Mapping spatial accuracy

Lex Comber

June 2016

Introduction

In the last section you created a spatially distributed measure of overall accuracy. In this you created a variable that recorded whether the *observed* class in the field was the same as the class *predicted* by the classification algorithm. It recorded a series of `1`s and `0`s to indicate whether the classes matched or not. These were analysed using classic logistic regression (Generalized Linear Models) to get *a*spatial measures of accuracy and then using Geographically Weighted logistic Regression to generate spatially distributed measures.

In this section you will develop further measures of accuracy that are commonly used in remote sensing: User and Producer accuracies and an estimate of the Kappa statistic $\hat{\kappa}$, also known as k-hat. A full description of these can be found in the classic Congalton (1991) paper at http://uwf.edu/zhu/evr6930/2.pdf.

Again we will calculate these measures from first principles as probabilities and then develop *geographically weighted* versions.

Data set up

As before, you will need to load the data from he **`github`** resource

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

```
## [1] "data"   "roilib"
```

And have a look at the what you have

```
ls()
```

```
## [1] "data"   "roilib"
```

```
head(data.frame(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
```

And the code from the *Introduction* can be re-used to create the correspondence matrix

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

Then as before, calculate User and Producer accuracies - you should have all of this code already!

```
# 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"
```

Calculate the overall accuracy:

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

and check the final table:

```
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
```

Recap: User Accuracy

So from the table in the `tab` variable you can see that for example Grazing land has a User Accuracy of 0.535. User accuracy seeks to provide a measure of per-class accuracy that describes how probable it is that a pixel (or segmented object) of that has been labelled as *Grass* will actually be that class if it was visited in the field for example. It indicates the errors of commission (inclusion) and for the potential user of the map it indicates the probability of correctly finding the class indicated on the map present on the ground.

User accuracies, as stated in the *Introduction* session, can be estimated using a logistic regression to analyse the reference data against the classified data in the following way:

$$P(y = 1) = logit(b_0 + b_1 x_1)(eqn1)$$

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`).

In the *Introduction* session you used a GLM (Generalized Linear Model) approach to calculate the probabilities associated with Grazing land User accuracy.

Recall that the approach for User accuracy was as follows:

1. for class `x`, create a `data.frame` containing:

a) the data locations where the remote sensing indicated class `x` [0 or 1]
b) the data locations where the field visit indicated class `x` [0 or 1]

2. construct a GLM of the extent to which the field class was predicted by the remote sensing class (*Observed* was predicted by *Predicted*)
3. Manipulate the resulting coefficients (`sum` and the `alogit` function) to determine the User accuracy (remembering that User accuracy is $P(FS = 1|RS = 1)$

The code for doing this is for the class of *Grazing Land* is repeated below:

```r
# 1. Create a data.frame
class.list <- unique(data$Boolean_RS)[order(unique(data$Boolean_RS))]
# 'G' is for Grazing Land
class <- class.list[2]
# have a look!
class
```

```
## [1] G
## Levels: B G U V W
```

```r
# 1a where the RS indicated the class
rs.class <- (data$Boolean_RS == class) * 1
# 1b where the FS indicated the class
fs.class <- (data$Boolean_FS == class) * 1
# join together
fsrs <- data.frame(cbind(fs.class,rs.class))
```

Construct the GLM:

```r
# 2. GLM for User Accuracy
mod1 <- glm(fs.class~rs.class,data = fsrs,family= binomial)
```

Now manipulate the coefficients:

```r
# 3. Define and apply the alogit function
alogit <- function(x){exp(x)/(1+exp(x))}
mod.coefs <- mod1$coefficients
mod.coefs[2] <-sum(mod.coefs)
# P(y = 1|x = 1)
mod.user <- alogit(mod.coefs[2])
cat("user accuracy:", round(mod.user, 3))
```

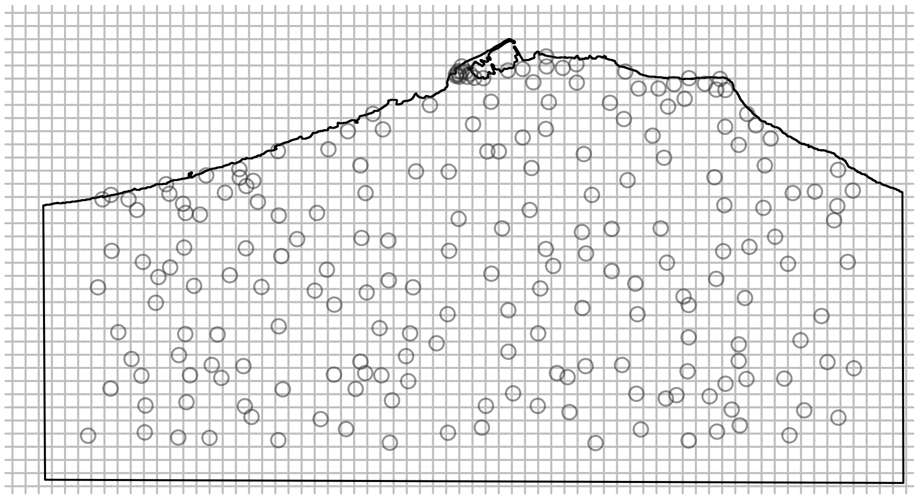
```
## user accuracy: 0.535
```

GW User Accuracy

So the User accuracy is the probabilities arising from a GLM constructed from binary variables (i.e. containing `1`s and `0`s) describing whether the class was *observed* in the field (`fs.class`) and whether the class was *predcited* by the remote sensing classification (`rs.class`). In the global analysis all 210 data points were used to construct the model.

Recall that in Geographically Weighted approaches, local models are constructed, in this case at each location on a grid shown below. The `fsrs` variable created above can be used to construct `SpatialPointsDataFrame` object using the coordinates in the `data` variable and used as input to the GW GLM function, `ggwr`:

```r
fsrs.spdf <- SpatialPointsDataFrame(coords = data[,2:3],
  data = data.frame(fsrs))
grid <- SpatialPoints(expand.grid(x=seq(295000,363000,by=1000),
  y=seq(3610000,3646000,by=1000)))
plot(grid, cex = 0.7, col = "grey")
plot(roilib, add = T)
plot(fsrs.spdf, add = T, pch = 1, col = "#25252580")
```



At each location, the data under the kernel are weighted by their distance to the kernel centre and used to construct a *local* GLM from which local probabilities such as User accuracy can be computed. In this case the nearest 15% of the data points are be used to construct a GLM at each location on the grid. Note that the kernel bandwidth can also be set as a distance. You should examine the help pages for the `ggwr` function to see the parameters that it takes.

```
bw = 0.15
```

The GW model can be constructed. Note the similarity in the parameters that the `glm` and `ggwr` functions take:

```
gwr.mod <- ggwr(fs.class~rs.class, data = fsrs.spdf,
  adapt = bw,fit.points=grid, family= binomial)
```

You can examine the GW model (`gwr.mod`) and the SpatialDataFrame (SDF) of the GW model:

```
gwr.mod
```

```
## Call:
## ggwr(formula = fs.class ~ rs.class, data = fsrs.spdf, adapt = bw,
##     fit.points = grid, family = binomial)
## Kernel function: gwr.Gauss
## Adaptive quantile: 0.15 (about 31 of 210 data points)
## Fit points: 2553
## Summary of GWR coefficient estimates at fit points:
##                Min. 1st Qu. Median 3rd Qu.   Max.
## X.Intercept. -4.566  -2.664 -2.197  -1.883 -1.621
## rs.class      2.016   2.316  2.481   2.643  3.130
```

```
head(data.frame(gwr.mod$SDF))
```

```
##      sum.w X.Intercept. rs.class dispersion working_resids      x       y
## 1 59.74886    -2.018269 2.524138          1             NA 295000 3610000
## 2 59.99208    -2.009465 2.524937          1             NA 296000 3610000
## 3 60.02202    -1.999324 2.525970          1             NA 297000 3610000
## 4 59.63173    -1.986672 2.527510          1             NA 298000 3610000
## 5 58.72686    -1.970674 2.529622          1              NA 299000 3610000
## 6 58.32436    -1.956735 2.530850          1              NA 300000 3610000
##   optional
## 1     TRUE
## 2     TRUE
## 3     TRUE
## 4     TRUE
## 5     TRUE
## 6     TRUE
```

And the coefficients can be manipulated in the same way as before but this time from the `gwr.mod$SDF`:

```
coefs <- data.frame(gwr.mod$SDF)[,2:3]
coefs[,2] <- rowSums(coefs)
# P(x = 1|y = 1)
gwr.user <- alogit(coefs[,2])
```

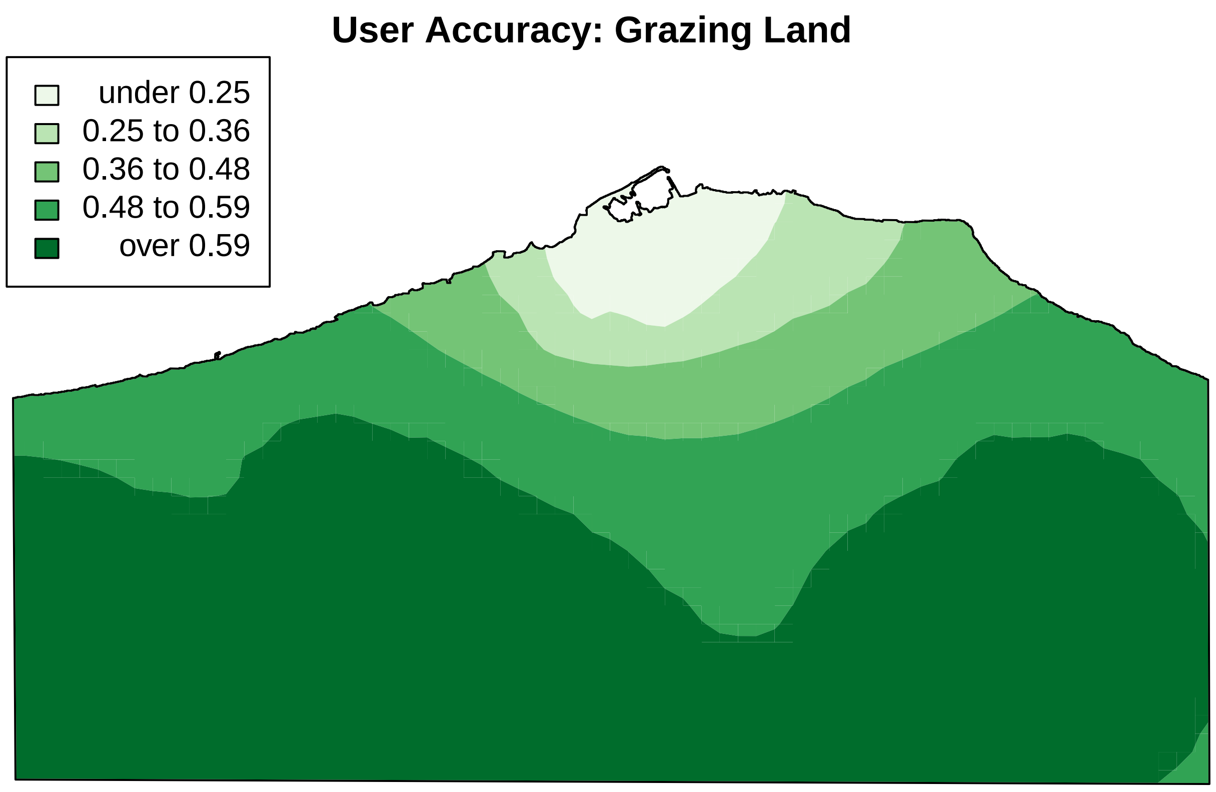
The spatial variation in the coefficients is indicated by the distribution of User accuracy values:

```
summary(gwr.user)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1342  0.4794  0.5751  0.5351  0.6205  0.7088
```

Here we can see that there is evidence of considerable variation - compare the 1st and 3rd quartiles of the distribution. As for the Overall Accuracy in the last session, these can be mapped:

```
shades = auto.shading(gwr.user, n=5,cols=brewer.pal(5,"Greens"),
  cutter=rangeCuts, digits = 2)
gwr.spdf = SpatialPixelsDataFrame(gwr.mod$SDF, data.frame(gwr.user))
par(mar = c(0,0,1,0))
level.plot(gwr.spdf,shades)
lib.masker = poly.outer(gwr.spdf, roilib, extend = 100)
add.masking(lib.masker)
plot(roilib, add = T)
choro.legend(297000, 3650000,shades)
title("User Accuracy: Grazing Land")
```



So the *global* measure of User accuracy was 0.535 and *locally* this varies from 0.479 to 0.62:

```r
# global
round(as.vector(mod.user), 3)
```

```
## [1] 0.535
```

```r
# local
round(summary(gwr.user)[c(2,5)],3)
```

```
## 1st Qu. 3rd Qu.
##   0.479   0.620
```

Recap: Producer Accuracy

So from the table in the `tab` variable you can see that for example Grazing land has a Producer Accuracy of 0.59. Producer accuracy provides an estimate of the probability that a reference pixel is correctly labelled in the classified data. So for example, that a pixel labelled as *Grass* in the field will actually be that class in the data. It indicates the errors of omission (exclusion) and for the producer of the map it indicates the probability that the features of interest are omitted from the classified data.

Producer accuracies, as stated in the *Introduction* session, can be estimated using a logistic regression to analyse the reference data against the classified data in the following way:

$$P(x = 1) = logit(b_0 + b_1 y_1)(eqn2)$$

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. This generates the probability that the classified data is the class (i.e. is `TRUE` or `equals 1`), given that the reference data is the class (i.e. also `equals 1`).

Again recall that in the *Introduction* session you used a GLM (Generalized Linear Model) to calculate the probabilities associated with Grazing land Producer accuracy and the approach was as follows:

1. for class `x`, create a `data.frame` containing:

a) the data locations where the remote sensing indicated class `x` [0 or 1]
b) the data locations where the field visit indicated class `x` [0 or 1]

2. construct a GLM of the extent to which the remote sensing class was predicted by the reference or field class (*Predicted* was predicted by *Observed*)
3. Manipulate the resulting coefficients (`sum` and the `alogit` function) to determine the Producer accuracy (remembering that Producer accuracy is $P(RS = 1|FS = 1)$

The code for doing this is for the class of *Grazing Land* is repeated below, with Step 1 having already been done for the User accuracy above so the GLM can be constructed:

```
# 2. GLM for Producer Accuracy
mod2 <- glm(rs.class~fs.class,data = fsrs,family= binomial)
```

Now manipulate the coefficients:

```
# 3. Define and apply the alogit function
mod.coefs <- mod2$coefficients
mod.coefs[2] <-sum(mod.coefs)
# P(x = 1|y = 1)
mod.producer <- alogit(mod.coefs[2])
cat("Producer accuracy:", round(mod.producer, 3))
```

```
## Producer accuracy: 0.59
```

GW Producer Accuracy

So the Producer accuracy is the probabilities arising from a GLM constructed from binary variables (i.e. containing `1`s and `0`s) describing whether the class was *predicted* in the remote sensing classification (`rs.class`) and whether the class was *observed* by the field (`fs.class`) . In the global analysis all 210 data points were used to construct the model.

As for GW User accuracy, local models are constructed at each location on the grid using the `fsrs.spdf` `SpatialPointsDataFrame` using the nearest 15% of the data points. The GW Producer model is as follows:

```
gwr.mod <- ggwr(rs.class~fs.class, data = fsrs.spdf,
  adapt = bw,fit.points=grid, family= binomial)
```

And again, you can examine the GW model (`gwr.mod`) and the SpatialDataFrame (SDF) of the GW model:

```
gwr.mod
```

```
## Call:
## ggwr(formula = rs.class ~ fs.class, data = fsrs.spdf, adapt = bw,
##     fit.points = grid, family = binomial)
## Kernel function: gwr.Gauss
## Adaptive quantile: 0.15 (about 31 of 210 data points)
## Fit points: 2553
## Summary of GWR coefficient estimates at fit points:
##                  Min. 1st Qu. Median 3rd Qu.   Max.
## X.Intercept. -2.418  -2.257 -2.096  -1.859 -1.624
## fs.class      2.016   2.316  2.481   2.643  3.130
```

```
head(data.frame(gwr.mod$SDF))
```

```
##      sum.w X.Intercept. fs.class dispersion working_resids      x       y
## 1 59.74886    -2.238048 2.524138          1             NA 295000 3610000
## 2 59.99208    -2.240152 2.524937          1             NA 296000 3610000
## 3 60.02202    -2.242691 2.525970          1             NA 297000 3610000
## 4 59.63173    -2.246137 2.527510          1             NA 298000 3610000
## 5 58.72686    -2.250661 2.529622          1             NA 299000 3610000
## 6 58.32436    -2.253822 2.530850          1             NA 300000 3610000
##   optional
## 1     TRUE
## 2     TRUE
## 3     TRUE
## 4     TRUE
## 5     TRUE
## 6     TRUE
```

The coefficients can be manipulated in the same way as before:

```
coefs <- data.frame(gwr.mod$SDF)[,2:3]
coefs[,2] <- rowSums(coefs)
# P(x = 1|y = 1)
gwr.producer <- alogit(coefs[,2])
```

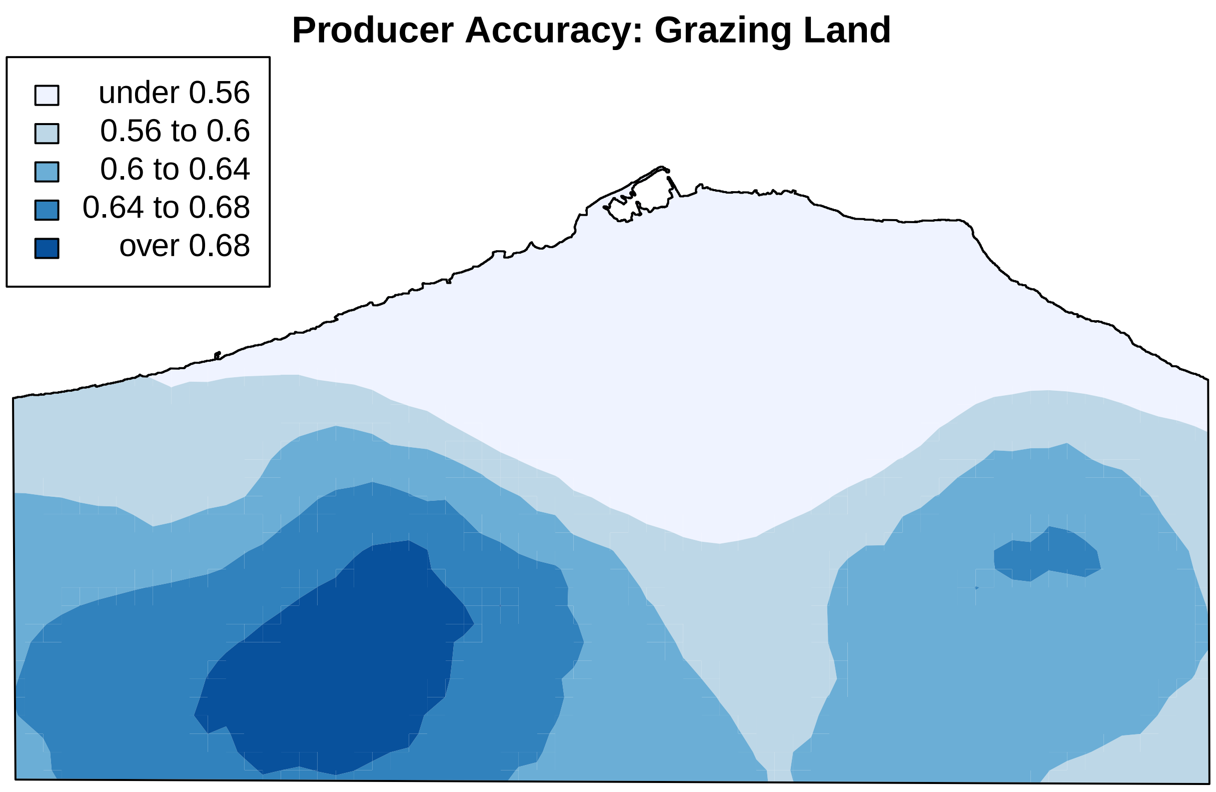
The spatial variation in the coefficients is indicated by the distribution of User accuracy values:

```
summary(gwr.producer)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.5195  0.5733  0.6078  0.6084  0.6386  0.7198
```

Here there is less variation than for the User accuracy from the 1st and 3rd quartiles of the distribution and these can be mapped:

```
shades = auto.shading(gwr.producer, n=5,cols=brewer.pal(5,"Blues"),
  cutter=rangeCuts, digits = 2)
gwr.spdf = SpatialPixelsDataFrame(gwr.mod$SDF, data.frame(gwr.user))
par(mar = c(0,0,1,0))
level.plot(gwr.spdf,shades)
lib.masker = poly.outer(gwr.spdf, roilib, extend = 100)
add.masking(lib.masker)
plot(roilib, add = T)
choro.legend(297000, 3650000, shades)
title("Producer Accuracy: Grazing Land")
```



So the *global* measure of Producer accuracy was 0.59 and *locally* this varies from 0.479 to 0.639:

```
# global
round(as.vector(mod.producer), 3)
```

```
## [1] 0.59
```

```
# local
round(summary(gwr.producer)[c(2,5)],3)
```

```
## 1st Qu. 3rd Qu.
##   0.573   0.639
```

Summary

In this section you have applied Geographically Weighted approaches to generated *local* models of user, producer and Overall accuracies. These spatially distributed measures can be mapped. In the next section you will create develop some code to automate these operations in functions.