Geographically Weighted Regression in R

Lex Comber and Paul Harris

15 May 2017

Contents

Overview

In this session we will build on the regression work in the last session to develop a geographically weighted regression analyses. We will need to do some mapping (rather than plotting) as the analyses we will develop are spatial. This session will cover a number of areas and topics:

- Introduce Geographically Weighted Regression;
- Formally describe GWR, including the critical steps bandwidth selection and weighting;
- Apply GWR to the Liudaogou data in order to model Phosphorus;
- Map the spatially distributed coefficient estimates generated by GWR;
- Identify important further considerations when using GWR.

You will need the data and packages that you installed and used in the last R session.

Geographically Weighted Regression: background

Geographically Weighted Regression (GWR) was first described by Brunsdon et al (1996) and then more foully covered by Fotheringham et al (2002). It has at its basis the idea that processes and relationships between variables may, in some circumstances, vary over space, and that understanding the nature of this variation can reveal important aspects of the process under consideration.

The major issue with modelling spatial data is the lack of independence of the spatial objects and the assumption of stationarity in the processes being modelled. These are well recognised in socio-economic domains, but are also relevant within bio-geographical, environmental and ecological ones and are reflected in Tobler's dictum (the First Law of Geography) that: '*everything is related to everything else, but near things are more related than distant things*' (Tobler 1970). However, despite numerous methodological advances in addressing such spatial modelling problems, whose lineage from the late 1970s can be loosely traced through spatial statistics texts from Journel and Huigbrechts (1978) to Cressie (1993), to Chilès and Delfiner (1999) and to Cressie and Wilke (2011), policy related research is still routinely informed by non-spatial modelling practices which are heroic at best, and ill-advised at worst.

Focusing on regression models, the main drawback in assuming observations are independent is that any spatial dependencies turn up in the residuals, as they are not explicitly added to the model. For area-based data, ways to account for this include Besag's (1974) conditionally autoregressive model or Anselin's (1988) spatially autoregressive model. For point-based data, geostatistical regressions are commonly used, where spatial dependencies are modelled directly via the variogram with a number of recent advances (Goovaerts 2009). Spatial information can also be accounted for by expressing the regression's coefficients as functions of the spatial coordinates (Casetti 1982; Gorr and Oligschaeger 1994), permitting a model of relationship nonstationarity. GWR (Brunsdon et al. 1996) was a major advance in this area. It uses a continuous distance-decay weighting scheme to accommodate the spatial structures in the data and the resultant localised regression coefficients can be mapped. Brunsdon et al. (1998) and Harris et al. (2015) provide some simple inferential mechanisms to determine whether the coefficients exhibit significant spatial variation. Some of the advances in the GWR method can be found in Nakaya et al. (2005), Wheeler (2007), Huang et al. (2010), Harris et al. (2011), Brunsdon et al. (2012) and Silva and Fotheringham (2015).

In summary, a key issue in ecological modelling is that *dependence* and some form of *non-stationarity* are endemic in spatial data. When spatial non-stationarity occurs the predictor and response variables are expected to change over space and the relationship between the distribution of a certain species and predictor variables may change over space. Mechanisms to handle these characteristics of spatial data in modelling is an important challenge. This paper accommodates spatial nonstationarity using the GWR model in order to demonstrate how to address process heterogeneity in relationships.

GWR: formal description

In overview, GW approaches use a moving window or kernel that passes through the study area. At each location being considered, data under the window are used to make a local calculation of some kind, such as a regression. The data are weighted by their distance to the kernel centre and in this way GW approaches construct a series of models at discrete locations in the study area.

For the geographically weighted kernels, an adaptive bi-square weighting function was applied. This generates higher weights at locations very near to the kernel centre relative to those towards the edge. For each data point (P_j) under the kernel with a given bandwidth, a weight $w_{i,j}$ is calculated based on its distance to the centre of the kernel (K_i) as follows:

$$w_{i,j} = 1 - ((d_{i,j})^2/b^2) \tag{1}$$

where $d_{i,j}$ is the distance in metres from the centre of the kernel K_i to the data point P_j and b is the bandwidth.

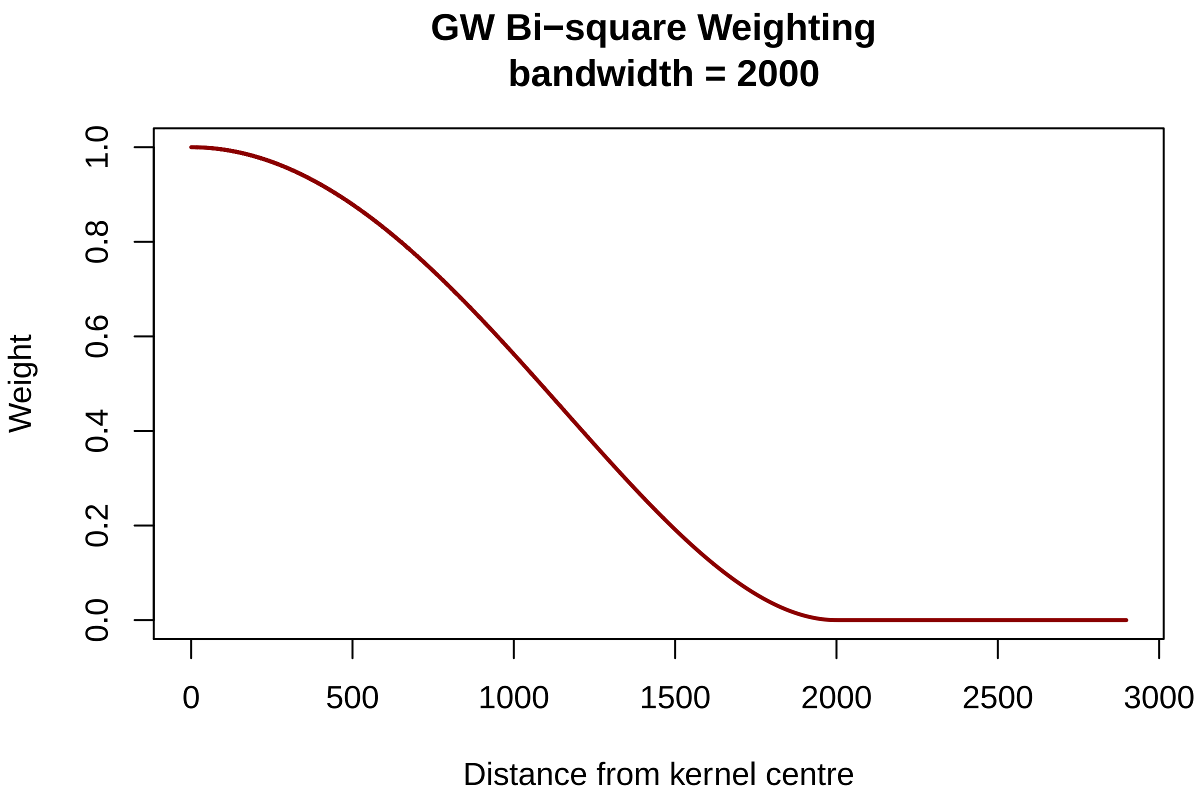
An optimum kernel bandwidth for GWR can be found by minimising a model fit diagnostic. Options include a leave-one-out cross-validation (CV) score (Bowman 1984; Brunsdon et al. 1996). This optimises model prediction accuracy and the Akaike Information Criterion (AIC) (Akaike 1973; Fotheringham et al. 2002) optimises model parsimony by trading off prediction accuracy and complexity.

The standard GWR model is:

$$y_i = \beta_{i0} + \sum_{m}^{k=1} \beta_{ik} x_{ik} + \epsilon_i \tag{2}$$

where y_i is the response variable at location i, x_{ik} is the value of the k^{th} predictor variable at location i, m is the number of predictor variables, β_{i0} is the intercept term at location i, β_{ik} is the local regression coefficient for the k^{th} predictor variable at location i and ϵ_i is the random error at location i. The weighting results in data nearer to the kernel centre making a greater contribution to the estimation of local regression coefficients at each local regression calibration point i.

There are a number of options for weighting, as described in Gollini et al (2015). The bi-square is illustrated below:



GWR in R

There are a number of GWR packages for R. The best are `spgwr` and `GWmodel` and each has their own advantages:

- `GWmodel` is run by the the original GWR development team and has the widest range of GW functionality including different weighting types, distance measures, and GW operations: PCA, robust and basic GWR, ridge compensated GWR to handle local collinearity - one of the major criticisms of GWR.
- `spgwr` has been developed by Roger Bivand and has less functionality although it runs quicker.

You should install, load and then explore both packages but this exercise will use the functions from the `GWModel` package as it a greater range of tools and functions:

```r
install.packages("spgwr", dep = T)
install.packages("GWmodel", dep = T)
```

Then

```r
library(GWmodel)
library(spgwr)
```

GWR analysis

The first thing you will need to do is to generate the `SpatialPointsDataFrame` that was called `data.sp` of the Liudaogou data in the last exercise. If you did not save this session then this can be recreated using the commands below. First download the data :

Spatial Data

```r
library(repmis)
# load the data
source_data("https://github.com/lexcomber/CAS_GW_Training/blob/master/Liudaogou.RData?raw=True")
source_data("https://github.com/lexcomber/CAS_GW_Training/blob/master/boundary.RData?raw=True")
```

Then create the spatial data. Notice that number of packages are being installed:

```r
library(GISTools)
library(tidyverse)
library(plyr)
# define a projection
proj. <- CRS("+proj=tmerc +lat_0=0 +lon_0=108 +k=1 +x_0=500000
             +y_0=0 +ellps=krass +units=m +no_defs ")
# define coordinates
coords <- data[,3:2]
# create a SpatialPointsDataFrame
data.sp <- SpatialPointsDataFrame(coords,
    data = data.frame(data),
    proj4string = proj.)
```

Define the regression parameters

Now we define the regression model again based on the previous exercise, which will will then be used to help specify the GWR bandwidth. Here, all the data are used to form an initial model and the `stepAIC` function is used to select the model with the best fit. This will then be used for all subsequent models.

```r
terms <- names(data)[c(4:19)]
regmod <- paste(terms[1], "~")
for ( j in c(3:16) ) {
  if ( j != 16 ) regmod <- paste(regmod, terms[j],  "+")
  if ( j == 16 ) regmod <- paste(regmod, terms[j])
}
regmod <- as.formula(regmod)
step.i <- stepAIC(lm(regmod, data), trace = F)
regmod <- as.formula(step.i)
regmod
```

The first stage in any GW analysis is to determine the size of the kernel or moving window. This is known as the kernel *bandwidth*. The bandwidth can be fixed or adaptive. A fixed bandwidth specifies a distance, and at each location for which a local analysis is being undertaken (regression in this case) data points falling within that distance are includes. An alternative is the adaptive bandwidth. Here, the nearest n points are included. In `spgwr` this is expressed as a proportion between [0, 1] and in `GWmodel` is expressed as an integer. The code below uses `bw.gwr` from `GWmodel` to select the bandwidth.

```r
library(GWmodel)
bw <- bw.gwr(regmod, data = data.sp,
  kernel = "bisquare", adaptive = TRUE, approach = "AIC")
```

In this case the bandwidth is 215.

Run GWR

Having determined the optimal bandwidth, this can then be used within a GWR function:

```r
gwr.mod <- gwr.basic(regmod, data = data.sp, bw = bw,
  kernel = "bisquare", adaptive = T)
```

And the outputs can be examined. Here we are interested in the coefficient estimates and how they compare to the OLS. Specifically we are interested in how the distribution of these local coefficients compare with the global ones. The `gwr` summary provides this information:

```r
gwr.mod
```

	OLS (global)	GWR 1stQ	GWR median	GWR 3rdQ	P-value
(Intercept)	-0.1349	-0.1449	-0.1037	-0.0594	0.000
SOCgkg	0.0124	0.0084	0.0146	0.0269	0.000
ClayPC	-0.0048	-0.0136	-0.0039	-0.0006	0.033
SiltPC	0.0025	0.0018	0.0023	0.0037	0.000
NH4Ngkg	0.0017	0.0003	0.0011	0.0019	0.010
N2P	0.3783	0.2645	0.3599	0.3988	0.000
Aspect	0.0001	0.0000	0.0000	0.0001	0.127
SoilTypeMein	0.0172	0.0033	0.0146	0.0462	0.277
SoilTypeRed	-0.0362	-0.0379	-0.0041	0.0161	0.017
SoilTypeWarp	0.0139	-0.0135	0.0165	0.0542	0.556

Here we can see that for some of the coefficient estimates there is considerable spatial variation. For example, consider SOCgkg - this soil organic carbon - it varies from 0.0084 to 0.0269 locally , while the global value is 0.0124. Other coefficient estimates do not vary so radically: for example, N2P - this is the ratio of Nitrogen to Phosphorus - it varies from 0.2645 to 0.3988 locally , while the global value is 0.3783but proportionately this is large. So here we can see that the global model produced by the OLS regression may overestimate the association of certain covariates with the variable of interest in some places and under-estimate it in others. The next section maps these.

Mapping the outputs

We can explore the spatial variation in the coefficient estimates. In this case the data were collected at relatively regularly spaced sample locations, making the mapping and visualizations of spatial distributions relative easy. Other options are available including mapping over regression points to generate a GWR surface. These are illustrated in the next section along with other additional GWR considerations. Here we are interested in mapping the GWR coefficient estimates. There are a number of options for doing this including:

- `choropleth` in the `GISTools` package;
- the `tmap` package;
- `spplot` in `sp`. You can and should explore the vignettes in R for these mapping functions but here we will stay with the `ggplot` function.

The outputs of the `gwr.basic` function are multiple but include `SDF` which is typically the same format as the input - in this case a `SpatialPointsDataFrame`. You will probably not want to use all of the variables at this stage, but instead focus on the coefficient estimates for the 22 variables used as inputs. You should examine these. In the code below these are written into a variable called `gwr.sp`:

```
# the names function is not run here
# but you could try it!
# names(gwr.mod$SDF)
class(gwr.mod$SDF)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
gwr.sp <- gwr.mod$SDF[,1:10]
```

You could examine this using the `summary` function:

```
summary(gwr.sp)
```

To support the `ggplot` function it is helpful to have the coordinates as explicit variables included in the `data.frame` of the `SpatialPointsDataFrame`. These are defined using the `coordinates` function and then combined using the `spCbind` function:

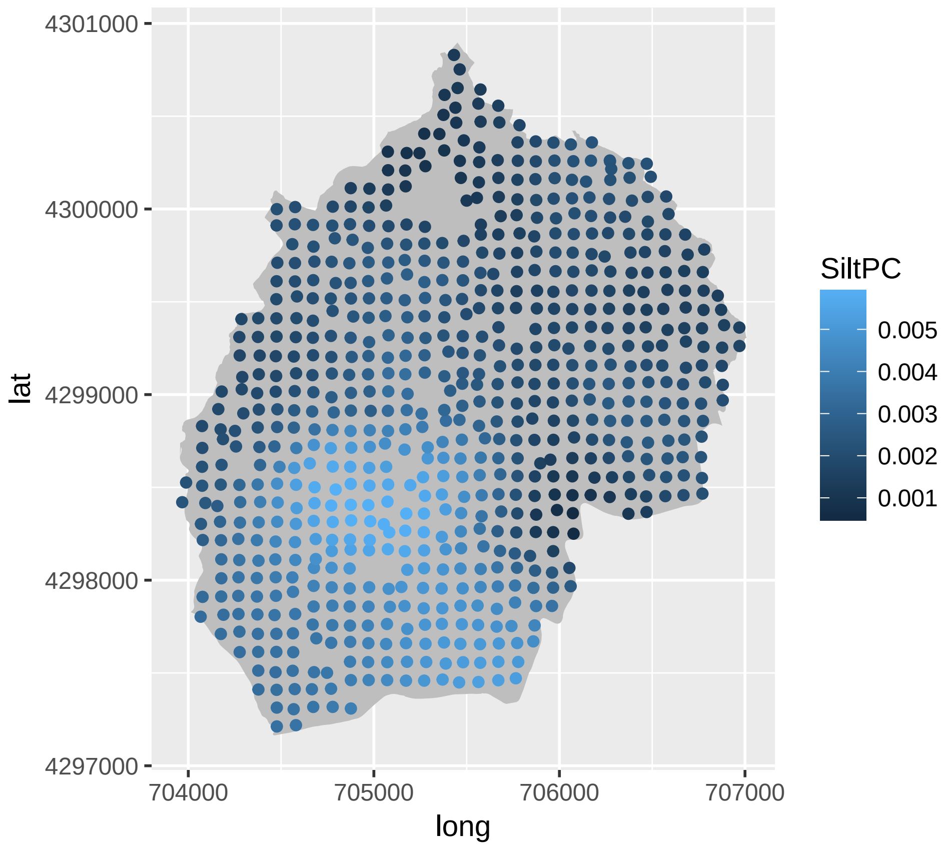
```
X <- coordinates(gwr.sp)[,1]
Y <- coordinates(gwr.sp)[,2]
gwr.sp <- spCbind(gwr.sp, data.frame(X, Y))
```

We will also require the boundary data to add context to the mapping. As before, this will need to be converted into a format suitable for input to the ggplot function:

```
boundary@data$id = rownames(boundary@data)
boundary.points = fortify(boundary, region="id")
boundary.df = join(boundary.points, boundary@data, by="id")
```

Then this can be plotted using ggplot. In the below a plot has been defined that plots the SiltPC coefficient estimates and the variation in the degree to which changes in SiltPC are associated with changes in TPPC:

```
ggplot(boundary.df) +
  geom_polygon(aes(x=long, y=lat), colour=NA, fill="grey") +
  coord_equal() +
  geom_point(data = gwr.sp@data, aes(x = X, y = Y, colour = SiltPC)) +
  scale_colour_gradient()
```



TASK: It might be more relevant to examine the covariates that were found to be significant predictors of TPPC. Your task is identify these from gwr.mod and then to map their distributions.

One thing you might want to to do is to consider how to map multiple outputs from ggplot onto a single plot window. There is a script here http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)/ that you may find useful. We modified to the code below so that it took a list of plots.

```
multiplot2 <- function(plot.list, file, cols=3, layout=NULL) {
  library(grid)
  # Make a list from the ... arguments and plotlist
  numPlots = length(plot.list)
  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                     ncol = cols, nrow = ceiling(numPlots/cols))
  }
 if (numPlots==1) {
    print(plots[[1]])
  } else {
    # Set up the page
    grid.newpage()
    pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

    # Make each plot, in the correct location
    for (i in 1:numPlots) {
      # Get the i,j matrix positions of the regions that contain this subplot
      matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

      print(plot.list[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                          layout.pos.col = matchidx$col))
    }
  }
}
```

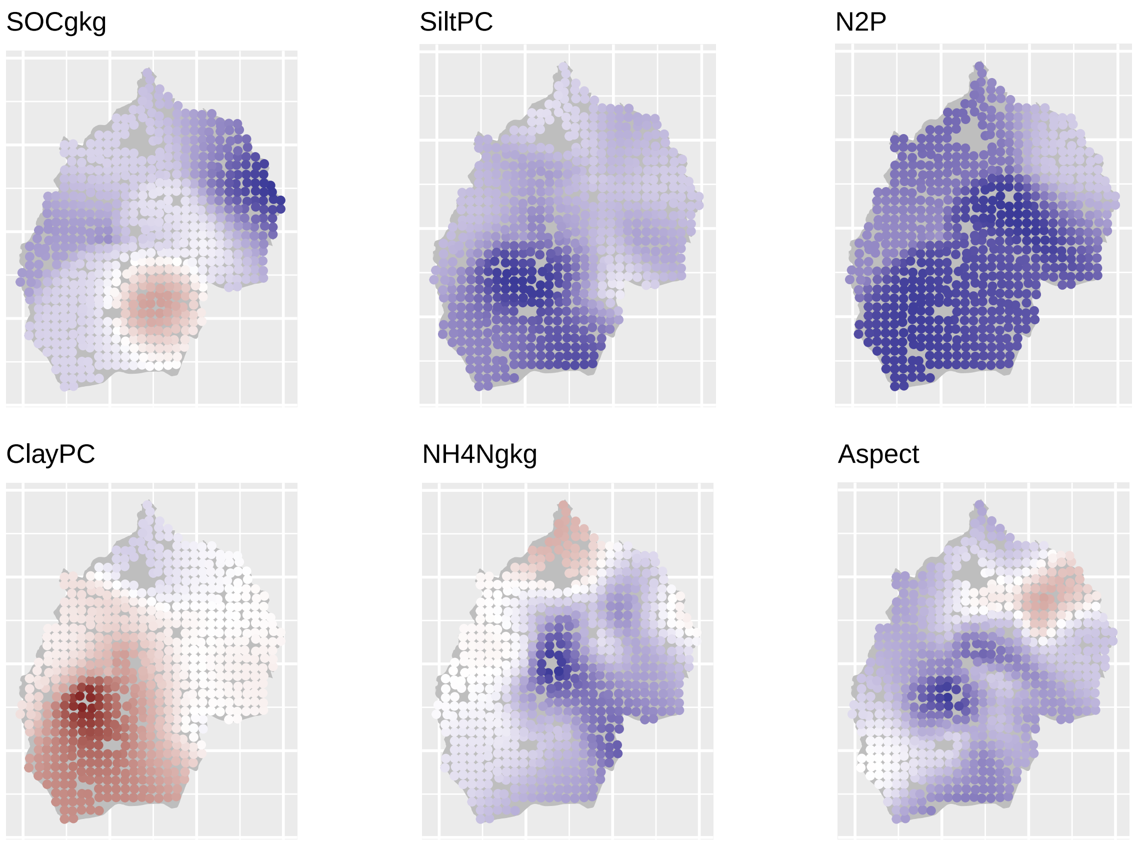
Now it is possible to define a nicer plot and wrap into into a function. The below is an adaptation of the `ggplot` routine above. This can be used to generate plots for all the variables.

```
# plot function
my.ggplot.func <- function(gwr.sp,i, size = 1, tit = ""){
    p <- ggplot(boundary.df) +
    geom_polygon(aes(x=long, y=lat), colour=NA, fill="grey") +
    coord_equal() +
    geom_point(data = gwr.sp@data, aes(x = X, y = Y,
            colour = gwr.sp@data[i]), size = size) +
    scale_colour_gradient2() +
    labs(subtitle = paste(tit, names(gwr.sp@data[i]), sep="")) +
    theme(axis.title.x=element_blank(),
                axis.text.x=element_blank(),
                axis.ticks.x=element_blank(),
                axis.title.y=element_blank(),
                axis.text.y=element_blank(),
                axis.ticks.y=element_blank(),
            legend.position = "none")

    return(p)
}
```

These functions - `my.ggplot.func` and `multiplot2` - can then be applied to all variables (but there would be many of them to map) or just to the significant variables. Notice that the shading describes negative coefficient estimates in red (the more red the more negative), positive ones in blue (again the more positive the more blue), and coefficient estimates close to zero are shaded in white.

```
p2 <- my.ggplot.func(gwr.sp, 2)
p3 <- my.ggplot.func(gwr.sp, 3)
p4 <- my.ggplot.func(gwr.sp, 4)
p5 <- my.ggplot.func(gwr.sp, 5)
p6 <- my.ggplot.func(gwr.sp, 6)
p7 <- my.ggplot.func(gwr.sp, 7)
multiplot2(list(p2,p3,p4, p5, p6, p7), cols = 3)
```



Robust vs. Basic GWR

You may have noticed that the gwr function used above was named `gwr.basic`. One of the issues in any regression is dealing with the effect of any outliers. An outlier is an observation, a value at a sample point, with large residual. That is it is an observation whose dependent-variable value is unusual given its value on the predictor variables. An outlier may indicate a sample peculiarity or may indicate a data entry error or other problem. The locally linear estimates of a basic GWR are unlikely to be as robust to outliers as are the globally linear estimates of linear regression. Outliers will be more prone to in- accurate parameter estimates and can artificially increase local variability and mask key features in local data structures.

To identify and reduce the effect of outliers in GW regression, various robust extensions have been proposed, as described in Fotheringham et al. (2002). The first robust model re-fits a GW regression with a filtered data set that has been found by removing observations that correspond to large externally studentized residuals of an initial GW regression fit. The second robust model, iteratively down-weights observations that correspond to large residuals. Harris et al (2010) described the application and benefits of robust GWR approaches.

The `gwr.robust` function deals with outliers with options to do this by filtering or by re-fitting. First recall the form of the `gwr.basic` model that was run before:

```
gwr.mod <- gwr.basic(regmod, data = data.sp, bw = bw,
  kernel = "bisquare", adaptive = T)
```

This generated spatial distributions of the local coefficient estimates as that were shown in a table along with global ones and global p-values.

Now, the GWR analysis can be re-run using a robust form as follows:

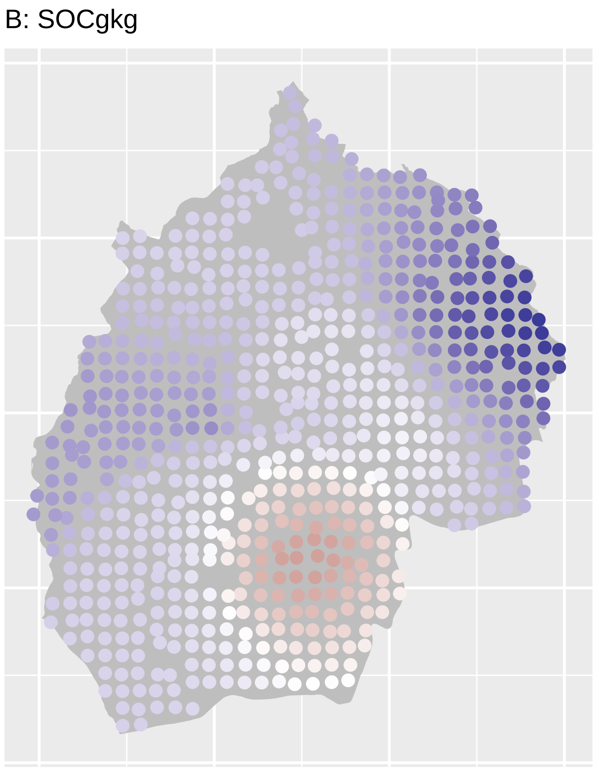
```
gwr.rob <- gwr.robust(regmod, data = data.sp, bw = bw, filtered = F,
  kernel = "bisquare", adaptive = T)
```

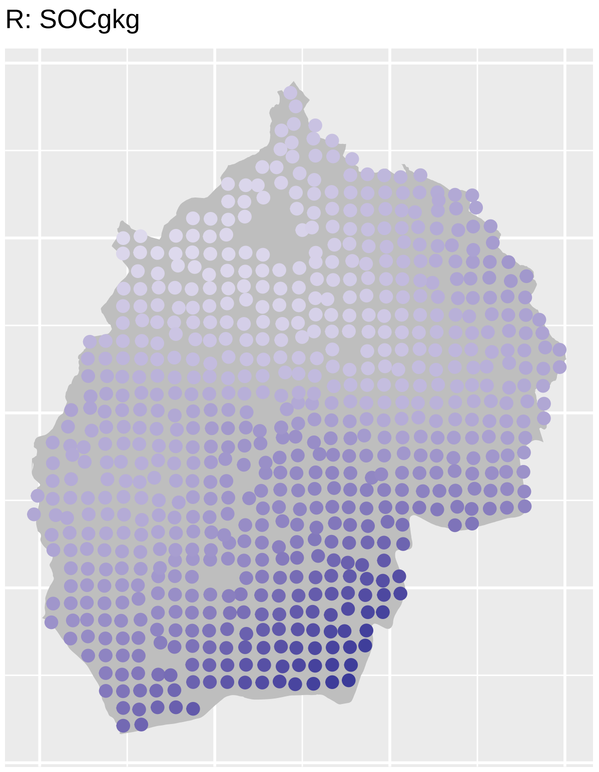
The results can be compared with the results of the `gwr.basic` function in the `gwr.mod` object as in the table below. Here is clear that for some covariates outliers are a problem: consider the differences between the distributions of the coefficient estimates for the basic and robust versions of SOCgkg, ClayPC and SoilTypeMein. For other covariates outliers is less of a problem (for example, SiltPC, NH4Ngkg N2P, Aspect, etc).

	Basic 1stQ	Basic median	Basic 3rdQ	Robust 1stQ	Robust median	Robust 3rdQ
Intercept	-0.1449	-0.1037	-0.0594	-0.0782	-0.0611	-0.0458
SOCgkg	0.0084	0.0146	0.0269	0.0249	0.0325	0.0437
ClayPC	-0.0136	-0.0039	-0.0006	-0.0056	-0.0033	-0.0007
SiltPC	0.0018	0.0023	0.0037	0.0018	0.0022	0.0026
NH4Ngkg	0.0003	0.0011	0.0019	0.0001	0.0004	0.0014
N2P	0.2645	0.3599	0.3988	0.1882	0.2439	0.2823
Aspect	0.0000	0.0000	0.0001	0.0000	0.0001	0.0001
SoilTypeMein	0.0033	0.0146	0.0462	-0.0008	0.0078	0.0154
SoilTypeRed	-0.0379	-0.0041	0.0161	-0.0010	0.0082	0.0173
SoilTypeWarp	-0.0135	0.0165	0.0542	-0.0097	0.0045	0.0305

We can map these differences and compare the spatial distributions of the coefficients under robust and basic GWR models. The example below compares SOCgkg:

```
# First, create the BASIC variable
gwr.b <- gwr.mod$SDF[,1:10]
X <- coordinates(gwr.b)[,1]
Y <- coordinates(gwr.b)[,2]
gwr.b <- spCbind(gwr.b, data.frame(X, Y))
# Then the Robust variable
gwr.r <- gwr.rob$SDF[,1:10]
X <- coordinates(gwr.r)[,1]
Y <- coordinates(gwr.r)[,2]
gwr.r <- spCbind(gwr.r, data.frame(X, Y))
# Finally map
pb <- my.ggplot.func(gwr.b, 2, size = 1.75, tit = "B: ")
pr <- my.ggplot.func(gwr.r, 2, size = 1.75, tit = "R: ")
multiplot2(list(pb,pr), cols = 2)
```





Other Considerations

There are many many considerations that apply to regressions and to GWR - we do not have time to cover the ones we know about in a short workshop. However there are a number of things that you should be aware of that are briefly covered here

1. It is possible to run GWR over a surface of regression points. These can be a spatial points data frame and passed to the GWR function. The resultant SDF output will over those points rather than the original spatial data. This can be good when seeking to develop a regular surface of gridded outputs.
2. GWR automatically computes Euclidean distances between input data locations. This means that toy should be aware of 2 things: If the data are in degrees then the `longlat=F` needs to be set to `TRUE` in the `gwr.basic` function so that great circle distances are used. Secondly, it is possible to pass in other distances such as network distances as well as others such as Manhattan distances. These need to be specified and passed to `dMat` parameter.
3. Collinearity amongst the predictor variables in global regressions is an issue. However, because of the way that GWR uses local subsets of the data, there is the potential for local collinearity to be present even when none is observed locally. The `GWmodel` package includes locally-compensated ridge GWR model for dealing with this: see '?gwr.lcr for a vignette on how to use it.

Code

All of the analyses and mappings were undertaken in R, the free open source statistical software. The RMarkdown script used to produce this manuscript, including all the code used in the analysis and to produce the mapped figures, can be found at https://github.com/lexcomber/CAS_GW_Training

References

Akaike H (1973). Information Theory and an Extension of the Maximum Likelihood Principle. In BN Petrov, F Csaki (eds.), *2nd Symposium on Information Theory*, pp. 267–281. Akademiai Kiado, Budapest.

Anselin L (1988). *Spatial Econometrics: Methods and Models*, Dordrecht: Kluwer Academic

Besag, J (1974). Spatial interaction and the statistical analysis of lattice systems (with discussion). *Journal of the Royal Statistical Society B* 36: 192-225.

Brunsdon C, Charlton M, Harris P (2012). Living with Collinearity in Local Regression Models. In *Proceedings of the 10th International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences*. Brasil.

Brunsdon C, Fotheringham AS, Charlton M (1996). Geographically Weighted Regression: A Method for Exploring Spatial Nonstationarity. *Geographical Analysis*, 28, 281–289.

Casetti E (1982). Drift analysis of regression parameters: An application to the investigation of fertility development relations, *Modeling & Simulation* 13: 961-966.

Chiles, J.-P. and P. Delfiner (1999). *Geostatistics - modelling spatial uncertainty.*

Cressie, N (1993). *Statistics for Spatial Data.* New Jersey, John Wiley.

Cressie, N and Wilke CK (2011) *Statistics for Spatio-Temporal Data*, New Jersey, John Wiley.

Fotheringham, A. S., Brunsdon, C., & Charlton, M. (2002). Geographically weighted regression: the analysis of spatially varying relationships. John Wiley & Sons.

Gollini, I., Lu, B., Charlton, M., Brunsdon, C., & Harris, P. (2015). GWmodel: an R package for exploring spatial heterogeneity using geographically weighted models. *Journal of Statistical Software*, 63 (17), 1–50.

Goovaerts P (2009). Medical Geography: A promising field of application for geostatistics. *Mathematical Geosciences* 41: 243-264

Gorr WL, Olligschlaeger AM (1994). Weighted Spatial Adaptive Filtering: Monte Carlo Studies and Application to Illicit Drug Market Modeling. *Geographical Analysis*, 26: 67-87.

Harris P, Fotheringham AS, Juggins S (2010) Robust geographically weighed regression: a technique for quantifying spatial relationships between freshwater acidification critical loads and catchment attributes. *Annals of the Association of American Geographers* 100(2): 286-306

Harris P, Brunsdon C, Fotheringham AS (2011). Links, comparisons and extensions of the geographically weighted regression model when used as a spatial predictor. *Stochastic Environmental Research and Risk Assessment* 25: 123-138

Journel, A. G. and C. J. Huijbregts (1978). *Mining geostatistics.* London, Academic Press.

Nakaya, T., Fotheringham, A.S., Brunsdon, C. and Charlton, M., 2005. Geographically weighted Poisson regression for disease association mapping. *Statistics in Medicine*, 24(17), pp.2695-2717.

Tobler WR (1970). A computer movie simulating urban growth in the Detroit region. *Economic Geography*, 46: 234-240

Wheeler D (2007). Diagnostic Tools and a Remedial Method for Collinearity in Geographically Weighted Regression. *Environment and Planning A*, 39(10), 2464–2481.