Data preprocessing 3/3, change detection, and dealing with vector data

Your attempt to detect clouds in the imagery at the end of session 2 should at least have revealed a small cloud in the Landsat 5 image. Such clouds cannot be corrected for but need to be masked manually. In snow-free areas, clouds can easily be detected using unsupervised techniques based on two unique spectral properties of clouds: they feature a high albedo in the visible spectrum and a rather low temperature. The ratio of radiances in the blue and thermal band is able to combine these properties in a single layer.

```
clouds2011 <- (blue2011rad / the2011rad)
plot (clouds2011) ## All pixels in the cloud have values > 13
```

Identify threshold

To convert this ratio into a binary mask layer, we need to define and apply a threshold beyond which a pixel is considered to be a cloud. The subjectively chosen threshold of 13 is able to separate cloud pixels from non-cloud pixels.

```
clouds2011 <- clouds2011 > 13
plot (clouds2011)
```

Reclassify binary Raster layer and apply the mask

The binary cloud layer has to be reclassified to a mask layer. For this purpose, we replace the the pixels values indicating the presence of a cloud by NA and the pixel values indicating no cloud by 1.

```
clouds2011 <- reclassify (clouds2011, matrix (c (0, 1, 1, NA), 2, 2, byrow=T))
```

This mask layer is then multiplied with the image pixels. In consequence, all NA pixels in the mask layer become NA in the image while the other pixels remain unchanged.

```
ls52011dps.topo <- ls52011dps.topo * clouds2011
plotRGB (ls52011dps.topo, 4, 3, 2, stretch="lin", colNA="black")
```

The subset of the 2014 image has no visible clouds. To make sure that this visual assessment is indeed true, we repeat this process for the Landsat 8 image. Because we have converted the raw DN values directly to TOA reflectance, we still need to calculate the radiance values for the blue band. The required coefficients are listed the meta data. Table 1.1 shows that the combination of the two thermal bands of Landsat 8 covers approximately the spectral range of the Landsat 5 thermal band. Dealing with radiances, we hence have to summarize the thermal bands to produce a raster layer that is comparable to the Landsat 5 thermal band.

```
blue2014rad <- ls82014.subs[[1]] * 1.2626E-02 - 63.12837
clouds2014 <- blue2014rad / (thea2014rad + theb2014rad)
plot (clouds2014)
```

Q3.1: Can you use the same threshold for the 2014 image? If yes, why?

Yes, because we have converted the raw DNs to radiances. Radiances are quantified in W*m-2*sr-1 and are not dependent on the respective sensor.

Q3.2: Apply the threshold onto the clouds2014 layer. Is the visual assessment that there are no clouds substantiated?

Still no clouds...

Write image to file

The processed image can be written with the function writeRaster() to a file that can be imported to any GIS or remote sensing software. The image can be written to various file types that are listed in the documentation of the function.

Q3.3: Which file types can be generated with writeRaster?

*.grd (native raster package format), *.asc (ESRI ASCII), *.sdat (SAGA GIS grid), *.rst (IDRISI format), *.nc (netCDF, if the ncdf package is installed), *.tif (GeoTiff, if the rgdal package is installed), *.envi (ENVI header labelled), *.bil (ESRI header labelled), *.img (Erdas Imagine images)

Here, we use the GeoTiff format that is most convenient for many purposes.

```
writeRaster (ls52011dps.topo, "ls52011dps_topo.tif", format="GTiff")
writeRaster (ls82014dps.topo, "ls82014dps_topo.tif", format="GTiff")
```

Identifying water bodies

Now we have completed all pre-processing steps that are required for a joint analysis of data taken with different sensors, on different dates, or at different locations. This allows us to proceed to the original question 'how has the surface of Lake Shasta and Lake Trinity changed from 2011 to 2014?'

To answer this question, we first calculate the normalized difference water index (NDWI) for both years. This index highlights areas with high wetness or moisture (i.e., water bodies) in the image. Subsequently, we calculate the difference between the two NDWI layers to identify areas with changing water content. The NDWI is calculated as normalized difference between the reflectance values in the green and the SWIR range following equation 3.1

$$NDWI = \frac{Green - SWIR}{Green + SWIR} \tag{3.1}$$

```
ndwi2014 <- (ls82014dps.topo[[2]] - ls82014dps.topo[[5]]) /
            (ls82014dps.topo[[2]] + ls82014dps.topo[[5]])
ndwi2011 <- (ls52011dps.topo[[2]] - ls52011dps.topo[[5]]) /
            (ls52011dps.topo[[2]] + ls52011dps.topo[[5]])
```

The two NDWI layers are then subtracted

```
change <- ndwi2011 - ndwi2014
plot (change)
```

Again, we need to define a threshold in terms of the `change` magnitude to separate altered from unaltered pixels. Instead of defining the threshold after visual examination of the `change` image, we use a histogram analysis for a more reliable estimation. For this purpose, we plot the density histogram of the difference values. Assuming that the majority of pixels is unchanged, pixel values beyond the inflexion point of the distribution curve indicate a change.

```
plot (density (change)) ## the inflexion point is near 0.3
loss <- change > 0.3
plot (loss)
```

We are further interested in the total area of the surface decrease. This area can be calculated with different approaches. We can either convert the loss layer to a vector data set, write it to a shapefile, and calculate the area

in a GIS software. This conversion is easily feasible in R using the functions rasterToPolygons() and shapefile() (you will learn how to use shapefile() later), but rasterToPolygons() takes a large computational effort and thus a long processing time.
A faster and more efficient way to answer this question is to determine the number of change-pixels and calculate the area from this number and the area of a single pixel.

```
lossval <- getValues (loss) ## writes the pixel values to a table
table (lossval) ## determines the number of pixels per value
```

Q3.4: How can you determine the area covered by a single pixel on the ground?

This characteristic is the spatial resolution of the imagery that can be requested through the res() function.

```
area <- table (lossval)[2] * 30 * 30 / 10000 ## change area in hectares
```

Handling vector data

We can easily import and export shapefiles using the shapefile() function that will be introduced in an upcoming session. Besides this option, the raster-packge offers the option to use the getData() function to directly download vector (and also some raster) data sets from open repositories into your R workspace. Among the available data are administrative units that we will now use to determine the water surface loss per county.

```
adm <- getData ("GADM", country="USA", level=2) ## level indicates the
                       ## hierarchical level of adminstrative units
plot (adm)
```

From this data set, which covers the full US territory, we need to extract the outline of Shasta county and Trinity county. The vector data are stored in R as SpatialPolygonsDataFrame that is similar in handling to a regular data frame. To extract the two counties (i.e., to create a subset of the data frame), we need to know the row numbers in the data table corresponding to the counties.

```
View (adm@data)
```

Q3.5: Which row numbers have Shasta county and Trinity county?

Shasta county is in row 228, Trinity county in row 236.

```
shasta.county <- adm[ROW_NUMBER_OF_SHASTA_COUNTY,]
trinity.county <- adm[ROW_NUMBER_OF_TRINITY_COUNTY,]
```

R has no 'on the fly' projection ability. Spatial data that are analyzed jointly thus have to have the same spatial projection. This is not true for the county data and our image derived layers. Hence, the vector data have to be projected to the coordinate reference system of the images. This can be done with the spTransform() function for vector data and with the projectRaster() function for raster data.

```
crs (shasta.county) ## Geographic coordinates, WGS 84
crs (loss) ## UTM Zone 10, WGS 84
shasta.county.utm <- spTransform (shasta.county, crs (loss))
trinity.county.utm <- spTransform (trinity.county, crs (loss))
crs (shasta.county.utm)
```

The mask() function can then be used to 'clip' the loss layer with the county polygons.

```
shasta.loss <- mask (loss, shasta.county.utm)
plot (shasta.loss)
trinity.loss <- mask (loss, trinity.county.utm)
plot (trinity.loss)
```

Q3.6: What is the loss area in each of the two counties?

Shasta county = 4790.61 ha, Trinity county = 3280.68 ha

Visualization

R is not the best choice for creating nice maps, but some basic visualization is possible. Many of the common graphic functions can be directly applied to the raster data. We use these functions to create an overview map that shows the loss (in yellow) with the RGB image in the background. The map is written as *.pdf to your working directory.

```
loss <- reclassify (loss, matrix (c (0, NA), 1, 2)) ## make non-change pixels
                                                    ## transparent
pdf ("map.pdf", 5, 4, useDingbats=F) ## open a *.pdf
par (pin=c (4.01, 3), cex=0.7) ## set the size of the panel
plot (loss,  legend=F) ## plot the loss layer with coordinate axes
plotRGB (ls82014dps.topo, 4, 3, 2, stretch="lin", add=T) ## plot the background
plot (loss, col='#FFFF00', add=T, legend=F) ## plot the loss area
legend ("bottomright", pch=22, pt.bg="yellow","Decrease") ## add a legend
scalebar (20000, xy=c (530000, 4495000), type="bar", label=c (0, 10, 20),
          below="km") ## add a scalebar
dev.off() ## close the *.pdf
```