







Special Issue

# gwverse: A Template for a New Generic Geographically Weighted R Package

Alexis Comber<sup>1,2</sup>, Martin Callaghan<sup>3</sup>, Paul Harris<sup>4</sup>,  
Binbin Lu<sup>5</sup>, Nick Malleson<sup>1,2</sup>, and Chris Brunsdon<sup>6</sup>

<sup>1</sup>School of Geography, University of Leeds, Leeds, UK, <sup>2</sup>Leeds Institute for Data Analytics, University of Leeds, Leeds, UK, <sup>3</sup>School of Computing, University of Leeds, Leeds, UK, <sup>4</sup>Sustainable Agriculture Sciences, North Wyke, Rothamsted Research, Okehampton, UK, <sup>5</sup>School of Remote Sensing and Information Engineering, Wuhan University, Wuhan, China, <sup>6</sup>National Centre for Geocomputation, Maynooth University, Maynooth, Ireland

*GWR is a popular approach for investigating the spatial variation in relationships between response and predictor variables, and critically for investigating and understanding process spatial heterogeneity. The geographically weighted (GW) framework is increasingly used to accommodate different types of models and analyses, reflecting a wider desire to explore spatial variation in model parameters and outputs. However, the growth in the use of GWR and different GW models has only been partially supported by package development in both R and Python, the major coding environments for spatial analysis. The result is that refinements have been inconsistently included within GWR and GW functions in any given package. This paper outlines the structure of a new gwverse package, that may over time replace GWmodel, that takes advantage of recent developments in the composition of complex, integrated packages. It conceptualizes gwverse as having a modular structure, that separates core GW functionality and applications such as GWR. It adopts a function factory approach, in which bespoke functions are created and returned to the user based on user-defined parameters. The paper introduces two demonstrator modules that can be used to undertake GWR and identifies a number of key considerations and next steps.*

## Introduction

This paper describes the structure of a new over-arching R package called gwverse that includes some – but not all – packages for different geographically weighted tools. The aim in doing this is twofold. First, to reimagine the functionality of the GWmodel package (Lu et al., 2014; Gollini et al., 2015) that can be used for geographically weighted analyses of different kinds in R, including regression. Second, and just as importantly, to include within the new framework, structures that facilitate the development and integration of user-defined geographically weighted tools, able to draw from the core functionality provided by gwverse. The reasons for doing this

Correspondence: Alexis Comber, University of Leeds, Leeds, U.K.  
e-mail: a.comber@leeds.ac.uk

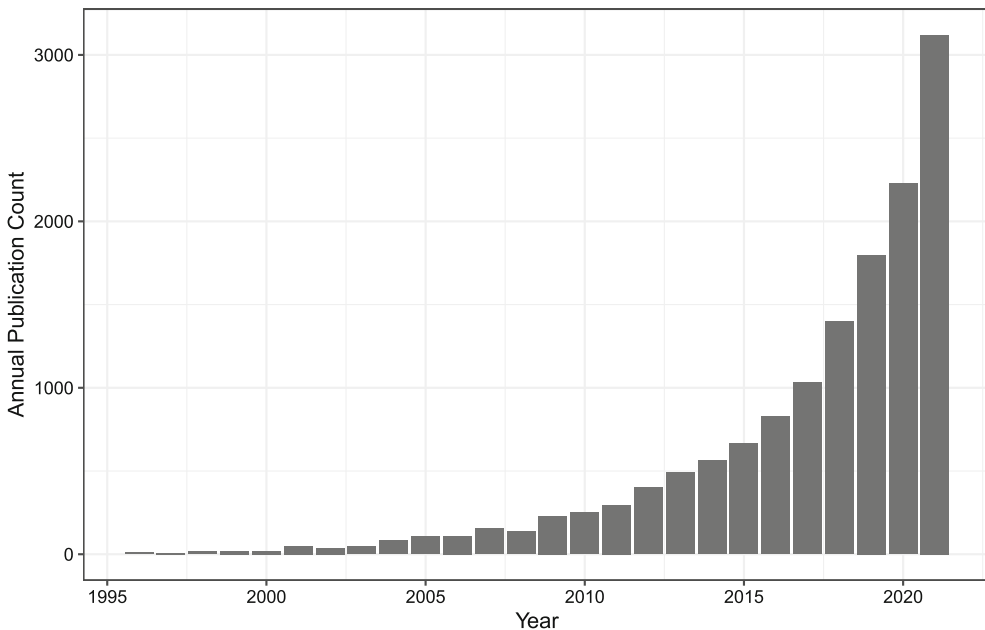
Submitted: September 30, 2021. Revised version accepted: May 24, 2022.

are to propose a framework that better supports users in undertaking such analyses, and critically, allows developers to easily create and benchmark their own geographically weighted tools.

Geographically Weighted Regression (GWR, Brunson, Fotheringham, and Charlton, 1996) investigates the spatial variation in relationships between response and predictor variables. It reflects a desire to shift away from global whole map regressions (Openshaw, 1996) such as those estimated by ordinary least squares (OLS), and including those that account for error spatial dependence, such as regressions estimated by restricted maximum likelihood. GWR arose due to broader interests in investigating and understanding process spatial heterogeneity. As described in Brunson, Fotheringham, and Charlton (1996), the origins of GWR can be found in locally weighted regression (Cleveland, 1979) and local likelihood estimation (Tibshirani and Hastie, 1987) and it emerged in parallel with other developments in local forms of spatial analysis, such as Getis and Ord’s *G* family of statistics (summarized in Getis and Ord, 2010) and the LISA framework (Anselin, 1995).

GWR is increasingly being used for spatial analyses. A search of the Scopus database (<https://www.scopus.com>) in April 2022 for the phrases “GWR,” “Geographically Weighted,” and “Geographically Weighted Regression” in titles, abstracts and keywords indicated 15,480 records, with sharp increases in recent years (see Fig. 1).

This proliferation has been driven by a four main factors (Comber et al., 2022). First, is the increase in the generation, provision and availability of spatial data (i.e. data with some form of location attached), and their ability to support inherently spatial analyses (i.e. analyses that explicitly accommodate the spatial properties of data). Second, is a broader recognition by researchers from different quantitative domains of the benefits of quantifying spatial patterns in data, say through some kind of spatially informed cluster analysis or regression technique, and in doing so handle spatial dependencies in the data or the model parameters themselves. This



**Figure 1.** Geographically Weighted Regression publication numbers, 1996 to 2021, as listed on <https://www.scopus.com>.

has in part been driven by the evolving adoption of the First Law of Geography as invoked by Tobler (1970) as a guiding principle, which in essence describes process spatial autocorrelation (typically dependency in the data), and process spatial heterogeneity (typically dependency in model parameters)<sup>1</sup>. GWR is a method that was designed to support the latter, but indirectly addresses the former (Harris, 2019). Third, is the relative simplicity and conceptual elegance of GWR as a spatial model, which has helped to fuel its popularity. OLS regression is the basic modeling approach (modeling 101), and the creation of local regression models calibrated from data under a moving window, as GWR does, is conceptually intuitive to understand. Fourth, as a result GWR has been implemented in a number of GISs (e.g. ESRI's ArcGIS); in R packages such as `spgwr` (Bivand et al., 2020), `mgwrsar` (Geniaux and Martinetti, 2018), `GWLeIast` (Yoneoka, Saito, and Nakaoka, 2016), `gwrr` (Wheeler, 2013), `GWmodel` (Lu et al., 2014; Gollini et al., 2015), `McSpatial` (McMillen, 2013) and `lctools` (Kalogirou and Kalogirou, 2020); in Python packages such as `PySAL` (Rey and Anselin, 2010) and `mgwr` (Oshan et al., 2019); and in standalone implementations such as `GWR3` (Charlton, Fotheringham, and Brunson, 2003), `GWR4` (Nakaya et al., 2014), and `MGWR` (Li et al., 2019).

GWR itself has been refined to accommodate extensions found in standard regression, such outlier-resistant Harris, Stewart Fotheringham, and Juggins (2010), heteroskedastic (Fotheringham, Brunson, and Charlton, 2002; Páez, Uchida, and Miyamoto, 2002a, b), ridge (Wheeler, 2007; Gollini et al., 2015), LASSO (Wheeler, 2009) and elastic net forms (Li and Lam, 2018; Alexis Comber and Harris, 2018). Further extensions include time in the form of geographically and temporally weighted regression (GTWR) (Huang, Wu, and Barry, 2010; Fotheringham, Crespo, and Yao, 2015), area to point regression (Murakami and Tsutsumi, 2015), multiple scales of analysis (Yang, 2014; Fotheringham, Yang, and Kang, 2017), spatially variable model specification (Comber et al., 2018) and the use of different distance metrics (Lu et al., 2016).

A secondary tranche of developments has seen the use of the Geographically Weighted (GW) framework as a generic structure to accommodate different types of models and analyses. Again this reflects a desire to explore spatial variation in model parameters or its components and to move away from global, “whole map” approaches. Examples include GW principal components analysis (PCA) (Harris, Brunson, and Charlton 2011), GW descriptive statistics (Brunson, Fotheringham, and Charlton, 2002), GW discriminant analysis (Brunson, Fotheringham, and Charlton, 2007; Foley and Demšar, 2013), GW correspondence matrices (Comber et al. 2018), GW structural equation models (Comber et al. 2017), GW evidence combination (Comber et al., 2016), GW Variograms (Harris, Charlton, and Stewart Fotheringham, 2010), GW network design (Harris et al., 2014), GW Kriging (Harris, Charlton, and Stewart Fotheringham, 2010; Harris, Brunson, and Stewart Fotheringham, 2011), GW visualization techniques (Dykes and Brunson, 2007), and more recently GW artificial neural networks (Du et al., 2020; Hagenauer and Helbich, 2022) and GW machine learning (Chen et al., 2018; Li, 2019; Quiñones, Goyal, and Ahmed, 2021; Xu et al., 2021). In each of these developments, the moving window or kernel is still used to generate local data subsets that are weighted by their distance to the kernel center, as is done in GWR, thereby providing local inputs to the model, analysis or evaluation being applied. These various GW models demonstrate a generic, open, and continually evolving technical framework that is being used to explore spatial heterogeneities from a wide range of disciplines in the natural and social sciences.

The growth in the use of GWR and in GW models of different kinds, as well as the refinements to GWR, has been supported to some degree by package development in both

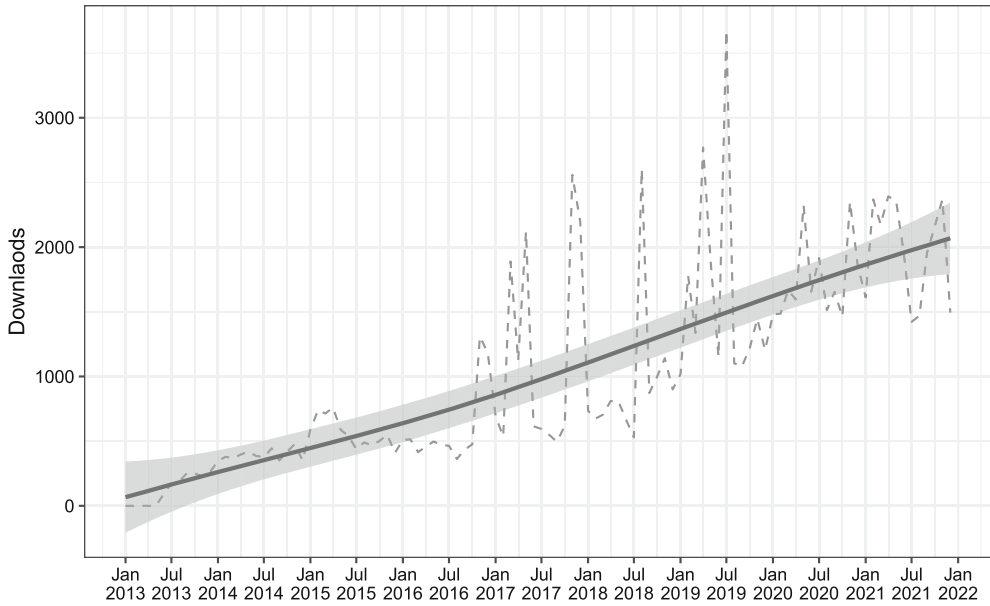
R and Python. However, much of the development has taken place on a piecemeal basis, extending current functionality, without consideration of any overarching schema, nor of more recent developments in thinking around the composition of complex, integrated packages that incorporate a *function factory* approach. The aim of this paper is to critically examine the developments in the package offering the greatest range of GWR- and GW-related functionality, the `GWmodel` R package (Lu et al., 2014; Gollini et al., 2015), to propose an organizational framework within which a new GWR / GW R package will be developed, and to illustrate the first iteration of this in a new `gwverse` R package. In so doing, the paper seeks to describe a comprehensive ecology for undertaking GWR and other GW models, that is also able to support the generation of user-defined GW tools.

## Background: The current `GWmodel` R package

The `GWmodel` R package provides the most comprehensive suite of GWR- and GW-related tools. It contains various forms of GWR, some of which have both basic and outlier resistant forms, some with local statistical tests and diagnostics, a generalized linear model form, some with options for flexible choices of distance metrics (Lu et al., 2016) and a generalized linear model form (Fotheringham, Brunson, and Charlton, 2002). It also contains a number of different functions based on the GW scheme, including tools for GW descriptive statistics, GW PCA and GW discriminant analysis. As of April 2022, the `GWmodel` package has been downloaded more than 206,898 times since it was released on CRAN in 2013 (as recorded on the CRAN download counts web page and the BioConductor site). Monthly CRAN downloads are shown in Fig. 2, indicating the increasing attraction of the package to users from a wide range of disciplines. Additionally, the package functionality has been constantly extended to accommodate refinements and requests for tools from users and the package management team. For example, modules have been incorporated over the last five years to support GWR with large-scale datasets (Murakami et al., 2020), multiscale GWR (Lu et al., 2017, 2018) as well as functions for GTWR and revised algorithms for GW discriminant analysis and GW PCA.

One of the development problems, that occurs with many projects managed by people in their spare or part-funded time, is that this growth in package functionality mostly occurs on a piecemeal basis, without the over-arching organization of the package being considered or revised from its original structure. An example of this is that the `GWmodel` manual is some 85 pages in length (Gollini et al., 2015), with varying depth of detail and two vignettes, published with the launch of the package (Lu et al., 2014; Gollini et al., 2015). The `GWmodel` help pages are similarly inconsistent. Some functions have examples with in-depth explanations and some do not. Many of the recent developments in package functionality do not have vignettes, have ones that are inconsistent or have not been described in the help pages to sufficient depth (there are also some help pages where the example does not work). The result is that there is little to guide the user about which functions to use and how to use them, especially for the newer functions. This inconsistency is shown in Table 1, which indicates the various refinements and specification options that have been incorporated into different functions in `GWmodel` as part of these developments. The full description of the package is contained in the Appendix. What is clear from Table 1 is that while the complexity of the package has increased, allowing more refined approaches to GWR for example, this refinement has not been uniform across the main groups of GWR functionality, or for that matter any GW model.

A further critical consideration for `GWmodel` is that currently it only supports analyses of spatial data in `sp` format (Pebesma and Bivand, 2005). In the `sp` data model, spatial



**Figure 2.** Monthly downloads of the GWmodel package from CRAN, the Comprehensive R Archive Network. [Colour figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)].

**Table 1.** The Presence of GWmodel Package Functionality by the Main Groups of Related Specification Options, where Applicable

Option	GW summary statistics	GW principal components analysis	GW regression	GW generalized linear models	GW discriminant analysis
Flexible distance metric	Yes	Yes	Yes	Yes	Yes
Five kernel functions	Yes	Yes	Yes	Yes	Yes
Fixed/adaptive bandwidth	Yes	Yes	Yes	Yes	Yes
Bandwidth optimization	Yes*	Yes	Yes	Yes	Yes
Robust choice for outliers	Yes	Yes	Yes	No	No
Heteroskedastic errors	–	–	Yes	No	–
Ridge term	–	–	Yes	No	–
F- Tests (Leung)	–	–	Yes	–	–
Monte Carlo Tests	Yes	Yes	Yes	No	No
Bootstrap SE estimation	No	No	Yes	No	No
Local coefficient t-tests	–	–	Yes	Yes	No
Multiscale extension	–	–	Yes	No	No
Space-time	No	No	Yes	No	No
High performance	No	No	Yes	No	No

Note: For mean/median only.

objects can be thought of containing a data table of attributes and a list structure of geometric information for the different kinds of spatial objects (e.g. `SpatialPointsDataFrame`, `SpatialPolygonsDataFrame` etc), where each row in the data frame is associated with an individual component or element of the geometric information. The `sp` class of objects is broadly analogous to shapefile formats (lines, points, areas) and raster or grid formats.

There have been a number of key developments in the R ecosystem for spatial analysis as discussed in Bivand (2021). The `sp` package is in the process of being deprecated and is being replaced by a new class of spatial object called `simple features` as implemented in the `sf` package (Pebesma, 2018). This encodes spatial data in a way that conforms to formal standards defined in the ISO 19125-1:2004 standard and defines a model that represents geometry in Well-Known Text (WKT) format. Spatial objects in `sf` appear as a data table but with an extra `geometry` column that contains the WKT geometrical information. The geometry (called an `sfc` or simple feature column) can be used in geometric operations. Additionally, the `sf` structure follows a “tidy” framework (Wickham, 2014) and can be used with both the new native piping syntax and the `magrittr` pipe to undertake `dplyr` data wrangling operations, for example. The deprecation of `sp` and its replacement by `sf` has required packages be updated to incorporate the new standard in geographical representation in R, which `GWmodel` has not done.

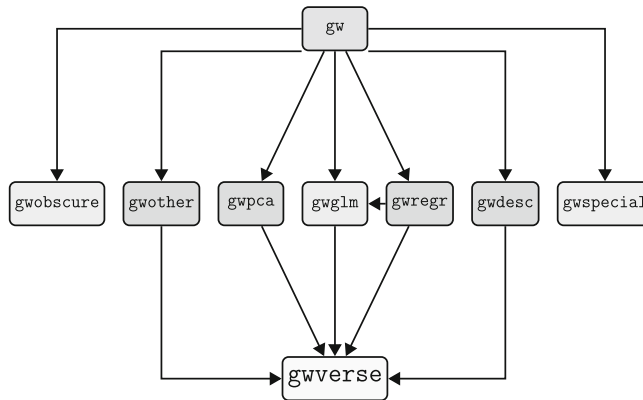
Hence, due to its complexity, over-flowing structures, inconsistent application of refinements, and the fact that it has not been revised to work with `sf` format spatial objects, the `GWmodel` package is ripe for an overhaul. The need for this is enhanced because of the ever-growing popularity of GWR and the GW framework and the increasing use of `GWmodel` to undertake these analyses. The next section sketches out the form that this overhaul could take for a new `gwverse` package and considers *function factories* as an approach for doing this.

## Proposal: `gwverse` – a template for a new GW package

### The basic idea

The basic idea for the `gwverse` package is to implement a modular package structure. Such structures are seen in packages such as `tidyverse`, which when called loads a number of `tidyverse`-related packages. However, it does not load all `tidyverse`-related packages, because this may take time, and occupy resources. It loads more than the absolute basic `dplyr` package (e.g., it loads `ggplot`) but not `feather`. In a similar way, we propose to have an over-arching package called `gwverse` that loads many – but not all – GW-related packages. The structure has, at its core, a package called `gw` that provides general helper functions for a GW analysis, but essentially provides a toolkit to be used in the construction of other GW modules. It includes tools for building GW functions, but not the functions themselves.

It is unlikely that people other than those developing GW tools will load the `gw` package directly, rather it is implicitly loaded (imported) when GW packages are loaded. Our proposed structure and module dependencies are shown in Fig. 3. In this, the boxes represent packages, and the directional arrows imply “makes use of” or “draws functionality from.” So for example, `gwregr` (for GWR), `gwprca` (for GW PCA) and `gwdesc` (for GW descriptive statistics) will all use functions contained in `gw`. The mid-layer packages are individual GW applications, and all of these make use of `gw`. When called, the `gwverse` package will load up several commonly used packages, although not all. The role of `gwrglm` (for GW generalized linear models) is slightly different as it will extensively borrow from the standard (Gaussian response) GWR code



**Figure 3.** The proposed gwverse structure. [Colour figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)].

in `gwregr`, and hence also to load and run `gwgml` will require `gwregr` to be loaded as a dependency as well. The packages `gwobscure` and `gwspecial` indicate yet to be developed GW packages (e.g. GW structural equation models [Comber et al. 2018] or multiscale GW discriminant analysis [Comber et al., 2021]), and future developments and refinements, (e.g. the use of local bandwidths and tests for heterogeneity [Páez, Uchida, and Miyamoto, 2002a], or exploration of the geographical weights [Farber and Páez, 2007]), and so are not loaded via `gwverse`.

In this structure, the use of `gw` as the core package is essential to provide a consistent interface to all of the other functions, whereas `gwverse` provides a convenient wrapper for the modules.

### GW models

The GW framework, for regression, or any other analysis, has at its core a set of fundamental operations: the identification of nearby observations to the location being considered (i.e. observations under the kernel) and calculation of weights for those observations based on their distance to the location (i.e. the kernel center). The precise form of the functions that are used to undertake these operations will depend on user-specified choices about:

- kernel bandwidth type: a fixed bandwidth of a uniform size (distance), or an adaptive one in which size (distance) varies but the number of observations is fixed or uniform;
- kernel function shape: the form of the distance weighting, with choices of Gaussian, exponential, bisquare, tricube, or boxcar and many more.

After the kernel bandwidth type and shape have been defined, they can be applied to extract and weight local data in some kind of GW analysis. For example, in GWR they are used to create a series of local regressions (returning local coefficients and other regression related outputs), in a GW discriminant analysis they are used to determine the local posterior class probabilities, in a GW PCA they are used to determine the local components, local loadings and local scores.

The general operation of a given GW model such as GWR has two stages:

1. Determination of the kernel bandwidth size (whether fixed or adaptive) typically through some form of optimal evaluation;
2. Application of the optimal bandwidth in the final model.

The nature of these stages are specific to the particular GW model and can depend on whether or not some objective function exists (typically whether or not the model can predict). In this respect a regression might use a leave-one-out cross validation (CV) or an Akaike Information Criterion (AIC) metric to evaluate different bandwidth sizes. A discriminant analysis might use metrics commonly generated from a correspondence analysis (matrix) such as overall classification accuracy or the Kappa statistic.

This can be illustrated by considering the case for GWR. A GWR analysis requires the optimal bandwidth to be determined and then used to generate the local regressions. A linear bandwidth search would have the following sequence, after user decisions about bandwidth type and shape:

```

1. for each potential bandwidth
2. | for each regression point / location
3. | | identify the nearest n observations under the kernel
4. | | calculate the observations weights (bisquare, Gaussian etc)
5. | end
6. | evaluate the fit of the local regression
   | (via an overall CV or AIC diagnostic)
7. end
8. find the best performing bandwidth

```

Each GWR proceeds by undertaking steps 2 to 5 for the given bandwidth<sup>2</sup>. For a different GW model such as GW discriminant analysis, the outline is the broadly the same but with different localized models and fit measures in step 6.

And of course, many of these steps require a number of inputs: step 3 requires the distances between the location being considered and each observation suggesting the need for a distance matrix of some kind; step 3 also requires a specific function to identify nearby observations depending on the bandwidth type (fixed or adaptive); step 4 requires a weighting function that is also specific to the bandwidth type. The point being that a number of generic functions and data structures are required by any GW model, although they are used in different ways to support different types of bandwidth evaluation and final analyses.

### A function factory approach

An alternative to the `for` loop approach above is to take a function factory approach in combination with functionals. A `functional` is a function that has a function as its input and returns a vector as its output. They are commonly used alternatives to `for` loops because they are faster<sup>3</sup> and more flexible. A `function factory` is a function that returns a function. They have the advantages of allowing values to be precomputed within them (such as the distance matrix mentioned above), saving computation time, of supporting a multilevel design approach that more closely reflect the structure of the problem being addressed (e.g. wrapping user defined kernel and bandwidth choices within steps 3 and 4 above) and this way allow the complexity of the problem to be partitioned in into more easily understood (and testable) chunks (Wickham, 2019). Examples of current R packages that take this approach include MCMC (Geyer, 2020).

Their key advantage is that functions generated by function factories have an enclosing environment that is an execution environment of the function factory. This allows, for example, the names of functions in the enclosing environment to be associated with different



function bodies, have different values in different functions generated by function factories (e.g., a CV or an AIC evolution function in the GWR example above.) Thus the “enclosing environment of the manufactured function is unique and constant” (Wickham, 2019, section 10.2.4).

The for loop GWR schema presented above can be replaced with a function factory approach, in combination with functionals, to define a function with the `gwreg` module in Fig. 3.

The first stage is to determine the GWR bandwidth. A function factory is used to create a function that returns a function to evaluate a single bandwidth, that encloses a distance matrix and the data needed, along with user defined bandwidth choices:

```
single_bw_gwr = function(spatial_data, adaptive, kernel_shape, evaluation) {
  ### input data related
  1. create distance matrix from spatial_data
  ### core related (i.e. functions from gw)
  2. create the 'get nearby observations' function (adaptive parameter)
  3. create the 'weight nearby observations' (kernel_shape and adaptive parameters)
  ### application related (i.e. function from gwreg)
  4. define the evaluation function (evaluation parameter)
  ### output
  5. define the function to be returned
  function(bandwidth, spatial_data, formula) {
    create matrix of nearby locations for each observation
      (using the nearby function, distance matrix and bandwidth);
    create matrix of weights for each observation
      (using weight function, nearby locations matrix and bandwidth);
    return the results of the evaluation function given the formula
      (over the weighted data at each location);
  }
}
```

This is broadly equivalent in functionality to steps 2 to 6 in the for loop schema above. Once defined, this function can be used to evaluate a single bandwidth, for a given regression model / equation as specified in the `formula` parameter, and it returns the evaluation measure for that bandwidth:

```
my_gwr_bandwidth_function = single_bw_gwr(spatial_data, adaptive = TRUE,
                                           kernel_shape = "bisphere", evaluation = "AIC")
my_gwr_bandwidth_function(bw = 100, spatial_data, formula)
```

More usually the bandwidth evaluation function is used to determine the best bandwidth using an optimize function or through a linear search:

```
# optimise
optimise(my_gwr_bandwidth_function, c(10, nrow(spatial_data)),
        spatial_data, formula, maximum = FALSE)
# linear search
bws_adaptive = 10:nrow(spatial_data)
bws_res = sapply(bws_adaptive,
                function(x) my_gwr_bandwidth_function(x, spatial_data, formula))
bws_adaptive[which.min(bws_res)]
```

Having determined the optimal bandwidth, the second stage is to undertake the GWR analysis. This has a similar structure to the schema for the `single_bw_gwr` outline above, with the final operation in part 5 being replaced with:

```
return the coefficients of the local regression given the formula
  (for the weighted data at each location);
```

### **gwverse 0.0.1**

This approach has been used to create two new packages that provide a simple proof of concept of the new `gwverse`: the core package `gw` and a GWR package `gwregr`. These can be installed and used to undertake a GWR analysis in R, with fixed or adaptive bandwidth types, different kernel weights, and evaluated by either CV or corrected AIC.

The `gw` module contains three functions:

- `gw_get_nearby` which returns a function that identifies the observations nearby to the regression point under consideration for a given bandwidth. A different function is returned for adaptive and fixed bandwidths. The returned function takes an observation index, distance matrix and bandwidth as inputs and returns a vector of nearby observation indices.
- `gw_get_weight` which returns a function for different kernel weights: Gaussian, bisquare, tricube, exponential and boxcar. Again, a different function is returned for adaptive and fixed bandwidths. The returned function takes a bandwidth and a vector of distances to nearby observations as inputs. It returns a vector of weights for nearby observations.
- `gw_do_weight` which applies the selected weight function within an `apply` call within the function generated by the function factory. It takes as inputs, an index of observations, a bandwidth, a matrix of nearby observations for a given bandwidth, a distance matrix and the weight function. It returns a vector of weights for all observations.

The `gwregr` module contains four functions:

- `gw_get_lm_eval` which returns an evaluation function to evaluate the GWR results for a given bandwidth. This can be specified as “AIC” or “CV.” The returned function takes as inputs the data frame of the input spatial data, a formula, the matrix of nearby locations and the matrix of their weights. It returns an AIC or CV value.
- `gw_single_bw_gwr` which returns a function to evaluate a single GWR bandwidth. It takes as input point or polygon spatial dataset in `sf` format, containing the attributes to be modeled, a logical value to indicate whether an adaptive or fixed bandwidth distance is being used, the type of distance weighting to be used, and evaluation method for the local model, either “AIC” or “CV.” The returned function generates an evaluation measure.
- `gw_do_local_lm` which undertakes the local weighted regression using an `apply` function. It takes a vector weights (pertaining to an observation point), the formula and a flat data frame of the spatial data as inputs and returns a vector of coefficient estimates for the observation point being considered. This is only used after the bandwidth has been determined, and not in bandwidth selection.
- `gw_regr` which returns a function to undertake GWR once the optimal bandwidth has been defined. It takes as input parameters, the bandwidth value, the spatial dataset in `sf` format with the attributes to modeled, a formula, a logical value to indicate whether an adaptive or fixed bandwidth types is being used and the kernel shape. The returned function generates an  $n \times m$  matrix of coefficients at each location ( $n$ ) as specified in the formula ( $m$ ).

## A gwverse walkthrough

The `gw` and `gwverse` packages need to be installed. They can be installed from GitHub using the `devtools` package and loaded into the current R session, with `gwregr` loading `gw`:

```
library(devtools)
install_github("gwverse/gw")
install_github("gwverse/gwregr")
library(gwregr)
```

The `gwregr` package has dependences on the `gw` and `sf` packages for the geographically weighted generic framework and for spatial data respectively. The `gw` package contains with the `liudaogou` soils dataset (Wang, Zhang, and Huang, 2009). This has been used in a number of GWR-related studies (Comber et al., 2018, 2022) and can be loaded as follows:

```
data(liudaogou)
```

The dataset records a number of soil measurements at 689 observations in small subcatchment in China, including soil total nitrogen percentage (TNPC), here taken as the response variable, and five predictor variables: soil organic carbon (SOCgkg), percentage clay (ClayPC) and silt (SiltPC), nitrate nitrogen (NO3Ngkg) and ammonium (NH4Ngkg). In both Comber et al. (2018, 2022) the data were transformed TNPC, SOCgkg, NO3Ngkg and NH4Ngkg are transformed using natural logs and ClayPC is square root transformed. These transformations have been undertaken in this version of the dataset.

The first thing in any GWR is bandwidth selection. A function for doing this returned by the `gw_single_bw_gwr` function and the code below does this for an adaptive bandwidth and a bisquare kernel, applied over the `liudaogou` data, using AIC as the evaluation criteria:

```
gwr_bw_func = gw_single_bw_gwr(liudaogou, adaptive=TRUE, kernel="bisquare",
eval="AIC")
```

After defining a formula, the function can be run for a given bandwidth and returns the evaluation value (in this case the AIC score):

```
formula = as.formula(TNPC ~ SOCgkg + ClayPC + SiltPC + NO3Ngkg + NH4Ngkg)
gwr_bw_func(bw =100, formula)
```

```
## [1] 1081.702
```

Notice how no data needs to be passed to the function, as this is held in the function environment. The function environment bindings can be explored:

```
library(rlang)
# environments
env_print(gwr_bw_func)
```

```
## <environment: 0x7fe6658926d8>
## Parent: <environment: namespace:gwregr>
## Bindings:
## * eval_func: <fn>
```

## Geographical Analysis

```
## * weight_func: <fn>
## * nearby_func: <fn>
## * df: <df[,10]>
## * dist_mat: <dbl[,689]>
## * adaptive: <lgl>
## * kernel: <chr>
## * eval: <chr>
```

This indicates the objects and items that are bound to the function and we can examine individual environment bindings such as the data frame (`$df`) or the weighting function (`$weight_func`), in this case specified as bi-square:

```
fn_env(gwr_bw_func)$weight_func
```

```
## function (bw, dists)
## {
##   bw = max(dists)
##   (1 - (dists/bw)^2)^2
## }
## <bytecode: 0x7fe665375bd8>
## <environment: 0x7fe665370ac8>
```

A GWR analysis can proceed with this function. The optimal bandwidth can be determined using the optimization strategy or via a linear search. The `optimise` function returns the bandwidth (minimum) and fit value (objective):

```
opt = optimise(gwr_bw_func, c(10,nrow(liudaogou)),formula=formula, maximum=FALSE)
opt
```

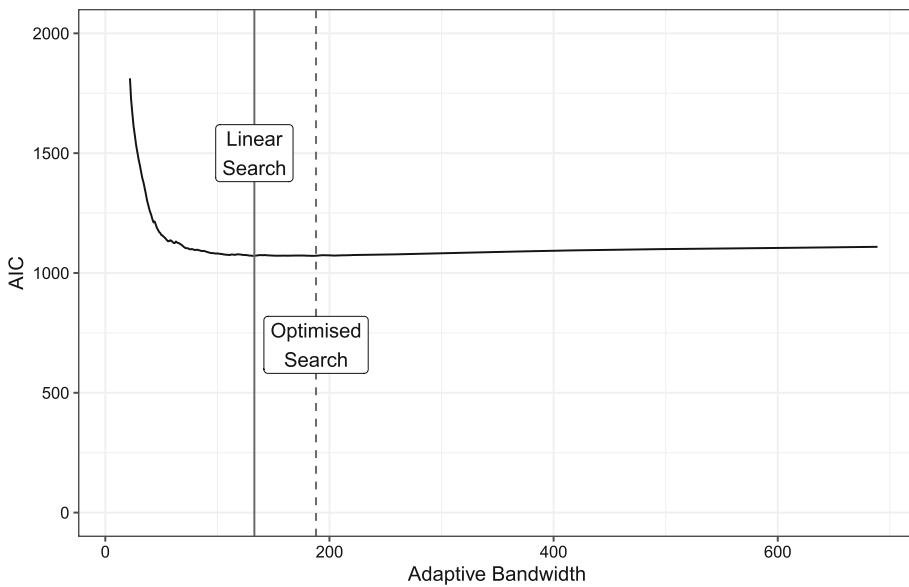
```
## $minimum
## [1] 188.034
##
## $objective
## [1] 1071.415
```

A linear search of all bandwidths is slower but should always be undertaken. This is because any optimization may determined local rather than global minima, which is the case here.

```
# create a vector of adaptive bandwidths
bws_adaptive = 10:nrow(liudaogou)
# apply the function to vector of bandwidths
bws_res = sapply(bws_adaptive, function(x) gwr_bw_func(x, formula))
bws_adaptive[which.min(bws_res)]
```

```
## [1] 133
```

The AIC scores of the individual bandwidths with the “optimised” bandwidth and the one determined through a linear search can be plotted as in Fig. 4.



**Figure 4.** The AIC scores of the bandwidths with the bandwidth found through an optimized search (dashed line) and a linear search (solid line), with the y-axis limited to highlight the difference. [Colour figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)].

Finally a GWR analysis can be undertaken using the best bandwidth and the coefficient estimates investigated as shown in the code below:

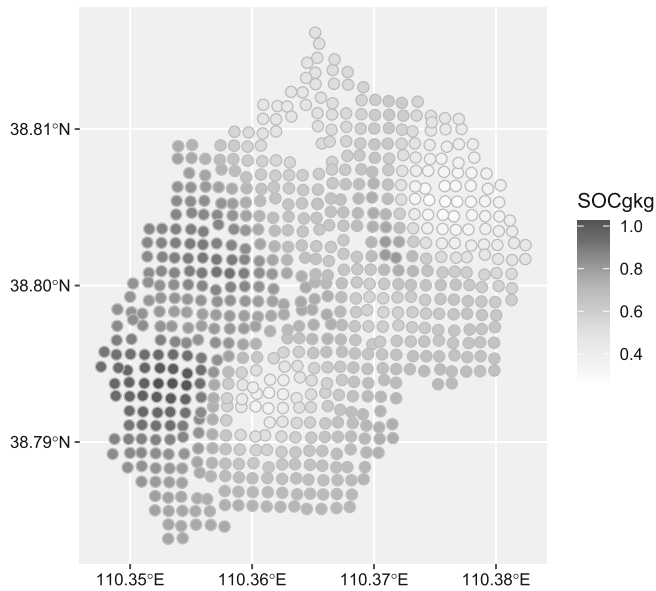
```
# extract the bandwidth
bw = bws_adaptive[which.min(bws_res)]
# define the GWR function
gwr_func = gw_regr(bw, formula, liudaogou, adaptive = T, "bisquare")
# apply to the data
coef_mat = gwr_func(formula)
# rename and examine
colnames(coef_mat) = c("Intercept", all.vars(formula)[-1])
round(apply(coef_mat, 2, summary), 3)
```

```
##      Intercept  SOCgkg  ClayPC  SiltPC  NO3Ngkg  NH4Ngkg
## Min.      -4.348  0.276  -0.272  -0.011  -0.349  -1.024
## 1st Qu.    -2.286  0.588  -0.053  0.006  -0.001  -0.360
## Median     -1.838  0.674  -0.009  0.009  0.090  -0.165
## Mean       -1.849  0.674  0.013  0.011  0.134  -0.209
## 3rd Qu.    -1.401  0.784  0.064  0.017  0.224  -0.021
## Max.       -0.492  1.027  0.377  0.031  0.761  0.587
```

And the results can be mapped as in the example in Fig. 5.

### A gwverse pipeline

A pipelined analysis can also be undertaken and a second example dataset is used to illustrate this. The Dublin voter data is described in Kavanagh (2004) and was used by Harris, Brunson,



**Figure 5.** The GWR coefficient estimates for SOCgkg. [Colour figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)].

and Charlton (2011), as well as being one included in the `GWmodel` package (Lu et al., 2014). This can be loaded from the `gw` package as in the code below. It has a number of variables over 322 electoral divisions in Dublin, Ireland, including percentages of voter turnout (`GenE12004`) as the dependent variable, high social class (`SC1`), unemployed (`Unempl`), without any formal educational (`LowEduc`) and different age groups (`Age18_24`, `Age25_44` and `Age45_64`).

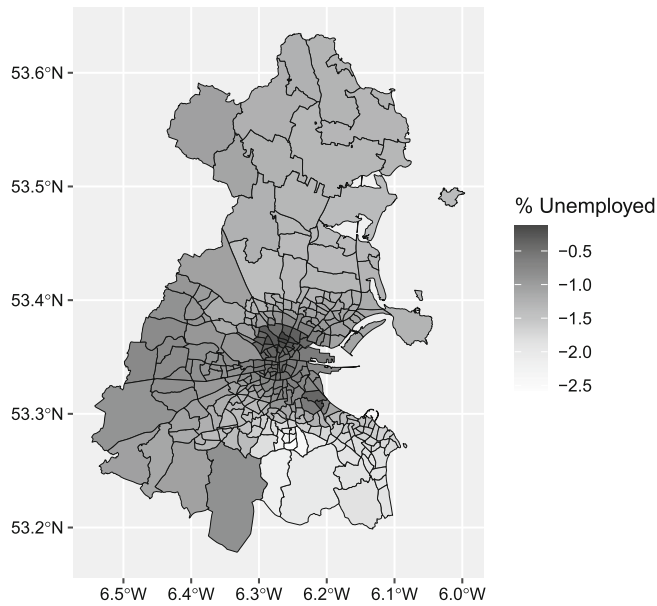
The code below loads the data and then, as an initial step, declares a regression formula and bandwidth evaluation function:

```
load("dubvotes.rda")
formula = as.formula(GenE12004 ~ SC1 + Unempl + Age18_24 + Age25_44 + Age45_64)
gwr_bw_func = gw_single_bw_gwr(dubvotes, adaptive=TRUE, kernel="bisquare",
eval="AIC")
```

These can then be put into a pipeline to find the optimal bandwidth from a set of potential bandwidths, using a linear search, and before finally returning the GWR function.

```
# potential bandwidths
tibble(bw = 10:nrow(dubvotes)) %>%
# evaluate them with the bandwidth evaluation function
rowwise() %>%
mutate(fit = gwr_bw_func(bw, formula)) %>%
as_tibble() %>%
# find the best fit and output
slice(which.min(fit)) %>%
select(bw) %>% unlist() %>% as.vector() %>%
# create the GWR function
gw_regr(formula, dubvotes, adaptive = T, "bisquare") -> gwr_func
```

The GWR function generated by the pipeline is lazy. That is, it is only when the function is called that the full sequence of pipeline operations are evaluated and their results bound to the



**Figure 6.** The GWR coefficient estimates for the percentage unemployed covariate. [Colour figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)].

function. The code below demonstrates this and the one of the coefficient estimates is mapped in Fig. 6.

```
# lazy elements are included in the GWR function
env_print(gwr_func)

## <environment: 0x7fe66593eb38>
## Parent: <environment: namespace:gwregr>
## Bindings:
## * weight_func: <fn>
## * nearby_func: <fn>
## * df: <df[,12]>
## * dist_mat: <dbl[,322]>
## * bw: <lazy>
## * formula: <lazy>
## * adaptive: <lgl>
## * kernel: <chr>

fn_env(gwr_bw_func)$bw

## NULL

# the function is run
coef_mat = gwr_func(formula)
colnames(coef_mat) = c("Intercept", all.vars(formula)[-1])
```

```
# now the lazy elements are populated  
fn_env(gwr_func)$bw
```

```
## [1] 81
```

### Some observations about `gwverse`

There are a number of observations associated with this approach as illustrated through this very simple package development:

1. The `gwverse` provides a consistent framework for undertaking different GW analyses including GWR.
2. Functions in the core `gw` module are never called directly by the user. Instead they are called from the modules for specific GW applications like GWR in the `gwregr` package.
3. The returned functions bind what they need within their environment. This makes them quicker than conventional approaches despite being larger in working memory.
4. The modularization promotes cleaner and consistent coding, allowing for “all options” of GWR as in Table 1 in the future.
5. The idea in `gwverse` is that all of the user-end GW functions (e.g. for regression, PCA, etc) are defined with this function factory approach, to ensure consistency of argument names (etc.) between functions.

Additionally, there is a need for consistent naming conventions. Function factories produce *anonymous functions* – that is, the body of a function but unassigned to a name. The name is created when the assignment happens. This means that although the function factory approach guarantees consistency in interface, naming is down to self-discipline. We suggest three naming rules:

- Everything is lower case. This is easier to use, as you do not have to remember whether a function is called `GWModel` or `GWmodel` for example;
- Spaces in names are represented as “\_”;
- All key functions begin with `gw_` - this helps on autocomplete in RStudio.

There are many potential areas of further development some which are discussed below, such as tidy considerations, whether functions should be made pipe-able. For example, in the `gwregr` pipeline, the data is not necessarily the first argument, as is normally the case. The pipe to the `gw_regr` function above required the bandwidth as the critical parameter to be passed to it, and this is the first argument in `gw_regr`.

### Discussion and conclusion

Here we propose a broad framework for the development of geographically weighted methods for spatial data analysis. Below, some implications of this proposal are considered. As proposers we expect to lead some of the initial contributions – we see development of the core `gw` worktools, followed by further development of the package for basic GWR (`gwregr`) as the likely first contributions, moving on to geographically weighted generalized linear model tools, such as Poisson and binomial regression. Other priorities include geographically weighted descriptive statistics and PCA. We would encourage others working in specified fields to contribute – for



example the remote sensing community may develop GW correspondence analysis or discriminant analysis approaches for classification, ecologists may develop geographically weighted redundancy analysis, GW variance partitioning, or GW canonical correspondence analysis.

The function factory approach provides a versatile framework but also there are technical considerations. When function factory techniques are used, specific “building block” functions are “bound in,” for ease of use. Using this approach to create the functions makes the specification of building blocks open and explicit, as well as consistent. An alternative may be to specify the building block functions as arguments to the main function. However, as well as ending up with a very complex argument list, there are issues relating to passing values to the building block functions, as well as handling the R dot-dot-dot ( . . . ) parameter syntax and partial parameter name matches (Geyer, 2020). There are, however some potential problems which must be considered. For example, when binding very large environments to a function (such as a large spatial database), this involves making a copy of that database – which could lead to storage or memory issues. There is a need for a set of guidelines on good practice for the use of function factories.

There are issues in code development to be addressed. For example in the *GWmodel* package, extensive use of linking R to C++ was made, to enable code to run faster. This would also be useful in this proposal – but a consistent approach to incorporating compiled C++ routines would be needed, and in addition to a GW core package (*gw*) providing R tools, a similar set of core procedures in C++ may be needed. In addition, many users and developers of geographically weighted methods use Python rather than R. Some consideration of interoperability could allow some degree of collaboration – working together to some extent could become possible via the use of the *reticulate* library, which allows Python code to run within R. A longer-term goal may be to provide a suite of Python modules mirroring the *gwverse* approach, encouraging the same development framework to run in parallel for the R and Python user communities.

Also, interoperability between the *gwverse* approach and other R packages and package families can be considered. For example, many users are now trained to work primarily with pipeline operators in R (either the `%>%` operator from *magrittr*, or the native `|>` operator) and designing *gwverse* functions to combine simply and intuitively with tools from other packages is an important consideration. This involves thought about which argument in *gwverse* functions should be first, and what form the returned value of the functions take. Ideally, one would wish *gwverse* functions to combine easily with functions from *sf* and data manipulation tools from *tidyverse*. The use of the functionality in the *purrr* and *tidymodels* packages will also be explored as the *gwverse* framework and the constituent packages are developed (and some of the refinements in Table 1 are incorporated). This will be interesting as GW approaches, because of their very nature, do not nest well into classic machine learning approaches and many *purrr* functions are higher level versions of the base R counterparts currently used in *gwverse* and may be less efficient. However, any nested *apply* functions may be more transparently coded using *map* functions from *purrr*.

Some organizational issues also require consideration. In particular we are proposing a family of R packages, maintained on GitHub, with periodic updates to CRAN. Although the approach here involves – and indeed encourages – collaboration, the project will require curation (much as CRAN does, but on a much smaller scale), and some structure for this needs to be agreed. This needs to address standards for *gwverse* package contribution – for example, checking that the package properly meets the function naming requirements set out above, and assessing whether any attached vignettes are well written and with sufficient content, and checking whether

CRAN's requirements are met when versions are submitted there. There may also be issues of managing contributions – for example if two contributors simultaneously propose packages for the same (or at least overlapping) GW techniques. Another related issue may be to create guidelines for developers, possibly combined with a 'how-to' manual outlining the use of GitHub, and the agreed curation framework. In this way users who have a question with no current means of investigation could be encouraged to become developers.

In conclusion, in this paper we have presented a modular framework for developing a consistent and interoperable family of R packages for geographically weighted analytical methods. We advocate the use of functionals and function factories as key principles in this framework. This offers a number of advantages: it facilitates the creation of packages having a consistent interface – so that for example an argument to a function specifying bandwidth always has the same name and the data supplied always has the same format. In addition, the use of a GW core package ensures that procedures appearing in several geographically weighted methods do not have to be recreated repeatedly (and possibly inconsistently) in several packages. With current trends suggesting an increasing interest in a number of new approaches, such as space-time weighting and multiscale models, perhaps now is an advantageous time to provide a consistent set of tools for software development. Potentially this brings together disparate GW working groups (of both users and developers) together under the same framework with mutual benefit to all, allowing rapid development of new GW models with a more streamlined community review process.

## Acknowledgements

This research is funded by the U.K. Natural Environment Research Council grant number NE/S009124/1 (Comber), grant number NE/N007433/1 (Harris) and by the British Academy under the Urban Infrastructures of Well-Being programme, grant number UWB190190 (Comber and Malleson). The data and code used to create all the results and figures are available at <https://github.com/gwverse/gwverseGAPaper>.

## Notes

- 1 In practice it is not always possible to distinguish between autocorrelation and heterogeneity, and so we have to accept that (paraphrasing Cressie, 1991, p. 114) “[o]ne person's correlated error structure may be another person's varying parameters.”
- 2 In reality a GWR evaluation by CV or AIC does not require the local model to be created only the observation weights at each location to be determined.
- 3 Actually they promote tighter coding and the avoidance of temporary data structures.

## Appendix: the current structure of the `GWmodel` package

### Overview

`GWmodel` includes functions to calibrate and estimate a wide range of techniques based on geographical weighting. These include: summary statistics, principal components analysis, discriminant analysis and various forms of regression; some of which are provided in basic and outlier resistant forms.

However the manual is, at the time of writing is long, some 85 pages in length. It is also organized alphabetically, and while the write-ups conform to the CRAN guidelines, they can be

hard to follow. For some of the more complex techniques there is little to guide the user as to which functions to use. This appendix provides a structured overview of the `GWmodel` library. It has been divided into sections, each section containing a group of related functions. Each section is headed with a summary table, giving the name of each function and a one-line description of its action. Below each table a bulleted list containing slightly more extended, but brief, descriptions of the function.

## Datasets

There are several built-in datasets. Most of these are used in the example code which is provided at the end of each function description. However, you can also use them to practice on, or as test data in their own right. With the exception of `Georgia` these are all in Spatial Polygon or Spatial Point Data Frame `sp` formats.

- `Dubvoter`: Voter turnout and social characters data in Greater Dublin for the 2002 General election and the 2002 census. Note that this dataset was originally thought to relate to 2004, so for continuity we have retained the associated variable names.
- `EWHP`: A house price dataset for England and Wales from 2001 with nine hedonic (explanatory) variables.
- `EWOutline`: Outline (`SpatialPolygonsDataFrame`) of the England and Wales house price data `EWHP`
- `Georgia`: Census data from the county of Georgia, United States
- `GeorgiaCounties`: The Georgia census data with boundaries for mapping
- `LondonBorough`: Outline (`SpatialPolygonsDataFrame`) of London boroughs for the `LondonHP` data.
- `LondonHP`: A house price data set with 18 hedonic variables for London in 2001.
- `USElect`: Results of the 2004 U.S. presidential election at the county level, together with five socio-economic (census) variables. This data can be used with `GW Discriminant Analysis`.

## Service functions

- `gw.dist`: Calculate a distance vector(matrix) between any `GW` model calibration point(s) and the data points.
- `gw.weight`: Calculate a weight vector(matrix) from a distance vector(matrix).
- `gwr.write`: This function writes the calibration result of function `gwr.basic` to a text file
- `gwr.write.shp`: This function writes the calibration result of function `gwr.basic` to a shapefile

## GW descriptive statistics

- `bw.gwss.average`: A function for automatic bandwidth selections to calculate `GW` summary averages, including means and medians, via a `CV` approach.
- `gwss`: This function calculates basic and robust `GWSS`. This includes geographically weighted means, standard deviations and skew. Robust alternatives include geographically weighted medians, interquartile ranges and quantile imbalances. This function also calculates basic geographically weighted covariances together with basic and robust geographically weighted correlations.

- `gwss.montecarlo`: This function implements Monte Carlo (randomization) tests for the GW summary statistics found in `gwss`.
- `gw.pcplot`: This function provides a geographically weighted parallel coordinate plot for locally investigating a multivariate data set. It has an option that weights the lines of the plot with increasing levels of transparency, according to their observation's distance from a specified focal/observation point.
- `gwpca.glyph.plot`: This function provides a multivariate glyph plot of GWPCA loadings at each output location.

### **GW (gaussian) regression**

- `bw.gwr`: A function for automatic bandwidth selection to calibrate a basic GWR model.
- `gwr.basic`: This function implements basic GWR.
- `gwr.robust`: This function implements two robust GWR models.
- `gwr.hetero`: This function implements a heteroskedastic GWR model
- `gwr.bootstrap`: This function implements bootstrap methods to test for coefficients.
- `gwr.montecarlo`: This function implements a Monte Carlo (randomization) test to test for significant (spatial) variability of a GWR model's parameters or coefficients.
- `gwr.t.adjust`: Given a set of p-values from the pseudo t-tests of basic GWR outputs, this function returns adjusted p-values using: (a) Bonferroni, (b) Benjamini-Hochberg, (c) Benjamini-Yekutieli and (d) Fotheringham-Byrne procedures.
- `gwr.model.selection`: This function selects one GWR model from many alternatives based on the AICc values.
- `gwr.model.sort`: Sort the results from the GWR model selection function `gwr.model.selection`.
- `gwr.model.view`: This function visualizes the GWR models from `gwr.model.selection`.
- `gwr.mink.approach`: This function implements the Minkowski approach to select an optimum distance metric for calibrating a GWR model.
- `gwr.mink.matrixview`: This function visualizes the AICc/CV results from the `gwr.mink` approach.
- `gwr.mink.pval`: These functions implement heuristics to select the values of  $p$  from two intervals:  $(0, 2]$  in a backward direction and  $(2, \infty)$  in a forward direction.

### **Generalized GWR**

- `bw.ggwr`: A function for automatic bandwidth selection to calibrate a generalized GWR model.
- `ggwr.basic`: This function implements generalized GWR.
- `ggwr.cv`: This function finds the CV score for a specified bandwidth for generalized GWR. It can be used to construct the bandwidth function across all possible bandwidths and compared to that found automatically.
- `ggwr.cv.contrib`: This function finds the individual CV score at each observation location, for a generalized GWR model, for a specified bandwidth. These data can be mapped to detect unusually high or low CVs scores.

### **Locally compensated ridge GWR**

- `bw.gwr.lcr`: A function for automatic bandwidth selection for `gwr.lcr` via a cross-validation approach only.

- `gwr.lcr`: To address possible local collinearity problems in basic GWR, GWR-LCR finds local ridge parameters at affected locations (set by a user-specified threshold for the design matrix condition number).
- `gwr.lcr.cv`: This function finds the CV score for a specified bandwidth for GWR-LCR. It can be used to construct the bandwidth function across all possible bandwidths and compared to that found automatically.
- `gwr.lcr.cv.contrib`: This function finds the individual cross-validation score at each observation location, for a GWRLCR model, for a specified bandwidth. These data can be mapped to detect unusually high or low cross-validations scores.
- `gwr.collin.diagno`: This function provides a series of local collinearity diagnostics for the independent variables of a basic GWR model.

### ***Multiscale GWR***

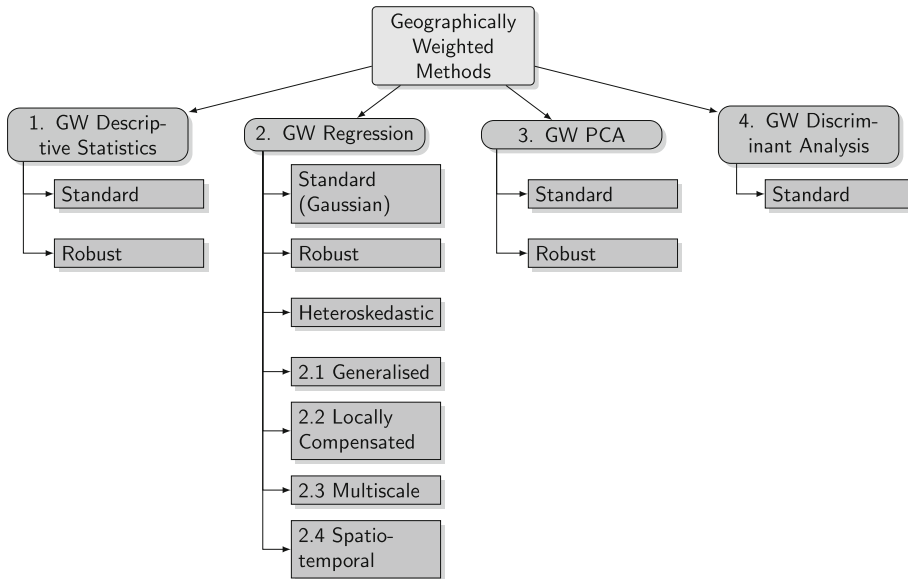
- `gwr.mixed`: This function implements mixed (semi-parametric) GWR.
- `gwr.multiscale`: This function implements multiscale GWR to detect variations in regression relationships across different spatial scales. This function can not only find a different bandwidth for each relationship but also (and simultaneously) find a different distance metric for each relationship (if required to do so).

### ***Geographically and temporally weighted regression***

- `bw.gtwr`: A function for automatic bandwidth selection to calibrate a Geographically and Temporally Weighted Regression (GTWR) model.
- `gtwr`: A function for calibrating a GTWR model.

### ***Geographically weighted principal components analysis***

- `bw.gw pca`: A function for automatic bandwidth selection to calibrate a basic or robust GWPCA via a CV approach only
- `gw pca`: This function implements basic or robust GWPCA.
- `gw pca . check . components`: The function interacts with the multivariate glyph plot of GWPCA loadings.
- `gw pca . cv`: This function finds the CV score for a specified bandwidth for basic or robust GWPCA. It can be used to construct the bandwidth function across all possible bandwidths and compared to that found automatically.
- `gw pca . cv . contrib`: This function finds the individual CV score at each observation location, for a GWPCA model, for a specified bandwidth. These data can be mapped to detect unusually high or low CV scores
- `gw pca . montecarlo . 1`: This function implements a Monte Carlo (randomization) test for a basic or robust GW PCA with the bandwidth prespecified and constant. The test evaluates whether the GW eigenvalues vary significantly across space for the first component only.
- `gw pca . montecarlo . 2`: This function implements a Monte Carlo (randomization) test for a basic or robust GW PCA with the bandwidth automatically reselected via the CV approach. The test evaluates whether the GW eigenvalues vary significantly across space for the first component only.



**Figure A1.** An ontology of GW approaches currently in GWmodel. [Colour figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)].

**Geographically weighted discriminant analysis**

- `bw.gwda`: A function for automatic bandwidth selection for GW Discriminant Analysis using a CV approach only
- `gwda`: A function to implement GW discriminant analysis.

**References**

Anselin, L. (1995). “Local Indicators of Spatial Association – LISA.” *Geographical Analysis* 27(2), 93–115.

Bivand, R. S. (2021). “Progress in the R Ecosystem for Representing and Handling Spatial Data.” *Journal of Geographical Systems* 23(4), 515–46.

Bivand, R., D. Yu, T. Nakaya, M.-A. Garcia-Lopez, and M. R. Bivand. (2020). “Package ‘Spgwr.’” R Software Package.

Brunsdon, C., A. S. Fotheringham, and M. E. Charlton. (1996). “Geographically Weighted Regression: A Method for Exploring Spatial Nonstationarity.” *Geographical Analysis* 28(4), 281–98.

Brunsdon, C., A. S. Fotheringham, and M. Charlton. (2002). “Geographically Weighted Summary Statistics – A Framework for Localised Exploratory Data Analysis.” *Computers, Environment and Urban Systems* 26(6), 501–24.

Brunsdon, C., S. Fotheringham, and M. Charlton. (2007). “Geographically Weighted Discriminant Analysis.” *Geographical Analysis* 39(4), 376–96.

Charlton, M., A. S. Fotheringham, and C. Brunsdon. (2003). *GWR 3: Software for Geographically Weighted Regression*. Maynooth: NCG, National University of Ireland Maynooth.

Chen, L., C. Ren, B. Zhang, Z. Wang, and Y. Xi. (2018). “Estimation of Forest Above-Ground Biomass by Geographically Weighted Regression and Machine Learning with Sentinel Imagery.” *Forests* 9(10), 582.

Cleveland, W. S. (1979). “Robust Locally Weighted Regression and Smoothing Scatterplots.” *Journal of the American Statistical Association* 74(368), 829–36.

- Comber, A., C. Brunson, M. Charlton, G. Dong, R. Harris, B. Lu, Y. Lü, et al. (2022). “A Route Map for Successful Applications of Geographically Weighted Regression.” *Geographical Analysis*. <https://doi.org/10.1111/gean.12316>
- Comber, A., C. Fonte, G. Foody, S. Fritz, P. Harris, A.-M. Olteanu-Raimond, and L. See. (2016). “Geographically Weighted Evidence Combination Approaches for Combining Discordant and Inconsistent Volunteered Geographical Information.” *GeoInformatica* 20(3), 503–27.
- Comber, A., and P. Harris. (2018). “Geographically Weighted Elastic Net Logistic Regression.” *Journal of Geographical Systems* 20(4), 317–41.
- Comber, Alexis, Ting Li, Yihe Lü, Bojie Fu, and Paul Harris. 2017. “Geographically Weighted Structural Equation Models: Spatial Variation in the Drivers of Environmental Restoration Effectiveness.” In *Societal Geo-Innovation. 20th AGILE Conference Proceedings*. Wageningen, Netherland.
- Comber, A., N. Malleon, H. N. T. Thuy, T. B. Quang, M. Kieu, H. H. Phe, and P. Harris. (2021). “Multiscale Geographically Weighted Discriminant Analysis.” In *GIScience 2021 Short Paper Proceedings*. UC Santa Barbara. <https://doi.org/10.25436/E2pp4f>
- Comber, A., Y. Wang, Y. Lü, X. Zhang, and P. Harris. (2018). “Hyper-Local Geographically Weighted Regression: Extending GWR through Local Model Selection and Local Bandwidth Optimization.” *Journal of Spatial Information Science* 17, 63–84.
- Cressie, N. (1991). *Statistics for Spatial Data*. New York: John Wiley & Sons.
- Du, Z., Z. Wang, S. Wu, F. Zhang, and R. Liu. (2020). “Geographically Neural Network Weighted Regression for the Accurate Estimation of Spatial Non-Stationarity.” *International Journal of Geographical Information Science* 34(7), 1353–77.
- Dykes, J., and C. Brunson. (2007). “Geographically Weighted Visualization: Interactive Graphics for Scale-Varying Exploratory Analysis.” *IEEE Transactions on Visualization and Computer Graphics* 13(6), 1161–8.
- Farber, S., and A. Páez. (2007). “A Systematic Investigation of Cross-Validation in GWR Model Estimation: Empirical Analysis and Monte Carlo Simulations.” *Journal of Geographical Systems* 9(4), 371–96.
- Foley, P., and U. Demšar. (2013). “Using Geovisual Analytics to Compare the Performance of Geographically Weighted Discriminant Analysis Versus Its Global Counterpart, Linear Discriminant Analysis.” *International Journal of Geographical Information Science* 27(4), 633–61.
- Fotheringham, A. S., C. Brunson, and M. Charlton. (2002). *Geographically Weighted Regression: The Analysis of Spatially Varying Relationships*. New York: John Wiley & Sons.
- Fotheringham, A. S., R. Crespo, and J. Yao. (2015). “Geographical and Temporal Weighted Regression (GTWR).” *Geographical Analysis* 47(4), 431–52.
- Fotheringham, A. S., W. Yang, and W. Kang. (2017). “Multiscale Geographically Weighted Regression (MGWR).” *Annals of the American Association of Geographers* 107(6), 1247–65.
- Geniaux, G., and D. Martinetti. (2018). “A New Method for Dealing Simultaneously with Spatial Autocorrelation and Spatial Heterogeneity in Regression Models.” *Regional Science and Urban Economics* 72, 74–85.
- Getis, A., and J. K. Ord. (2010). “The Analysis of Spatial Association by Use of Distance Statistics.” In *Perspectives on Spatial Data Analysis*, 127–45. Berlin, Heidelberg: Springer.
- Geyer, C. J. (2020). “MCMC Package Example (Version 0.9.7).” <https://cran.rediris.es/web/packages/mcmc/vignettes/demo.pdf>
- Gollini, I., B. Lu, M. Charlton, C. Brunson, and P. Harris. (2015). “GWmodel: An R Package for Exploring Spatial Heterogeneity Using Geographically Weighted Models.” *Journal of Statistical Software* 63(17), 1–50.
- Hagenauer, J., and M. Helbich. (2022). “A Geographically Weighted Artificial Neural Network.” *International Journal of Geographical Information Science* 36(2), 215–235.
- Harris, P. (2019). “A Simulation Study on Specifying a Regression Model for Spatial Data: Choosing Between Autocorrelation and Heterogeneity Effects.” *Geographical Analysis* 51(2), 151–81. <https://doi.org/10.1111/gean.12163>
- Harris, P., C. Brunson, and M. Charlton. (2011). “Geographically Weighted Principal Components Analysis.” *International Journal of Geographical Information Science* 25(10), 1717–36.

- Harris, P., C. Brunson, and A. Stewart Fotheringham. (2011). "Links, Comparisons and Extensions of the Geographically Weighted Regression Model When Used as a Spatial Predictor." *Stochastic Environmental Research and Risk Assessment* 25(2), 123–38.
- Harris, P., M. Charlton, and A. Stewart Fotheringham. (2010). "Moving Window Kriging with Geographically Weighted Variograms." *Stochastic Environmental Research and Risk Assessment* 24(8), 1193–209.
- Harris, P., A. Clarke, S. Juggins, C. Brunson, and M. Charlton. (2014). "Geographically Weighted Methods and Their Use in Network Re-Designs for Environmental Monitoring." *Stochastic Environmental Research and Risk Assessment* 28(7), 1869–87.
- Harris, P., A. Stewart Fotheringham, and S. Juggins. (2010). "Robust Geographically Weighted 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.
- Huang, B., B. Wu, and M. Barry. (2010). "Geographically and Temporally Weighted Regression for Modeling Spatio-Temporal Variation in House Prices." *International Journal of Geographical Information Science* 24(3), 383–401.
- Kalogirou, S. (2019). *lctools: Local Correlation, Spatial Inequalities, Geographically Weighted Regression and Other Tools*. R package version 0.2-7. <https://CRAN.R-project.org/package=lctools>.
- Kavanagh, A. (2004). "Turnout or turned off? electoral participation in Dublin in the 21st century." *Journal of Irish Urban Studies* 3(2), 1–24.
- Li, K., and N. S. N. Lam. (2018). "Geographically Weighted Elastic Net: A Variable-Selection and Modeling Method Under the Spatially Nonstationary Condition." *Annals of the American Association of Geographers* 108(6), 1582–600.
- Li, L. (2019). "Geographically Weighted Machine Learning and Downscaling for High-Resolution Spatiotemporal Estimations of Wind Speed." *Remote Sensing* 11(11), 1378.
- Li, Z., T. Oshan, S. Fotheringham, W. Kang, L. Wolf, H. Yu, and W. Luo. (2019). *MGWR 1.0 User Manual*. Tempe, USA: Spatial Analysis Research Center, Arizona State University.
- Lu, B., C. Brunson, M. Charlton, and P. Harris. (2017). "Geographically Weighted Regression with Parameter-Specific Distance Metrics." *International Journal of Geographical Information Science* 31(5), 982–98.
- Lu, B., M. Charlton, C. Brunson, and P. Harris. (2016). "The Minkowski Approach for Choosing the Distance Metric in Geographically Weighted Regression." *International Journal of Geographical Information Science* 30(2), 351–68.
- Lu, B., P. Harris, M. Charlton, and C. Brunson. (2014). "The GWmodel r Package: Further Topics for Exploring Spatial Heterogeneity Using Geographically Weighted Models." *Geo-Spatial Information Science* 17(2), 85–101.
- Lu, B., W. Yang, Y. Ge, and P. Harris. (2018). "Improvements to the Calibration of a Geographically Weighted Regression with Parameter-Specific Distance Metrics and Bandwidths." *Computers, Environment and Urban Systems* 71, 41–57.
- McMillen, D. (2013). *McSpatial: Nonparametric Spatial DataAnalysis*. R package version 2.0. <http://CRAN.R-project.org/package=McSpatial>.
- Murakami, D., and M. Tsutsumi. (2015). "Area-to-Point Parameter Estimation with Geographically Weighted Regression." *Journal of Geographical Systems* 17(3), 207–25.
- Murakami, D., N. Tsutsumida, T. Yoshida, T. Nakaya, and L. Binbin. (2020). "Scalable GWR: A Linear-Time Algorithm for Large-Scale Geographically Weighted Regression with Polynomial Kernels." *Annals of the American Association of Geographers*, 111(2), 459–480.
- Nakaya, T., M. Charlton, P. Lewis, C. Brunson, J. Yao, and S. Fotheringham. (2014). "GWR4 user manual: GWR4 Windows application for geographically weighted regression modelling." [https://raw.githubusercontent.com/gwrtools/gwr4/master/GWR4manual\\_409.pdf](https://raw.githubusercontent.com/gwrtools/gwr4/master/GWR4manual_409.pdf)
- Openshaw, S. (1996). "Developing GIS relevant zone based spatial analysis methods". In *Spatial Analysis: Modelling in a GIS Environment*, edited by P. Longley and M. Batty, 55–73. New York: John Wiley & Sons.
- Oshan, T. M., Z. Li, W. Kang, L. J. Wolf, and A. S. Fotheringham. (2019). "Mgwr: A Python Implementation of Multiscale Geographically Weighted Regression for Investigating Process Spatial Heterogeneity and Scale." *ISPRS International Journal of Geo-Information* 8(6), 269.



- Páez, A., T. Uchida, and K. Miyamoto. (2002a). "A General Framework for Estimation and Inference of Geographically Weighted Regression Models: 1. Location-Specific Kernel Bandwidths and a Test for Locational Heterogeneity." *Environment and Planning A* 34(4), 733–54.
- Páez, A., T. Uchida, and K. Miyamoto. (2002b). "A General Framework for Estimation and Inference of Geographically Weighted Regression Models: 2. Spatial Association and Model Specification Tests." *Environment and Planning A* 34(5), 883–904.
- Pebesma, E. J. (2018). "Simple Features for R: Standardized Support for Spatial Vector Data." *The R Journal* 10(1), 439.
- Pebesma, E., and R. S. Bivand. (2005). "S Classes and Methods for Spatial Data: The Sp Package." *R News* 5(2), 9–13.
- Quiñones, S., A. Goyal, and Z. U. Ahmed. (2021). "Geographically Weighted Machine Learning Model for Untangling Spatial Heterogeneity of Type 2 Diabetes Mellitus (T2d) Prevalence in the USA." *Scientific Reports* 11(1), 1–13.
- Rey, S. J., and L. Anselin. (2010). "PySAL: A Python Library of Spatial Analytical Methods." In *Handbook of Applied Spatial Analysis*, 175–93. Heidelberg: Springer.
- Tibshirani, R., and T. Hastie. (1987). "Local Likelihood Estimation." *Journal of the American Statistical Association* 82(398), 559–67.
- Tobler, W. R. (1970). "A Computer Movie Simulating Urban Growth in the Detroit Region." *Economic Geography* 46(sup1), 234–40.
- Wang, Y., X. Zhang, and C. Huang. (2009). "Spatial Variability of Soil Total Nitrogen and Soil Total Phosphorus Under Different Land Uses in a Small Watershed on the Loess Plateau, China." *Geoderma* 150(1–2), 141–9.
- Wheeler, D. (2013). "Gwrr: Fits Geographically Weighted Regression Models with Diagnostic Tools." <https://Cran.r-Project.org/Web/Packages/Gwrr/Index.html>
- Wheeler, D. C. (2007). "Diagnostic Tools and a Remedial Method for Collinearity in Geographically Weighted Regression." *Environment and Planning A* 39(10), 2464–81.
- Wheeler, D. C. (2009). "Simultaneous Coefficient Penalization and Model Selection in Geographically Weighted Regression: The Geographically Weighted Lasso." *Environment and Planning A* 41(3), 722–42.
- Wickham, H. (2014). "Tidy Data." *Journal of Statistical Software* 59(10), 1–23.
- Wickham, H. (2019). *Advanced R*. Boca Raton: Chapman & Hall/CRC Press.
- Xu, S., Q. Zhao, K. Yin, G. He, Z. Zhang, G. Wang, M. Wen, and N. Zhang. (2021). "Spatial Downscaling of Land Surface Temperature Based on a Multi-Factor Geographically Weighted Machine Learning Model." *Remote Sensing* 13(6), 1186.
- Yang, W. (2014). "An Extension of Geographically Weighted Regression with Flexible Bandwidths." PhD thesis, University of St Andrews.
- Yoneoka, D., E. Saito, and S. Nakaoka. (2016). "New Algorithm for Constructing Area-Based Index with Geographical Heterogeneities and Variable Selection: An Application to Gastric Cancer Screening." *Scientific Reports* 6(1), 1–7.