Introduction

Lex Comber

June 2016

Introduction

In this first session you will become familiar with the data and with some mapping operations in R using the `GISTools` package. Specifically you will:

- open a point dataset (shapefile) of a field validation exercise
- inspect the attributes and spatial properties of the data
- generate a correspondence matric and some classic measures of accuracy
- generate some surfaces

Background

Between 2008 and 2012, the late Prof Pete Fisher and I supervised a Libyan PhD student, Dr Abdulhakim Khmag who was undertaking research on *Fuzzy change detection*, using a case study in Libya. He was awarded his thesis in January 2013. For his research Abdulhakim generated a fuzzy land cover map of the area around Tripoli in Libya containing 5 classes: *Bare ground*, *Grazing land* , *Urban*, *Vegetation* and *Woodland*. He undertook some field work in 2010 and collected data to validate the land cover map. He selected 210 locations using a stratified random sample. In this the study area was divided into blocks, from which an average of 10 locations per block was selected randomly. At each location, Abdulhakim recorded the proportions of the different land cover classes for the area covered by the 30m pixel. He was the first research to collect fuzzy ground data. The data for this study includes the values recorded by Abdulhakim and the fuzzy class memberships from the land cover map. For most of this workshop we will work with the crisp, Boolean class derived from the memberships.

Data

First you will need to load some R packages:

```
install.packages("repmis", dep = T)
install.packages("GISTools", dep = T)
install.packages("spgwr", dep = T)
```

A zip file containing the data used in this practical can be downloaded from Lex Comber's GitHub site **https://github.com/lexcomber/LexTrainingR/blob/master/GW_Accuracy_Data.zip**. You could save this to your working directory and the load it directly. You can also check your current directory using the `getwd()` function.

Then the packages can be called and the data can be loaded into R after you have set the working directory using the `setwd()` function. The example from my computer is below:

```
library(GISTools)
library(spgwr)
setwd("/Users/geoaco/Desktop/my_docs_mac/leeds_work/workshop/datacode")
lib <- readShapePoly("libya.shp")
data <- read.csv("all_data_new2.csv")
```

Alternatively the code below loads into your working directory directly from a `RData` file also on the site.

```
library(GISTools)
library(spgwr)
library(repmis)
source_data("http://github.com/lexcomber/LexTrainingR/blob/master/SpatialAccuracy.RData?raw=True")
```

```
## [1] "lib"  "data"
```

The first thing to do is to examine the data. You will need to point R to your working directory which can be done through the menu or can be specified. You should examine the 'data' variable:

```
ls()
```

```
## [1] "data" "lib"
```

```
head(data)
```

```
##   PointID   East   North Urban_FS Vegetation_FS Woody_FS Grazing_FS
## 1       1 301847 3631819   0.2500         0.250    0.250       0.25
## 2       2 302491 3632155   0.0000         0.250    0.000       0.00
## 3       3 303834 3631818   0.0000         0.000    0.750       0.00
## 4       4 304480 3631008   0.2500         0.625    0.000       0.00
## 5       5 306691 3632967   0.6875         0.250    0.000       0.00
## 6       6 308175 3630784   0.0000         0.750    0.125       0.00
##   Bare_FS Boolean_FS Urban_RS Vegetation_RS Woody_RS Grazing_RS Bare_RS
## 1  0.0000          U    0.103         0.189    0.673      0.000   0.032
## 2  0.7500          B    0.256         0.036    0.387      0.000   0.321
## 3  0.2500          W    0.000         0.076    0.216      0.053   0.651
## 4  0.1250          V    0.112         0.372    0.215      0.185   0.110
## 5  0.0625          U    0.265         0.473    0.147      0.000   0.112
## 6  0.1250          V    0.000         0.365    0.312      0.143   0.175
##   Boolean_RS
## 1          V
## 2          B
## 3          W
## 4          V
## 5          V
## 6          V
```

This shows that the are a number of continuous and categorical variables. The first few columns of data indicate the ID (`PointID`) and the easting and northing of the data point.

The next set of 5 variables (columns 4 to 8), all with the suffix `_FS` are the fuzzy memberships to the 5 land cover classes as recorded in the field (FS for *Field Survey*). Column 9 records the Boolean or crisp class observed in the field. It is useful to consider this as the *Observed* or *Reference* data.

The third group of variables with the suffix `_RS` are the outputs from a *Remote Sensing* analysis using a Fuzzy Set classification (columns 10 to 14) with a Boolean / crisp class label in column 15.

You will mainly be using the Boolean data in columns 9 and 15.

Map the data

It is instructive to map the data to see what we have and the `eat` and `north` variables in `data` can be used to create a `SpaatialPointsDataFrame` class of object. These are defined in the `sp` package which is loaded with the `GISTools` package.

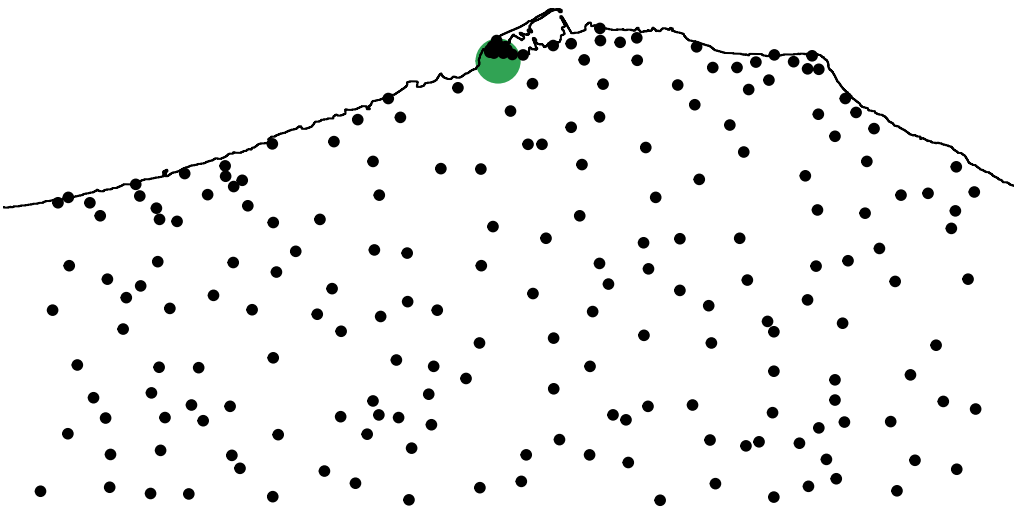
```
# Convert data to SPDF - spatial data
# 1.define a projection - see http://spatialreference.org/
lyb.proj <- CRS("+proj=utm +zone=33 +ellps=WGS84 +datum=WGS84 +units=m +no_defs ")
# 2. then use this to project the data from its coordinates
data.spdf <- SpatialPointsDataFrame(coords = data[,2:3],
  data = data, proj4string = lyb.proj)
```

This can be mapped but is not very informative without context:

```
# This has not been run but you could do it!
plot(data.spdf, pch = 1)
```

A shapefile of Libya provides a bit of background and the location of Tripoli adding further context:

```
### define Tripoli for Figure 1 with dummy data
x <- 329266.6
y <- 3640653
loc <-cbind(x,y)
Tripoli <- SpatialPointsDataFrame(coords = loc, data = data[1,])
### Final Figure
plot(data.spdf, pch = 20, col = "white")
plot(Tripoli, pch = 19, add = T, cex = 3, col = "#31A354")
plot(data.spdf, add = T, pch = 20, cex = 1, col = "Black")
plot(lib, add = T)
```



The correspondence matrix

The table below shows the accuracy / confusion / error / validation matrix arising from the field data collection exercise. Overall accuracy is calculated from the diagonal and off-diagonal elements in the confusion matrix. User and producer accuracies are calculated from the diagonals and the marginal row and column totals. Overall accuracy describes the proportion of the total number of pixels that have the same class in the reference classified data for all classes. User and producer accuracies describe the errors related to individual classes.

So first, populate the correspondence matrix:

```
tab <- table(data$Boolean_RS, data$Boolean_FS)
```

This generates a table of *Predicted* or *Classified* (rows) against *Observed* (columns). Have a look:

```
tab
```

```
##
##      B  G  U  V  W
##   B 18  8  7  2  4
##   G  3 23  3  8  6
##   U  0  0 27  1  2
##   V  0  4  7 31  5
##   W  0  4  2 18 27
```

This can be made more user-friendly by adding some class names ad some marginal totals:

```
class.names.long <- c("Bare", "Grazing", "Urban", "Vegetation", "Woodland")
rownames(tab) <- class.names.long
colnames(tab) <- class.names.long
tab <- cbind(tab, rowSums(tab))
tab <- rbind(tab, colSums(tab))
rownames(tab)[6] <- "Total"
colnames(tab)[6] <- "Total"
```

Again have a look:

```
tab
```

```
##            Bare Grazing Urban Vegetation Woodland Total
## Bare         18       8     7          2        4    39
## Grazing       3      23     3          8        6    43
## Urban         0       0    27          1        2    30
## Vegetation    0       4     7         31        5    47
## Woodland      0       4     2         18       27    51
## Total        21      39    46         60       44   210
```

Then the marginal totals and diagonals can be used to determine User and Producer accuracies - see Congalton (1991) for a full explanation - the paper can be downloaded from http://uwf.edu/zhu/evr6930/2.pdf.

```
# Users accuracy
tmp <- vector(mode = "numeric", length = 6)
for (i in 1:5) {
    tmp[i] <- tab[i,i] / tab[i,6]
    }
tab <- cbind(tab, zapsmall(tmp, 3))
colnames(tab)[7] <- "Users"
# Producers accuracy
tmp <- vector(mode = "numeric", length = 7)
for (i in 1:5) {
    tmp[i] <- tab[i,i] / tab[6,i]
    }
tab <- rbind(tab, zapsmall(tmp, 3))
rownames(tab)[7] <- "Producers"
```

Then calculate the overall accuracy and include in the resultant element

```
tab[7,7] <- sum(diag(table(data$Boolean_FS,
  data$Boolean_RS)))/sum(table(data$Boolean_FS, data$Boolean_RS))
```

And print if you want to (note the use of the `round` function to round the table values)

```
round(tab, 2)
```

```
##             Bare Grazing Urban Vegetation Woodland Total Users
## Bare       18.00    8.00  7.00       2.00     4.00    39  0.46
## Grazing     3.00   23.00  3.00       8.00     6.00    43  0.54
## Urban       0.00    0.00 27.00       1.00     2.00    30  0.90
## Vegetation  0.00    4.00  7.00      31.00     5.00    47  0.66
## Woodland    0.00    4.00  2.00      18.00    27.00    51  0.53
## Total      21.00   39.00 46.00      60.00    44.00   210  0.00
## Producers   0.86    0.59  0.59       0.52     0.61     0  0.60
```

The table could be written to a file if you want:

```
write.csv(tab, file = "Table1.csv")
```

Correspondence, Probabilities and Regression

The *Overall*, *User* and *Producer* accuracies can be considered as probabilities: the probability that any pixel is correctly classified (*Overall*), the probability that any pixel of any *Observed* class in the field is correctly predicted (*Producer*), and the probability that any *Predicted* class in the classified data is correctly predicted (*User*). These can also be estimated using ordinary logistic regressions. First, a logit function needs to be defined to transform any value, q:

$$logit(q) = exp(q)/(1 + exp(q))(eqn1)$$

Overall Accuracy

Overall accuracy, Ao, can be estimated using a reduced logistic regression model that only contains an intercept term, b:

$$Ao = logit(b)(eqn2)$$

This returns an estimate of the probability of Ao being equal to 1. The code to do this using a *Genalized Linear Model* (GLM) is shown below:

```
# Define the a logit function
alogit <- function(x){exp(x)/(1+exp(x))}
# Create a binary variable where
# the Field data match the Predicted data
res <- vector(length = dim(data)[1])
for (i in 1: dim(data)[1]) {
    if (data$Boolean_RS[i] == data$Boolean_FS[i]) {
        res[i] <- 1
    }}
# Then use a GLM to do a logistic regression
mod0 <- glm(res~1,family= binomial)
mod.coefs <- mod0$coefficients
mod.coefs[2] <-sum(mod.coefs) #logit ea+c
mod.ov <- alogit(mod.coefs[2]) #n1
cat("overall accuracy:", mod.ov)
```

```
## overall accuracy: 0.6
```

This seeks to predicts the degree to which the data is equal to 1.

User and Producer Accuracies

For *User* and *Producer* accuracies we need a different model. It is useful to consider a specific example, in this case the class of Grazing Land. From the table, we can see that it can be represented by binary vectors of 210 elements, with the Field data (reference) a vector with 39 elements scored as 1, the classified data (predicted) having 43 elements scored as 1, with 23 data elements scored as 1 in common.

User accuracy can be estimated using a logistic regression to analyse the reference data against the classified data in the following way:

$$P(y = 1) = logit(b_0 + b_1 x_1)(eqn3)$$

where $P(y = 1)$ is the probability that the reference land-cover class, y, is correctly predicted by the classified data, x_1, b_0 is the intercept term and b_1 is the slope. this generates the probability that the reference data is the class (i.e. is `TRUE` or `equals 1`), given that the classified data is the class (i.e. also `equals 1`).

The producer accuracy is estimated by inverting the response and explanatory variables:

$$P(x = 1) = logit(b_0 + b_1 y_1)(eqn4)$$

where $P(x = 1)$ is the probability that the classified land-cover class is correctly predicted by the reference data, y_1, b_0 is the intercept term and b_1 is the slope.

For the example of grazing land the code to do this is as follows. Note that the first stage is to create the binary vectors `fs.class` and `rs.class` for denote the field survey and remote sensing classes:

```
class.list <- unique(data$Boolean_RS)[order(unique(data$Boolean_RS))]
# NB: i = 2 below specifies 'G' for Grazing Land - look at class.list
# change i to analyse other classes
i = 2
class <- class.list[i]
fs.class <- (data$Boolean_FS == class) * 1 # y above
rs.class <- (data$Boolean_RS == class) * 1  # x above
# Combine these for use in the GLMs
fsrs <- data.frame(cbind(fs.class,rs.class))
```

These data can then be used as inputs to the GLM below. Note that the terms / inputs to the GLM are inverted in the code below:

```
# User Accuracy
mod1 <- glm(fs.class~rs.class,data = fsrs,family= binomial)
mod.coefs <- mod1$coefficients
mod.coefs[2] <-sum(mod.coefs)
# P(x = 1|y = 1)
mod.user <- alogit(mod.coefs[2])
cat("user accuracy:", round(mod.user, 3))
```

```
## user accuracy: 0.535
```

```
# Producer - invert terms
mod2 <- glm(rs.class~fs.class,data = fsrs,family= binomial)
mod.coefs <- mod2$coefficients
mod.coefs[2] <-sum(mod.coefs) #logit ea+c
mod.prod <- alogit(mod.coefs[2])
cat("producer accuracy:", round(mod.prod, 3))
```

```
## producer accuracy: 0.59
```

Summary

You have been introduced to the data and the usual way of developing measures of accuracy for remote sensing products such as land cover. The statistical bases for the probabilistic accuracy measures that are commonly derived from the correspondence matrix have been shown. In the next section you will start to develop spatially distributed measures of accuracy.