    -b L7_20323_20030218 \
                    -f LE72030232003049SGS00/L71203023_02320030218_B10.TIF \
                    -f LE72030232003049SGS00/L71203023_02320030218_B20.TIF \
                    -f LE72030232003049SGS00/L71203023_02320030218_B30.TIF \
                    -f LE72030232003049SGS00/L71203023_02320030218_B40.TIF \
                    -f LE72030232003049SGS00/L71203023_02320030218_B50.TIF \
                    -f LE72030232003049SGS00/L72203023_02320030218_B70.TIF
rsgislibvarsxml.py -i LandsatProcessingTemplate_Filenames.xml \
                    -o L7_20323_20030218_PreProcessing.xml \
```

```
68                  -v 18 \
69                  -v 2 \
70                  -v 2003 \
71                  -v 22.4606180
```

Finally, you run each of the XML scripts generated and using the rsgisexe command to process your imagery.

3.3 Linear Spectral Unmixing

RSGISLib provides commands for unconstrained and constrained linear unmixing which is commonly used to understand the proportion of endmembers contributing to the reflectance of each pixel. In the case of linear spectral unmixing then the combination is linear (i.e., additive) and provides the proportion of the reflectance of each input pixel corresponding the endmembers provided. The key to spectral unmixing is the selection of suitable endmembers which need to correspond with the extremes of the feature space of the data you are unmixing. You can define up to $n-1$ (n is the number of image bands) endmembers but is commonly less.

Using multispectral data within the UK, common endmembers are:

- Photosynthetic Vegetation
- Non-photosynthetic Vegetation
- Shade / Water

Therefore, unmixing for these three parameters will provide an output image with 3 image bands:

1. Proportion of photosynthetic vegetation
2. Proportion of non-photosynthetic vegetation
3. Proportion of shade and water

3.3.1 Defining End Members

There are a number of automated methods for defining end members (see literature) but in this case a manual selection of the end members is undertaken by defining a region (polygon) for each endmember using a shapefile (i.e., using QGIS or ArcMap).

If imagery is well corrected to surface reflectance with a standardised sun and view angle then a common set of endmembers can be used across a set of images but if imagery is poorly corrected (i.e., top of atmospheric reflectance or at sensor radiance) then end members need to be individually defined per image.

Once a set of polygons (one for each endmember has been define) then the following XML command with RSGISLib can be executed to generate a matrix file (.mtxt) where a single spectral profile is calculated for each region and saved as a matrix.

```
<!-- A command to extract the pixel values for
     regions to a matrix file as columns which
     can be used as endmembers for unmixing
-->
<rsgis:command algor="zonalstats" option="endmembers"
               image="image.env" vector="polygons.shp"
               output="output.mtxt" method="polyContainsPixel |
               polyContainsPixelCenter | polyOverlapsPixel |
               polyOverlapsOrContainsPixel | pixelContainsPoly |
               pixelContainsPolyCenter | adaptive | envelope" />
```

So for example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
    Description:
        XML File for execution within RSGISLib
    Created by **ME** on Sat Mar 30 18:12:47 2013.
    Copyright (c) 2013 **Organisation**. All rights reserved.
-->

<rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
    <rsgis:command algor="zonalstats" option="endmembers"
```

```
                    image="./L7ETM_530N035W_20100620_AtCor_osgb_masked.kea"
                    vector="./EndMembers.shp" output="./Endmembers"
                    method="polyContainsPixelCenter" />
</rsgis:commands>
```

3.3.2 Unmixing the Scene

Using the matrix file representing the end members the scene can be unmixed using either of the following XML commands.

```
<!-- A command to undertake an unconstrained linear spectral
     unmixing of the input image for a set of endmembers -->
<rsgis:command algor="imagecalc" option="unconlinearspecunmix"
               image="image.env" output="image" endmembers="matrix.mtxt"
               [gain="float" offset="float" format="GDAL Format"
               datatype="Byte | UInt16 | Int16 | UInt32 | Int32 |
                         Float32 | Float64"] />
<!-- A command to undertake a partially constrained linear spectral
     unmixing of the input image for a set of endmembers where the
     sum of the unmixing will be approximately 1 -->
<rsgis:command algor="imagecalc" option="consum1linearspecunmix"
               image="image.env" output="image" endmembers="matrix.mtxt"
               weight="float" [gain="float" offset="float" format="GDAL Format"
               datatype="Byte | UInt16 | Int16 | UInt32 | Int32 |
                        Float32 | Float64"] />
```

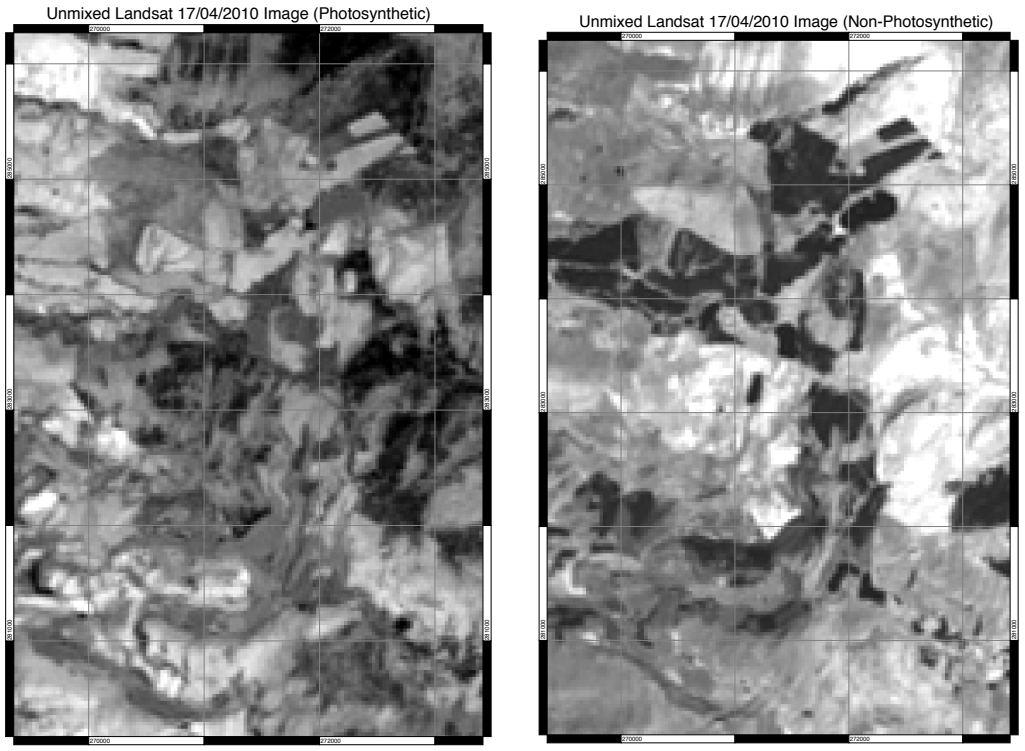
The first command is a completely unconstrained approach and will commonly produce results which are unrealistic as the combination (mixture) does not add up to 1 and some values could be negative which is of course impossible (a pixel cannot be made up of a negative amount of photosynthetic vegetation, for example). The second is partially constrained as the combination (mixture) must add up to 1 but does still allow negative values. There is also a version of least squares (the mathematical method used to solve the unmixing problem) which does not produce negative values (non-negative least squares) but this version is not yet working within RSGISLib.

Therefore, it is recommend that the partially constrained algorithm (which is the implementation ENVI uses) is used as shown below where as the imagery is at-

mospherically corrected the same endmembers have been used on multiple images.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
    Description:
        XML File for execution within RSGISLib
    Created by **ME** on Sat Mar 30 18:12:47 2013.
    Copyright (c) 2013 **Organisation**. All rights reserved.
-->

<rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
    <rsgis:command algor="zonalstats" option="endmembers"
                   image="./L7ETM_530N035W_20100620_AtCor_osgb_masked.kea"
                   vector="./EndMembers.shp" output="./Endmembers"
                   method="polyContainsPixelCenter" />
    <rsgis:command algor="imagecalc" option="consum1linearspecunmix"
                   image="./L7ETM_530N035W_20100620_AtCor_osgb_masked.kea"
                   output="L7ETM_530N035W_20100620_AtCor_osgb_masked_unmixed.kea"
                   endmembers="./Endmembers.mtxt" weight="35" format="KEA"
                   datatype="Float32" />
    <rsgis:command algor="imagecalc" option="consum1linearspecunmix"
                   image="./L7ETM_530N035W_20100417_AtCor_osgb_masked.kea"
                   output="L7ETM_530N035W_20100417_AtCor_osgb_masked_unmixed.kea"
                   endmembers="./Endmembers.mtxt" weight="35" format="KEA"
                   datatype="Float32" />
</rsgis:commands>
```



(a) (b)

Figure 3.1: Example of a landsat scene which has been unmixed using linear spectral unmixing.

Chapter 4

Examples – Image Registration

Chapter 5

Examples – Vectors

5.1 Zonal Statistics

Chapter 6

Examples - Image Segmentation

6.1 Iterative Elimination Algorithm

The segmentation algorithm (**?**) is based on generating spectrally similar units with a minimum object size.

The algorithm consists of a number of steps

1. Select image bands and stack images
2. Stretch image data
3. Find unique cluster within feature space (KMeans)
4. Assign pixels to clusters
5. Clump the image
6. Eliminate small segments

The KMeans clusters takes just a single image where all the bands are used as input so if multiple images are required to be inputted then they need to be stacked and the bands which are to be used selected. As a Euclidean distance is used within

the feature space the image is stretched such that all the pixel values are within the same range (i.e., 0–255).

A clustering algorithm is then used to identify the unique colours within the image, in this case a KMeans clustering is used but other clustering algorithms could also be used instead. The image pixels are then assigned to the clusters (classifying the image) and the image clumped to find the connected regions of the image.

The final step is an iterative elimination of the small segments, starting with the single pixels and going up to the maximum size of the segments specified by the user.

Therefore, there are two key parameters within the algorithm:

1. the number of cluster centres identified by the KMeans clustering
2. the minimum size of the segments

6.1.1 XML Code

```
<rsgis:command algor="imageutils" option="stretch" image="$FILEPATH"
               output="$PATH/$FILENAME_stretched.kea" ignorezeros="yes"
               stretch="LinearStdDev" stddev="2" format="KEA" />

<rsgis:command algor="imagecalc" option="bandmaths" output="$PATH/$FILENAME_mask.kea"
               format="KEA" expression="b1==0?0:1" >
        <rsgis:variable name="b1" image="$FILEPATH" band="1" />
</rsgis:command>

<rsgis:command algor="imageutils" option="mask"
               image="$PATH/$FILENAME_stretched.kea"
               mask="$PATH/$FILENAME_mask.kea"
               output="$PATH/$FILENAME_stretched_masked.kea"
               maskvalue="0" outputvalue="0" format="KEA" />

<rsgis:command algor="commandline" option="execute"
               command="rm $PATH/$FILENAME_mask.kea" />
<rsgis:command algor="commandline" option="execute"
               command="rm $PATH/$FILENAME_stretched.kea" />

```

```
<rsgis:command algor="imagecalc" option="kmeanscentres"
               image="$PATH/$FILENAME_stretched_masked.kea"
               output="$PATH/$FILENAME_clusters" numclusters="60" maxiterations="200"
               degreeofchange="0.25" subsample="1" initmethod="diagonal_range_attach" />

<rsgis:command algor="segmentation" option="labelsfromclusters"
               image="$PATH/$FILENAME_stretched_masked.kea"
               output="$PATH/$FILENAME_clusters.kea"
               clusters="$PATH/$FILENAME_clusters.gmtxt"
               ignorezeros="yes" format="KEA" proj="IMAGE" />

<rsgis:command algor="segmentation" option="elimsinglepxls"
               image="$PATH/$FILENAME_stretched_masked.kea"
               clumps="$PATH/$FILENAME_clusters.kea"
               temp="$PATH/$FILENAME_clusters_singlepxls_tmp.kea"
               output="$PATH/$FILENAME_clusters_nosinglepxls.kea"
               ignorezeros="yes" format="KEA" proj="IMAGE" />

<rsgis:command algor="commandline" option="execute"
               command="rm $PATH/$FILENAME_clusters.kea" />
<rsgis:command algor="commandline" option="execute"
               command="rm $PATH/$FILENAME_clusters_singlepxls_tmp.kea" />

<rsgis:command algor="segmentation" option="clump"
               image="$PATH/$FILENAME_clusters_nosinglepxls.kea"
               output="$PATH/$FILENAME_clumps.kea" nodata="0"
               format="KEA" inmemory="no" proj="IMAGE" />

<rsgis:command algor="commandline" option="execute"
               command="rm $PATH/$FILENAME_clusters_nosinglepxls.kea" />

<rsgis:command algor="segmentation" option="rmsmallclumpsstepwise"
               image="$PATH/$FILENAME_stretched_masked.kea"
               clumps="$PATH/$FILENAME_clumps.kea"
               output="$PATH/$FILENAME_clumps_elim.kea"
               minsize="50" maxspectraldist="200000"
               format="KEA" inmemory="no" proj="IMAGE" />

<rsgis:command algor="commandline" option="execute"
               command="rm $PATH/$FILENAME_stretched_masked.kea" />
<rsgis:command algor="commandline" option="execute"
```

```
62                command="rm $PATH/$FILENAME_clumps.kea" />
63
64  <rsgis:command algor="segmentation" option="relabelclumps"
65                image="$PATH/$FILENAME_clumps_elim.kea"
66                output="$PATH/$FILENAME_clumps_elim_final.kea"
67                format="KEA" inmemory="no" proj="IMAGE" />
68
69  <rsgis:command algor="commandline" option="execute"
70                command="rm $PATH/$FILENAME_clumps_elim.kea" />
71
72  <rsgis:command algor="segmentation" option="meanimg"
73                image="$FILEPATH" clumps="$PATH/$FILENAME_clumps_elim_final.kea"
74                output="$PATH/$FILENAME_clumps_elim_mean.kea"
75                format="KEA" inmemory="no" proj="IMAGE" />
76
77  <rsgis:command algor="imageutils" option="popimgstats"
78                image="$PATH/$FILENAME_clumps_elim_mean.kea" ignore="0" pyramids="yes" />
```

To use the script provided you need to use the rsgislibxml.py command which replaces the $FILEPATH with the file path of the input image (found by rsgislibxml.py within the input directory) $PATH with the provided directory path and $FILENAME with the name of the input file. An example of this command is given below:

```
rsgislibxml.py -i RunSegmentationTemplate.xml \
               -o Segmentation.xml -p ./Segments \
               -d ./Data/ -e .kea -r no -t single
```

Once the command above has been executed then the segmentation can be run using the rsgisexe command:

```
rsgisexe -x Segmentation.xml
```

The resulting segmentation will have produced 3 output files

1. *clusters.gmtxt – Cluster centres.
2. *clumps_elim_final.kea – Segment clumps.
3. *clumps_elim_mean.kea – Mean colour image using segments.

Following the segmentation the it is recommend that you make sure that the clumps file is defined as a thematic file, as demonstrated in the following piece of

python:

```
#!/usr/bin/env python

import sys
from osgeo import gdal

ds = gdal.Open(sys.argv[1], gdal.GA_Update)
for bandnum in range(ds.RasterCount):
    band = ds.GetRasterBand(bandnum + 1)
    band.SetMetadataItem('LAYER_TYPE', 'thematic')
```

Finally, use the gdalcalcstats command to populate the image with an attribute table, histogram and colour table (set -ignore 0 as 0 is the background no data value).

```
setthematic.py L7ETM_530N035W_clumps_elim_final.kea
gdalcalcstats L7ETM_530N035W_clumps_elim_final.kea -ignore 0
```

Chapter 7

Examples – Raster GIS

7.1 Populating Segment Statistics

To populate the segments with statistics (i.e., Mean for each spectral band) there is a command with the rastergis part of the RSGISLib software. Examples of this are shown within the XML code below, note the text given for each band is the names of the output columns.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
    Description:
        XML File for execution within RSGISLib
    Created by **ME** on Thu Mar 21 09:25:21 2013.
    Copyright (c) 2013 **Organisation**. All rights reserved.
-->

<rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">

    <rsgis:command algor="rastergis" option="popattributestats"
                   clumps="L7ETM_530N035W_Classification.kea"
                   input="L7ETM_530N035W_20100417_AtCor_osgb_masked.kea" >
        <rsgis:band band="1" mean="MayBlue" stddev="MaySDBlue" />
        <rsgis:band band="2" mean="MayGreen" stddev="MaySDGreen" />
        <rsgis:band band="3" mean="MayRed" stddev="MaySDRed" />
        <rsgis:band band="4" mean="MayNIR" stddev="MaySDNIR" />
```

```
        <rsgis:band band="5" mean="MaySWIR1" stddev="MaySDSWIR1" />
        <rsgis:band band="6" mean="MaySWIR2" stddev="MaySDSWIR2" />
    </rsgis:command>

    <rsgis:command algor="rastergis" option="popattributestats"
                   clumps="L7ETM_530N035W_Classification.kea"
                   input="L7ETM_530N035W_20100620_AtCor_osgb_masked.kea" >
        <rsgis:band band="1" mean="JuneBlue" stddev="JuneSDBlue" />
        <rsgis:band band="2" mean="JuneGreen" stddev="JuneSDGreen" />
        <rsgis:band band="3" mean="JuneRed" stddev="JuneSDRed" />
        <rsgis:band band="4" mean="JuneNIR" stddev="JuneSDNIR" />
        <rsgis:band band="5" mean="JuneSWIR1" stddev="JuneSDSWIR1" />
        <rsgis:band band="6" mean="JuneSWIR2" stddev="JuneSDSWIR2" />
    </rsgis:command>

    <rsgis:command algor="rastergis" option="popattributestats"
                   clumps="L7ETM_530N035W_Classification.kea"
                   input="Nant_y_Arian_DEM_30m.kea" >
        <rsgis:band band="1" min="MinDEM" mean="MaxDEM"
                             mean="MeanDEM" stddev="StdDevDEM" />
    </rsgis:command>

</rsgis:commands>
```

If you are going to use a indices and other derived information within your classification it is quite often a good idea to set up a python script to calculate those indices and write them back to the image rather than over complicating your classification script. An example of this is shown below.

```
#!/usr/bin/env python

import sys
from rios import rat
import numpy
import osgeo.gdal as gdal


#Input file.
fname = "L7ETM_530N035W_Classification.kea"
ratDataset = gdal.Open( fname, gdal.GA_Update )

```

```
13 print "Import Columns."
14 MayBlue = rat.readColumn(ratDataset, "MayBlue")
15 MayGreen = rat.readColumn(ratDataset, "MayGreen")
16 MayRed = rat.readColumn(ratDataset, "MayRed")
17 MayNIR = rat.readColumn(ratDataset, "MayNIR")
18 MaySWIR1 = rat.readColumn(ratDataset, "MaySWIR1")
19 MaySWIR2 = rat.readColumn(ratDataset, "MaySWIR2")
20
21 JuneBlue = rat.readColumn(ratDataset, "JuneBlue")
22 JuneGreen = rat.readColumn(ratDataset, "JuneGreen")
23 JuneRed = rat.readColumn(ratDataset, "JuneRed")
24 JuneNIR = rat.readColumn(ratDataset, "JuneNIR")
25 JuneSWIR1 = rat.readColumn(ratDataset, "JuneSWIR1")
26 JuneSWIR2 = rat.readColumn(ratDataset, "JuneSWIR2")
27
28 MeanDEM = rat.readColumn(ratDataset, "MeanDEM")
29
30 MayNIR.astype(numpy.float32)
31 MayRed.astype(numpy.float32)
32 JuneNIR.astype(numpy.float32)
33 JuneRed.astype(numpy.float32)
34 MayBlue.astype(numpy.float32)
35 JuneBlue.astype(numpy.float32)
36
37 print "Calculate Indices."
38 MayNDVI = (MayNIR - MayRed) / (MayNIR + MayRed)
39 JuneNDVI = (JuneNIR - JuneRed) / (JuneNIR + JuneRed)
40
41 MayWBI = MayBlue/MayNIR
42 JuneWBI = JuneBlue/JuneNIR
43
44 rat.writeColumn(ratDataset, "MayNDVI", MayNDVI)
45 rat.writeColumn(ratDataset, "JuneNDVI", JuneNDVI)
46 rat.writeColumn(ratDataset, "MayWBI", MayWBI)
47 rat.writeColumn(ratDataset, "JuneWBI", JuneWBI)
```

Chapter 8

TODO

Chapter 9

TODO

Chapter 10

TODO