Introduction to ARCSI for generating Analysis Ready Data (ARD)

Pete Bunting
Aberystwyth University
Earth Observation and Ecosystem Dynamics Group
Department of Geography and Earth Sciences
pfb@aber.ac.uk

ACSEM

PRIFYSGOL ABERYSTWYTH UNIVERSITY

Contents

Chapter 1

Introduction

This course aims to provide an introduction to the Atmospheric and Radiometric Correction of Satellite Imagery (ARCSI) software developed by Pete Bunting (pfb@aber.ac.uk). ARCSI aims to be provide as automatic as possible processing chain for generating analysis ready earth observation imagery from optical sensors (e.g., Landsat, Sentinel-2, Rapideye etc.). The primary task for this analysis is the application of an atmospheric correction for which the the 6S model (Vermote et al., 1997) is used and accessed through the python interface Py6S (Wilson, 2013).

1.1 Software

ARCSI is build on top of a number of other packages:

1. Remote Sensing and GIS Software Library (RSGISLib; `http://www.rsgislib.org`)
2. Geospatial Data Abstraction Library (GDAL; `http://www.gdal.org`)
3. Py6S (`https://py6s.readthedocs.io`)
4. KEA file format (`http://www.kealib.org`)
5. python-fmask (`http://pythonfmask.org`)
6. RIOS (`http://www.rioshome.org`)
7. scikit-learn (`http://scikit-learn.org`)

1.1.1 RSGISLib

The Remote Sensing and GIS Software Library (RSGISLib; Bunting et al., 2014) was originally designed to just provide the functionality we required for our own research, where it wasn't available in existing software packages. However, RSGISLib has evolved into a set of Python modules providing a wide range of functionality which is scriptable, creating a flexible system for data analysis and batch processing. The available module are:

- Image Calibration
- Classification
- Elevation
- Image Calculations
- Image Filtering
- Image Morphology
- Image Registration
- Image Utilities
- Histogram Cube
- Raster GIS
- Image Segmentation
- Tools
- Vector Utilities
- Zonal Statistics

1.1.2 GDAL

At its core the Geospatial Data Abstraction Library (GDAL; `http://www.gdal.org`) provides software to convert between many image file format but it has been extended to provide a set of utilities for processing image data. The most useful utilities are:

- `gdalinfo` – report information about a file.
- `gdal_translate` – Copy a raster file, with control of output format.
- `gdalwarp` – Warp an image into a new coordinate system.
- `gdaladdo` – Add overviews to a file.
- `gdalbuildvrt` – Build a Virtual Raster (VRT) from a list of datasets.

- `gdal_contour` – Contours from DEM.
- `gdaldem` – Tools to analyse and visualise DEMs.
- `gdal_merge.py` – Build a quick mosaic from a set of images.
- `gdal_rasterize` – Rasterise vectors into raster file.
- `gdal_proximity.py` – Compute a raster proximity map.
- `gdal_polygonize.py` – Generate polygons from raster.
- `gdal_sieve.py` – Raster Sieve filter.
- `gdal_fillnodata.py` – Interpolate to fill no-data regions.
- `gdalmanage` – Identify, copy, rename and delete raster.
- `gdalcompare.py` – Compare two images and report on differences.

1.1.3 Py6S

The Py6S module (Wilson, 2013) provides an easy to use and convenient way in which to run the 6S radiative transfer model (Vermote et al., 1997) from within the python scripting language. This model is used to model the atmospheric component of the reflectance to perform the atmospheric correction.

1.1.4 KEA file format

The KEA file format developed by Bunting and Gillingham (2013) and named after the New Zealand bird (Kea) is a HDF5 based image file format with a GDAL driver. Therefore, the format can be used in any software using GDAL, provided the KEA library is available. It offers support for large raster attribute tables and uses zlib based compression to provide small file sizes. The development was funded and supported by Landcare Research, New Zealand.

1.1.5 Python-FMask

Python-FMask (`http://pythonfmask.org`) is a python implementation of the FMask cloud masking algorithm (Zhu and Woodcock, 2014; Zhu et al., 2015) for Landsat 4 TM, 5 TM, 7 ETM+ and 8 and Sentinel-2. It would be noted that this cloud masking approach is much less reliable for Sentinel-2 than it is for the Landsat sensors due to the availability of a thermal channel.

1.1.6 RIOS

The Raster Input and Output (I/O) simplification (RIOS; Gillingham and Flood, 2013, `http://www.rioshome.org`) library is a set of Python modules which makes it easier to write raster processing code in Python. Built on top of GDAL, it handles the details of opening and closing files, checking alignment of projections and raster grid, stepping through the raster in small blocks, etc., allowing the programmer to concentrate on implementing the solution to the problem rather than on how to access the raster data and detail with the spatial header. It also offers functions for reading and writing Raster Attribute Tables.

1.1.7 scikit-learn

Scikit-learn (Pedregosa et al., 2011) is a library of machine learning algorithms accessible from within Python. Scikit-learn provides functionality under the following headings.

- Classification
- Regression
- Clustering
- Dimensionality reduction
- Model selection
- Preprocessing

RSGISLib uses scikit-learn to solve a number of classification and clustering problems.

1.2 Computing Environment

The software used for this course are accessed from the command line. Whilst GUI tools are easier to learn, if you are not familiar with the command line, the command line is more powerful for batch processing and/or creating processing chains. Although the software listed is cross-platform (e.g., Windows, Linux, OS X), ARCSI is currently only tested and extensively used under UNIX-like operating systems (MacOS and Linux). Building tools using the UNIX environment scales well to HPCs, the majority of which run a UNIX-like operating system.

1.3 Software Installation

Binary downloads are made available through the conda package management system (`http://conda.pydata.org`), which allows multiple 'environments' to be created. With conda, you can create, export, list, remove, and update environments that have different versions of Python and/or packages installed in them. Switching or moving between environments is called activating an environment. You can even share an environment file with a coworker. See the Conda documentation: `http://conda.pydata.org/docs/using/envs.html`

1.3.1 MacOS and Linux

Install Conda

To install the software you need for this worksheet you need to first install conda on either MacOS or Linux, via using the miniconda installation package (`http://conda.pydata.org/miniconda.html`) – you require the Python 3.5, 64-bit version. You will have downloaded either:

- `Miniconda3-latest-MacOSX-x86_64.sh` – MacOS
- `Miniconda3-latest-Linux-x86_64.sh` – Linux

If you already have conda installed then do not install it again but go to section 1.3.1, you can check by running:

```
conda --version
```

If a version is printed out to the terminal then it is installed, otherwise an error will be provided stating that the `conda` command is not available.

Once downloaded, open a Terminal window (e.g., Figure 1.1) and navigate to where the downloaded file is stored on your system (e.g., `cd Downloads`). If you are not familiar with a UNIX terminal see Appendix A.

Once you have navigated to file, you will need to change permissions so it can be executed:

```
chmod a+x ./Miniconda3-latest-MacOSX-x86_64.sh
```

and then run is as shown below, by typing the name of the script:

```
./Miniconda3-latest-MacOSX-x86_64.sh
```

It will ask a number of questions:

Figure 1.1: A terminal window.

1. Do you want to install Miniconda – `Press Enter`
2. It'll show you the license, press `spacebar` to scroll through. Type `Yes` to confirm you agree to the license.
3. Confirm the installation location, the default is your home directory and it is suggested that you use this. – `Press Enter`
4. It will ask if it can append the miniconda installation to your `PATH`. The default is Yes, therefore – `Press Enter`

Conda should now be installed, close all your Terminal windows and open a new one, you can test conda is working by running the following command:

```
conda --version
```

Setup Conda Environment

To install the package you require for this worksheet, you should now run the following commands:

First, setup an environment (`osgeo`) for this worksheet:

```
conda create --name osgeo-env-v1 python=3.5
```

It will install a set of default packages, just agree (press `y`). To activate the environment you run the following command, you will need to do that each time you open a new Terminal.

```
source activate osgeo-env-v1
```

You will be able to tell which environment you are using as to the left of your command prompt it will name it, `(osgeo-env-v1)`, in brackets; for example:

```
(osgeo-env-v1) Petes-MacBook-Pro:Downloads pete$
```

Now you have an environment, you are ready to install the software. It is good practice to create a new environment for each project you undertaken, then you can ensure that the appropriate version of each package is maintained throughout the project – different projects can require different versions of packages.

Run the following to install the software you require:

```
conda install -c conda-forge arcsi
conda install -c conda-forge python-fmask
conda install -c conda-forge tuiview
conda install -c conda-forge scikit-learn matplotlib h5py
```

When these are installed they will also download the dependencies those software require, therefore agree to the messages requesting that other packages are also installed.

If there are any problems with your installation (e.g., a missing package) then this can often be resolved by updating you system:

```
conda update -c conda-forge --all
```

To check that the software has been installed, run the following:

```
arcsi.py --version
```

You should have at least version `3.1`. Also, run tuiview, which will start the viewer (Figure 1.2).

```
tuiview
```

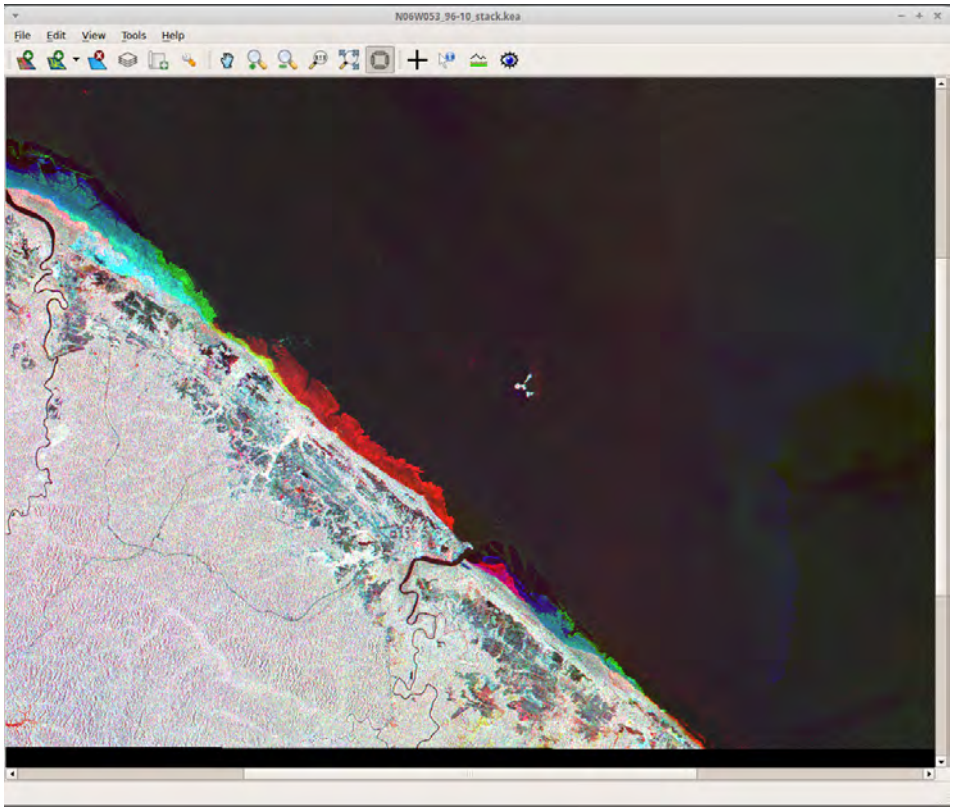

Figure 1.2: The TuiView image viewer.

1.3.2 Windows

While you can install RSGISLib and the other tools on Windows, RSGISLib is primarily tested on Linux and MacOS therefore we recommend using one these of systems. On Windows 10 you can install Linux alongside Windows, please note this does not provide a graphic interface so TuiView will need to be installed, via Conda, under Windows rather than through the Linux installation. Alternatively, you can install a Linux within Windows using a virtual machine. VirtualBox is a free and open source package which enables this but VMWare is a commercial alternative.

Chapter 2

Getting Help / Documentation

2.1 Online Documentation

Online documentation for the latest version of ARCSI is provided on the website (`http://www.rsgislib.org/arcsi`) and should be your first point of call.

2.2 Blog

You will also have the online blog `https://spectraldifferences.wordpress.com` a useful source of examples of specific problems and data types.

2.3 Mailing List

We have a mailing list `https://groups.google.com/forum/#!forum/rsgislib-support` where you can communicate with others using ARCSI and associated tools, such as RSGISLib, if you have specific questions or think there is bug or problem with the software. We do our best to answer emails on this list in a prompt manner. To post (`rsgislib-support@googlegroups.com`) on this mailing list you first need to register.

2.4 Code Repository

If you would like to see the ARCSI source code or submit to our issues list this is done through the bitbucket service, `https://bitbucket.org/petebunting/arcsi`.

Chapter 3

Getting Started

3.1 Background

3.1.1 Radiance

Optical data recorded in a particular wavelength region λ and obtained from the data provider, should be given in units of radiance ($L_\lambda W\ m^{-2} sr^{-1} \mu^{-1}$). In order to compress (i.e., reduce the file size), the image is typically provided with a gain and offset to convert the pixel value, commonly referred to as the digital number (DN), to radiance where:

$$L_\lambda = (\text{gain} \times \text{DN}) + \text{offset} \tag{3.1}$$

One of the first steps in processing that ARCSI will undertaken is to transform the input image into 'at sensor radiance'.

3.1.2 At Sensor Reflectance

At sensor reflectance, also referred to as top of atmosphere (TOA) reflectance, is a standard and easily calculated ratio (Equation 3.2) of the incoming radiant energy (light) from the sun (ESUN) and the corresponding radiance measured by the sensor. The radiance measured at the sensor differs from the incoming signal due to the reflectance of the Earth surface and the atmosphere (or part of the atmosphere) the signal has transmitted through. Although providing a standard measure and common range of values (0 - 1), the reflectance measurement includes the reflectance from the atmosphere and the ground surface and therefore images taken at different times are not directly comparable. At sensor reflectance is calculated as:

$$\rho = \frac{\pi \cdot L_\lambda \cdot d^2}{\text{ESUN}_\lambda \cdot \cos(\theta_s)} \tag{3.2}$$

where λ is the wavelength, ρ_λ is the spectral (planetary or TOA) reflectance for wavelength λ, L_λ is the spectral radiance (W m^{-2} sr^{-1} μm^{-1}) d is the Earth-Sun distance in astronomical units, ESUN is the mean solar exoatmospheric irradiance in units of W m^{-2} μm^{-1}, and $\cos(\theta_s)$ is the solar zenith angle.

3.1.3 Surface Reflectance

Surface reflectance, also called 'bottom of atmosphere reflectance' is the ratio of incoming radiance (i.e., from the sun) with the radiance that is measured by the sensor without *any* atmospheric effect and should be equivalent to the signal measured if the sensor was at ground level or there was no atmosphere. To derive this measurement, the effect of the atmosphere needs to be removed from the at sensor radiance measured at the sensor. There are a number of options for this:

- Empirical Line Calibration
- Dark Object Subtraction
- Modelling Atmospheres

The Empirical Line Calibration (Smith and Milton, 2010) is commonly used to correct high-resolution airborne imagery but requires that ground data of bright and dark targets be captured at the time of the overflight. Dark Object Subtraction methods (Chavez, 1996) are relatively simple and require relatively little inputs so can be easily applied to all image data but do not produce the most reliable and consistent results. It is, therefore, the method used when the others are not available. Modelled Atmospheric Correction Methods (Vermote et al., 1997) model reflection, absorption and scattering by the atmosphere and commonly used models include 6S (Vermote et al., 1997), LOWTRAN, MODTRAN, FLAASH, ATCOR and HYCOR. These models require many parameters to be known or estimated and can, therefore, be complex to apply. However, for lower resolution imagery or where ground spectra for targets are not available, it is the best solution.

The following flowchart (Figure 3.1) provides a recommended decision process you can go through to decide the method of atmospherical correction you should follow with you imagery.

ARCSI provides tools for undertaking a dark objective subtraction and creating and applying a modelled atmosphere to earth observation data.

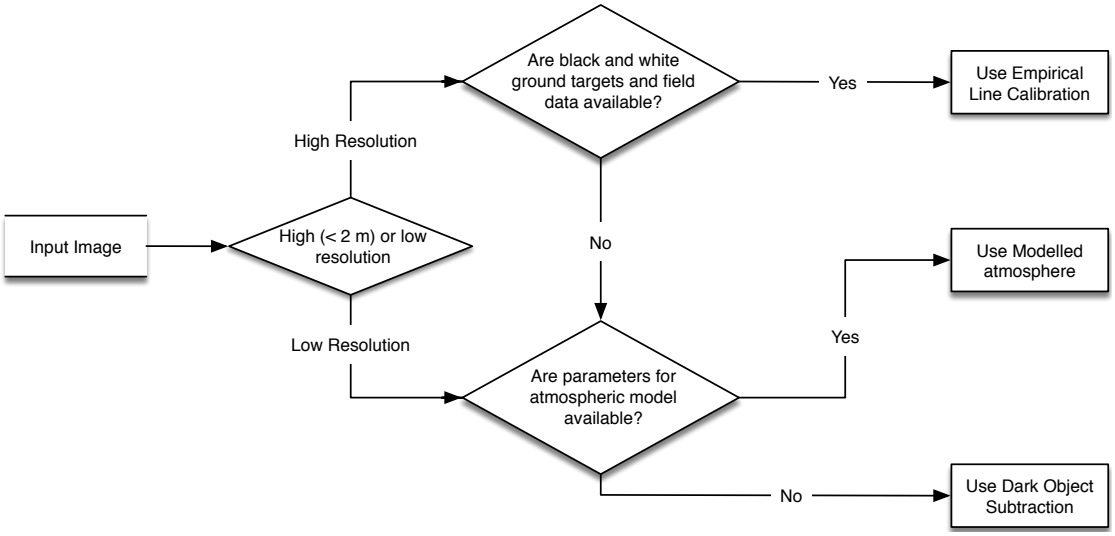

Figure 3.1: Decision flowchart outlining suggesting a method of deriving a surface reflectance product.

3.1.4 Standardised Reflectance

The method of Shepherd and Dymond (2003) was implemented to provide a standardised reflectance product. Standardised reflectance refers to a product which is normalised for the solar and sensor view angles and in this case topography. In terms of providing a topographic correction, this method only works for images where the solar elevation is between 50 and 70 degrees (i.e., from late spring to early autumn) but does not produce artefacts outside of this range. Standardised reflectance is defined using the following equation

$$\rho_h^{dir} = \frac{\pi L}{\frac{E^{dif}}{\gamma} + \beta E^{dif}} \tag{3.3}$$

where ρ_h^{dir} is the direct reflectance for a horizontal surface, L is the radiance at the bottom of the atmosphere, E^{dir} is the direct component of the irradiance and E^{dif} is the diffuse component, the ratio β is evaluated using a bidirectional reflectance distribution model but in the study a constant of 1 was used (Shepherd and Dymond, 2003):

$$\gamma = \frac{cos(i) + cos(e)}{cos(i_h) + cos(e_h)} \tag{3.4}$$

where i and e are the incidence, and exitance angles on an inclined surface, and i_h and e_h are the incidence and exitance angles on a horizontal surface (Figure 3.2).

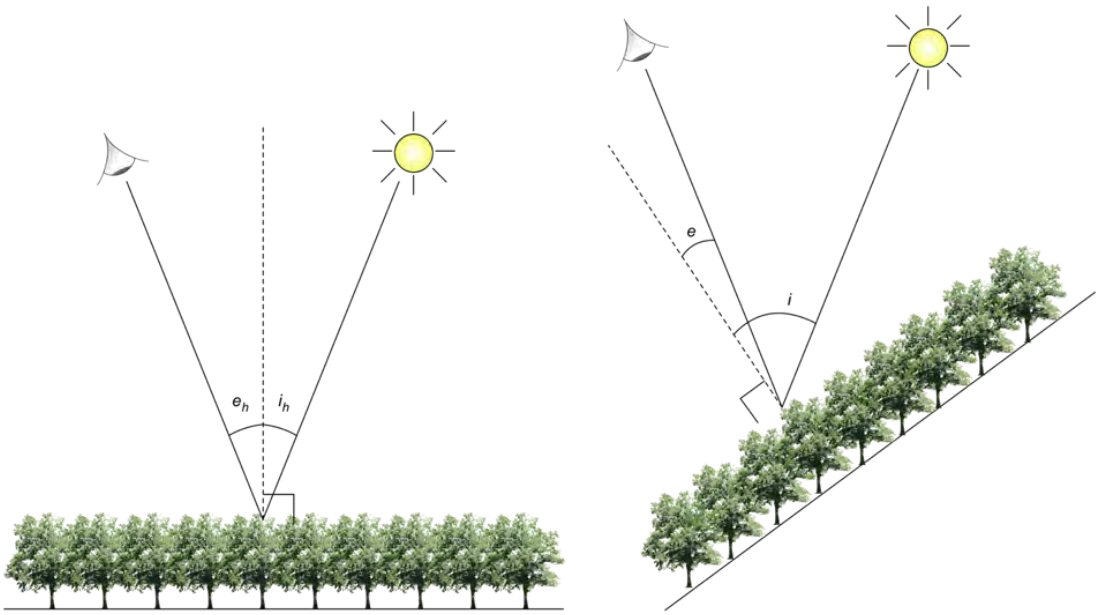

Figure 3.2: Standardised reflectance normalised for topographic and view angle (adapted and redrawn from Shepherd and Dymond (2003)

3.2 6S Parameters

To calculate surface reflectance the 6S radiative transfer model will be used. The 6S model has a number of inputs which need to be specified, specifically

- Atmospheric Profile
- Aerosol Profile
- Ground Reflectance
- Geometry
- Altitude
- Wavelength
- 6S Outputs

For detailed information on the parameterisation of 6S visit the Py6S website `http://py6s.readthedocs.org/en/latest/params.html`.

3.2.1 Atmospheric Profile

A number of standard profiles are available NoGaseousAbsorption, Tropical, MidlatitudeSummer, MidlatitudeWinter, SubarcticSummer, SubarcticWinter and USStandard1962. These can all be specified via the terminal within ARCSI. Additionally, user specified values of water and ozone within the vertical path within the atmosphere can also be specified. The amount of water within the vertical water column has the largest effect of the SWIR channels.

3.2.2 Aerosol Profile

The aerosol profile and particularly the aerosol optical depth (AOD) / aerosol optical thickness (AOT) make a significant difference to the reflectance of the visible bands following atmospheric correction.

A number of standard profiles are available NoAerosols, Continental, Maritime, Urban, Desert, BiomassBurning and Stratospheric. These are all available within ARCSI, from the terminal (`arcsi.py`). Alternatively, a user defined set of values can be specified for the proportion of dust, water, oceanic and soot like aerosols. The total of the proportions needs to add up to 1, for example oceanic = 0.75 and dust = 0.25 or oceanic = 0.35, water = 0.35 and dust = 0.30.

Additionally, the AOT also needs to be specified. This can be specified as a single value to be used across the whole scene via the ARCSI command line utility. The amount of AOT has a significant effect of the visual reflectance of the Landsat scene during atmospheric correction.

3.2.3 Ground Reflectance

The reflectance of a ground target needs to be specified for the model to be executed on. The ground surface can either be a Lambertian or bidirectional reflectance distribution function (BRDF) surface. 6S has a number of implemented BRDF models but they need to be parameterised. Within ARCSI the standard Lambertian surface targets Green Vegetation, Clear Water, Sand and Lake Water are available.

3.2.4 Geometry

This is the geometry of the sensor and is specified by ARCSI from the header file(s) associated with the input image.

3.2.5 Altitude

This is the altitude of the sensor and ground surface. Within ARCSI only the approximate altitude of the ground surface needs to be specified and the sensor altitude is already defined internally. The surface altitude can also be provided via the inclusion of a DEM image file as a command to ARCSI. Where a DEM is provided ARCSI will subset, resample and if needed reproject the DEM is match the pixels of the inputted image. Surface altitude is important is it defines the depth of the atmosphere and therefore the the effect the atmosphere has on the signal (Figure 3.3).

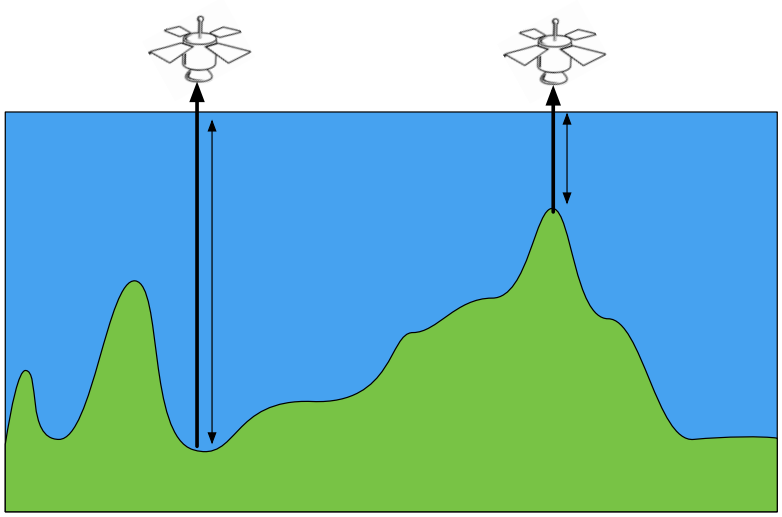

Figure 3.3: Sensor Response functions for the WorldView-2 sensor.

3.2.6 Wavelength

The wavelength is the response function of the sensor band being corrected (e.g., Figure 3.4). ARCSI specifies the response function(s) for each wavelength of the sensor.

3.2.7 6S Outputs

6S returns a set of coefficients, which are printed to the console during execution of ARCSI, for converting at sensor radiation to surface reflectance. The 6S coefficients aX, bX and cX are applied to the input image using the following equations:

$$y = aX \times (\text{measured radiance}) - bX \tag{3.5}$$

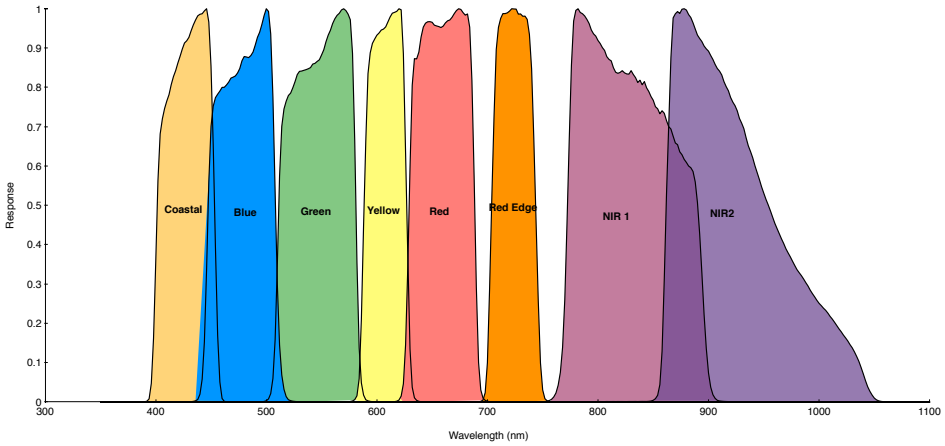

Figure 3.4: Sensor Response functions for the WorldView-2 sensor.

$$\text{surface reflectance} = \frac{y}{1.0 + cX \times y} \tag{3.6}$$

Chapter 4

Starting with ARCSI

ARCSI supports a range of sensors but this tutorial will focus on Landsat and Sentinel-2. The majority of examples will be given using Landsat data but these can be changed to Sentinel-2 by just updating the sensor specified.

ARCSI is executed using the command line tool `arcsi.py`. If you run the command without any options then some example commands will be presented. To find help on the options available you can run `arcsi.py -h`.

4.1 Getting a dataset

However, before running arcsi you will need an image to test against. I have provided the image `LT05_L1TP_203024_19950815_20180217_01_T1.tar.gz` covering an areas of South Wales and South West England including the city of Bristol.

Before running ARCSI you need to extract the Landsat scene from the `tar.gz` archive file. You could use the command

```
tar -zxf LT05_L1TP_203024_19950815_20180217_01_T1.tar.gz
```

or

```
arcsiextractdata.py -f ./LT05_L1TP_203024_19950815_20180217_01_T1.tar.gz -o ./
```

to extract the contents to the current directory but it would be good to have a the contents of the `tar.gz` file extracted into a directory for that scene. ARCSI provides a command to do that `arcsiextractdata.py`. The command operates on all archive files that data is likely to be provided. While is can be applied to a single file it also has the

option (shown below) to be applied to an input directory (in this case RAW) where all the archives within the input directory will be extracted to the output directory.

```
arcsiextractdata.py -i ./RAW/ -o Inputs/
```

To use ARCSI to generate a surface reflectance product (alongside top of atmosphere and radiance) the simplest method is to specify the parameters via the command line as shown below:

```
arcsi.py -s ls5tm -f KEA --stats -p RAD TOA SREF --aot 0.25 \
--surfacealtitude 0.4 -o ./Outputs \
-i LT05_L1TP_203024_19950815_20180217_01_T1/LT05_L1TP_203024_19950815_20180217_01_T1_MTL.txt
```

-s ls5tm specifies that it is a Landsat 5 TM scene which is being processed.

-f KEA specifies that the output image file format should be KEA. Currently, only the KEA format is provided as an option but output files can be converted to another format using the `gdal_translate` command.

–**stats** specifies that the output images should be populate with statistics and pyramids – makes display much faster

-p RAD TOA SREF specifies the output products to be generated. You only need to specify what you want, e.g., SREF but if other products (e.g., RAD) are required then these will also be produced even if they are not specified.

–**aeropro Maritime** specifies that the 'Maritime' aerosol profile should be used.

–**atmospro MidlatitudeSummer** specifies that the 'MidlatitudeSummer' atmosphere profile should be used.

–**aot 0.25** specifies that an AOT value of 0.25 will be used for the correction.

–**surfacealtitude 0.4** specifies that the surface altitude (ground elevation) used for the correction is 400 m or 0.4 km. The value specified needs to be in KM.

-o ./Outputs specifies the output directory where all output file will be outputted to.

-i `LT05_L1TP_203024_19950815_20180217_01_T1/LT05_L1TP_203024_19950815_20180217_01_T1_MTL.txt` is the input images header file.

Once you have run the command (shown above) open the images within TuiView and compare the surface reflectance, radiance and TOA images, Figure 4.1. Please remember that you can open multiple TuiView windows (File > New Window) and tile them (File > Tile Windows...). Using the band combination NIR, SWIR1 and RED is recommended. Have a look at a number of different land cover and also try to find similar features around clouds and away from clouds (highly productive fields, for example; bright orange in the recommended band combination), what do you notice about the image pixel values? Also, ARCSI multiples the output pixel values by 1000 (this can be edited using the

`--scalefac` switch) so 100 % reflectance will have a value of 1000 and 1 % will have a value of 10.

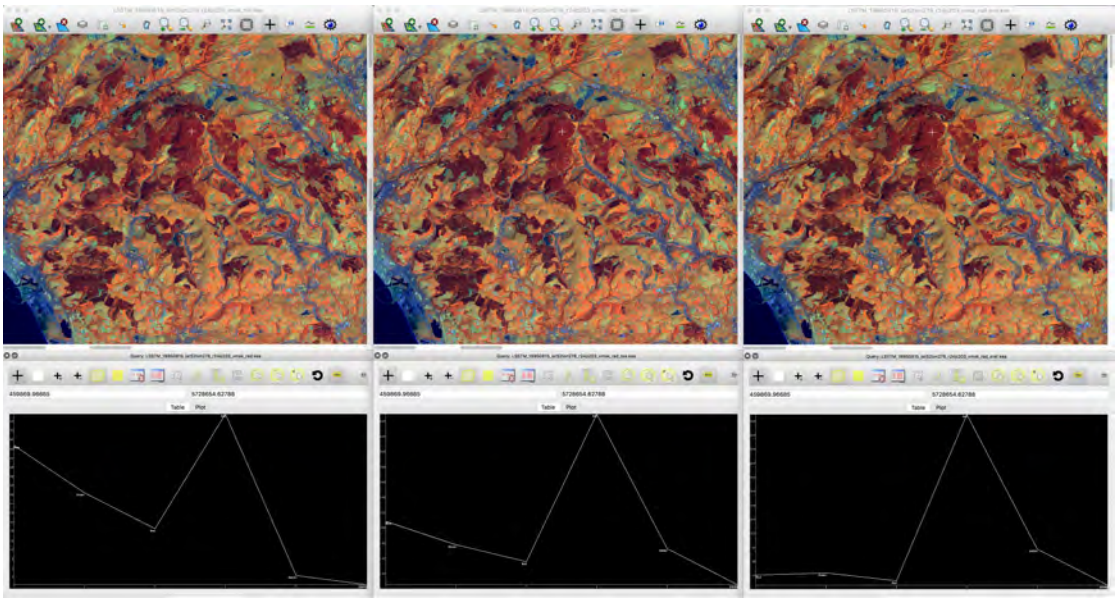

Figure 4.1: Comparison of spectral curves using TuiView for Radiance, TOA and SREF.

You should have noticed that they are different and that is due to the fact that AOT is variable across the scene. Within this worksheet we will just look at using a single value of AOT for correcting the scene but ARCSI already has some functionality to create an AOT surface which will be something for the future, particularly for scenes with high cloud cover.

4.2 Changing AOT and Vertical Water Content

You have now atmospherically corrected a scene with a single set of parameters we now need to considered what parameters should you use. You will now undertake a basic sensitivity analysis for AOT and vertical water content to see what effect they have on your atmospherically corrected result.

4.2.1 Changing values of AOT

Valid values of AOT range from 0.0 to 1.0, it is suggested that you run ARCSI using the following AOT values:

- 0.05
- 0.1

- 0.15
- 0.2
- 0.25
- 0.3
- 0.4
- 0.5
- 0.75
- 0.95

For example,

```
# AOT 0.05
arcsi.py -s ls5tm -f KEA --stats -p RAD TOA SREF --aot 0.05 \
--surfacealtitude 0.4 -o ./OutputsAOT005 \
-i LT05_L1TP_203024_19950815_20180217_01_T1/LT05_L1TP_203024_19950815_20180217_01_T1_MTL.txt

# AOT 0.1
arcsi.py -s ls5tm -f KEA --stats -p RAD TOA SREF --aot 0.1 \
--surfacealtitude 0.4 -o ./OutputsAOT01 \
-i LT05_L1TP_203024_19950815_20180217_01_T1/LT05_L1TP_203024_19950815_20180217_01_T1_MTL.txt

# AOT 0.5
arcsi.py -s ls5tm -f KEA --stats -p RAD TOA SREF --aot 0.5 \
--surfacealtitude 0.4 -o ./OutputsAOT05 \
-i LT05_L1TP_203024_19950815_20180217_01_T1/LT05_L1TP_203024_19950815_20180217_01_T1_MTL.txt

# AOT 0.95
arcsi.py -s ls5tm -f KEA --stats -p RAD TOA SREF --aot 0.95 \
--surfacealtitude 0.4 -o ./OutputsAOT95 \
-i LT05_L1TP_203024_19950815_20180217_01_T1/LT05_L1TP_203024_19950815_20180217_01_T1_MTL.txt
```

Use TuiView to compare the results (e.g., Figure 4.2). What changes do you observe? Make notes on these changes with screenshots. Also, what do you notice around areas of cloud cover? You may find it useful to use the band combination RED, GREEN and BLUE for display as the AOT most effects the visible channels. Figure 4.2 is displayed using the visible bands and you can (hopefully) see that the first (AOT 0.05) looks slight 'hazy', while the middle image looks quite clear with 'strong' colours and the image on the right (AOT 0.95) is over corrected with almost all the visible reflectance removed from the visible bands.

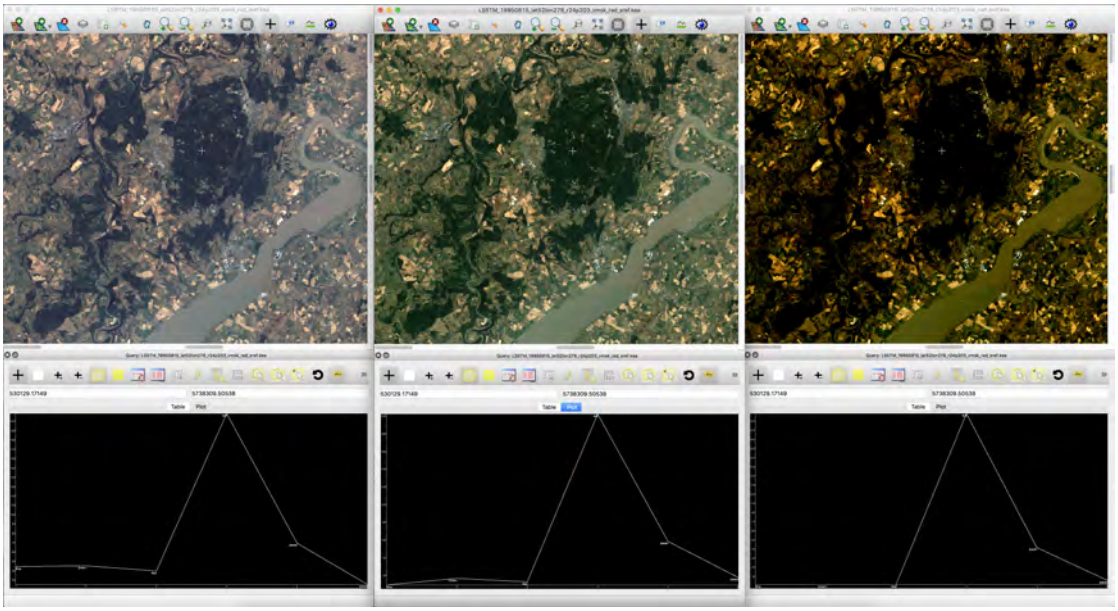

Figure 4.2: Comparison of spectral curves for SREF images with AOT values of 0.05, 0.5 and 0.95. The image is displayed using the RED, GREEN and BLUE image bands. You can see the image on the left looks a little 'hazy' and the righthand image is over corrected with much of the reflectance removed (i.e., black/dark areas of the image).

4.2.2 Changing values of water content

The water content within the vertical column is supplied in units of g/cm^2. A typical value is 3.6, however the amount of ozone (with units cm-atm) also needs to be specified but in this instance you a constant value of 0.9 throughout. Therefore, it is suggested that you run ARCSI using the following Water Column values:

- 0.5
- 1.0
- 2.0
- 3.0
- 4.0
- 5.0
- 6.0
- 8.0
- 9.0

- 10.0

For example,

```
# Water Content 0.5
arcsi.py -s ls5tm -f KEA --stats -p RAD TOA SREF --aot 0.25 \
--surfacealtitude 0.4 --atmoswater 0.5 --atmosozone 0.9 -o ./OutputsWater05 \
-i LT05_L1TP_203024_19950815_20180217_01_T1/LT05_L1TP_203024_19950815_20180217_01_T1_MTL.txt

# Water Content 3
arcsi.py -s ls5tm -f KEA --stats -p RAD TOA SREF --aot 0.25 \
--surfacealtitude 0.4 --atmoswater 3 --atmosozone 0.9 -o ./OutputsWater3 \
-i LT05_L1TP_203024_19950815_20180217_01_T1/LT05_L1TP_203024_19950815_20180217_01_T1_MTL.txt

# Water Content 9
arcsi.py -s ls5tm -f KEA --stats -p RAD TOA SREF --aot 0.25 \
--surfacealtitude 0.4 --atmoswater 9 --atmosozone 0.9 -o ./OutputsWater9 \
-i LT05_L1TP_203024_19950815_20180217_01_T1/LT05_L1TP_203024_19950815_20180217_01_T1_MTL.txt
```

Again, compare the outputted images to see the effect of editing these values has on the shape of the spectral curves and the spectral values. Which wavelengths have been most effected by this parameter?

4.3 Inversion for AOT

With time and expert knowledge it would be possible to derive an AOT value for each scene manually or in some regions of the world (e.g., Australia use an AOT of 0.05 Gillingham et al., 2012) it has been found that a constant can be used. However, for most regions a constant is not viable as the atmosphere is too variable (Wilson et al., 2014) and manually selecting values can be difficult and is time consuming. Therefore, deriving AOT from the image itself is desirable.

ARCSI provides a method of doing this using a dark object subtraction (DOS) to estimate the surface reflectance in blue channel. 6S is then numerically inverted to identify an AOT value which derives a surface reflectance value as close a possible to the estimated. To execute this functionality the following command is used:

```
arcsi.py -s ls5tm -f KEA --stats -p RAD DOSAOTSGL SREF METADATA \
-o ./OutputsAOTInv  --dem ./UKSRTM_90m.kea --tmpath ./tmp \
-i LT05_L1TP_203024_19950815_20180217_01_T1/LT05_L1TP_203024_19950815_20180217_01_T1_MTL.txt
```

Where:

–**tmpath ./tmp** is a directory where temporary files can be outputted during the processing, these will be deleted afterwards.

–**dem ./UKSRTM_90m.kea** is an elevation model for the UK, in this case the 90 m SRTM product. The higher the resolution of the DEM available the better and this will be used in place of the `--surfacealtitude` command line option. A look up table (LUT) for surface altitude will be created and used for producing the final correction as well. Note. the DEM will be resampled and reprojected to match the input image. Please ensure the projection of the DEM is well defined and the DEM no data value is defined, if it is not defined within the header file (recommended) then the `--demnodata` switch can be used.

In the information printed to the console you will be able to see what AOT value is identified (i.e., 0.35), however in this case I've asked for the METADATA product to be produced which will produced a metadata file associated with image file containing information from the input header file and produced from the processing stages. Alongside, identifying the AOT value this analysis will use the DEM to build a look up table (LUT) for elevation at steps of 100 m. This will help minimise atmospheric variability due to topography.

Chapter 5

More Advanced Usage

5.1 Operational Command – i.e., the one to use

ARCSI has a number of other products and options which can be applied. However, for Landsat imagery the following command which be what is normally applied – please note this command will commonly take up to or over an hour of processing time for a single scene, depending on the hardware being used.

```
arcsi.py -s ls5tm -p CLOUDS DOSAOTSGL STDSREF METADATA -o ./OutputsStdSREF/ \
--stats --format KEA --tmpath ./tmp --dem ./UKSRTM_90m.kea \
--keepfileends  stdsref.kea clouds.kea meta.json \
-i LT05_L1TP_203024_19950815_20180217_01_T1/LT05_L1TP_203024_19950815_20180217_01_T1_MTL.txt
```

Where:

-p CLOUDS DOSAOTSGL STDSREF METADATA – Specifies the products which are to be produced, in this case standardised reflectance, estimation of AOT via a dark object subtractions, undertake a cloud mask and produce metadata file.

-o ./OutputsStdSREF/ – directory where outputs will be outputted.

–**stats** – statistics and pyramids (overviews) should be created for visualisation.

–**format KEA** – output file formation (i.e., KEA)

–**tmpath ./tmp** – directory where temporary outputs will be written and then deleted if processing is successful completed. Within the tmp directory a directory with the same name as the output image is created so multiple files can use the same tmp directory.

–**dem ./UKSRTM_90m.kea** – the elevation model to be used for the processing.

–**keepfileends stdsref.kea clouds.kea meta.json** – The processing can create a lot of output files, which you may or may not want to keep. This option allows you to specify the file endings of the files you want to keep (i.e., all others are deleted).

Once you have produced this product compare it to the images you previously produced. If you open both images within Tuiview together and flick between between you should see the effect of the standardisation (i.e., topographic correction; Figure 5.1).

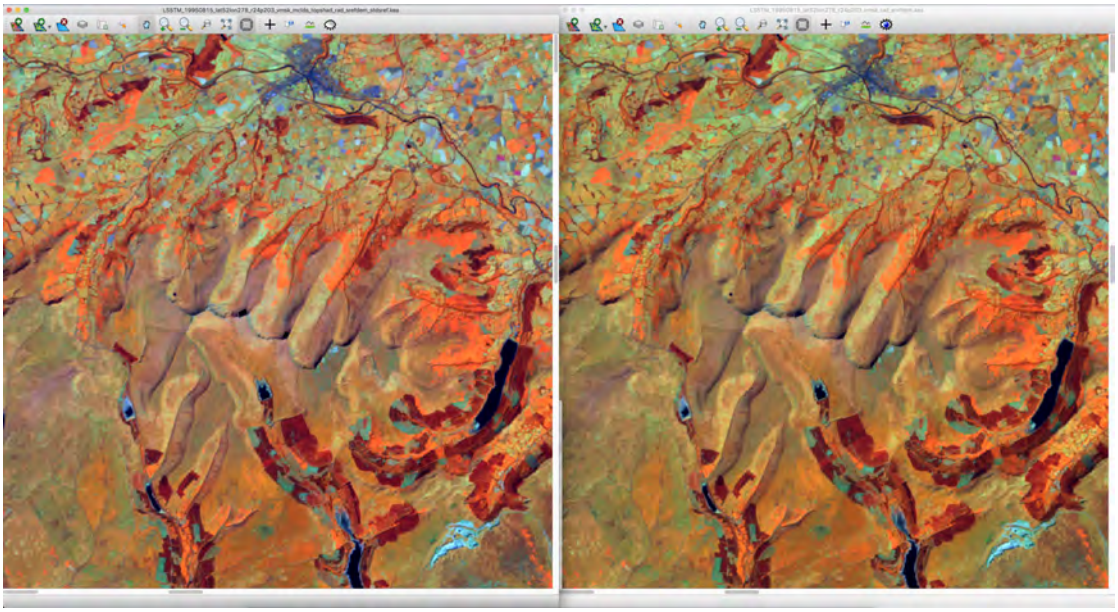

Figure 5.1: Comparison of the image corrected to standardised reflectance (i.e., topographically corrected; Left) and Surface reflectance (right).

Please note that for these examples, a lower resolution DEM (i.e., 90 m SRTM) is being used to avoid larger than necessary files to be downloaded etc. You should use the highest resolution and best quality DEM/DSM that you have available.

5.2 Operational Command with More Outputs

Depending on your use case you may want to keep more outputs, in this case the topographic shadow, saturated pixel and valid extent masks.

```
arcsi.py -s ls5tm -p CLOUDS DOSAOTSGL STDSREF SATURATE TOPOSHADOW FOOTPRINT \
-o ./OutputsStdSREF2/ --stats --format KEA --tmpath ./tmp --dem ./UKSRTM_90m.kea \
--k  clouds.kea meta.json sat.kea toposhad.kea valid.kea stdsref.kea \
-i LT05_L1TP_203024_19950815_20180217_01_T1/LT05_L1TP_203024_19950815_20180217_01_T1_MTL.txt
```

5.3 Not Applying the Generated Image Masks

By default the masks generated by ARCSI are applied to the image being processed, removing areas of imagery from the dataset. However, in some cases, it is useful to output the image data without the masks applied, supplying the masks to the end user who can then decide what they wish to apply to the images. In ARCSI this is done using the `--fullimgouts` switch, when this switch is included two versions of the output image are created one with the masks applied and the other without.

```
arcsi.py -s ls5tm -p CLOUDS DOSAOTSGL STDSREF SATURATE TOPOSHADOW FOOTPRINT \
-o ./OutputsStdSREFFull/ --stats --format KEA --tmpath ./tmp --dem ./UKSRTM_90m.kea \
--fullimgouts --k  clouds.kea meta.json sat.kea toposhad.kea valid.kea stdsref.kea \
-i LT05_L1TP_203024_19950815_20180217_01_T1/LT05_L1TP_203024_19950815_20180217_01_T1_MTL.txt
```

5.4 Re-projection

Another operation which can be included with your ARCSI command is re-projection where the output datasets are re-projected into another coordinated system and projection. There are two switches are needed for this, `--projabbv` specifies the string which is added to the output file name to specify the output projection (e.g., osgb for the UK Ordnance Survey Great Britain National Grid) and `--outproj4` or `--outwkt` to specify the output project using either a WKT or Proj4 string in a text file.

The WKT string for the UK Ordnance Survey National Grid is:

```
PROJCS["OSGB 1936 / British National Grid",
GEOGCS["OSGB 1936",
    DATUM["OSGB_1936",
        SPHEROID["Airy 1830",6377563.396,299.3249646,
            AUTHORITY["EPSG","7001"]],
        TOWGS84[446.448,-125.157,542.06,0.15,0.247,0.842,-20.489],
        AUTHORITY["EPSG","6277"]],
    PRIMEM["Greenwich",0,
        AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,
        AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4277"]],
PROJECTION["Transverse_Mercator"],
PARAMETER["latitude_of_origin",49],
PARAMETER["central_meridian",-2],
PARAMETER["scale_factor",0.9996012717],
PARAMETER["false_easting",400000],
PARAMETER["false_northing",-100000],
```

```
UNIT["metre",1,
    AUTHORITY["EPSG","9001"]],
AXIS["Easting",EAST],
AXIS["Northing",NORTH],
AUTHORITY["EPSG","27700"]]
```

You can find these representations of projections from websites such as `https://epsg.io` and `http://spatialreference.org`.

The following command will undertake the same analysis as above but the output will be reprojected as OSGB.

```
arcsi.py -s ls5tm -p CLOUDS DOSAOTSGL STDSREF SATURATE TOPOSHADOW FOOTPRINT \
-o ./OutputsStdSREFOSGB/ --stats --format KEA --tmpath ./tmp --dem ./UKSRTM_90m.kea \
--projabbv osgb --outwkt BritishNationalGrid.wkt \
--k  clouds.kea meta.json sat.kea toposhad.kea valid.kea stdsref.kea \
-i LT05_L1TP_203024_19950815_20180217_01_T1/LT05_L1TP_203024_19950815_20180217_01_T1_MTL.txt
```

When you re-project, ARCSI will create a pixel grid which is on a whole pixel grid, i.e., the origin coordinate (top-left corner) will be a whole number rather than have a decimal point or something weird and cumbersome.

5.5 Clear-Sky Product

The next option which can be applied is termed 'clear-sky'. This product aims to identify regions which have large continuous areas of clear-sky data (i.e., no clouds). Specifically, where there are regions of scattered cloud the gaps between the cloud are no always useful for many products and therefore we want to remove them.

The 'clear-sky' product can produced by simply adding `CLEARSKY` to the list of products.

```
arcsi.py -s ls5tm -p CLOUDS DOSAOTSGL STDSREF SATURATE TOPOSHADOW FOOTPRINT CLEARSKY \
-o ./OutputsStdSREFClearSky/ --stats --format KEA --tmpath ./tmp --dem ./UKSRTM_90m.kea \
-i LT05_L1TP_203024_19950815_20180217_01_T1/LT05_L1TP_203024_19950815_20180217_01_T1_MTL.txt
```

5.6 Thermal bands

Some sensors (e.g., Landsat) have thermal channels and ARCSI can process these alongside the other bands but only provides a 'at sensor' / 'top of atmosphere' product. For Landsat this product is produced if you select the `CLOUDS` or `CLEARSKY` products are the thermal data is needed as part of the cloud masking. However, if you would

like to product this separately without cloud masking then add `THERMAL` to the list of products.

```
arcsi.py -s ls5tm -p DOSAOTSGL SREF THERMAL \
-o ./OutputsThermal/ --stats --format KEA --tmpath ./tmp --dem ./UKSRTM_90m.kea \
-i LT05_L1TP_203024_19950815_20180217_01_T1/LT05_L1TP_203024_19950815_20180217_01_T1_MTL.txt
```

5.7 Sentinel-2

Sentinel-2 has basically the same options as Landsat 4-8, the main difference is that Landsat has a thermal channel(s) while Sentinel-2 provides images bands at different resolutions. The `THERMAL` product is therefore not available for Sentinel-2.

ARCSI, currently, outputs a single image for each product and therefore the image bands for Sentinel-2 need to be resampled to match one another. By default ARCSI resamples the 20 m bands to 10 m while the `--resample2lowres` switch can be used to produce a 20 m resolution output (i.e., the 10 m bands will be resampled).

Another option which ARCSI provides when resampling the 20 m bands to 10 m you can apply a 'sharpening'. The sharpening product aims to enhance the 20 m image bands using the 10 m bands to produce a sharper stacked 10 m product and works through the application of local linear regression models. The method was first proposed by Dymond and Shepherd (2004) for pan-sharpening Landsat-7 imagery and has subsequently been extended for Sentinel-2. Within ARCSI it is applied to the radiance image where the 20 m bands have been oversampled to 10 m resolution image using a nearest neighbour interpolation. A 7×7 pixel filter is applied to the 10 m image stack where within the 7×7 window a linear regression is performed independently for each lower resolution band to each higher resolution band. The linear model for each lower resolution image band with the best fit, above 0.5, is then used to predict the image pixel value for the band. If not fit above 0.5 is identified then the image pixel value is not altered.

5.7.1 Basic Sentinel-2 Commands

I have provided a sample Sentinel-2 scene over North Wales, just cutting off North of Aberystwyth (`S2A_MSIL1C_20170617T113321_N0205_R080_T30UVD_20170617T113319.SAFE`). The following command provides are basic surface reflectance product with the bands resampled to 10 m, without sharpening:

```
arcsi.py -s sen2 -f KEA --stats -p DOSAOTSGL SREF METADATA \
-o ./OutputsSen2_10m  --dem ./UKSRTM_90m.kea --tmpath ./tmp \
-i S2A_MSIL1C_20170617T113321_N0205_R080_T30UVD_20170617T113319.SAFE/MTD_MSIL1C.xml
```

The following command produces a 20 m resolution output:

```
arcsi.py -s sen2 -f KEA --stats -p DOSAOTSGL SREF METADATA \
-o ./OutputsSen2_20m  --dem ./UKSRTM_90m.kea --tmpath ./tmp --resample2lowres \
-i S2A_MSIL1C_20170617T113321_N0205_R080_T30UVD_20170617T113319.SAFE/MTD_MSIL1C.xml
```

Finally, the following command produces a 10 m resolution product where the 20 m bands have also been sharpened, as described above. Compare this product to the first Sentinel-2 product you produced using Tuiview flicking between the layers. The difference can be difficult to see in places (which is the idea!) but where you have strong boundaries (e.g., field boundaries) and you display the scene using a band combination which includes the SWIR or Red-Edge bands (e.g., NIR, SWIR1, RED) then the improvement through sharpening should be very evident.

```
arcsi.py -s sen2 -f KEA --stats -p DOSAOTSGL SREF METADATA SHARP \
-o ./OutputsSen2_10mSharp  --dem ./UKSRTM_90m.kea --tmpath ./tmp \
-i S2A_MSIL1C_20170617T113321_N0205_R080_T30UVD_20170617T113319.SAFE/MTD_MSIL1C.xml
```

5.7.2 Advanced Sentinel-2 Commands

When running through Sentinel-2 to create an Analysis Ready Data (ARD) product the following command or similar (see Landsat examples above) is used.

```
arcsi.py -s sen2 -p CLOUDS DOSAOTSGL STDSREF SHARP METADATA -o ./OutputsSen2ARD/ \
--stats --format KEA --tmpath ./tmp --dem ./UKSRTM_90m.kea \
--keepfileends  stdsref.kea clouds.kea meta.json \
-i S2A_MSIL1C_20170617T113321_N0205_R080_T30UVD_20170617T113319.SAFE/MTD_MSIL1C.xml
```

Please note, this can take a while to run through and can be up to an hour or so.

5.7.3 Older Sentinel-2 Scenes rather than Granules

Some older Sentinel-2 scenes available from ESA are not provided as individual granules but a collection of granules which make the whole image. To process these in ARCSI the scene needs to be broken up into individual granules. ARCSI provides the command `arcsisplitsen2granules.py` to perform this operation. The input is the input SAFE file directory (i.e., scene) while the output is the directory were the granule SAFE files are to be written.

Chapter 6

Batch Processing

One of the most exciting things about these data (i.e., Landsat and Sentinel-2) is that they are global and freely available. However, downloading and processing large amounts of this data is challenging. However, ARCSI has some commands and tools which can make handling these data easier and feasible. If you have a high performance computer (HPC) available then you can process some very large datasets using these tools. For example, I have produced landsat composites globally for all the mangrove coastlines of the world. That required the processing of over 15000 landsat 5 and 7 scenes. Using the tools shown below this was undertaken in only a few days using around 100 processing cores.

When a large number of scenes require processing then commands that automate steps are desirable and greatly simplifies the process. ARCSI provides a number of commands which are useful for batch processing:

arcsisortlandsat.py sorts the landsat files into individual directories for each sensor (i.e., LS 5 TM) and also builds a standard directory structure for processing.

arcsiextractdata.py extracts the contents of archives, with each archive being extracted into an individual directory.

arcsibuildcmdslist.py builds the `arcsi.py` commands for each scene creating a shell script.

The files provided for this example are shown below, but others could be downloaded and added to the directory (all files need to be within the same directory), a mixture of landsat scenes is OK but not a mixture of different sensors (e.g., Sentinel-2 and Landsat) process the different sensors seperately.

```
LT05_L1TP_204023_19940718_20170113_01_T1.tar.gz
LT05_L1TP_204024_19940616_20170114_01_T1.tar.gz
LT05_L1TP_204024_19940718_20170113_01_T1.tar.gz
```

6.1 Sorting Landsat Scenes

If you are processing landsat scenes from different sensors (e.g., LS5, LS7 and LS8) you might find it useful to separate the downloaded files by sensor. This can be done using the `arcsisortlandsat.py` command. However, this is optional as the `arcsibuildcmdslist.py` command can detect the landsat sensor version (see below).

6.2 Extracting Data

I find it useful to create the following directory structure for my processing:

```
Inputs
Outputs
RAW
tmp
```

where the download archives are within the RAW directory. Note. the `arcsisortlandsat.py` command can create this structure for you.

To extract all scenes the `arcsiextractdata.py` is used as shown below:

```
> arcsiextractdata.py -i RAW/ -o Inputs/
```

Once the files are extracted the directory structure will look like the following:

```
> ls RAW/
LT05_L1TP_204023_19940718_20170113_01_T1.tar.gz
LT05_L1TP_204024_19940616_20170114_01_T1.tar.gz
LT05_L1TP_204024_19940718_20170113_01_T1.tar.gz

> ls */Inputs
Inputs/
LT05_L1TP_204023_19940718_20170113_01_T1
LT05_L1TP_204024_19940616_20170114_01_T1
LT05_L1TP_204024_19940718_20170113_01_T1
```

6.3 Building ARCSI Commands

Now the data have been extracted the relevant `arcsi.py` commands, one for each input file, need to be created. If you had a large number of input files this would be very time consuming and error prone to do manually, therefore we will use a script to automate it.

Notice that these are very similar to the individual commands that you previously executed but now provide inputs to the `arcsibuildcmdslist.py` command which selects a number of input files and generate a single shell script output.

```
arcsibuildcmdslist.py -s ls5tm -f KEA --stats -p CLOUDS DOSAOTSGL STDSREF \
--outpath ./Outputs --dem ../UKSRTM_90m.kea \
--keepfileends stdsref.kea clouds.kea \
--tmpath ./tmp -i ./Inputs -e "*MTL.txt" -o LSARCSICmds.sh
```

Following the execution of this command the following file will have been created `LSARCSICmds.sh`. This file contain the `arcsi.py` commands to be executed. Open the file and take a look, you will notice that all the file paths have been convert to absolute paths which means the file can be executed from anywhere on the system as long as the input files are not moved.

If you have a selection of landsat scenes from different versions of the sensor or wish to have a generic command you can use for all then if you define the sensor as `LANDSAT` then the function will automatically derive the landsat sensor version. For example:

```
arcsibuildcmdslist.py -s LANDSAT -f KEA --stats -p CLOUDS DOSAOTSGL STDSREF \
--outpath ./Outputs --dem ../UKSRTM_90m.kea \
--keepfileends stdsref.kea clouds.kea \
--tmpath ./tmp -i ./Inputs -e "*MTL.txt" -o LSARCSICmds.sh
```

6.4 Executing ARCSI Commands

To execute the `arcsi.py` commands the easiest methods is to run each in turn using the following command:

```
> sh LSARCSICmds.sh
```

This will run each of the commands sequentially. However, most computers now have multiple processing cores and to take advantage of those cores we can use the GNU `parallel` command line tool (`http://www.gnu.org/software/parallel/`). Taking advantage of those cores means that processing can be completed much quicker and more efficiently.

```
> parallel -j 4 < LSARCSICmds.sh
```

The switch `-j 4` specifies the number of processing cores which can be used for this processing. If no switches are provided then all the cores will be used, ensure that you don't request more resources than you have available. Please note, until all processing for a scene has completed, nothing will be printed to the console.

Once you have completed your processing you should clean up your system to remove any files you don't need for any later processing steps. In most cases you will only need to keep the original archives (so you can reprocess the RAW data at a later date if required) and the SREF product with relevant masks and metadata. It is recommended that you also retain the scripts you used for processing and a record of the commands you used for a) reference if you need to rerun the data and b) as documentation for the datasets so you know what parameters and options were used.

6.5 Sentinel-2

Processing Sentinel-2 and other sensors is identical to the steps shown above. However, one caution of processing Sentinel-2 is that it difficult to uniquely identify the image header file. The correct header file is always within the top level of the SAFE file structure. To ensure that the correct file is found you need to use the `-d / --depth` switch which controls the depth in the file structure to which the search is carried out.

As an example, the following command was used to process several thousand Sentinel-2 granules for the UK.

```
arcsibuildcmdslist.py -i ./Inputs -o ./Sen2ARCSICmds.sh \
-e "*MTD*.xml" -d 1 -f KEA --stats --outpath ./Outputs \
-p RAD SATURATE TOPOSHADOW TOA CLOUDS DOSAOTSGL SREF STDSREF SHARP FOOTPRINT METADATA \
-s sen2 --checkouts --fullimgouts --outproj4 ./osgb_proj4.prj \
--projabbv osgb --tmpath ./tmp --dem ./SRTM_1arc_UK.kea
```

Chapter 7

Downloading Landsat and Sentinel-2 Data

One of the biggest challenges of undertaking large data processing tasks is always downloading and getting hold of the data. Here, ARCSI can help again – with a lot of help from Google!

Google provide, what are referred to as, Cloud Buckets with the Sentinel-2 and Landsat image archives through which these data are freely available and accessible.

- Landsat Bucket: `https://cloud.google.com/storage/docs/public-datasets/landsat`
- Sentinel-2 Bucket: `https://cloud.google.com/storage/docs/public-datasets/sentinel-2`

To view Sentinel-2 and Landsat-8 imagery so you can see what is available and know the row/path or granules you are interested in then the following site is a useful reference `https://search.remotepixel.ca`.

7.1 Search and Find Data

Alongside the datasets, Google also make CSV files with the associated meta-data for all the images stored available for download. This is very useful as it allow us to search and find the scenes/granules we might be interested in.

ARCSI provides commands to automatically download and build a local database of these meta-data.

```
# Setup Landsat Database
arcsisetuplandsatdb.py -f landsatdb-20180216.db
```

```
# Setup Sentinel-2 Database
arcsisetupsen2db.py -f sentinel2db-20180216.db
```

Setting up these databases can take a few minutes. However, once they are set up you can use the commands `arcsigenlandsatdownlst.py` and `arcsigensen2downlst.py` to generate a list of scenes to download using the Google Cloud Tools (`https://cloud.google.com/sdk/`).

Querying Landsat is perform on a per row/path basis while for Sentinel-2 it is per granule. Therefore, if you require more than one row/path or granule you'll need to run the command multiple times but this can be done with a simple script if the number of rows/paths or granules is large.

Both commands let you specify a date range of interest, if no range is specified then the whole range is considered. An upper cloud threshold (i.e., cloud percentage less then XX%) can also be specified. For landsat you can also specify the sensor, spacecraft and collection and where nothing is specified it is assumed all are of interest.

When considering landsat it is recommended that only collection 'T1' are normally considered as 'T2', 'RT' and 'PRE' can have some limitations in terms of registration etc. More information is available from here: `https://landsat.usgs.gov/landsat-collections`.

To download image for the French Guiana coast to assess mangrove change then the following commands could be used:

```
arcsigenlandsatdownlst.py -f landsatdb-20180216.db -p 227 -r 56 --cloudcover 70 \
--collection T1 --startdate 1989-01-01 --enddate 1991-12-31 \
-o ./Dwnlds_r227_p56_1990_T1.sh --outpath './Dwnlds/1990' --lstcmds --multi

arcsigenlandsatdownlst.py -f landsatdb-20180216.db -p 227 -r 56 --cloudcover 70 \
--collection T1 --startdate 1994-01-01 --enddate 1996-12-31 \
-o ./Dwnlds_r227_p56_1995_T1.sh --outpath './Dwnlds/1995' --lstcmds --multi

arcsigenlandsatdownlst.py -f landsatdb-20180216.db -p 227 -r 56 --cloudcover 70 \
--collection T1 --startdate 1999-01-01 --enddate 2001-12-31 \
-o ./Dwnlds_r227_p56_2000_T1.sh --outpath './Dwnlds/2000' --lstcmds --multi

arcsigenlandsatdownlst.py -f landsatdb-20180216.db -p 227 -r 56 --cloudcover 70 \
--collection T1 --startdate 2004-01-01 --enddate 2006-12-31 \
-o ./Dwnlds_r227_p56_2005_T1.sh --outpath './Dwnlds/2005' --lstcmds --multi

arcsigenlandsatdownlst.py -f landsatdb-20180216.db -p 227 -r 56 --cloudcover 70 \
--collection T1 --startdate 2009-01-01 --enddate 2011-12-31 \
-o ./Dwnlds_r227_p56_2010_T1.sh --outpath './Dwnlds/2010' --lstcmds --multi

arcsigenlandsatdownlst.py -f landsatdb-20180216.db -p 227 -r 56 --cloudcover 70 \
--collection T1 --startdate 2014-01-01 --enddate 2016-12-31 \
-o ./Dwnlds_r227_p56_2015_T1.sh --outpath './Dwnlds/2015' --lstcmds --multi
```

Note. the `--lstcmds` switch requests that the full commands are outputted rather than just the urls and `--multi` specifies that multiple connections are used for the download.

7.1.1 Perform Download

To perform the download requires that the Google Cloud tools are downloaded and set up on your machine such that the `gsutils` commands is available on your system. Currently, these commands use Python 2.7 and you are most likely using Python 3.X for ARCSI/RSGISLib etc. thankfully conda allows us to create different environments each with different versions of python and associated packages and switch between them.

To create a new conda environment with python 2.7 then the following command can be used:

```
conda create -n googcloud python=2.7
```

To activate that environment run

```
source activate googcloud
```

You can download the Google Cloud tools (SDK) from `https://cloud.google.com/sdk/` and follow the instructions for its installation.

Once you have the tools installed you should then be able to run the script (e.g., `Dwnlds_r227_p56_2015_T1.sh`) to download the data, created by the `arcsigenlandsatdownlst.py` or `arcsigensen2downlst.py` commands.

An example command to download a scene will look like:

```
gsutil -m cp -r gs://gcp-public-data-landsat/LE07/01/227/056/LE07_L1TP_227056_19990810_20170217_01_T1 \
./Dwnlds/2000
```

Make sure the output directory exists but once the download has finished you can then use ARCSI as you would have done, the data is not within an archive so you can skip the data extraction step.

Chapter 8

Other Useful Bits/Bobs

`arcsibuildfilenameslu.py` is a command which allows you to build a look up table (LUT) referencing the input archive name to the output file name generated by ARCSI.

`arcsifindnotprocessed.py` is a command which can find scenes which have no been processed yet (i.e., there is not output files).

`arcsichecksen2ver.py` is a command which can check that there is only one version of each Sentinel-2 scene, sometimes there can be multiple version of files within a repository (i.e., processed from RAW at different times with different processor versions).

`arcsiremoveduplicates.py` is a command for checking for and removing duplicate files – similar to the command above for Sentinel-2.

`arcsibuildmultifilelists.py` create identify images which are captured within the same path. Functionality which is not yet covered within this tutorial, ARCSI are the option of processing paths ensuring the images within the paths are processed such that they follow each other without hard boundaries. This command generates the scene lists as a text file ready to be put into arcsi.py.

`arcsimpi.py` when processing paths, multiple processing cores can be used using `arcsi.py` this only works for shared memory machines (e.g., normal desktops, virtual machines etc.). For HPC clusters, `arcsimpi.py` provides the option of using MPI to access multiple nodes/cores across the HPC. If you want to use this functionality, it is probably best to email the mailing list and we can provide you some support.

Chapter 9

Conclusions

Hopefully, this tutorial has reminded you of the basics of atmospheric correction and shown you how to implement that functionality within ARCSI, both basic functionality and more complex batch processing and generation of analysis ready data (ARD) products. Using the materials from this tutorial worksheet you should now be in a position to process thousands of landsat and Sentinel-2 scenes if you have the computational resource available.

Don't forget there is the mailing list `https://groups.google.com/forum/#!forum/rsgislib-support`, which is often quite active, if you need any help or support.

Bibliography

Bunting, P., Clewley, D., Lucas, R. M., Gillingham, S., Jan. 2014. The Remote Sensing and GIS Software Library (RSGISLib). Computers and Geosciences 62, 216–226.

Bunting, P., Gillingham, S., 2013. The KEA image file format. Computers and Geosciences 57, 54–58.

Chavez, P. S., 1996. Image-based atmospheric corrections-revisited and improved. Photogrammetric Engineering And Remote Sensing.

Dymond, J. R., Shepherd, J. D., Mar. 2004. The spatial distribution of indigenous forest and its composition in the Wellington region, New Zealand, from ETM+ satellite imagery. Remote Sensing Of Environment 90 (1), 116–125.

Gillingham, S., Flood, N., 2013. Raster I/O Simplification Python Library. URL `https://bitbucket.org/chchrsc/rios/overview`

Gillingham, S., Flood, N., Gill, T. K., 2012. On determining appropriate aerosol optical depth values for atmospheric correction of satellite imagery for biophysical parameter retrieval: requirements and limitations under Australian conditions. International Journal Of Remote Sensing 34 (6), 2089–2100.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830.

Shepherd, J. D., Dymond, J. R., 2003. Correcting satellite imagery for the variance of reflectance and illumination with topography. International Journal Of Remote Sensing 24 (17), 3503–3514.

Smith, G. M., Milton, E. J., Nov. 2010. The use of the empirical line method to calibrate remotely sensed data to reflectance. International Journal Of Remote Sensing 20 (13), 2653–2662.

Vermote, E., Tanre, D., Deuze, J., Herman, M., Morcrette, J., Jan. 1997. Second Simulation of the Satellite Signal in the Solar Spectrum, 6S: An overview. IEEE Transactions of Geoscience and Remote Sensing 35 (3), 675–686.

Wilson, R. T., Feb. 2013. Py6S: A Python interface to the 6S radiative transfer model. Computers and Geosciences 51, 166–171.

Wilson, R. T., Milton, E. J., Nield, J. M., 2014. Spatial variability of the atmosphere over southern England, and its effect on scene-based atmospheric corrections. International Journal of Remote Sensing.

Zhu, Z., Wang, S., Woodcock, C. E., Mar. 2015. Improvement and expansion of the Fmask algorithm: cloud, cloud shadow, and snow detection for Landsats 4–7, 8, and Sentinel 2 images. Remote Sensing Of Environment 159, 269–277.

Zhu, Z., Woodcock, C. E., Sep. 2014. Automated cloud, cloud shadow, and snow detection in multitemporal Landsat data: An algorithm designed specifically for monitoring land cover change. Remote Sensing Of Environment 152, 217–234.

Appendix A

A brief introduction to the UNIX terminal

Linux is a UNIX like operating system which is widely used, for example the Android operating system running on most mobile phones uses the Linux kernel and most servers forming the internet are running a flavour of Linux/UNIX. In fact, the majority of you are using a UNIX/Linux operating system every day as while Android is Linux MacOS and iOS run an a flavour of UNIX based on the BSD (Berkeley Software Distribution) UNIX. Linux/UNIX is generally a free operating system which is very stable and secure. There are many application which can be run the operating system and it can be used in place of Windows, either as MacOS or various desktop flavours of Linux (e.g., Mint `https://www.linuxmint.com`, Ubuntu `https://www.ubuntu.com`, CentOS (`https://www.centos.org`), of which there are hundreds.

In this tutorial we will be looking at using the UNIX Terminal (Figure 1.1), alternatively called the 'Command Line'. This is a text based interface to the computer system, which allow manipulation of the files and directories of the file system and actions to be carried out on those files, including in the form of scripts. Scripts an simple a formal, computer readable, method of writing down a series of actions (steps) the computer is to carry out in sequence. You can do everything that you can typically do through a graphical user interface (GUI) using the terminal and when scripted it is more reproducible and can be more easily applied to large datasets through batch processing.

The majority of high performance computing (HPC) environments and cloud services such as Amazon and Google use the Linux operating system and in many cases only provide a terminal driven interface through which to configure and install your applications. Knowledge and ability to use a terminal interface on Linux is therefore tremendously useful.

A.1 Working with the file system

When you open a Terminal window you will almost certainly start in your home directory, this is the directory structure where you are expected to store your files.

A.1.1 Listing your current directory

The first command you need is to list the file in the directory you are currently working in. You might find it useful to have the graphical file browser open alongside the terminal as you go through these steps.

To list the files in your home directory, run the following command:

```
ls
```

Compare what you see in your graphical file browser.

Most commands and options which are accessed via things called 'switches', this come straight after the command (with a space after the command) with a dash ('`-`') and then a letter or double-dash ('`--`') and then a word.

A useful switch on the `ls` command is `-l`, which lists the files with information, such as size and date of modification.

```
ls -l
```

Again, compare to your graphical file browser (list view). You can have more than one switch on a command, for example the `-h` switch on the `ls` command is useful when combined with `-l` is it converts the files sizes to a human readable unit. Try:

```
ls -l -h
```

note that you can combine switches, so `-lh` is the same as `-l -h`, try:

```
ls -lh
```

If you want to sort files by time then add `-t`

```
ls -lht
```

A.1.2 Where am I?

A common problem is not knowing where you are within the file system, in a graphical interface you can usually click back to home and start navigating again and you have

an address bar on the top of the screen telling you where you are. There is a command for to providing you the same information as is within the address bar:

```
pwd
```

Compare what `pwd` has printed to the terminal screen and what is in your graphical browser.

A.1.3 Moving Directories

Within your graphical interface you will be used to double clicking on a directory icon and the interface updating to show you the contents of that directory. Obviously, you can do the same thing on the terminal, using the '`cd`' (change directory) command.

Try moving to the directory Desktop:

```
cd Desktop
```

Check where you are in the file system using the `pwd` command and then list the files within this directory using the `ls` command. Compare this to the graphical file browser.

Once you have moved into a directory you will want to go back (i.e., as you would it you pressed the back button in your graphical interface). On the terminal this is done using two dots (`..`), which means move one directory back. Try it and then run `pwd` and `ls` to check you are back in your home directory.

```
cd ..
pwd
ls
```

You can provide a whole path to the `cd` command, referring to multiple directories. For example,

```
cd Desktop
pwd
cd ../Documents
pwd
```

If you want to get back to your home directory where ever you are within the directory structure you can just type `cd` without specifying a directory path.

```
cd
```

It is also useful to know that you can navigate to your home directory using the ~ symbol:

```
cd ~
```

and that it can be used to navigate to a path starting from your home directory, for example:

```
cd ~/Downloads
```

A.1.4 Creating a new directory

The command to create a new directory is `mkdir`. To create a new directory run the command providing the name of the new directory as input. For example, run the following starting in your home directory.

```
cd Desktop
ls
mkdir Test
ls
cd Test
pwd
```

If you have a whole directory path you want to create then you can use the `-p` switch which will create all the directories list in the path provided which don't currently exist. For example, run the following.

```
cd
cd Desktop
mkdir -p Test/Test1/Test2/Test3/Test4
ls
cd Test
pwd
ls
cd Test1
pwd
ls
cd Test2/Test3/Test4
pwd
```

To navigate back to the Desktop you can run either:

```
cd ../../../../../
```

or

```
cd
cd Desktop
```

or

```
cd ~/Desktop
```

A.1.5 Case Sensitive

You should be aware that file and directory names are case sensitive and therefore `Test` and `test` would be two different files or directories on your system. Try:

```
cd test
```

This fails as the directory you created earlier is `Test`.

A.1.6 Copying Files and Directories

For illustration download an existing file, for example an image e.g., `LS8_20130511_lat450lon16985_r91p7`
These instructions assume that the file will be downloaded to the 'Downloads' directory. You can list the contents of your Downloads directory using the following command:

```
ls ~/Downloads
```

Check what has been listed using your graphical file browser.

To copy a file or directory you need to use the `cp` command, for example to copy the `LS8_20130511_lat450lon16985_r91p75_sref.kea` to the Test folder you have created on your Desktop you can use the following command:

```
cp ~/Downloads/LS8_20130511_lat450lon16985_r91p75_sref.kea ~/Desktop/Test
ls ~/Desktop/Test
```

Check in your graphical file browser what has happened with the files.

Lets copy the `Test` directory to your documents folder, you can do this using the `-R` switch, which causing the `command` to recurse down a directory structure.

```
cd ~/Desktop
cp -R ./Test ~/Documents
ls ~/Documents
cd ~/Documents/Test/Test1/Test2/Test3/Test4
pwd
```

Check what you have in your graphical file browser.

A.1.7 Moving Files and Directories

Moving rather than copying files and directories means than the data is moved rather than a copy made. For example, lets move the copy of `LS8_20130511_lat450lon16985_r91p75_sref.kea` in the directory `Documents\Test` to the `Desktop`, run the following:

```
ls ~/Documents/Test
mv ~/Documents/Test/LS8_20130511_lat450lon16985_r91p75_sref.kea ~/Desktop

ls ~/Documents/Test
ls ~/Documents/Desktop
```

Let's move the test directory structure to your Downloads folder, run the following commands.

```
cd ~/Documents
ls
mv Test ~/Downloads
ls
cd ~/Downloads
ls
cd Test
pwd
```

Again, check in your graphical file browser so you can see what has happened.

A.1.8 Removing a Directory

Now you have created some directories and moved things about you've got a bit of mess to clear up so lets look at deleting files and directories. Deleting is done using the `rm` (remove) command.

Let's first delete the KEA image file we can left on the Desktop:

```
cd ~/Desktop
ls
rm LS8_20130511_lat450lon16985_r91p75_sref.kea
ls
```

Check in your graphical file browser that the file has been removed.

Similar to the copy command, to delete a directory we need to use the `-R` command to recurse through the directory structure deleting it all, let's delete the Test folder on the Desktop:

```
cd ~/Desktop
ls
rm -R Test
ls
```

Check in your graphical file browser that the directory has been removed. Let's do the same in your Downloads directory.

```
cd ~/Downloads
ls
rm -R Test
ls
```

A.1.9 Decompressing files

The most common compression format under UNIX is a tar.gz file, where the `tar` command has been used to create an archive and the `gzip` command has been used to compress the archive. This can all be accomplished using the `tar` command. For example, using the `LC82040242013139LGN01.tar.gz` file downloaded from the USGS Earth Explorer (`https://earthexplorer.usgs.gov`).

```
tar -zxf LC82040242013139LGN01.tar.gz
```

The switches are: `z` uncompress using gzip, `x` extract from archive and `f` from file.

A.2 Shell Scripts

A shell script ('the shell' is another term for the terminal) is a simple way of listing the command you wish to run in a test file and then running them in a single step. A shell script is just a simple text file, so open a text editor and save an empty file on your Desktop called `testscript.sh`, we use the file extension `.sh` to indicate a shell script. Within the script save the following:

```
echo "Hello World"
```

To run the script (ensure it is saved in your text editor) and then type the following on the terminal:

```
cd ~/Desktop
sh ./testscript.sh
```

Edit the script again so it reads:

```
echo "Hello World"

ls
cd ..
pwd
cd Documents
pwd
ls
cd ~/Desktop
mkdir -p Test/Test1/Test2/Test3/Test4
cd Test/Test1/Test2/Test3/Test4
pwd
```

This can be useful for documenting the commands you are running so it is reproducible but it can also be used for batch processing using a command or python script which is doing something to our input data (e.g., images) and we have a lot of processes, for example we could create a shell script similar to the one below and when it runs it'll run each of the images in turn.

```
python ImageProcessingCmd.py inputimg1.kea
python ImageProcessingCmd.py inputimg2.kea
python ImageProcessingCmd.py inputimg3.kea
python ImageProcessingCmd.py inputimg4.kea
python ImageProcessingCmd.py inputimg5.kea
python ImageProcessingCmd.py inputimg6.kea
```

Note. Python scripts have the file extension `py`.

A.3 Conclusions

You should now be able to move around the Linux/UNIX Terminal moving from one directory to another and manipulating files. There is a lot more that you can do which is out of scope here but there are loads of online resources and materials you can use. However, just spending time using the Terminal and command line tools is the best way to learn, there are few short cuts to perseverance.