Putting it all Together

Lex Comber and Paul Harris

July 2016

Data set up

You will need to load the data.

This can be done locally 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")
roilib <- readShapePoly("lib_clip.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] "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
```

Putting it all together with Loops and Functions

So far we have examined overall accuracy and per class User and Producer accuracies individually, showing the original *aspatial* or *global* measure, and then a summary of the distribution of the related *geographically weighted* values. The accuracy surfaces have then been mapped using the `level.plot` function in the `GISTools` package.

The code above has been developed and described step by step to walk you through the process.

It might be useful to develop functions to automate some of these operations. And then perhaps to combine some of these functions so that for any given class, a number of accuracy measures are returned. The code in this section starts to do this.

Overall Accuracy

Remember that Overall Accuracy is calculated from the sum of the diagonals in the accuracy / confusion / error / validation matrix that compares *Predicted* against *Observed* classes, divided by the total number of data points.

```
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
    }}
```

This can be calculated from the data directly:

```
cat("overall accuracy:", sum(res)/length(res))
```

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

Or from a logistic regression and a the `alogit` function:

```
mod.ov <- glm(res~1,family= binomial)
mod.coefs <- mod.ov$coefficients
mod.coefs[2] <-sum(mod.coefs)
alogit <- function(x){exp(x)/(1+exp(x))}
mod.ov <- alogit(mod.coefs[2])
cat("overall accuracy:", mod.ov)
```

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

And if a SpatialPointsDataFrame object is created this can be used in a GW approach with the ggwr function. First some variables and parameters need to be set:

```
bw = 0.15
grid <- SpatialPoints(expand.grid(x=seq(295000,363000,by=1000),
  y=seq(3610000,3646000,by=1000)))
res.spdf <- SpatialPointsDataFrame(coords = data[,2:3],
  data = data.frame(res))
```

Then the GW model can be constructed:

```
gwr.mod <- ggwr(res~1, data = res.spdf, adapt = bw,
  fit.points = grid, family= binomial)
gwr.ov <- alogit(data.frame(gwr.mod$SDF)[,2])
```

And the variation in the distribution of Overall Accuracy values examined:

```
summary(gwr.ov)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.5717  0.5962  0.6057  0.6053  0.6104  0.6446
```

Create a Function

A function to do all of this can be assembled from the above code snippets.

First of all it would be useful to create a SpatialPointsDataFrame of the all of the original data - this is the kind of dataset that you might bring as shapefile to this kind of analysis:

```
spdf <- SpatialPointsDataFrame(coords = data[,2:3], data = data.frame(data))
head(data.frame(spdf))
```

```
##   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 East.1 North.1 optional
## 1          V 301847 3631819     TRUE
## 2          B 302491 3632155     TRUE
## 3          W 303834 3631818     TRUE
## 4          V 304480 3631008     TRUE
## 5          V 306691 3632967     TRUE
## 6          V 308175 3630784     TRUE
```

Then define a function that takes this `spdf` as input and returns a `SpatialGridDataFrame` with the results of the geographically weighted analysis:

```
gw.accuracy <- function(spdf, Field.class = "Boolean_FS",
    RS.class = "Boolean_RS", bw = 0.15, grid=grid, family= binomial){
  # compare predicted and observed (classified and field)
  res <- as.vector(spdf@data[RS.class] == spdf@data[Field.class]) * 1
  # notice how the line of code above replaces the specification of
  # the  res vector, the for loop etc
  # Commented Out: A-spatial overall accuracy
  # cat("Overall accuracy:",sum(res)/length(res), "\n")
  # GW approach
  alogit <- function(x){exp(x)/(1+exp(x))}
  gwr.mod <- ggwr(res~1, data = spdf, adapt = bw,
    fit.points = grid, family= binomial)
  gwr.ov <- alogit(data.frame(gwr.mod$SDF)[,2])
  # Commented Out: Summary of the GW variation
  # cat("GW overall accuracy:", summary(gwr.ov))
  # create SpatialPixelsDF to return from the function
  tmp <- as(gwr.mod$SDF, "SpatialPixels")
  gw.spdf <-SpatialPixelsDataFrame(tmp, data.frame(gwr.ov))
  return(gw.spdf)
}
```

And then this function can be run:

```
tmp <- gw.accuracy(spdf, Field.class = "Boolean_FS",
    RS.class = "Boolean_RS", bw = 0.15, grid, family= binomial)
```

And the results in `tmp` can be evaluated:

```
summary(tmp)
```

```
## Object of class SpatialPixelsDataFrame
## Coordinates:
##        min     max
## x  294500  363500
## y 3609500 3646500
## Is projected: NA
## proj4string : [NA]
## Number of points: 2553
## Grid attributes:
##   cellcentre.offset cellsize cells.dim
## x            295000     1000        69
## y           3610000     1000        37
## Data attributes:
##      gwr.ov
##  Min.   :0.5717
##  1st Qu.:0.5962
##  Median :0.6057
##  Mean   :0.6053
##  3rd Qu.:0.6104
##  Max.   :0.6446
```

This shows that are is NOT much variation in the Overall Accuracy - examine the 1st and 3rd quartile range. There is no point in mapping this!

Create a Mapping Function

And a mpping function can be dfined to map `SpatialPixelsDataFrame` objects:

```
gw.mapping <- function(grd, index = 1, n = 4, cols=brewer.pal(n=4,'Reds'),
    bounding.poly = roilib, x = 297000, y = 3650000, tit = "My Title") {
  z = data.frame(grd)[, index]
  zz = z[!is.na(z)]
  shades <- auto.shading(zz, n=n, cols = cols)
  level.plot(grd, index = index, shades = shades)
  masker = poly.outer(grd, bounding.poly, extend = 100)
  add.masking(masker)
  plot(bounding.poly, add = T)
  choro.legend(x, y,shades)
  title(tit)}
```

Hint the `locator` function can be used to identify the `x` and `y` coordinates for the `choro.legend` function.

So now there are two functions, with a number of default parameters that the user can change, that can be called to calculate and then map overall accuracy. This is in the general form:

```
# this code will not run!
tmp <- gw.accuracy(spdf)
gw.mapping(tmp)
```

User and Producer Accuracies

So far we have just mapped User and Producer accuracies for the class of `Grazing Land`. This class was chosen to exemplify the GW methods because it *does* exhibit spatial variation in these accuracies. But this might be the case for all classes. So, in this section we will first construct a function to compute the the GW User and Producer accuracies for all classes. Then we will examine the spatial variation and select per-class accuracies to map.

The stages in this are:

1. For each class, construct a variable pair of remote sensing (*Predcited*) class and field (*Observed*) class:
2. Compute the GW User accuracy
3. Compute the GW Producer accuracy
4. Add the results to a data frame
5. Create a SpatialPixelsDataFrame
6. Evaluate the variation of the GW accuracy measures

```
# define the classes
class.list <- unique(data$Boolean_RS)[order(unique(data$Boolean_RS))]
# pass this into a loop
for (i in 1:length(class.list) ){
  class <- class.list[i]
  # 1. Construct the variable pair
```

```
  # RS indicates the class
  rs.class <- (data$Boolean_RS == class) * 1
  # FS indicates the class
  fs.class <- (data$Boolean_FS == class) * 1
  # join together
  fsrs <- data.frame(cbind(fs.class,rs.class))
  # convert to SPDF
  fsrs.spdf <- SpatialPointsDataFrame(coords = data[,2:3],
    data = data.frame(fsrs))
  # 2. GW User accuracy
  # define a bandwidth
  bw = 0.15
  # construct GW model
  gwr.mod <- ggwr(fs.class~rs.class, data = fsrs.spdf,
    adapt = bw,fit.points=grid, family= binomial)
  coefs <- data.frame(gwr.mod$SDF)[,2:3]
  coefs[,2] <- rowSums(coefs)
  alogit <- function(x){exp(x)/(1+exp(x))}
  gwr.user <- alogit(coefs[,2])
  # 3. GW Producer accuracy
  gwr.mod <- ggwr(rs.class~fs.class, data = fsrs.spdf,
    adapt = bw,fit.points=grid, family= binomial)
  coefs <- data.frame(gwr.mod$SDF)[,2:3]
  coefs[,2] <- rowSums(coefs)
  gwr.producer <- alogit(coefs[,2])
  # 4. Add these to the data frame
  # define some variable names
  tit.user <- sprintf("%s-User", class)
  tit.producer <- sprintf("%s-Producer", class)
  df <- data.frame(gwr.user, gwr.producer)
  # name the df
  names(df) <- c(tit.user, tit.producer)
  # and combine
  if(i ==1) df.res <- df
  if(i > 1) df.res <- data.frame(df.res, df)
}
```

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

```
summary(df.res)
```

```
##      B.User         B.Producer        G.User         G.Producer
##  Min.   :0.2720   Min.   :0.7048   Min.   :0.1342   Min.   :0.5195
##  1st Qu.:0.4118   1st Qu.:0.8119   1st Qu.:0.4794   1st Qu.:0.5733
##  Median :0.4642   Median :0.8480   Median :0.5751   Median :0.6078
##  Mean   :0.4577   Mean   :0.8505   Mean   :0.5351   Mean   :0.6084
##  3rd Qu.:0.5297   3rd Qu.:0.8967   3rd Qu.:0.6205   3rd Qu.:0.6386
##  Max.   :0.5737   Max.   :0.9548   Max.   :0.7088   Max.   :0.7198
##      U.User         U.Producer        V.User         V.Producer
##  Min.   :0.6689   Min.   :0.4661   Min.   :0.4431   Min.   :0.3326
##  1st Qu.:0.8269   1st Qu.:0.5257   1st Qu.:0.6208   1st Qu.:0.4942
##  Median :0.9163   Median :0.5901   Median :0.6631   Median :0.5416
##  Mean   :0.8864   Mean   :0.5800   Mean   :0.6588   Mean   :0.5307
```

```
##   3rd Qu.:0.9533   3rd Qu.:0.6382   3rd Qu.:0.7068   3rd Qu.:0.5765
##   Max.   :0.9932   Max.   :0.7042   Max.   :0.7783   Max.   :0.6206
##       W.User          W.Producer
##   Min.   :0.3620   Min.   :0.3271
##   1st Qu.:0.5310   1st Qu.:0.5403
##   Median :0.5447   Median :0.6252
##   Mean   :0.5449   Mean   :0.6105
##   3rd Qu.:0.5684   3rd Qu.:0.6922
##   Max.   :0.6375   Max.   :0.7707
```

And these can used to construct a `SpatialPixelsDataFrame` object:

```
tmp <- as(gwr.mod$SDF, "SpatialPixels")
gw.all.spdf <-SpatialPixelsDataFrame(tmp, data.frame(df.res))
```

Create a function

This can be wrapped up into a function that takes the `spdf` variable created above

```
spdf <- SpatialPointsDataFrame(coords = data[,2:3], data = data.frame(data))
```

and returns a `SpatialPixelsDataFrame` object:

```
user.prod.accuracy <- function(spdf, Field.class = "Boolean_FS",
    RS.class = "Boolean_RS", bw = 0.15, grid=grid, family= binomial){
  class.list <- unique(spdf@data[,RS.class])[order(unique(spdf@data[,RS.class]))]
  # pass this into a loop
  for (i in 1:length(class.list) ){
    class <- class.list[i]
    # 1. Construct the variable pair
    # RS indicates the class
    rs.class <- (data$Boolean_RS == class) * 1
    # FS indicates the class
    fs.class <- (data$Boolean_FS == class) * 1
    # join together
    fsrs <- data.frame(cbind(fs.class,rs.class))
    # convert to SPDF
    fsrs.spdf <- SpatialPointsDataFrame(coords = data[,2:3],
      data = data.frame(fsrs))
    # 2. GW User accuracy
    # define a bandwidth
    bw = 0.15
    # construct GW model
    gwr.mod <- ggwr(fs.class~rs.class, data = fsrs.spdf,
      adapt = bw,fit.points=grid, family= binomial)
    coefs <- data.frame(gwr.mod$SDF)[,2:3]
    coefs[,2] <- rowSums(coefs)
    alogit <- function(x){exp(x)/(1+exp(x))}
    gwr.user <- alogit(coefs[,2])
    # 3. GW Producer accuracy
    gwr.mod <- ggwr(rs.class~fs.class, data = fsrs.spdf,
      adapt = bw,fit.points=grid, family= binomial)
```

```
    coefs <- data.frame(gwr.mod$SDF)[,2:3]
    coefs[,2] <- rowSums(coefs)
    gwr.producer <- alogit(coefs[,2])
    # 4. Add these to the data frame
    # define some variable names
    tit.user <- sprintf("%s-User", class)
    tit.producer <- sprintf("%s-Producer", class)
    df <- data.frame(gwr.user, gwr.producer)
    # name the df
    names(df) <- c(tit.user, tit.producer)
    # and combine
    if(i ==1) df.res <- df
    if(i > 1) df.res <- data.frame(df.res, df)
  }
  tmp <- as(gwr.mod$SDF, "SpatialPixels")
  gw.spdf <-SpatialPixelsDataFrame(tmp, data.frame(df.res))
  return(gw.spdf)
}
```

And then this can be called

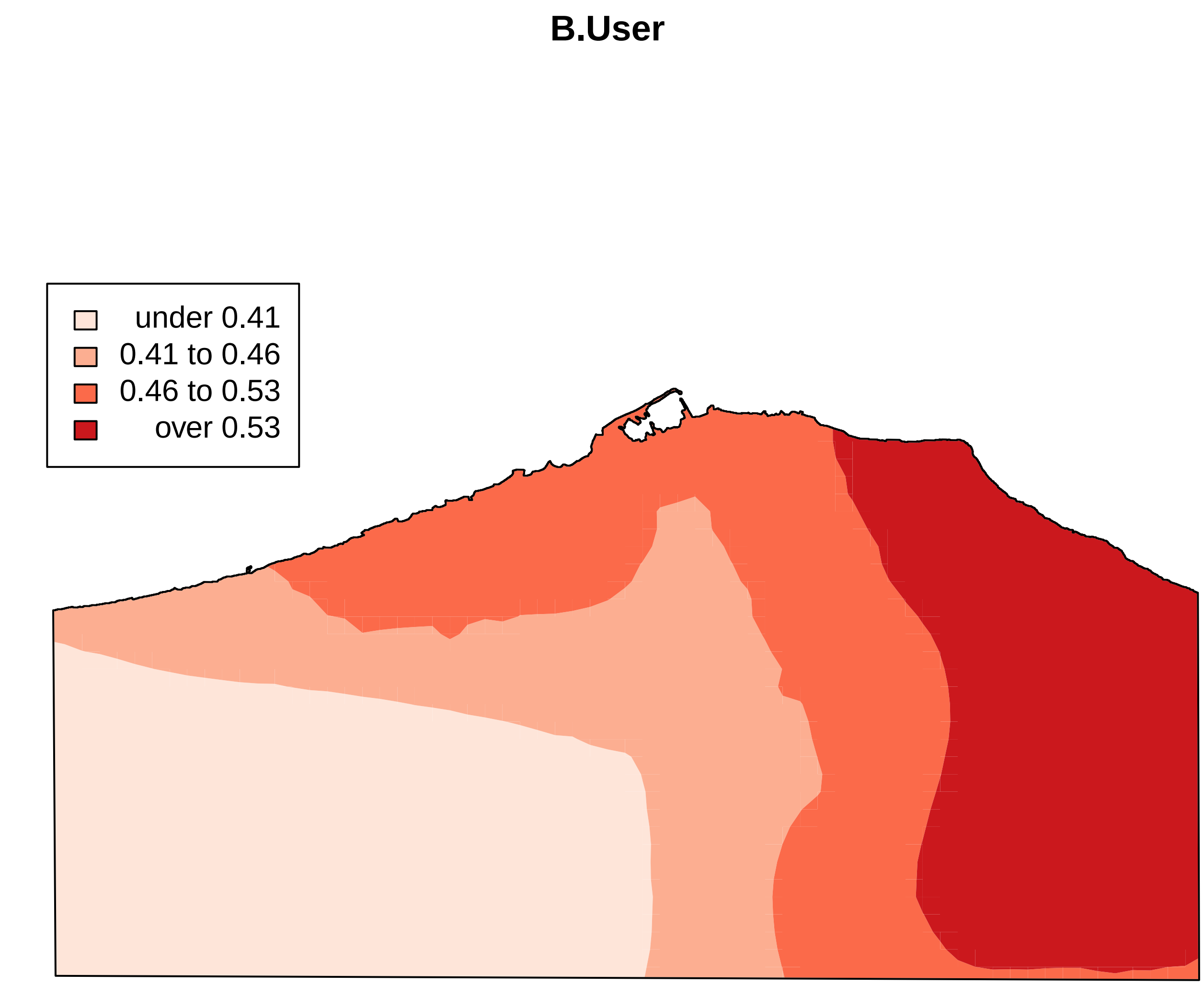
```
gwr.all.spdf <- user.prod.accuracy(spdf, Field.class = "Boolean_FS",
    RS.class = "Boolean_RS", bw = 0.15, grid=grid, family= binomial)
```

And the contents examined again:

```
summary(gwr.all.spdf@data)
```

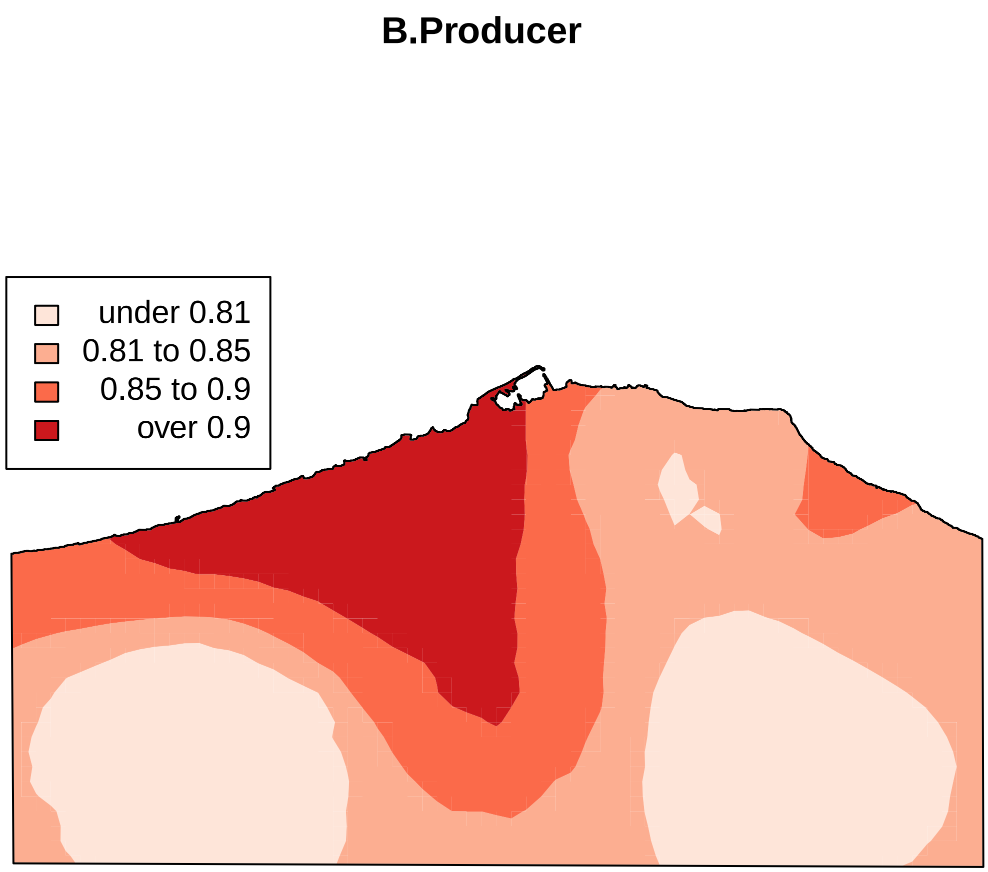
The elements of the `gwr.all.spdf` variable can be mapped using the function defined earlier:

```
par(mar = c(0,0,1,0))
gw.mapping(gwr.all.spdf, tit = names(gwr.all.spdf)[1])
```



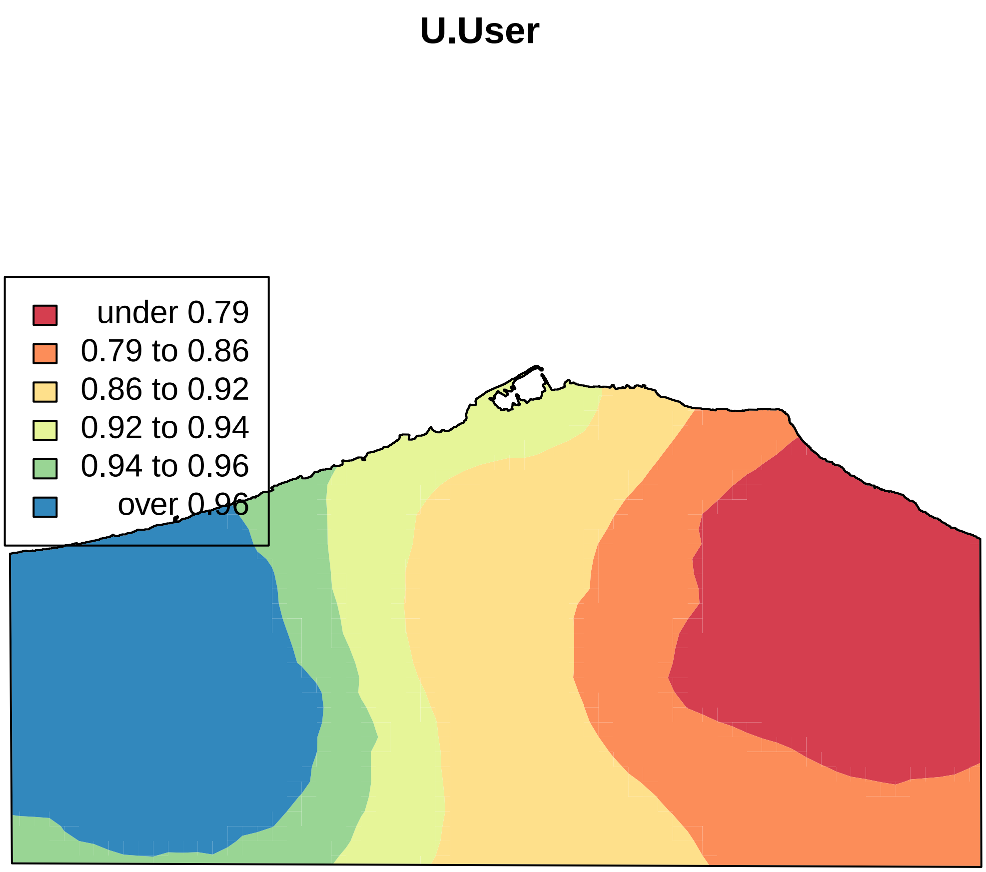
Of course the parameters can be adjusted:

```
gw.mapping(gwr.all.spdf, index = 2, tit = names(gwr.all.spdf)[2])
```



in different ways:

```
gw.mapping(gwr.all.spdf, n= 6, index = 5, tit = names(gwr.all.spdf)[5],
  cols = brewer.pal(6,'Spectral'))
```



or written to a file:

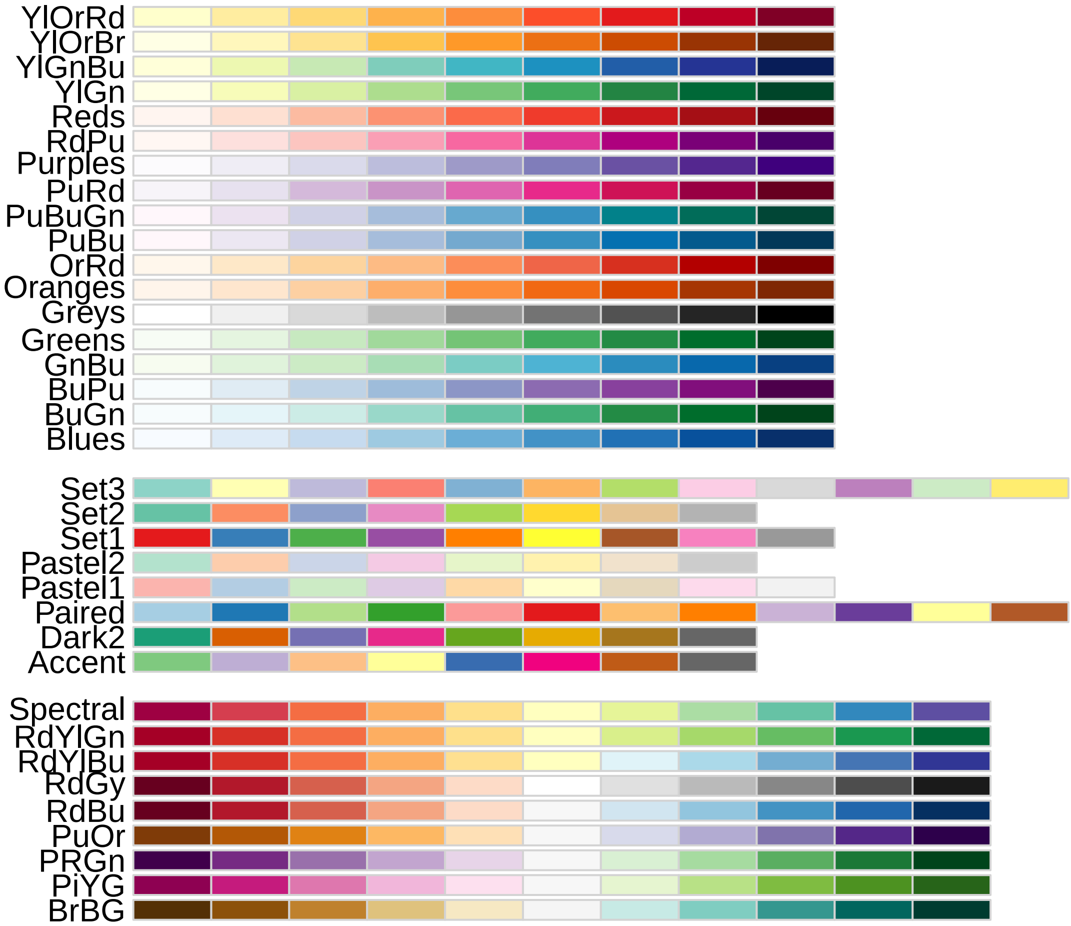
```
png(filename = "plot.png")
gw.mapping(gwr.all.spdf, index = 5, tit = names(gwr.all.spdf)[5],
  cols = brewer.pal(6,'YlOrRd'))
dev.off()
```

Or even put into a loop:

```
for (i in seq(2, 10, by = 2)) {
  gw.mapping(gwr.all.spdf, index = i, tit = names(gwr.all.spdf)[i])
}
```

And other shading schemes are available - see:

```
display.brewer.all()
```


YlOrRd
YlOrBr
YlGnBu
YlGn
Reds
RdPu
Purples
PuRd
PuBuGn
PuBu
OrRd
Oranges
Greys
Greens
GnBu
BuPu
BuGn
Blues
Set3
Set2
Set1
Pastel2
Pastel1
Paired
Dark2
Accent
Spectral
RdYlGn
RdYlBu
RdGy
RdBu
PuOr
PRGn
PiYG
BrBG