The R project and spatial data analysis: Open Source software and user/developer fusion*

Roger Bivand†

3rd March 2004

Abstract

R is a language and environment for statistical computing and graphics, and is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form. The R community also supports the contribution of packages adding functionality to the software, providing quality assurance services and on-line archives. The archives also contain executable binaries for a wide range of operating systems, including Windows, Mac OSX, and Unix/GNU Linux. Contributed packages distributed through the archive network also use Open Source licenses. The resulting communities of users and developers of these software resources express a fusion between developers and users, with the questions and issues raised by users, including beginning users, becoming vital input to developers seeking to extend functionality and remove infelicities.

The presentation covers the status of spatial data analysis in R, and prospects for future improvement, including further links with GIS. Because R is an implementation of S, it allows functionality to be developed rapidly, building on the security of well-proven numerical and graphics algorithms. But it is not only the practical aspect of the presence of geographical concerns within the R project that deserves attention: asserting this presence seems important for the discipline in maintaining our presence in applied empirical analysis, since much of the progress in cognate fields is taking place in Open Source arenas.

Keywords: spatial data analysis, open source software, statistical computing.

1 Introduction

This draft paper constitutes an introduction to the use of R and R project tools and features for developing spatial data analysis software, and for doing spatial data analysis. Since a good deal of information about R spatial projects is now available online, it will not be reviewed here at length, but links are provided. More space is devoted to a discussion of R as an environment for implementing data analysis

*3442. Open Source Software for Spatial Analysis: Salon C, Tuesday, 16 March 2004, 1300–1440.

†Economic Geography Section, Department of Economics, Norwegian School of Economics and Business Administration, Breiviksveien 40, N-5045 Bergen, Norway; Roger.Bivand@nhh.no

functions, covering specific aspects both of the language and the community environment growing around the language. There are plenty of languages that may be used for application building, but fewer include both low-level hooks for dynamically loadable compiled code and a very high level interpreted command line. Some of this subset are in active use in spatial data analysis, in particular Matlab and implementations of S, R and S-PLUS. Typically, the high level interpreted languages do not have dynamic loading hooks, and lower level languages with hooks are not convenient to use without scripting.

In order to set this discussion of application development in a specific context, a worked example of choropleth map class intervals will be presented in some detail. This allows us to see how spatial data can be handled in R, including reference system transformation, before moving on to exploring class intervals. Among the methods employed are multicolour join count statistics, a legacy technique perhaps deserving more attention. From a different direction, techniques for exploring for natural breaks are also examined, techniques already available in contributed packages in the R community. Since R is widely used for teaching and research in machine learning, it is often the case that well-tried code already exists for interesting methods without the overhead of coding from scratch. Being part of a data-analysis community does mean that it is possible to draw on the strengths of other contributors while simultaneously contributing functionality that may help others see that adding the spatial dimension to their analyses in a practical way can give them more domain-relevant insight. In addition, user comments, questions and bug reports constitute important and formative inputs to the development of contributed packages, and in this sense, the R project is a living example of user/developer fusion.

2 The R project

Changes in scientific computing applied to data analysis are less rapid than in many consumer fields. Underlying technologies are well-known, and have been available for decades. The visible changes are much more in the burgeoning of online and virtual scientific communities, and in the availability of cross-platform applications giving many more researchers, scientists and students access to implementations of methods that until recently only appeared in journal articles. Grunsky (2002) has drawn attention to the R data analysis and statistical programming environment as an emerging area of promise for the geosciences.

R (R Development Core Team, 2003) is an implementation of the S language initially written by Ross Ihaka and Robert Gentleman (Ihaka and Gentleman, 1996). S-PLUS is a proprietary implementation of S; since both R and S-PLUS are implementations of the same underlying language, they are often able to execute the same interpreted code. R follows most of the Blue and White books (Becker, Chambers and Wilks, 1988; Chambers and Hastie, 1992) that describe S, and also implements parts of the more recent Green Book (Chambers, 1998). R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form. It compiles and runs out of the box on a wide variety of Unix platforms and similar systems (including GNU/Linux). It also compiles and runs on Windows systems and MacOSX, and as such provides a functional cross-platform distribution standard for software and data. A good introduction is provided by Dalgaard (2002).

It is fair to say that the statistical and data analytic interests of the R community are catholic and rigorous, and partipants are enthusiastic, challenging the perceived barriers between proprietary and open source software in the interests of better, more timely, and more professional analysis in the proper sense of the word. Naturally, other and overlapping communities share these qualities; Matlab users interact and share code willingly, Python programmers do the same, as do users and developers working in a range of language environments. In all of these cases, users and developers share interests in data analysis, in which participants in different fields can drawn on each others' experience.

At the time of writing, searching the R site for "spatial" yielded 859 hits, almost double the 447 hits found in May 2002. As Ripley (2001) comments, some of the hesitancy that was observable in contributing spatial analysis packages to R was due to the availability of the S-PLUS SpatialStats module: duplicating existing work (including GIS integration) had not seemed fruitful. Over the recent period, however, a number of packages have been released on CRAN, the Comprehensive R Archive Network (`http://cran.r-project.org`), in all three areas of spatial data analysis (point patterns, continuous surfaces, and lattice data) as well as in accessing geographical data in common GIS formats.

An early package in point pattern and continuous surface analysis was contributed to R from the first edition of Venables and Ripley (2002) — now in its fourth edition. Initially written for S and S-PLUS, it was ported to R by Brian Ripley following work by Albrecht Gebhardt. S-PLUS itself has an add-on module for spatial statistics providing a wide range of useful functions (Kaluzny et al., 1996). Descriptions of some of the R packages available are given in notes in R News (Ripley, 2001; Bivand, 2001), while a more dated survey was made by Bivand and Gebhardt (2000), reflecting the situation at that time. R as a whole is experiencing rapid growth in the number of packages, and because it can be difficult to get an overview of relevant software, authors of spatial statistics software agreed to set up a website. This has been in operation since mid-2003, has an associated mailing list, and currently can be reached by searching Google with the string: "R spatial" (`http://sal.agecon.uiuc.edu/csiss/Rgeo/index.html`).

3 R as an environment for implementing data analysis functions

In the terminology used in the R project, the programming environment is provided as a program interpreting the language, managing memory, providing services, and running the user interface (command line, history mechanisms and graphics windows). In passing, it worth noting that the language supports the use of expressions as function arguments, allowing considerable fluency in command line interaction and function writing. There is a clearly defined interface to this program, permitting additional compiled functions to be dynamically loaded, and interpreted functions to be introduced into the list of known objects. By default on program startup, the **base** package is loaded, followed by autoloaded packages, and other packages are loaded at will. R distributions are accompanied by a small set of packages, available by default, and a larger recommended collection. Contributed packages provide a powerful vehicle for distributing additional code, and their structure encourages the extension of the catalogue of functions available to users. Venables and Ripley (2000) cover S programming in detail, including a description of packaging mechanisms in R; other information is to be found in online documentation, also included in the software distribution. Since R is open source, the utlimate documentation is the source code itself.

Packages are contributed in source form, and may also be distributed in compiled, binary, form. Most, although not all, source packages on CRAN are built as binaries in an automated process for Windows and Mac OSX platforms; on Unix and GNU/Linux platforms, users are more likely to have access to appropriate build trains. Issuing the `update.packages()` command on Windows and Mac OSX platforms in an R session online permits the updating of binary installed packages from CRAN (either the master site or a closer mirror); `install.packages()` with a character vector of package names as an argument installs the required binary packages from CRAN. On Unix and GNU/Linux platforms, the same functions update and install source packages, building them locally. These mechanisms provide users with one of the key benefits of open source development, access to software subject to rapid release cycles.

A minimal source package contains a directory of files of interpreted functions, and a directory of files of documentation of this code. All functions should be fully documented, and must be if the package is to be distributed through the CRAN. It is customary for the documentation to include example sections that can be executed to demonstrate what the function does; typically use is made of data sets distributed with the base package if possible. It is a requirement for distribution on CRAN that the examples run without error — not a guarantee that the implementations are correct scientifically, but at least that running them does not terminate with an error. If one chooses to use domain-specific data sets, then the package will contain a further directory with the necessary data files, which in turn are documented in the help file directory. Interpreted R code can of course be read within the context of the user interface, and functions may be edited, saved to user files, and sourced back into the program.

In some circumstances, it is desirable to move the internal computations of a function to a compiled language, although, as we will see, this is not an absolute requirement because the built-in internal functions themselves interface highly optimized and heavily debugged compiled code. In this case, a source directory will also be present, with C, C++, or Fortran 77 source files. Here it is sufficient to mention the possibility of dynamically loading used-compiled shared object code, and the usefulness of R header files in C in particular giving direct access R functions. R data objects may be readily passed to and from user functions, and R memory allocation mechanisms may be called from such user-compiled functions. If required, R code can even be executed in such user-compiled functions.

For instance, R provides a `factor` object definition for categorical variables, with a character vector of level labels and an integer vector of observation values pointing to the vector of levels. In the GRASS/R compiled interface using GRASS library functions (Bivand, 2000), moving categorical raster data between R and GRASS is accomplished fast and fully preserving labels by operating on R factor objects in C functions. Within R, functions are written to use object classes, for example the factor class, to test for object suitability, or in many modelling situations to convert factors into appropriate dummy variables. The use of classes in R for spatial data analysis is discussed in more depth by Bivand (2002).

Users can at will create new classes for which the class method despatch mechanism can be invoked. The `summary()` function appears to be a single function, but in fact calls appropriate summary functions based on the class of the first argument; the same applies to the `plot()` and `print()` functions. The extent to which spatial data analysis packages use class and method based functions varies, mostly depending on the age of the code and on the potential value of such revisions; status as of early 2003 is reviewed in Bivand (2003).

As has already been indicated indirectly, much of the added value of the R project extends beyond the standard functionality of the language and programming environment. The archive network is such an extension, as are the package checking mechanisms (in the **tools** package). Together with the test suites, they have been developed to facilitate quality control of the core system rather than user contributed packages, but because the same standards are used, the contributed packages also benefit in terms of organisation and coherence. The use of profiling will be demonstrated below, and is a typical side effect of the spillovers from the core team to users. Another useful spillover from package contributors to the core team is that all contributed packages on CRAN are checked nightly against the code bases of the released version of R, the patched version, and the development version — which will become the next full release. This means that the core team can track the effects of changes in the compute engine on a large body of code that has run without error at some previous point in time.

The **tools** package also contains functions to support literate statistical analysis, written to contain both documentation and the specification of procedures actually run to generate results (text, tabular and graphical). The `Sweave()` function runs on a file containing, in the LaTeX and R case, a marked-up mixture of LaTeX and R code, producing a file in LaTeX including verbatim code chunks, the output of the commands issued, and files with tables or graphics in suitable formats for inclusion. Leisch and Rossini (2003) argue that, with increasing dependence on settings needed to replicate results, for instance the specific random number generator and seed used, such literate analysis techniques are needed for reproducible research. This infrastructure has been extended and implemented as a less terse way of documenting packages contributed to R, in documents known as vignettes, and used extensively in the Bioconductor project (Gentleman, Rossini, Dudoit and Hornik, 2003). Vignettes also contain code that is checked on the same basis as help page examples. This paper has been written using the literate statistical analysis support available in R.

On the graphics side, R does not provide dynamic linked visualization, since the graphics model is based on drawing on one of a number of graphics devices. R does provide the more important tools for graphical data analysis, although no mapping is present as yet in general terms. The R spatial analysis website refered to above covers packages on CRAN for mapping using the legacy S format, using shapefiles, and using ArcInfo binary files, but each of these packages uses separate mechanisms and object representations at present. Work is progressing on the provision of panelled graphics in the **grid** and **lattice** packages. Graphics have in part been kept fairly simple because of cross-platform difficulties; they are extensible at the user level in many ways, but are more for viewing than for interaction.

4 Worked examples: choropleth map class intervals

Choropleth maps are among the more frequently used graphical display tools needed in applied spatial data analysis. In addition, this Association has published both landmark legacy papers, such as that by Jenks and Caspall (1971), and recent advances proposed by Armstrong, Xiao and Bennett (2003) in this area. They are of interest because of their broad application, because the media and the public see them as being easy to read, and because both the choice of class intervals and fill colours or hatchings can lead to misunderstanding by design or omission.

4.1 Getting spatial data into R

It is not the intention of these examples to attack substantive research questions, but rather to show how one might instrument such attacks using the base R language, base and contributed functions, and base graphics. The data set used is that underlying Jenks and Caspall (1971), the per acre value of gross farm products in dollars by county for Illinois in 1959. The county boundaries were downloaded from the US Census website in shapefile format for the 2000 census[1]. The data (Jenks and Caspall, 1971, p. 221) were added to the fields present in the associated DBF file, and the shapefile with added data exported in shapefile format from ArcGIS.

```
> library(maptools)
> ill59 <- read.shape("data/jenks71.shp")

Shapefile Type: Polygon   # of Shapes: 102

> ill59.df <- ill59$att.data
```

There are two contributed packages on CRAN for reading shapefiles, **shapefiles** and **maptools**; here we use **maptools**, loaded using the `library()` function. The shapes and the attribute data are read into a `"Map"` object using compiled code from shapelib[2]. The attribute data is next moved to a data frame for ease of access. From the metadata at the US Census website[3], we know that the spatial reference system of the shapefile is in geographical coordinates, using the NAD83 datum. We will need the area of the counties later on to calculate one of the measures proposed by Jenks and Caspall (1971), and visually it seems better to plot using a projection rather than the input geographical coordinates.

```
> library(SpatialCls)
> state <- as.character(ill59.df$STATE)
> county <- as.character(ill59.df$COUNTY)
> county.id <- paste(state, county, sep = "")
> ill.ll.nad83 <- "+proj=latlong +datum=NAD83"
> ill59.ll <- Map2Poly4(ill59, county.id, ill.ll.nad83)
```

[1] `http://www.census.gov/geo/cob/bdy/co/co00shp/co17_d00_shp.zip`
[2] `http://shapelib.maptools.org/`
[3] `http://www.census.gov/geo/www/cob/co_metadata.html`

The **SpatialCls** package (written by Edzer Pebesma with contributions from Barry Rowlingson and myself) is not yet released, but is under active development on SourceForge[4]. It provides object classes for spatial data, and it is hoped that other spatial packages will build on these once they stabilise. It also provides functions for transforming (including datum transformation) coordinates, based on PROJ.4 code and other files[5]. In order to specify a `"Polylist4"` object, we provide as arguments the `"Map"` object read from the shapefile, a vector of unique identifiers, and a character string defining the spatial reference system in the format used by PROJ.4 functions. These definitions are checked for validity when the `"Polylist4"` object is created.

It seems probable that Jenks and Caspall (1971) used one of the two Illinois State Plane projections in the figures of their paper (or some other Transverse Mercator projection and NAD27). The arguments for PROJ.4 transformation functions for the NAD27 / Illinois East were taken from listings distributed with the library.

```
> ill.spc.nad27 <- paste("+proj=tmerc +lat_0=36.66666666666666",
+     "+lon_0=-88.33333333333333 +k=0.999975 +x_0=152400.3048006096",
+     "+y_0=0 +ellps=clrk66 +datum=NAD27 +to_meter=0.3048006096012192",
+     "+units=us-mi")
> ill59.spc <- transform(ill59.ll, CoRS(ill.spc.nad27))
```

The `"Polylist4"` object `ill59.spc` now contains the Illinois county boundaries transformed to NAD27 Illinois East State Plane coordinates, and is shown in Figure 1. In order to extract the polygon areas, we can use the `areapl()` from the **splancs** package, but because it only accepts simple, not multiple, polygons, we first check to make sure that the counties are all meet this requirement.

```
> nParts <- unlist(lapply(ill59.spc@polygons, function(x) x@nParts))
> table(nParts)

nParts
  1
102

> library(splancs)

Spatial Point Pattern Analysis Code in S-Plus

 Version 2 - Spatial and Space-Time analysis

> area.m2 <- unlist(lapply(ill59.spc@polygons, function(x) areapl(x@coords)))
> sum(area.m2)

[1] 56357.4
```

Since no polygon has more than one part, we can proceed to apply `areapl()` to each of the polygons in the `"Polylist4"` object in turn; the sum in square miles falls between tabulated values for the state including or excluding water bodies, so the relative areas may be accepted.

```
> library(spdep)

spdep, version 0.2-12, 2004-02-26:
 a package for analysing spatial dependence,
 use help(get.spChkOption) for help on integrity checking
```

---

[4] `http://sourceforge.net/projects/r-spatial/`

[5] `http://www.remotesensing.org/proj`

```
> ill59.pl <- Map2poly(ill59)
> ill59.nb <- poly2nb(ill59.pl, row.names = county.id)
```

Because we will need a list of contiguous neighbours below, we generate it now using function `poly2nb()` from package **spdep**. The current release of **spdep** does not yet use the common spatial classes, so use is made of old-style classes, including the `"polylist"` and `"nb"` classes; relationships between new-style and old-style classes are discussed in Bivand (2003). In order to display the graph of the neighbours object `ill59.nb`, we need label points, here centroids, for the polygons.

```
> ill59.xy.ll <- data.frame(get.Pcent(ill59))
> names(ill59.xy.ll) <- c("x", "y")
> coordinates(ill59.xy.ll) <- c("x", "y")
> proj4string(ill59.xy.ll) <- CoRS("+proj=latlong +datum=NAD83")
> ill59.xy.spc <- transform(ill59.xy.ll, CoRS(ill.spc.nad27))
```

These are extracted from the original `"Map"` object, and need to be transformed to the NAD27 Illinois East State Plane spatial reference system too. We can now show the neighbour list graph displayed over the county boundaries (Figure 1).

```
> layout(matrix(c(1, 1, 2, 2), 2, 2))
> plot(ill59.spc)
> mtext("a)", 3, 1, adj = 0)
> plot(ill59.spc, border = "grey")
> plot(ill59.nb, coordinates(ill59.xy.spc), add = TRUE)
> mtext("b)", 3, 1, adj = 0)
> layout(matrix(1))
```

4.2 Criteria for class intervals

A number of different criteria have been advanced to assess the adequacy of choices of class intervals. With millions of possible alternatives for allocating the 101 unique values to 5 classes, some guidance is needed. Dent (1999, p. 148) proposes the goodness of variance fit measure, given by Armstrong, Xiao and Bennett (2003, p. 600) as:

$$GVF = 1 - \frac{\sum_{j=1}^{k} \sum_{i=1}^{N_j} (z_{ij} - \overline{z}_j)^2}{\sum_{i=1}^{N} (z_i - \overline{z})^2}$$

where the $z_i, i = 1, \ldots, N$ are the observed values, k is the number of classes, $\overline{z}_j$ the class mean for class j, and N_j the number of counties in class j. This can be coded as:

```
> gvf <- function(var, cols) {
+     sumsq <- function(x) sum((x - mean(x))^2)
+     sdam <- sumsq(var)
+     sdcm <- sum(tapply(var, factor(cols), sumsq))
+     res <- 1 - (sdcm/sdam)
+     res
+ }
```

This is a modification of the tabular accuracy index described but not formalised by Jenks and Caspall (1971), and specified by Armstrong, Xiao and Bennett (2003, p. 600) as:

$$TAI = 1 - \frac{\sum_{j=1}^{k} \sum_{i=1}^{N_j} |z_{ij} - \overline{z}_j|}{\sum_{i=1}^{N} |z_i - \overline{z}|}$$

which may be coded as:

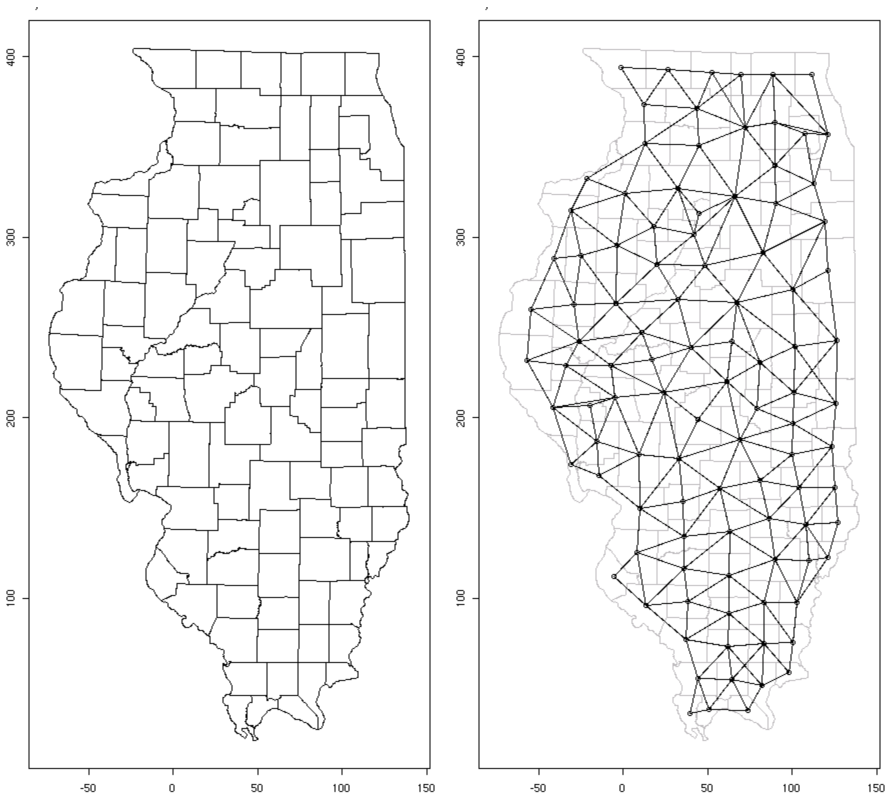

Figure 1: County boundaries (a) and polygon centroids linked by contiguous neighbour links (b), Illinois (Illinois East State Plane).

```
> tai <- function(var, cols) {
+     sumabs <- function(x) sum(abs(x - mean(x)))
+     x <- sumabs(var)
+     y <- sum(tapply(var, factor(cols), sumabs))
+     res <- 1 - (y/x)
+     res
+ }
```

This measure is extended to the overview accuracy index by Jenks and Caspall (1971, p. 237) by including the relative areas of the polygons (see also Armstrong, Xiao and Bennett, 2003, p. 600):

$$OAI = 1 - \frac{\sum_{j=1}^{k} \sum_{i=1}^{N_j} |z_{ij} - \bar{z}_j| a_{ij}}{\sum_{i=1}^{N} |z_i - \bar{z}| a_i}$$

where $a_i, i = 1, \ldots, N$ are the polygon areas, and as above the a_{ij} term is indexed over $j = 1, \ldots, k$ classes, and $i = 1, \ldots, N_j$ polygons in class j; it may be coded by passing rows of an array formed by binding together vectors of variable values and areas to a local function as:

```
> oai <- function(var, cols, area) {
+     sumabs1 <- function(x) sum(abs(x[, 1] - mean(x[,
+         1])) * x[, 2])
+     m <- cbind(as.numeric(var), as.numeric(area))
+     x <- sumabs1(m)
+     y <- sum(by(m, factor(cols), sumabs1))
+     res <- 1 - (y/x)
+     res
+ }
```

Armstrong, Xiao and Bennett (2003) provide a definition of the Jenks and Caspall (1971) boundary accuracy index. This is not replicated here, not are their *GEA* (geographic quantile) or *MIC* (Moran's cluster coefficient — a modified Moran's *I*). In order to accommodate at least some of the variation implied by the spatial configuration of the classes, use is made of the standardised value of the $Jtot$ measure, comparing the count of all county boundaries between counties belonging to different classes with the count that could have need expected with a random distribution of colours for the observed class sizes under non-free sampling (Cliff and Ord, 1981, pp. 13, 20). The `joincount.multi()` function in **spdep** is implemented based on Cliff and Ord (1981) and Upton and Fingleton (1985, pp. 164–170); the $Jtot$ measure is taken from (Cliff and Ord, 1981, p. 13) as:

$$Jtot = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} y_{ij}$$

where the w_{ij} are general weights matrix elements reflecting the spatial relationship between i and j (here binary: 1 if neighbours, 0 otherwise), and y_{ij} takes value 1 if i and j belong to different classes, 0 otherwise. The z-value of $Jtot$ is calculated by subtracting its expected value, and dividing by the square root of its variance.

These four measures are collected to provide a numerical summary of class interval performance for display. For convenience, each map display is pre-packaged as the map itself, a legend panel with class-intervals and class counts, the values of the summary measures, and an empirical cumulative distribution function with the class intervals marked and furnished with rectangles coloured to match the map itself — the function `illplot()` is not displayed here.

```
> jenks.tests <- function(var, cols, area, nb) {
+     jc <- joincount.multi(as.factor(cols), nb2listw(nb,
+         style = "B"))
+     res <- c(gvf(var, cols), tai(var, cols), oai(var,
+         cols, area), jc[nrow(jc), ncol(jc)])
+     res
+ }
```

4.3 Simple class intervals

Just to make sure that the data are unlikely to constitute a mixture of distributions, perhaps driven by an unobserved underlying driver, let us examine Figure 2. It seems reasonable to accept that the data do come from one distribution, even though the range is quite large, with the lowest and highest values almost an order of magnitude apart. This is related to differences in the composition of the farm products, about which we have no further information.

```
> summary(ill59.df$JENKS71)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  15.57   37.73   53.33   56.28   68.24  155.30

> hist(ill59.df$JENKS71, freq = FALSE, xlab = "", main = "")
> lines(density(ill59.df$JENKS71))
> rug(ill59.df$JENKS71)
```

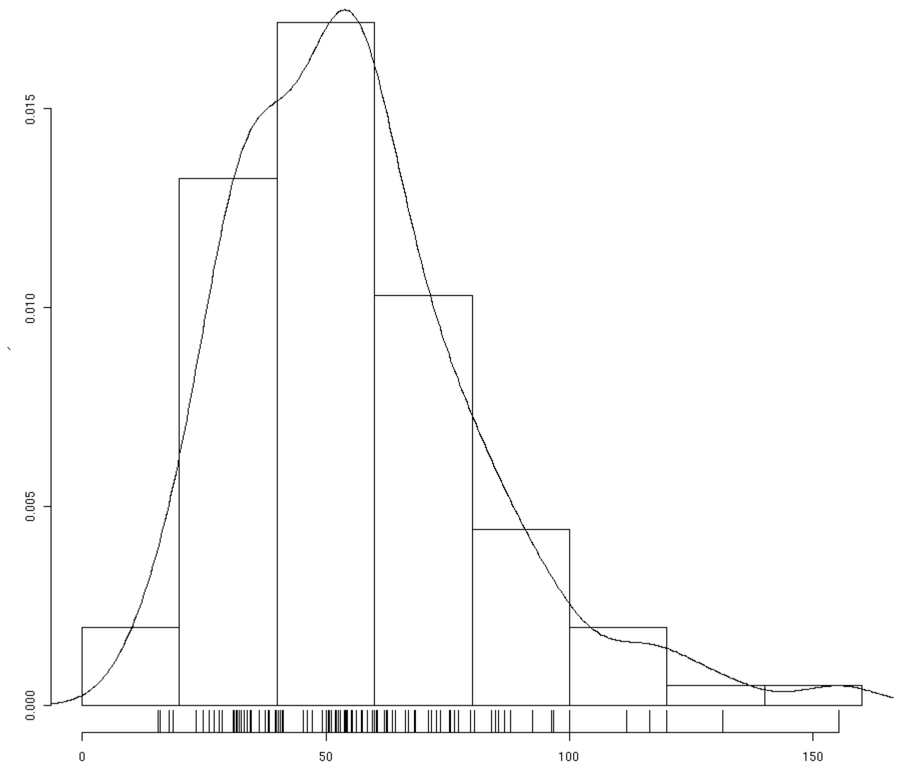

Figure 2: Per acre value of gross farm products in dollars by county for Illinois in 1959 — histogram and density plot.

Starting at the very beginning, we can replicate the original Jenks and Caspall (1971, Fig. 2, p. 216) map with equal class intervals in post of the interior of the data range, giving a lower narrow interval and an upper broad one. Assigning the breaks manually, we can make an ordered factor, which can next be examinied using `joincount.multi()`, which requires a factor argument, and will then use the factor levels to construct row labels.

```
> fig2.brks <- c(15.57, 25, 50, 75, 100, 155.3)
> j71.f <- as.ordered(cut(ill59.df$JENKS71, breaks = fig2.brks,
+     include.lowest = TRUE))
> joincount.multi(j71.f, nb2listw(ill59.nb, style = "B"))

                     Joincount  Expected  Variance  z-value
[15.6,25]:[15.6,25]    6.00000   0.77752   0.70289   6.2292
(25,50]:(25,50]       58.00000  29.07921  17.72416   6.8695
(50,75]:(50,75]       81.00000  44.62959  23.65512   7.4780
(75,100]:(75,100]     20.00000   4.71695   3.85509   7.7838
(100,155]:(100,155]    4.00000   0.77752   0.70289   3.8437
(25,50]:[15.6,25]     16.00000  10.57426   7.85557   1.9358
(50,75]:[15.6,25]      0.00000  13.06232   9.14169  -4.3202
(50,75]:(25,50]       40.00000  74.01980  43.06812  -5.1839
(75,100]:[15.6,25]     0.00000   4.35411   3.70320  -2.2626
(75,100]:(25,50]       0.00000  24.67327  17.44665  -5.9071
(75,100]:(50,75]      17.00000  30.47874  20.30299  -2.9914
(100,155]:[15.6,25]    0.00000   1.86605   1.66741  -1.4451
(100,155]:(25,50]      0.00000  10.57426   7.85557  -3.7728
(100,155]:(50,75]      8.00000  13.06232   9.14169  -1.6743
(100,155]:(75,100]    17.00000   4.35411   3.70320   6.5715
Jtot                  98.00000 187.01922  48.84968 -12.7366
```

For mapping, it will be more convenient to use `findInterval()` to assign data values to classes by breaks:

```
> cols <- findInterval(ill59.df$JENKS71, fig2.brks, all.inside = TRUE)
```

In the Illinois case, many of the counties are similar in area, with the result that the *TAI* and *OAI* values are typically close to each other when the intra-class area distributions match the total distribution, here for the class intervals of Figure 3:

```
> summary(area.m2)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  172.2   389.1   514.6   552.5   693.3  1186.0

> by(area.m2, factor(cols), summary)

INDICES: 1
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  181.8   268.8   361.7   339.8   426.9   446.9
---------------------------------------------------
INDICES: 2
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  202.9   364.8   444.2   466.6   572.9   872.5
---------------------------------------------------
INDICES: 3
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  172.2   432.9   604.9   627.3   796.9  1186.0
---------------------------------------------------
INDICES: 4
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  282.0   455.0   567.2   591.9   721.3  1148.0
---------------------------------------------------
INDICES: 5
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  336.6   529.0   589.3   637.0   778.2   957.1
```

Fortunately, a contributed R package, **RColorBrewer** gives by permission access to the ColorBrewer palettes described by Harrower and Brewer (2003) and otherwise accesible from the ColorBrewer website[6]. Figures in this paper are being kept to grey tones, so we choose the sequential palette `"Greys"`, with one less colour than the number of breaks. The breaks are also stacked for legend output.

```
> library(RColorBrewer)
> ncl <- length(fig2.brks) - 1
> pal <- brewer.pal(ncl, "Greys")
> stbrks <- stack.brks(fig2.brks)
```

```
> illplot(ill59.df$JENKS71, ill59.spc, cols, stbrks, pal,
+     area.m2, ill59.nb)
```

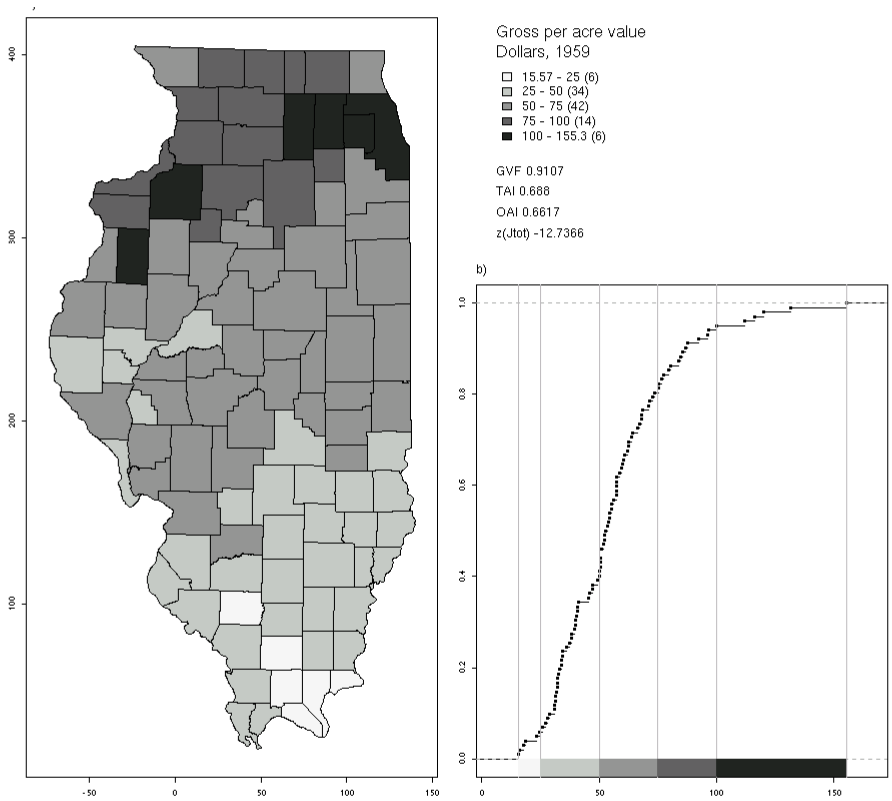

Figure 3: Jenks and Caspall (1971, Fig. 2, p. 216): a) choropleth map with approximately equal class intervals; b) empirical cumulative density plot with class intervals.

We can now examine the output on Figure 3. The left pane contains the choropleth map, the lower right pane a plot of the empirical cumulative distribution function with the class intervals marked. The legend pane contains a title, a legend with class intervals and class counts, and the values of the four summary measures.

The next display replicates Jenks and Caspall (1971, Fig. 5, p. 222), with almost equal class counts, using the `quantile()` function to find breaks for five classes. As we can see from Figure 4, the visual impact of the counties at the upper end of the distribution of the data values is strongly enhanced, and the *GVF* measure has a much lower value than in Figure 3.

[6] `http://colorbrewer.org`

```
> quant5.brks <- quantile(ill59.df$JENKS71, seq(0, 1, 1/5))
> cols <- findInterval(ill59.df$JENKS71, quant5.brks, all.inside = TRUE)
> ncl <- length(quant5.brks) - 1
> pal <- brewer.pal(ncl, "Greys")
> stbrks <- stack.brks(quant5.brks)

> illplot(ill59.df$JENKS71, ill59.spc, cols, stbrks, pal,
+     area.m2, ill59.nb)
```

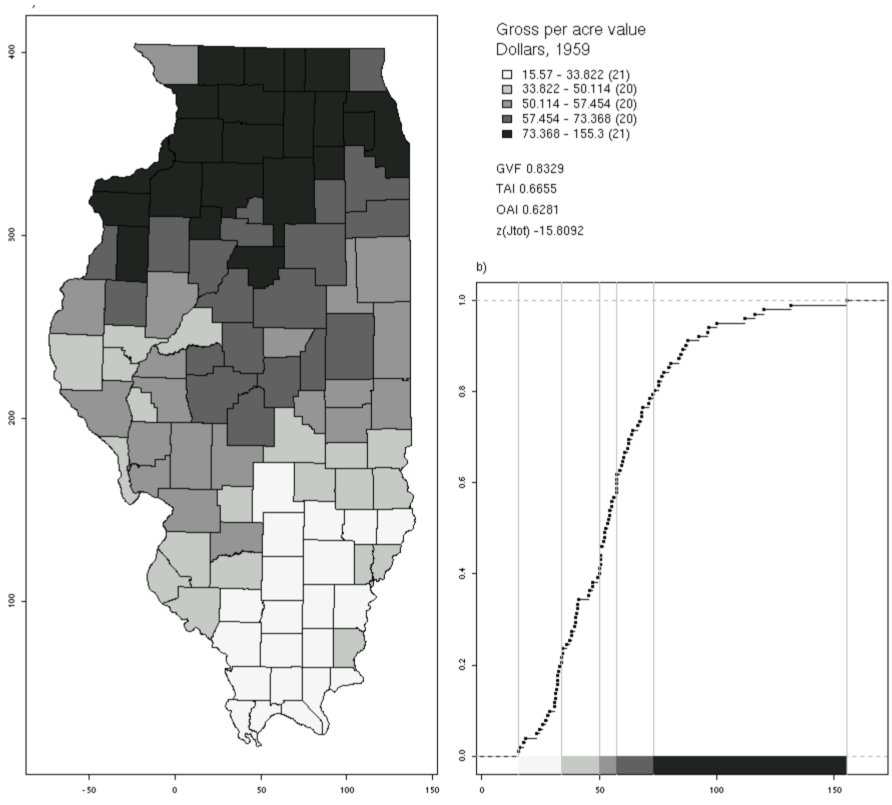

Figure 4: Jenks and Caspall (1971, Fig. 5, p. 222): a) choropleth map with approximately equal numbers of counties in each class; b) empirical cumulative density plot with class intervals.

4.4 Classification methods and class intervals

These two simple methods do not attempt to use information about the relative placings of the observed values to establish class intervals, emulating the manual natural breaks method. There are many clustering and classification techniques that may be used for this purpose; among these is K-means clustering, in R the `kmeans()` function. As described by Hartigan and Wong (1979) and reviewed in the context of other clustering and classification methods by Gentle (2002, pp. 237–255), the method attempts to minimise the variation within each class. It by default starts from a given number of randomly chosen centres from within the observed data values, and does not try to answer the question of the choice of number of classes. Since it uses the random number generator to start its search, we set the seed value before finding the breaks for five classes. It may be remarked that `kmeans()`

will only find a local minimum, and that its search is perhaps made more difficult when only one dimension is present, since the algorithms are written for data on a small number of dimensions, but not usually a single dimension.

```
> set.seed(20040316)
> km5.brks <- kmeans(ill59.df$JENKS71, 5)
> cols <- match(km5.brks$cluster, order(km5.brks$centers))
> ncl <- nrow(km5.brks$centers)
> pal <- brewer.pal(ncl, "Greys")
> stbrks <- matrix(unlist(tapply(ill59.df$JENKS71, factor(cols),
+     range)), ncol = 2, byrow = TRUE)

> illplot(ill59.df$JENKS71, ill59.spc, cols, stbrks, pal,
+     area.m2, ill59.nb)
```

As can be seen from Figure 5, this `kmeans()` choice of class intervals is quite similar to Figure 3. The classes do not necessarily abutt each other, and the gaps in some measure correspond to natural breaks. The *GVF* measure has a markedly higher value than in the case of quantile breaks, and slightly higher than in the case of equal class intervals.

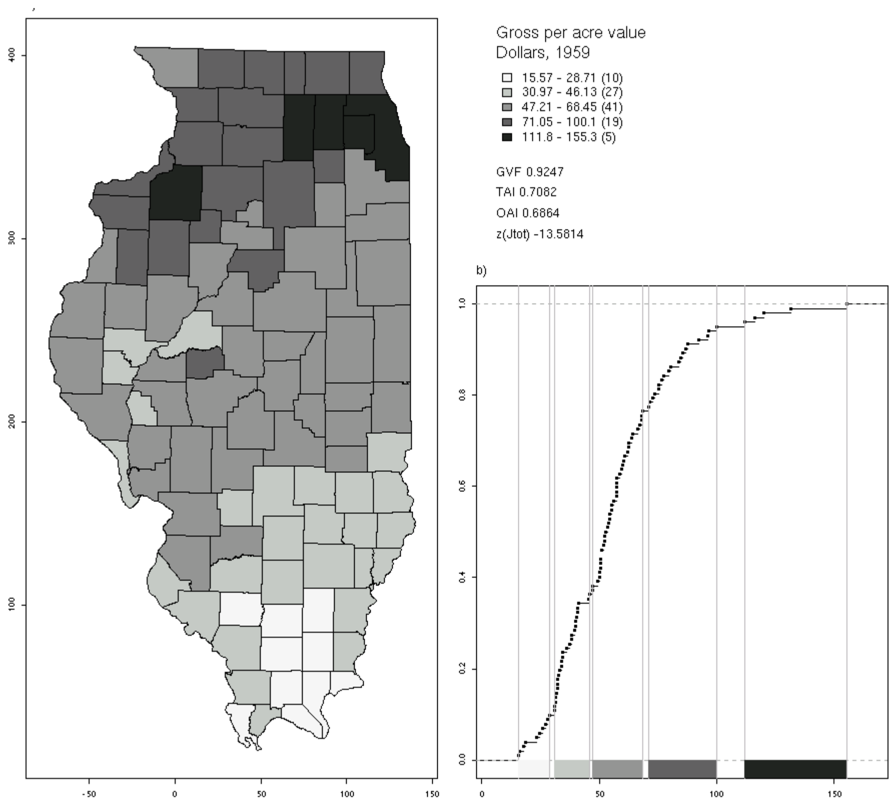

Figure 5: a) choropleth map constructed using `kmeans()` for five classes; b) empirical cumulative density plot with class intervals.

In order to examine more closely the impact of local minima and dependence on initial centre values, it may be of interest to see what happens when combined machine learning methods are applied. The **e1071** package includes a range of such techniques, including bagged clustering (Leisch, 1999), where bagged means bootstrap aggregated. This is implemented as the `bclust()` function, used here in a local version to permit the analysis of single-column problems. In `bclust()`, a partitioning cluster algorithm such as `kmeans()` is run repeatedly on bootstrap samples from the original data. The resulting cluster centers are then combined using the hierarchical cluster algorithm `hclust()`, using by default the `"average"` method. This means that we not only repeat `kmeans()` a number of times on the data in different order, but also use hierarchical clustering to choose which of the centres found is reported. Again, before starting, we set the seed:

```
> set.seed(20040316)
> library(e1071)
> source("data/bclust.R")
> bc5.brks <- bclust(ill59.df$JENKS71, centers = 5, verbose = FALSE)

Loading required package: mva
Loading required package: class

> cols <- match(bc5.brks$cluster, order(bc5.brks$centers))
> ncl <- nrow(bc5.brks$centers)
> pal <- brewer.pal(ncl, "Greys")
> stbrks <- matrix(unlist(tapply(ill59.df$JENKS71, factor(cols),
+     range)), ncol = 2, byrow = TRUE)

> illplot(ill59.df$JENKS71, ill59.spc, cols, stbrks, pal,
+     area.m2, ill59.nb)
```

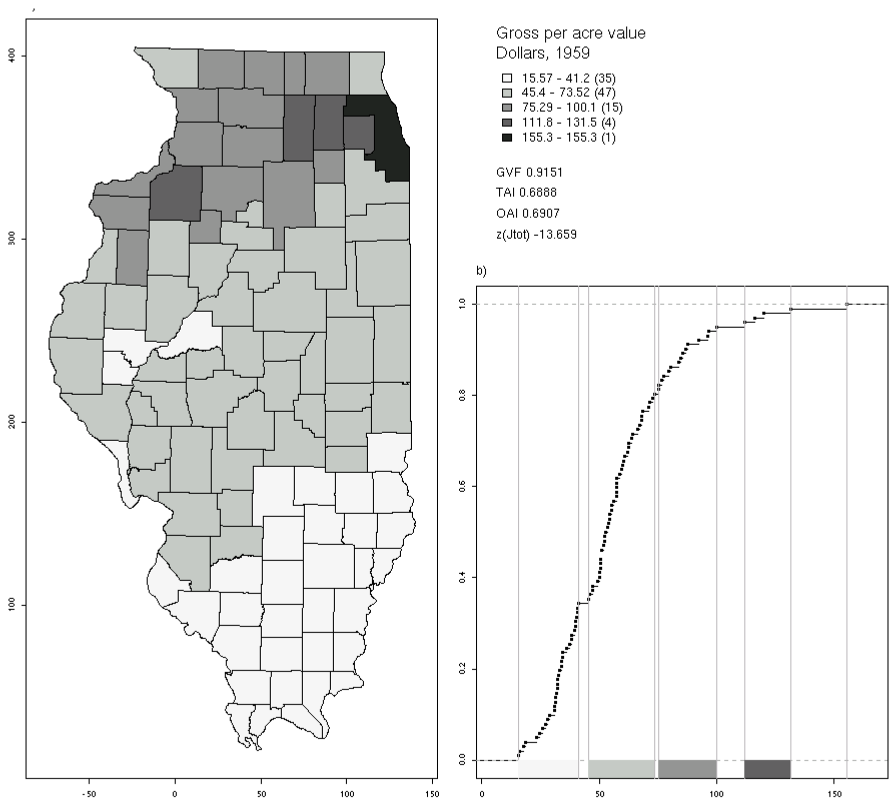

Figure 6: a) choropleth map constructed using bagged clustering for five classes; b) empirical cumulative density plot with class intervals.

While it is obvious that the use of genetic algorithms and a better choice of measures by Armstrong, Xiao and Bennett (2003) is a better and probably more robust approach, the output displayed on Figure 6 does suggest that raiding the machine learning armoury does seem to yield sensible results, looking much more like a natural breaks 5-class partition of the data set than those we have seen so far. The measures are not the strongest seen on any criterion, but the overall impression is subjectively pleasing. Bagged clustering results provide visually intuitive solutions, although they do point up the question of whether choosing five classes is a good solution, so for a single dimensional class interval explorer it may be necessary to jump between discrete numbers of classes as well as optimising clustering criteria.

To go beyond this, an attempt is made to explore other sets of breaks that can be constructed using `bclust()` by repeating its application a number of times and varying the number of classes. The numbers of classes (between 4 and 11) are permuted to add an extra level of randomisation. The measures values are stored in `res.score` in order to be able to choose solutions for display.

```
> set.seed(20040316)
> ks <- sample(c(rep(4:11, 25)))
> n <- length(ks)
> res <- vector(length = n, mode = "list")
> res.score <- matrix(NA, ncol = 5, nrow = n)
> colnames(res.score) <- c("k", "gvf", "tai", "oai", "jc")
> for (i in 1:n) {
+     bc <- bclust(ill59.df$JENKS71, centers = ks[i], verbose = FALSE)
+     res[[i]] <- bc
+     cols <- match(bc$cluster, order(bc$centers))
+     res.score[i, ] <- c(ks[i], jenks.tests(ill59.df$JENKS71,
+         cols, area.m2, ill59.nb))
+ }
> row.names(res.score) <- 1:n
> table(res.score[, 1])

 4  5  6  7  8  9 10 11
25 25 25 25 25 25 25 25

> res.score1 <- unique(res.score, MARGIN = 1)
> table(res.score1[, 1])

 4  5  6  7  8  9 10 11
14 16 15 18 19 25 23 24
```

While we calculated 200 solutions, it turns out that only 154 are unique, so that `bclust()` converged to the same values from different start positions on some occasions for given numbers of classes.

```
> jc.range <- tapply(res.score1[, 5], res.score1[, 1],
+     function(x) diff(range(x)))
> jc.range

       4        5        6        7        8        9       10 
7.771441 3.255746 1.351672 1.296399 1.151781 2.162290 1.860806 
      11 
2.277235 

> k.choice <- as.numeric(names(jc.range)[which.min(jc.range)])
> res.best <- res.score1[res.score1[, 1] == k.choice, ]
> summary(res.best[, 2:5])

      gvf              tai              oai               jc        
 Min.   :0.9467   Min.   :0.7510   Min.   :0.7420   Min.   :-13.63  
 1st Qu.:0.9566   1st Qu.:0.7760   1st Qu.:0.7699   1st Qu.:-13.18  
 Median :0.9643   Median :0.7864   Median :0.7814   Median :-12.88  
 Mean   :0.9618   Mean   :0.7843   Mean   :0.7770   Mean   :-12.96  
 3rd Qu.:0.9665   3rd Qu.:0.7911   3rd Qu.:0.7854   3rd Qu.:-12.74  
 Max.   :0.9726   Max.   :0.8138   Max.   :0.8115   Max.   :-12.48  
```

```
> best <- as.integer(row.names(res.best)[unique(apply(res.best[,
+     c(2, 4)], 2, which.max))])
> res.score[best, ]

          k         gvf         tai         oai          jc
  8.0000000   0.9726436   0.8138454   0.8115418 -13.0833802
```

The range of the $z(Jtot)$ values measured for the class intervals will be used as an ad-hoc criterion for choosing the number of classes. It appears that the range of the $z(Jtot)$ values (`jc.range`) at first falls sharply, and then begins to rise again. In this case, the number of class intervals with the smallest range in $z(Jtot)$ values is 8. The distributions of the measures in the summary for this number of classes are quite restricted. Figure 7 shows the map obtained by choosing the largest GVF value from among the unique values obtained for 8 classes.

```
> bc.best1 <- res[[best]]
> cols <- match(bc.best1$cluster, order(bc.best1$centers))
> ncl <- nrow(bc.best1$centers)
> pal <- brewer.pal(ncl, "Greys")
> stbrks <- matrix(unlist(tapply(ill59.df$JENKS71, factor(cols),
+     range)), ncol = 2, byrow = TRUE)

> illplot(ill59.df$JENKS71, ill59.spc, cols, stbrks, pal,
+     area.m2, ill59.nb, cex = 1)
```

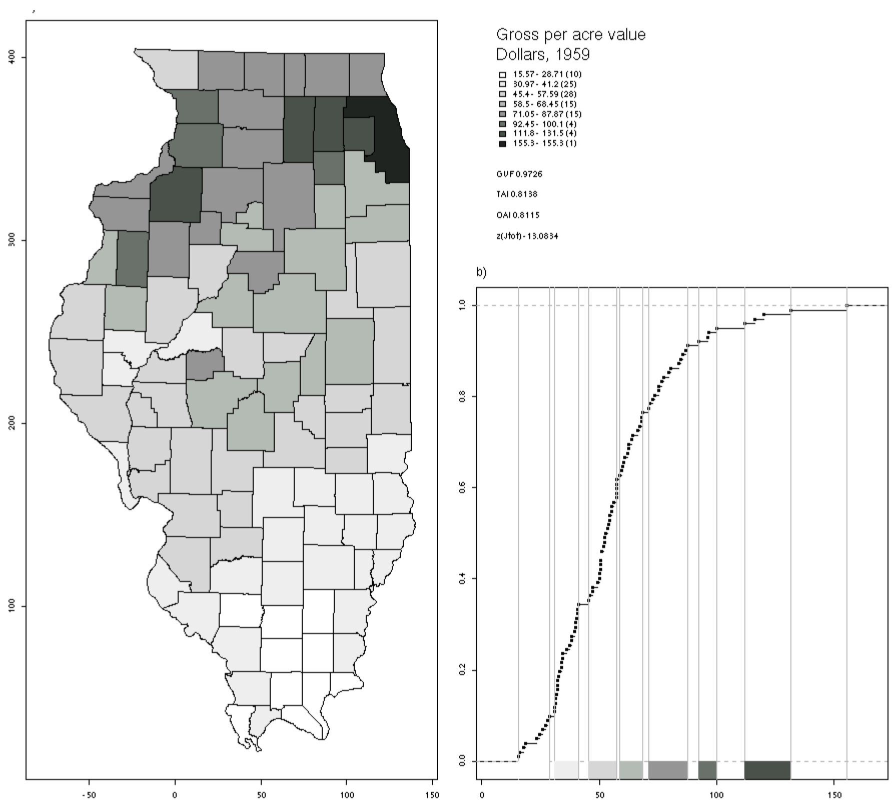

Figure 7: a) choropleth map constructed using bagged clustering for 8 classes — best on GCV, TAI and OAI; b) empirical cumulative density plot with class intervals.

The hierarchical clustering dendrogram from a single bagged clustering, used to combine the output of bootstrap aggregation, can also be examined by plotting the object returned by `bclust()` to try to establish an acceptable number of classes. This will not be examined here, but could arguably provide an interesting alternative to the naive, brute-force approach used above. Single runs of `bclust()` are not very costly in time, so that a `"bclust"` object might yield choices of bagged clustering class intervals for different numbers of classes, using access functions `clusters.bclust()` and `centers.bclust()`, setting the `k=` argument to the number of classes required. Here we choose the number with the smallest *Jtot* range found above.

```
> cols <- match(clusters.bclust(bc5.brks, k = k.choice),
+     order(centers.bclust(bc5.brks, k = k.choice)))
> ncl <- length(centers.bclust(bc5.brks, k = k.choice))
> pal <- brewer.pal(ncl, "Greys")
> stbrks <- matrix(unlist(tapply(ill59.df$JENKS71, factor(cols),
+     range)), ncol = 2, byrow = TRUE)

> illplot(ill59.df$JENKS71, ill59.spc, cols, stbrks, pal,
+     area.m2, ill59.nb, cex = 1)
```

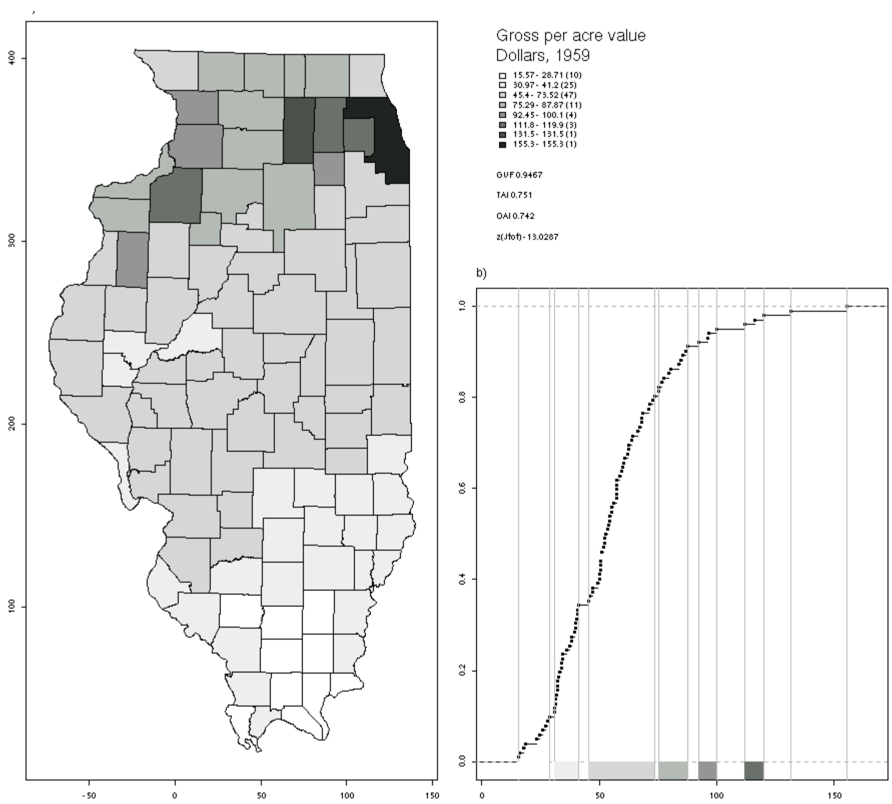

Figure 8: a) choropleth map constructed using bagged clustering for 8 classes retrieved from five class bagged clustering output; b) empirical cumulative density plot with class intervals.

Figure 8 shows the 8 class cut of the bagged clustering output originally generated for five centres. Quite a lot of searching across the spaces of different class intervals for different numbers of classes is undertaken in bagged clustering — given that the hierarchical clustering is being conducted on class centres from bootstrapped applications of `kmeans()`. By choosing different input settings for `bclust()` arguments, and using the output plot, it should be possible to provide ways of exploring a range of "natural breaks" intervals from a single `bclust()` run for a single and possible arbitrary number of intervals.

We could continue to prototype a class interval "wizard", but for now this will suffice to demonstrate how data analysis can proceed in R. It is quite typical that, by design or chance, it happens that existing code from other contributed packages, or from R core, suggests solutions that can be tried out, and very often modified or rejected, but at least provide comparisons with those derived from narrower disciplinary traditions. The combination of both modern applied statistics, often computational in form, and their explication in open code, turns out to be both stimulating and fruitful, and also challenges analysts of spatial data to benefit from progress in relevant neighbouring fields of study.

5 Opportunities for advancing spatial data analysis in R

There seem to be four main opportunities for advancing spatial data analysis in R. The first is the vitality and rapid development of R itself, now available on all major platforms, and supporting the release and distribution of contributed software packages through the archive network. This is assisted by the open source development model adopted by the R project, which certainly reduces cost of acquisition, and permits local customisation where needed. This model necessarily removes barriers between users and developers, consumers and producers, although it does not eliminate it. It is also noticable that take-up in underfunded teaching and research settings, such as developing countries, is occurring rapidly.

The second is the reproducible research thread present in communities like those using Matlab, S-PLUS, or R: certainly statisticians are used to having to reveal their journalled procedures. This applies in particular in biostatistics, where the consequences of the application of erroneous or inappropriate procedures can be serious. But it also extends to other areas of analysis where peer review and insight is sought and valued, including spatial data analysis, at least partly through joint work with epidemiologists.

The third opportunity is that it seems that relatively more applied statisticians are seeing spatial data analysis with more sympathy and interest, although they may be frustrated both by data formats (spatial position is much less systematically recorded than time) and by internal debates within quantitative geography or geostatistics. This opportunity means that if object structures needed to analyse spatial data can be provided and documented in a language environment that applied statisticians like and understand, then they can join in the development and implementation of methods of analysis.

The fourth opportunity stems from the spatial data analysis community itself, which is capable of working together and exchanging ideas. From a period in which geographical information systems, and later geocomputation and geographical information science have been the agenda-setters, there seems to be interest in trying things out, in expressing ideas in code, and in encouraging others to apply the coded functions in teaching and applied research settings. Having good ideas is no longer quite enough, their quality becomes apparent when they achieve adoption in applied fields, especially outside their fields of origin.

Acknowledgements

This work was funded in part by EU Contract Number Q5RS 2000 30183.

References

Armstrong, M. P., Xiao, N., Bennett, D. A., 2003. Using genetic algorithms to create multicriteria class intervals for choropleth maps. Annals, Association of American Geographers, 93 (3), 595–623.

Becker, R. A., Chambers, J. M., Wilks, A. R., 1988. The New S Language. Chapman & Hall, London, 702 pp.

Bivand, R. S., 2000. Using the R statistical data analysis language on GRASS 5.0 GIS data base files. Computers and Geosciences, 26 (9/10), 1043–1052.

Bivand, R. S., 2001. More on spatial data. R News, 1 (3), 13–17.

Bivand, R. S., 2002. Spatial econometrics functions in R: Classes and methods. Journal of Geographical Systems, 4 (4), 405–421.

Bivand, R. S., 2003. Approaches to Classes for Spatial Data in R. In: Hornik, K., Leisch, F., Zeilis, A. (Eds), Proceedings of the 3rd International Workshop on Distributed Statistical Computing, Vienna, Austria, 10 pp., (`http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/Bivand.pdf`).

Bivand, R. S., Gebhardt, A., 2000 Implementing functions for spatial statistical analysis using the R language. Journal of Geographical Systems, 2 (3), 307–317.

Chambers, J. M., 1998. Programming with Data. Springer, New York, 469 pp.

Chambers, J. M., Hastie, T. J., 1992. Statistical Models in S. Chapman & Hall, London, 608 pp.

Cliff, A. D., Ord, J. K., 1981. Spatial processes. Pion, London, 266 pp.

Dalgaard, P., 2002. Introductory Statistics with R. Springer, New York, 288 pp.

Dent, B. D., 1999, Cartography: thematic map design. McGraw-Hill, Boston, 417 pp.

Gentle, J. E., 2002. Elements of computational statistics. Springer, New York, 420 pp.

Gentleman, R., Rossini, A., Dudoit, S., Hornik, K., 2003. The Bioconductor FAQ. `http://www.bioconductor.org/`

Grunsky, E. C., 2002. R: a data analysis and statistical programming environment — an emerging tool for the geosciences. Computers & Geosciences, 28 (2), 1219–1222.

Harrower M. A., Brewer C. A., 2003. ColorBrewer.org: An Online Tool for Selecting Color Schemes for Maps. The Cartographic Journal, 40 (1), 27–37.

Hartigan, J. A., Wong, M. A., 1979. A K-means clustering algorithm. Applied Statistics, 28 (1), 100–108.

Ihaka, R., Gentleman, R., 1996. R: A Language for Data Analysis and Graphics. Journal of Computational and Graphical Statistics, 5 (3), 299–314.

Jenks, G. F., Caspall, F. C., 1971. Error on choroplethic maps: definition, measurement, reduction. Annals, Association of American Geographers, 61 (2), 217–244.

Kaluzny, S. P., Vega, S. C., Cardoso, T. P., Shelly, A. A., 1996. S-PLUS SPATIALSTATS user's manual version 1.0. Seattle: MathSoft Inc., 226 pp.

Leisch, F., 1999 Bagged clustering. Working Paper 51, SFB, Adaptive Information Systems and Modeling in Economics and Management Science, http://www.ci.tuwien.ac.at/~leisch/papers/wp51.ps

Leisch, F., Rossini, A., 2003. Reproducible statistical research. Chance, 16 (2), 46–50

R Development Core Team R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. http://www.R-project.org

Ripley, B. D., 2001. Spatial Statistics in R. R News, 1 (2), 14–15.

Upton, G. J. G., Fingleton, B., 1985. Spatial data analysis by example: point pattern and quatitative data. Wiley, Chichester, 410 pp.

Venables, W. N., Ripley, B. D., 2000. S Programming. Springer, New York, 264 pp.

Venables, W. N., Ripley, B. D., 2002. Modern Applied Statistics with S, 4th edn. Springer, New York, 495 pp.