An R Library for Geographically Weighted Regression

Chris Brunsdon

June 2, 2003

Spatial Analysis Research Group
Department of Geography
University of Newcastle-upon-Tyne
Newcastle-Upon-Tyne UK
NE1 7RU

1 Introduction

This document is a primer on the GWR library for the R statistical programming language. This library includes a number of routines for calibrating GWR models (Brunsdon, Fotheringham, and Charlton 1996), and also some visualization tools. It is assumed that the reader already has some knowledge of R. If not, a good starting point is the documentation available from the R web pages, and also Venables and Ripley (1999). The latter is actually based on S-Plus rather than R, but the two packages have much in common.

Where possible the use of the library is similar to the use of other R libraries - and as well as the GWR subroutines themselves, training data sets, help files and HTML help pages are also supplied. This document will demonstrate how to install the library, work through a few examples of its use and hint at some more advanced uses. A number of appendices are also provided, mainly to supply some technical details. There will be several practical examples in the text. Commands that the user must type in to R are shown in `this font`. When output from R is shown, it will be shown in the same font, but when possible it will be coloured in `red`.

2 Getting Started with the GWR Library

Like all R code, the GWR code has to be loaded after R has been started. To do this, type

```
source(gwr4)
```

at the prompt.

The core routine in the library is called `gwr`. To see how this is used, firstly we need some data. In this case, we will use crime and house price data from Columbus, Ohio (Anselin, Bera, Florax, and Yoon 1996). To load the data, enter

```
data(columbus)
```

This consists of 49 observations and 5 variables, stored in the data frame `columbus`. Typing

```
colnames(columbus)
```

gives the names of the five variables:

```
[1] "East"    "North"   "Crime"   "Homeval" "Income"
```

These are, respectively, the coordinates of the centroids of 49 zones in and around Columbus, the *per capita* crime rate, the average home value and the average annual income for each zone. The `gwr` routine in its basic form requires three arguments: `x`, `y` and `loc`. `x` is a matrix of predictor variables for the regression model, and `y` is the response variable. Finally `loc` is a two-column matrix containing the coordinates of each of the observations. The value returned is a list containing various items of information relating to the calibrated GWR model. Thus, to apply `gwr` to the columbus data, we must first extract these various columns from the data frame. One way of doing this is by array index subset selection:

```
x <- columbus[,c(3,5)]
y <- columbus[,4]
loc <- columbus[,c(1,2)]
```

We may now run the GWR routine. The result is stored in a variable called `model1`.

```
model1 <- gwr(x,y,loc)
```

This in itself is not very informative. However, `model1` can be "unpacked" to discover the results of the GWR analysis. As mentioned earlier, this variable is a *list* of named items. Two important items are called `est` and `se`. These are respectively the estimates and standard errors of the GWR coefficients at each observation's coordinates. These can be viewed by typing

```
model1$est
```

and

```
model1$se
```

The former gives a response as below:

```
      Intercept      Crime    Income
 [1,]  58.39927 -0.6904944 0.5086567
 [2,]  58.74560 -0.6713341 0.4082533
 [3,]  54.82317 -0.6338069 0.5912671
 [4,]  56.27808 -0.6350209 0.4710034
           .          .          .
           .          .          .
           .          .          .
```

It is possible to use these in further calculations - R is a programming language after all. Thus, it is possible to enter something like

```
pseudo.t <- model1$est / model1$se
```

and estimate a pseudo t-statistic to informally investigate departures from zero of local regression coefficients. Entering

```
pseudo.t
```

now would suggest that the `Crime` coefficient seems to differ notably from zero in more places than `Income`.

3 Visualizing GWR

The previous section demonstrated how the basic `GWR` function can fit local regression coefficients at the locations of the data points, but in many cases we wish to fit the coefficients at different data points. In particular, to view the GWR coefficients as continuous surfaces, it is helpful to fit them on the vertices of a regular grid. The first function needed for this is `nice.grid`. This takes a two column location matrix (such as `loc`) and returns another such matrix, whose points are the vertices of a grid. This is best shown visually. First enter

```
plot(loc)
```

to see the irregular point patterns of the data zone centroids. Now, create a set of regular grid points, store them in the variable `gr`, and plot these:

```
gr <- nice.grid(loc)
plot(gr)
```

The points in `gr` form a regular grid. The extent of the grid covers the original point pattern with some extra space around the margins. To see the relationship between the two sets of points enter

```
plot(gr,pch='+')
points(loc,pch=16)
```

Counting the "+" signs on this plot will show that the grid consists of a 25 x 25 arrangement of points. This may be altered by using the named argument `size` in `nice.grid`:

```
gr <- nice.grid(loc,size=c(40,40))
```

for example.

Next, we need to evaluate the GWR model at the locations in `gr`. This is done using the named argument `out.loc` in `gwr`:

```
model2 <- gwr(x,y,loc,out.loc=gr)
```

This creates a new variable `model2`. This variable is similar to `model1`, but with the local coefficient estimates and standard errors evaluated at points on the lattice `gr`. As before, it is possible to 'unpack' the GWR results, and to use them in further computations. However, as suggested earlier, it is also possible to visualize the local coefficient surfaces. To do this, the `plot` command is used. In its simplest form, this will require two arguments: the name of the variable containing the GWR result, and the name of the coefficient. The following example will plot the `Income` term from the model.

```
plot(model2,'Income',view='f1')
```

The plot created is a filled contour plot, sometimes referred to as a *level plot* (Cleveland 1993). The colour scheme runs from near-white (high) to green (low). An alternative view can be obtained using the `view` named argument. Here, setting this to the character `'s'` gives a surface representation in three-dimensional space.

```
plot(model2,'Income',view='s')
```

With surface views, it is also possible to alter the angle of viewing using `phi` and `theta`, as it is with the standard R function `persp`:

```
plot(model2,'Income',view='s',phi=40,border=NA)
```

In fact, all of the `persp` parameters may be used. If you have R version 1.0.0 or later, it is possible to label the plot axes using the parameter `xlab`, `ylab` and `zlab`:

```
plot(model2,'Income',view='s',phi=40,
     xlab='Easting',ylab='Northing',zlab='Income')
```

To plot the standard error surface, the `se` parameter should be set to `TRUE` or `T`. The standard error of the `Income` surface may be drawn using

```
plot(model2,'Income',se=T)
```

The view seen here is fairly typical, with the highest standard errors occurring at the edges of the grid.

Note that some areas in the grid are quite some distance from any data points. To verify this using the 'Crime' variable as a 'backdrop', one could enter

```
plot(model2,'Crime')
points(loc,pch=16)
```

This becomes even more apparent if the convex hull of the data points is added to the plot.

```
columbus.hull <- loc[chull(loc),]
polygon(columbus.hull)
```

In the above code, the indices of the points on the convex hull of `loc` are returned by the standard R function `chull`, in clockwise order. These are then used to select the coordinates of `loc` to be stored in the location matrix `columbus.hull`. Clearly, attention of the GWR analysis should be focused within the convex hull. Areas a long way away from this do not have any data very close to them, and the results of analysis are likely to be unreliable. It would be useful, therefore, to 'clip' the graphical representation around the convex hull. This is done using the `context` named argument:

```
plot(model2,'Income',context=function() cookie(columbus.hull))
```

Note the unusual form of the argument - this is because we are actually passing a *function* to the plot command, rather than a variable. This feature is also useful with the polygon outline of a geographical region, for example the mainland of the UK. An example of this will be given in section 5.

An alternative, but perhaps not as æsthetically pleasing method of 'trimming' the surfaces around the study area is to use the `mask` option:

```
plot(model2,'Income',mask=columbus.hull)
```

One advantage of the `mask` option is that it will work on surfaces `view='s'` whereas the `context` method will not.

Finally, the `plot` command for GWR models can be used in conjunction with all of R's other graphical facilities (except when `view = 'f1'`) For example, it is possible to create multiple panels in the same window using the `par(mfrow=...)` option. For this demonstration is is probably a good idea to resize the graphics window to by about twice as wide as it is tall.

```
par(mfrow=c(1,2))
plot(model2,'Crime',view='s',mask=columbus.hull)
plot(model2,'Crime',se=T,view='s',mask=columbus.hull)
```

4 Bandwidth Selection

In the previous sections, nothing has been mentioned about bandwidth selection for GWR models. In fact, in all of the examples until now, the bandwidth has been chosen automatically. Unless explicitly told what value the bandwidth should take, the value chosen by `gwr` is the standard distance of the points specified in `loc` - for a definition of this quantity see page 136 of Fotheringham, Brunsdon, and Charlton (2000). To see the value of bandwidth that has been chosen, simply print the model value returned from the `gwr` function. For example, entering

```
model1 <- gwr(x,y,loc)
model1
```

gives the readout

```
GWR model with variables : Intercept Crime Income
Bandwidth      =  8.07
Effective D.F. =  5.752
Corrected AIC  =  321.9
```

Here, alongside some other information we see that the bandwidth is 8.07 units.

This default bandwidth choice tends to oversmooth the data, although such a choice of bandwidth could be viewed as 'conservative', in the sense that any features of nonstationarity detected by this process are reasonably likely to be genuine, and not consequences of outlying or unusual data points. However, the bandwidth may also be chosen directly using the `bw` named argument in `gwr`. A much 'tighter' bandwidth than the default choice is used below:

```
model3 <- gwr(x,y,loc,out.loc=gr,bw=2)
plot(model3,'Income',view='f1')
```

A very large bandwidth is virtually equivalent to fitting a global model.

```
model4 <- gwr(x,y,loc,bw=1000000)
model4$est
```

Gives the output

```
      Intercept      Crime   Income
 [1,]  46.42818 -0.4848885 0.628984
 [2,]  46.42818 -0.4848885 0.628984
 [3,]  46.42818 -0.4848885 0.628984
 [4,]  46.42818 -0.4848885 0.628984
           .          .        .
           .          .        .
           .          .        .
```

Thus, although a GWR was fitted, the coefficient estimates are the same at all points. Indeed, entering

```
lm(Homeval~Crime+Income,data=columbus)
```

will verify that the coefficients agree with the those from the global model.

Another way to choose the bandwidth is via the *effective degrees of freedom* of the model. To explain this concept very briefly, a number of regression type models can be expressed in *hat matrix* form. That is, the predicted y-values and the observed y-values are related by an equation of the form

$$\hat{\mathbf{y}} = \mathbf{H}\mathbf{y} \, .$$

This is true for many kinds of nonparametric regression, and in particular it is true for GWR (Brunsdon, Fotheringham, and Charlton 1999). $\mathbf{H}$, the so-called hat matrix, has a number of important uses. For example, to estimate the variance of the error term in a GWR model, the expression

$$\hat{\sigma}^2 = \frac{\text{RSS}}{n - \text{Tr}(\mathbf{H})}$$

may be used. Note that this is very similar to the expression used in a global regression model:

$$\hat{\sigma}^2 = \frac{\text{RSS}}{n - p}$$

where p is the number of parameters in the model. In particular, in the GWR-based equation, Tr($\mathbf{H}$) corresponds to p. In fact, this association holds in a number of situations, and Tr($\mathbf{H}$) is often referred to as the *effective number of parameters* or *effective degrees of freedom* of the model - see for example Bowman and Azzalini (1997) or Hastie and Tibshirani (1990). Thus, each GWR model has an effective D.F. associated with it. This quantity need not be an integer - in fact usually it is not - but it gives a general idea of the flexibility of the model. Note that this is stored in the output attribute list from `gwr`, as the item labelled `df.used`. It is also shown when the model variable is listed. For example, when you entered `model1` earlier in this section, the effective D.F. was shown to be 5.752. This means there is slightly less "flexibility" in this model than in a global linear regression model with six explanatory variables. Now enter

```
model4
```

recall that this model had a bandwidth of 1,000,000 units - which on the scale of measurement used in `loc` is virtually infinite - and that the effective D.F. in this case is 3 - which agrees with the D.F. of the global model.

Using the `gwr.from.df` function it is possible to specify GWR models with a given effective D.F., instead of specifyingbandwidth. The function works like `gwr`, but takes a named argument `df` instead of `bw`. For example, to specify a model with 9 effective D.F., enter

```
model5 <- gwr.from.df(x,y,loc,out.loc=gr,df=9)
model5
```

The second line entered gives the readout

```
GWR model with variables : Intercept Crime Income
Bandwidth      =  5.145
Effective D.F. =  9
Corrected AIC  =  324.8
```

This verifies that the model really does have 9 effective D.F., and also shows that the bandwidth required for this is 5.145 units. A rough interpretation of the model flexibility here can be obtained by noting that if we apply the *expansion method* of Casetti (1972), and assume that each coefficient in the regression model is a linear function of geographical location, say $a + bu_i + cv_i$, where (u_i, v_i) is the location of observation i, then we end up with 9 parameters to be estimated. Thus, the degree of flexibility here is about the same as if each GWR surface is assumed to be a planar function. However, care must be taken to point out that this is not the same thing as saying that the surfaces actually *are* planar. To see this, enter

```
par(mfrow=c(2,2))
plot(model5,'Intercept',view='s',phi=40,border=NA)
plot(model5,'Income',view='s',phi=40,border=NA)
plot(model5,'Crime',view='s',phi=40,border=NA)
```

This is quite an important point. In some cases satisfactory GWR surfaces can be obtained with many less D.F. than the number required to model the most basic anisotropic parametric model in which all coefficients show spatial nonstationarity. This in turn leads to smaller standard errors in the surface estimates, and to smaller mean squared prediction errors for the y_i's.

5 Model Selection

An important part of any modelling procedure is selecting which model is most appropriate for a given data set. This is as much the case with GWR models as with any other kind. One useful approach to model selection theory is the use of the Akaike Information Criterion (AIC) (Akaike 1973). The underlying idea is to estimate the quantity

$$\int f(\mathbf{y}) \log \left(\frac{f(\mathbf{y})}{g(\mathbf{y})} \right) d\mathbf{y}$$

which measures the *information distance* between the model distribution g and the true distribution f (Kullback and Leibler 1951). By comparing this quantity for a number of competing models g_1 ... g_k we can decide which is 'closest to reality'. Unlike classical statistical inference, this doesn't involve a decision as to whether a hypothesis is 'true'. Such an assertion of absolute truth is certainly dubious in many social science situations, and arguably so in many other areas of study. Instead, this approach assumes that all models are 'wrong' in a strict sense, but that some are closer approximations to reality than others.

In 'hat matrix' situations generally, the AIC can be reasonably estimated by the expression

$$AIC_c = 2n \log(\hat{\sigma}) + \frac{n + \mathrm{Tr}(\mathbf{H})}{n + 2 - \mathrm{Tr}(\mathbf{H})}$$

(Hurvich and Simonoff 1998). This expression is computed by default in both `gwr` and `gwr.from.df`. It may be accessed as the named list item `aic` in the object returned by these routines, and it may be seen when the object is printed out. Entering

```
model5
```

gives the readout

```
GWR model with variables : Intercept Crime Income
Bandwidth      =  5.145
Effective D.F. =  9
Corrected AIC  =  324.8
Results are in gridded format
```

showing that the AIC for `model5` is 324.8. The phrase 'corrected' is used here to distinguish this estimate of AIC from a simpler one that is sometimes used:

$$AIC = 2n\log(\hat{\sigma}) + 2\mathrm{Tr}(\mathbf{H})$$

This expression is based on a simpler method of estimating the AIC, but has been found experimentally to favour undersmooth regression surfaces. It is important to distinguish between the two expressions - they should not be mixed when comparing models.

To decide which particular GWR model should be used, a number of competing models should be evaluated, and their AICs compared. Note that these competing models may differ with respect to the calibration bandwidth, the choice of explanatory variables or both of these. If a single model is to be selected as 'best' then this should be the one with the lowest AIC. For example, `model5` has an AIC of 324.8, whereas for `model1` the value is 321.9, suggesting that the model with the smaller effective D.F. (`model1`) is the better of the two.

For a given set of explanatory variables, the function `gwr.from.aic` will find the model whose bandwidth yields the smallest value of the AIC:

```
model6 <- gwr.from.aic(x,y,loc,out.loc=gr)
model6
```

in this case, we get the readout

```
GWR model with variables : Intercept Crime Income
Bandwidth      =  13.06
Effective D.F. =  4.054
Corrected AIC  =  321.2
Results are in gridded format
```

so that the best AIC obtainable from this model is 321.2, when the effective D.F. is 4.054, and the bandwidth is 13.06 units. This is only a marginal improvement on `model1`.

An alternative way of comparing the AICs for a number of competing models is to consider the *Akaike weights* for each model. If the AIC for model i is denoted by Δ_i then the Akaike weight for model i is defined as

$$w_i = \frac{\exp(-\Delta_i/2)}{\sum_i \exp(-\Delta_i/2)}$$

The w_i's sum to one, and can be thought of as the 'weight of evidence' in favour of each model. Using this approach, we obtain some idea of the relative likelihood that each model is 'best', rather than choosing a single candidate. It has been noted that w_i has a Bayesian interpretation - see for example Akaike (1994) and Akaike (1981). If we place equal prior probabilities on each of the competing models being the 'best', then the w_i's are the posterior probabilities that each model is 'best', in the light of the data.

For example, to compare `model4`, `model5` and `model6` we could enter

```
aic.list <- exp(-0.5 * c(model4$aic, model5$aic, model6$aic))
aic.list <- aic.list / sum(aic.list)
aic.list
```

This shows the respective posterior probabilities for `model4`, `model5` and `model6` to be about 0.42, 0.08 and 0.50 respectively. Recall that `model4` was effectively the global model. This result suggests that the model with the tights bandwidth (`model6`) is unlikely to be the best, but that the global model and `model5` are roughly equally likely. This is not a conclusive result, as it suggests that the most plausible GWR result is only a little more likely than the global model. One possible interpretation here is that if spatial non-stationarity is present, there is not a great deal of variation in the regression parameters. This is perhaps a reasonable interpretation here - the analysis only covers one urban area, and has quite a small number of observations, so perhaps one would not expect to detect any large-scale trends.

6 Working with Large Data Sets

One practical problem when working with GWR is that the data set to be analysed is often quite large. In the UK, for example, there are around 10,000 census wards, and GWR models analysing data for the whole country at this level of spatial aggregation is fairly common. I have tested the library on datasets of this size, and in general the software has coped[1]. The main difficulty has been the time taken to compute the standard errors, the effective D.F. and the AIC. Computationally, this is because all of these quantities require an estimate of $\hat{\sigma}^2$ or Tr($\mathbf{H}$) to be computed. Computing $\hat{\sigma}^2$ requires the residual sum of squares to be computed, which means that the GWR model has to be computed n times, in order to obtain a local regression estimate for each $\hat{y}_i$. Clearly, if n=10,000 this is quite a lot of computation. Similar levels of calculation are needed to compute Tr($\mathbf{H}$). To give a real life example, computing a 10,000 case by 4 variable problem takes around 20 minutes on my 450MHZ PC at the time of writing this primer. Note that `gwr.from.df` and `gwr.from.aic` iteratively re-evaulate GWR models, and could take somewhat longer than this.

However, the basic *estimation* procedure does not require $\hat{\sigma}^2$ or Tr($\mathbf{H}$) to be computed. Also, when the data set is this large, and the GWR coefficients are estimated on a grid, then the number of points at which the model is computed tends to be very much smaller than n. For example, the default grid size is 25 x 25 = 625 points. Thus, if it is only the coefficient estimates that are required, the computing time can be reduced by about 94%. To facilitate this, the `gwr` function has the optional named argument `quick`. Entering

```
gwr(x,y,loc,out.loc=gr,quick=T)
```

causes the `gwr` function to skip all computation except the basic coefficient estimation. The returned object is the same as for a normal call to `gwr`, except the information regarding the effective D.F., the AIC and the standard error estimates are all `NA`, R's missing value symbol. Grid-based output can be visualized with `plot` as before, although setting the named argument `se=T` will now result in an error message.

I would suggest initially using the `quick=T` option to explore a large dataset, and perhaps decide on an appropriate bandwidth by trial and error and visual inspection of the coefficient surfaces. Then, one could set the full-blown `gwr` routine running and go for a coffee. Alternatively, one could leave `gwr.from.df` or `gwr.from.aic` to run overnight.

[1] although you may need to expand R's run time memory requirement

References

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In B. Petrov and F. Csaki (Eds.), *2nd Symposium on Information Theory*, pp. 267–281. Budapest: Akademiai Kiado.

Akaike, H. (1981). Likelihood of a model and information criteria. *Journal of Econometrics 16*, 3–14.

Akaike, H. (1994). Implications of the informational point of view on the development of statistical science. In H. Bozdogan (Ed.), *Proceedings of the First US/Japan conference on the frontiers of statistical modelling: An informational approach*, Volume 3, Kluwer Academic, pp. 27–38. Dordrecht.

Anselin, L., A. Bera, R. Florax, and M. Yoon (1996). Simple diagnostic tests for spatial dependence. *Regional Science and Urban Economics 26*, 77–104.

Bowman, A. and A. Azzalini (1997). *Applied Smoothing Techniques for Data Analysis: The Kernel Approach with S-Plus Illustrations*. Oxford: Oxford University Press.

Brunsdon, C., A. S. Fotheringham, and M. Charlton (1996). Geographically weighted regression: A method for exploring spatial nonstationarity. *Geographical Analysis 28*, 281–289.

Brunsdon, C., A. S. Fotheringham, and M. Charlton (1999). Some notes on parametric significance tests for geographically weighted regression. *Journal of Regional Science 39*, 497–524.

Casetti, E. (1972). Generating models by the expansion method: Applications to geographic research. *Geographical Analysis 4*, 81–91.

Cleveland, W. S. (1993). *Visualizing Data*. New Jersey: Hobart Press.

Fotheringham, A., C. Brunsdon, and M. Charlton (2000). *Quantitative Geography: Perspectives on Spatial Analysis*. London: Sage.

Hastie, T. J. and R. J. Tibshirani (1990). *Generalized Additive Models*. London: Chapman and Hall.

Hurvich, C. M. and J. S. Simonoff (1998). Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion. *Journal of the Royal Statistical Society, Series B 60*, 271–293.

Kullback, S. and R. Leibler (1951). On information and sufficiency. *Annals of mathematical statistics 22*, 79–86.

Venables, W. N. and B. D. Ripley (1999). *Modern applied statistics with S-Plus* (3rd ed.). New York: Springer.