Progress in the R ecosystem for open source spatial analysis software

Roger Bivand

23 May 2019

Outline

- Introduction and background: why break stuff (why not)?
- Data input/output and representation: movement from legacy **sp** to standards-compliant **sf** representation; coordinate reference systems; developments similar to `GeoPandas` and `Shapely` in Python
- Spatial weights and measures of autocorrelation: software packages previously using the **sp** representation may add **sf** representation or replace **sp** with **sf**; **spdep** can use both for constructing spatial weights
- Spatial regression: model estimation and handling split out from **spdep** to **spatialreg**; fewer reverse dependencies, quicker changes

Introduction

The R ecosystem for open source spatial analysis software

- The R statistical programming language and environment has been used for analysing spatial data since its inception, partly building on its heritage from S and S-Plus.
- When the conceptualisation of spatial data was introduced in the **sp** package, it was expected that some packages would adopt its classes.
- Some years later, adoption rates had picked up strongly, as had use of the **sp**-based packages **rgdal** for input/output and **rgeos** for geometric manipulation of vector data.
- The packages depending on **sp** classes continue to require support as more adequate data representations are introduced in the **sf** and **stars** packages.

The `sf` package

- The **sf** package proides the input/output and geometry manipulation functionalities found in **rgdal** and **rgeos**, and an alternative class representation for vector data based on the Simple Features standard (Pebesma, Mailund, and Hiebert 2016; Pebesma 2018; Ucar, Pebesma, and Azcorra 2018)
- The **stars** package adds some facilities for handling spatio-temporal raster and vector data, building in part on work with the **spacetime** package.
- Finally, Pebesma and Bivand (2020) present a fuller picture of ongoing changes in the R spatial analysis ecosystem, going into more detail because Lovelace et al. (2019) provide an excellent introduction

Upstream geospatial software

- In addition there are many changes taking place in upstream geospatial software that will impact typical workflows.
- For vector data, it is very likely that the legacy shapefile file format will be replaced, probably by the geopackage format.
- One reason for this is that GPKG is designed to support multibyte string encoding.
- In addition, the representations of spatial/coordinate reference systems are being modernised, leading to breaking changes in most software.

Visualization packages

- Newer visualization packages, such as **tmap**, **mapview** and **cartography**, give broader scope for data exploration and communication.
- A thorough overview of modelling and analysis packages shows the considerable range of approaches now available in contributed packages and other R code present in supplementary material to published papers.
- This provides a helpful mechanism supporting reproducible research and hands-on reviewing in which readers can read the code and scripts used in calculating the results presented in published work.

Package dependencies

- Because contributed packages form an ecosystem, some packages are used by others in turn in dependency trees.
- Class representations of data are central, with the data frame conceptualisation shaping much of the whole R ecosystem.
- For the modelling infrastructure to perform correctly, the relationships between objects containing data and formula interfaces constructing model responses and matrices are crucial.
- Because both **sp** and **sf** provide similar interfaces, transition from **sp** to **sf** representations is convenient by design.

Splitting the spdep package

- The **spdep** package has been split into **spdep** for constructing spatial weights and exploratory spatial data analysis, and the new package **spatialreg** for spatial regression.
- At present **spatialreg** functions can also be accessed from the pre-split **spdep** code, but this deprecation period will be short; this reorganisation is similar in form to that taking place in PySAL.
- The **spdep** and **spatialreg** packages have been adapted to accommodate data held in **sf** classes in addition to **sp** classes, so both approaches are viable.
- In order to retain backward compatibility, other central packages may choose to handle the coexistence of **sp** and **sf** classes in the same way.

Splitting the `spdep` package

- The first step taken was to add **sf** interfaces to existing **spdep** functions, mostly for creating neighbour objects (completed version 1.0-2 13 February)
- The package split was announced as a github issue on `r-spatial/spdep` and on Twitter on 11 March
- With insight from Iñaki Ucar and Gábor Csárdi responding to my 9 March post on the `R-pkg-devel` mailing list, the reverse dependency functions were found using `pkgapi::map_package()`
- The deprecated split packages (**spdep** 1.1-2 and **spatialreg** 1.1-3) were published 1 & 5 April (see this post); the **spdep** model functions will be made defunct shortly

Comparative studies

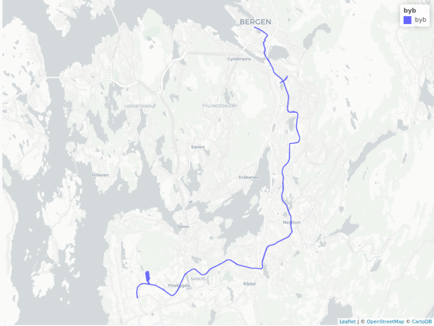
- The performance of the spdep package implementations of global and local measures of spatial autocorrelation have been compared with other implementations by Bivand and Wong (2018).
- The spatial regression model fitting functions now in the spatialreg and sphet packages were compared with other implementations by Bivand and Piras (2015).
- Methods for computing the log determinant in spatial regression using maximum likelihood and Bayesian estimation methods were surveyed in Bivand et al. (2013).
- Bivand et al. (2017) survey spatial multilevel model estimation approaches.

Data input/output and representation

Spatial data

Spatial data typically combine position data in 2D (or 3D), attribute data and metadata related to the position data. Much spatial data could be called map data or GIS data. We collect and handle much more position data since global navigation satellite systems (GNSS) like GPS came on stream 20 years ago, earth observation satellites have been providing data for longer. (Geocomputation with R may be useful, as may SDSR).

```
> suppressPackageStartupMessages(library(osmdata))
> library(sf)
## Linking to GEOS 3.7.2, GDAL 3.0.0, PROJ 6.1.0
> bbox <- opq(bbox = 'bergen norway')
> byb0 <- osmdata_sf(add_osm_feature(bbox, key = 'railway',
+   value = 'light_rail'))$osm_lines
> tram <- osmdata_sf(add_osm_feature(bbox, key = 'railway',
+   value = 'tram'))$osm_lines
> byb1 <- tram[!is.na(tram$name),]
> o <- intersect(names(byb0), names(byb1))
> byb <- rbind(byb0[,o], byb1[,o])
> library(mapview)
> mapview(byb)
```

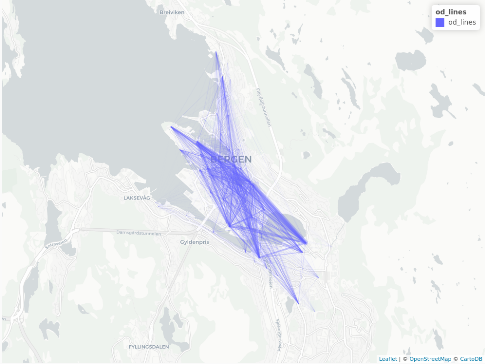


Data handling

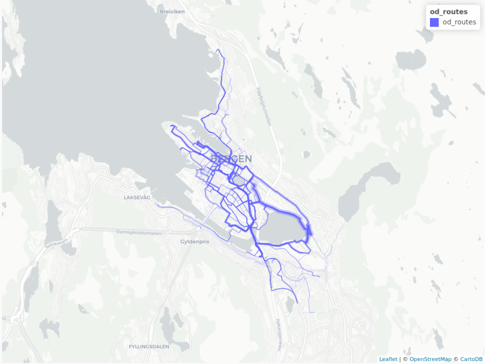
We can download monthly CSV files of city bike use, and manipulate the input to let us use the **stplanr** package to aggregate origin-destination data. One destination is in Oslo, some are round trips, but otherwise things are OK. We can use CycleStreets to route the volumes onto OSM cycle paths, via an API and API key. We'd still need to aggregate the bike traffic by cycle path segment for completeness.

```r
> bike_fls <- list.files("../bbs")
> trips0 <- NULL
> for (fl in bike_fls) trips0 <- rbind(trips0,
+   read.csv(file.path("../bbs", fl), header=TRUE))
> trips0 <- trips0[trips0[, 8] < 6 & trips0[, 13] < 6,]
> trips <- cbind(trips0[,c(1, 4, 2, 9)], data.frame(count=1))
> from <- unique(trips0[,c(4,5,7,8)])
> names(from) <- substring(names(from), 7)
> to <- unique(trips0[,c(9,10,12,13)])
> names(to) <- substring(names(to), 5)
> stations0 <- st_as_sf(merge(from, to, all=TRUE),
+   coords=c("station_longitude", "station_latitude"))
> stations <- aggregate(stations0, list(stations0$station_id),
+   head, n=1)
> suppressWarnings(stations <- st_cast(stations, "POINT"))
> st_crs(stations) <- 4326
> od <- aggregate(trips[,-(1:4)], list(trips$start_station_id,
+   trips$end_station_id), sum)
> od <- od[-(which(od[,1] == od[,2])),]
> library(stplanr)
> od_lines <- od2line(flow=od, zones=stations, zone_code="Group.1",
+   origin_code="Group.1", dest_code="Group.2")
> Sys.setenv(CYCLESTREET="xXxXXxXxXxXxXxXxX")
> od_routes <- line2route(od_lines, "route_cyclestreet",
+   plan = "fastest")
```

Origin-destination lines



Routed lines along cycle routes



Vector data

Spatial vector data is based on points, from which other geometries are constructed. Vector data is often also termed object-based spatial data. The light rail tracks are 2D vector data. The points themselves are stored as double precision floating point numbers, typically without recorded measures of accuracy (GNSS provides a measure of accuracy). Here, lines are constructed from points.

```
> all(st_is(byb, "XY"))
## [1] TRUE
> str(st_coordinates(st_geometry(byb)[[1]]))
##  num [1:14, 1:3] 5.33 5.33 5.33 5.33 5.33 ...
##  - attr(*, "dimnames")=List of 2
##  ..$ : chr [1:14] "4870663682" "331531217" "331531216" "331531215" ...
##   ..$ : chr [1:3] "X" "Y" "L1"
```

Representing spatial vector data in R (sp)

The **sp** package was a child of its time, using S4 formal classes, and the best compromise we then had of positional representation (not arc-node, but hard to handle holes in polygons). If we coerse **byb** to the **sp** representation, we see the formal class structure. Input/output used OGR/GDAL vector drivers in the **rgdal** package, and topological operations used GEOS in the **rgeos** package.

```
> library(sp)
> str(slot(as(st_geometry(byb), "Spatial"), "lines")[[1]])
## Formal class 'Lines' [package "sp"] with 2 slots
##   ..@ Lines:List of 1
##   .. ..$ :Formal class 'Line' [package "sp"] with 1 slot
##  .. .. .. ..@ coords: num [1:14, 1:2] 5.33 5.33 5.33 5.33 5.33 ...
##   .. .. .. .. ..- attr(*, "dimnames")=List of 2
##  .. .. .. .. .. ..$ : chr [1:14] "4870663682" "331531217" "331531216" "331
##   .. .. .. .. .. ..$ : chr [1:2] "lon" "lat"
##   ..@ ID   : chr "ID1"
```

Representing spatial vector data in R (sf)

The recent **sf** package bundles GDAL and GEOS (**sp** just defined the classes and methods, leaving I/O and computational geometry to other packages). **sf** used `data.frame` objects with one (or more) geometry column for vector data. The representation follows ISO 19125 (*Simple Features*), and has WKT (text) and WKB (binary) representations (used by GDAL and GEOS internally). The drivers include PostGIS and other database constructions permitting selection, and WFS for server APIs (**rgdal** does too, but requires more from the user).

```
> strwrap(st_as_text(st_geometry(byb)[[1]]))
## [1] "LINESTRING (5.333375 60.30436, 5.333386"
## [2] "60.30439, 5.333512 60.30463, 5.333664"
## [3] "60.30487, 5.3342 60.30559, 5.334472"
## [4] "60.30589, 5.334727 60.30613, 5.334901"
## [5] "60.30628, 5.33523 60.30652, 5.335494"
## [6] "60.30667, 5.335813 60.30682, 5.336282"
## [7] "60.30701, 5.336615 60.3071, 5.336872"
## [8] "60.30716)"
```

Baseline WKT and PROJ4

Spatial reference systems define how the geoid is viewed (prime meridian, ellipsoid, datum), and, if projected to the plane, where we are (central longitude, latitude, offsets, etc.). They also define the units - **sf** incorporates smart units handling. Projection (no datum change) and transformation are possible using PROJ and its `proj_api.h` interface directly (`rgdal::spTransform()` and `lwgeom::st_transform_proj()`), or through GDAL (`sf::st_transform()`). Migration to the new `proj.h` API has begun.

```
> (WKT <- st_crs(byb))
## Coordinate Reference System:
##   EPSG: 4326
##   proj4string: "+proj=longlat +datum=WGS84 +no_defs"
> strwrap(gsub(",", ", ", st_as_text(WKT)))
## [1] "GEOGCS[\"unknown\", DATUM[\"WGS_1984\","
## [2] "SPHEROID[\"WGS 84\", 6378137, 298.257223563,"
## [3] "AUTHORITY[\"EPSG\", \"7030\"]],"
## [4] "AUTHORITY[\"EPSG\", \"6326\"]],"
## [5] "PRIMEM[\"Greenwich\", 0, AUTHORITY[\"EPSG\","
## [6] "\"8901\"]], UNIT[\"degree\", 0.0174532925199433,"
## [7] "AUTHORITY[\"EPSG\", \"9122\"]],"
## [8] "AXIS[\"Longitude\", EAST], AXIS[\"Latitude\","
## [9] "NORTH]]"
> byb_utm <- st_transform(byb, crs=32632)
> st_crs(byb_utm)
## Coordinate Reference System:
##   EPSG: 32632
##  proj4string: "+proj=utm +zone=32 +datum=WGS84 +units=m +no_defs"
```

Escaping the WGS84 hub: PROJ 6 and OGC WKT2

Changes in the legacy PROJ representation and WGS84 transformation hub have been coordinated through the GDAL barn raising initiative. The syntax is changing to pipelines, but crucially WGS84 will often cease to be the pivot for moving between datums. A new OGC WKT is coming, and an SQLite EPSG file database is replacing CSV files. SRS will begin to support 3D by default, adding time too as SRS change.

```
> head(st_coordinates(st_geometry(byb)[[1]]), n=1)
##                   X        Y L1
## 4870663682 5.333375 60.30436  1
> x <- system(paste("echo 5.333375 60.30436 |",
+   "proj +proj=pipeline +ellps=WGS84",
+   "+step `projinfo -o PROJ -q epsg:32632`"),
+   intern=TRUE)
> as.numeric(strsplit(x, "\\t")[[1]])
## [1]  297436 6690941
> head(st_coordinates(st_geometry(byb_utm)[[1]]), n=1)
##             X       Y L1
## [1,] 297436.1 6690941  1
```

Representing spatial raster data in R (sf and stars)

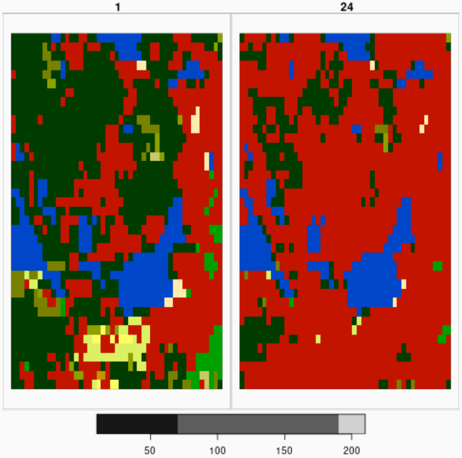
The new **stars** - Scalable, Spatiotemporal Tidy Arrays - package started looking at array structures and has built-in proxy data. Like **sf**, the development of **stars** has been supported by the R Consortium, and **stars** uses the infrastructure of **sf** to use GDAL for input/output and manipulation. In **sf**, the interface to the C++ GDAL library is based on **Rcpp**, which was not available when **rgdal** was written. The LandGIS ESA CCI land cover series may be downloaded (API coming), and displayed.

```
> fl <- "../../bml19/ESACCI-LC-L4-LCCS-Map-300m-P1Y-1992_2015-v2.0.7.tif"
> library(stars)
## Loading required package: abind
> LUC_stars <- read_stars(fl, proxy=TRUE)
> LUC_stars
## stars_proxy object with 1 attribute in file:
## $`ESACCI-LC-L4-LCCS-Map-300m-P1Y-1992_2015-v2.0.7.tif`
##   [1]  "../../bml19/ESACCI-LC-L4-LCCS-Map-300m-P1Y-1992_2015-
v2.0.7.tif"
##
## dimension(s):
##      from     to offset       delta
## x       1 129600   -180  0.00277778
## y       1  64800     90 -0.00277778
## band    1     24     NA          NA
##                               refsys point values
## x    +proj=longlat +datum=WGS8... FALSE   NULL
## y    +proj=longlat +datum=WGS8... FALSE   NULL
## band                           NA    NA   NULL
##
## x    [x]
## y    [y]
## band
```

Representing spatial raster data in R (sf and stars)

The **stars** package, with links to the **raster** package, is developing fast, and permits spatial subsets from a proxy object using an **sf** bounding box, here from the downloaded > 1GB global spatio-temporal LULC climate modelling file. We can add the correct RGB colours, but not yet the list of LULC categories (see also this issue on github)

```
> LUC_stars1 <- st_as_stars(LUC_stars[st_bbox(byb),,,c(1,24)])
> legs <- read.csv("../ESACCI-LC-Legend.csv", header = TRUE, sep = ";")
> plot(LUC_stars1, rgb=legs, main="")
```



Spatial weights and measures of autocorrelation

Neighbours

- The **spdep** package provides an `nb` class for neighbours, a list of length equal to the number of observations, with integer vector components.
- No-neighbours are encoded as an integer vector with a single element `0L`, and observations with neighbours as sorted integer vectors containing values in `1L:n` pointing to the neighbouring observations.
- This is a typical row-oriented sparse representation of neighbours. **spdep** provides many ways of constructing `nb` objects, and the representation and construction functions are widely used in other packages.

Weights

- `spdep` builds on the `nb` representation (undirected or directed graphs) with the `listw` object, a list with three components, an `nb` object, a matching list of numerical weights, and a single element character vector containing the single letter name of the way in which the weights were calculated.
- The most frequently used approach in the social sciences is calculating weights by row standardization, so that all the non-zero weights for one observation will be the inverse of the cardinality of its set of neighbours (`1/card(nb[[i]])`).

Data set (sf format)

We will be using election data from the 2015 Polish Presidential election in this chapter, with 2495 municipalities and Warsaw boroughs, and complete count data from polling stations aggregated to these areal units. The data are an `sf` object:

```
> library(sf)
> data(pol_pres15, package="spDataLarge")
> head(pol_pres15[, c(1, 4, 6)])
## Simple feature collection with 6 features and 3 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: 235157.1 ymin: 366913.3 xmax: 281431.7 ymax: 413016.6
## epsg (SRID):    NA
## proj4string:    +proj=tmerc +lat_0=0 +lon_0=18.99999999999998 +k=0.9993 +
5300000 +ellps=GRS80 +towgs84=0,0,0 +units=m +no_defs
##    TERYT                name       types
## 1 020101         BOLESŁAWIEC       Urban
## 2 020102         BOLESŁAWIEC       Rural
## 3 020103            GROMADKA       Rural
## 4 020104         NOWOGRODZIEC Urban/rural
## 5 020105           OSIECZNICA       Rural
## 6 020106 WARTA BOLESŁAWIECKA       Rural
##                        geometry
## 1 MULTIPOLYGON (((261089.5 38...
## 2 MULTIPOLYGON (((254150 3837...
## 3 MULTIPOLYGON (((275346 3846...
## 4 MULTIPOLYGON (((251769.8 37...
## 5 MULTIPOLYGON (((263423.9 40...
## 6 MULTIPOLYGON (((267030.7 38...
```

The spdep/spatialreg split

- Between early 2002 and April 2019, **spdep** contained functions for constructing and handling neighbour and spatial weights objects, tests for spatial autocorrelation, and model fitting functions.
- The latter have been split out into **spatialreg**, and will be discussed in the next section.
- **spdep** now accommodates objects represented using **sf** classes and **sp** classes directly, going beyond the explorations made in this vignette.

Contiguous neighbours

The `poly2nb()` function in **spdep** takes the boundary points making up the polygon boundaries in the object passed as the `pl=` argument. For each observation checks whether at least one (`queen=TRUE`, default), or at least two (rook, `queen=FALSE`) points are within `snap=` distance units of each other. The distances are planar in the raw coordinate units, ignoring geographical projections. Once the required number of sufficiently close points is found, the search is stopped.

```
> suppressPackageStartupMessages(library(spdep))
> system.time(nb_q <- poly2nb(pol_pres15, queen=TRUE))
##    user  system elapsed
##   1.345   0.040   1.391
> nb_q
## Neighbour list object:
## Number of regions: 2495
## Number of nonzero links: 14242
## Percentage nonzero weights: 0.2287862
## Average number of links: 5.708216
```

Contiguous neighbours

Pre-finding candidate contiguous neighbours may be helpful with larger objects, by finding intersecting bounding boxes, removing self-intersections and duplicate intersections (contiguities are by definition symmetric, if `i` is a neighbour of `j`, then `j` is a neighbour of `i`); the `foundInBox=` argument is used.

```
> f <- function(x) {
+   bb <- st_bbox(x)
+   mat <- cbind(c(bb[1], bb[1], bb[3], bb[3], bb[1]),
+                c(bb[2], bb[4], bb[4], bb[2], bb[2]))
+   st_polygon(list(mat))
+ }
> system.time({
+   fB1 <- st_intersects(st_as_sfc(lapply(st_geometry(pol_pres15), f)))
+   fB1a <- lapply(seq_along(fB1), function(i) fB1[[i]][fB1[[i]] > i])
+   fB1a <- fB1a[-length(fB1a)]
+   nb_sf_q1 <- poly2nb(pol_pres15, queen=TRUE, foundInBox=fB1a)
+ })
##    user  system elapsed
##   0.704   0.000   0.715
> all.equal(nb_q, nb_sf_q1, check.attributes=FALSE)
## [1] TRUE
```

Exchange of GAL files

Neighbour objects may be exported and imported in GAL format for exchange with other software, using `write.nb.gal()` and `read.gal()`. Using **reticulate**, it is possible to interoperate with the PySAL family of Python packages, first **libpysal** providing the basic weights handling infrastructure. As we can see, the percentage of non-zero neighbours is the same in both software systems.

```
> tf <- tempfile(fileext=".gal")
> write.nb.gal(nb_q, tf)
> library(reticulate)
> use_python(python='/usr/bin/python3')
> np <- import("numpy")
> libpysal <- import("libpysal")
> nb_gal_ps <- libpysal$io$open(tf)$read()
> nb_gal_ps$pct_nonzero
## [1] 0.2287862
```

Weights specification

Once neighbour objects are available, further choices need to made in specifying the weights objects. The `nb2listw()` function is used to create a `listw` weights object with an `nb` object, a matching list of weights vectors, and a style specification. Because handling no-neighbour observations now begins to matter, the `zero.policy=` argument is introduced. By default, this is `FALSE`, indicating that no-neighbour observations will cause an error, as the spatially lagged value for an observation with no neighbours is not available.

By convention, zero is substituted for the lagged value, as the cross product of a vector of zero-valued weights and a data vector, hence the name of `zero.policy`.

```
> args(nb2listw)
## function (neighbours, glist = NULL, style = "W", zero.policy = NULL)
## NULL
```

Weights specification

We will be using the helper function `spweights.constants()` below to show some consequences of varing style choices. It returns constants for a `listw` object, n is the number of observations, `n1` to `n3` are $n-1, \ldots$, `nn` is n^2 and S_0, S_1 and S_2 are constants, S_0 being the sum of the weights. There is a full discussion of the constants in Bivand and Wong (2018).

```
> args(spweights.constants)
## function (listw, zero.policy = NULL, adjust.n = TRUE)
## NULL
```

Weights specification

The **"B"** binary style gives a weight of unity to each neighbour relationship, and typically upweights units with no boundaries on the edge of the study area.

```
> lw_q_B <- nb2listw(nb_q, style="B")
> unlist(spweights.constants(lw_q_B))
##       n      n1      n2      n3      nn      S0
##    2495    2494    2493    2492 6225025   14242
##      S1      S2
##   28484  357280
```

Weights specification

The **"W"** row-standardized style upweights units around the edge of the study area that necessarily have fewer neighbours. This style first gives a weight of unity to each neighbour relationship, then divides these weights by the per unit sums of weights. Naturally this leads to division by zero where there are no neighbours, a not-a-number result, unless the chosen policy is to permit no-neighbour observations. We can see that S_0 is now equal to n.

```
> lw_q_W <- nb2listw(nb_q, style="W")
> unlist(spweights.constants(lw_q_W))[c(1,6:8)]
##           n          S0          S1          S2
##   2495.0000   2495.0000    957.5303 10406.4367
```

Global tests for autocorrelation (Morans I)

The implementation of Moran's I in **spdep** in the `moran.test()` function takes a vector of values `x=` and a `listw` object, and returns a list of `htest` (hypothesis test) objects defined in the **stats** package. The `randomisation=` argument indicates the underlying analytical approach used for calculating the variance of the measure.

```
> args(moran.test)
## function (x, listw, randomisation = TRUE, zero.policy = NULL,
##     alternative = "greater", rank = FALSE, na.action = na.fail,
##     spChk = NULL, adjust.n = TRUE, drop.EI2 = FALSE)
## NULL
```

Global tests for autocorrelation (Morans I)

The default for the `randomisation=` argument is `TRUE`, but here we will simply show that the test under normality is the same as a test of least squares residuals with only the intercept used in the mean model. The spelling of randomisation is that of Cliff and Ord (1973).

```
> mt <- moran.test(pol_pres15$I_turnout, listw=lw_q_B,
+                  randomisation=FALSE)
> if (broom_ok) broom::tidy(mt)[,1:4] else glance_htest(mt)
## # A tibble: 1 x 4
##   `Moran I statist~ Expectation Variance statistic
##               <dbl>       <dbl>    <dbl>     <dbl>
## 1             0.691   -0.000401 0.000140      58.5
```

Global tests for autocorrelation (Morans I)

The `lm.morantest()` function also takes a `resfun=` argument to set the function used to extract the residuals used for testing, and clearly lets us model other salient features of the response variable (Cliff and Ord 1981, 203).

```
> args(lm.morantest)
## function (model, listw, zero.policy = NULL, alternative = "greater",
##     spChk = NULL, resfun = weighted.residuals, naSubset = TRUE)
## NULL
```

Global tests for autocorrelation (Morans I)

To compare with the standard test, we are only using the intercept here, and as can be seen, the results are the same.

```
> ols <- lm(I_turnout ~ 1, pol_pres15)
> lmt <- lm.morantest(ols, listw=lw_q_B)
> if (broom_ok) broom::tidy(lmt)[,1:4] else glance_htest(lmt)
## # A tibble: 1 x 4
##   `Observed Moran~ Expectation Variance
##              <dbl>       <dbl>    <dbl>
## 1            0.691   -0.000401 0.000140
## # ... with 1 more variable: statistic[,1] <dbl>
```

Global tests for autocorrelation (Morans I)

The only difference between tests under normality and randomisation is that an extra term is added if the kurtosis of the variable of interest indicates a flatter or more peaked distribution, where the measure used is the classical measure of kurtosis.

```
> all.equal(3+e1071::kurtosis(pol_pres15$I_turnout, type=1),
+           moran(pol_pres15$I_turnout, listw=lw_q_B,
+                 n=nrow(pol_pres15), S0=Szero(lw_q_B))$K)
## [1] TRUE
```

Under the default randomisation assumption of analytical randomisation, the results are largely unchanged.

```
> mtr <- moran.test(pol_pres15$I_turnout, listw=lw_q_B)
> if (broom_ok) (tmtr <- broom::tidy(mtr)[,1:4]) else (tmtr <- glance_hte
## # A tibble: 1 x 4
##   `Moran I statist~ Expectation Variance statistic
##               <dbl>       <dbl>    <dbl>     <dbl>
## 1             0.691   -0.000401 0.000140      58.5
```

Global tests for autocorrelation (Morans I)

The PySAL **esda** package contains the `Moran` function reporting the same results, here under randomisation. The function returns results under normality, randomisation and by permutation simulation. Similar compatisons may be made for other global measures; for details see Bivand and Wong (2018).

```
> esda <- import("esda")
> np$random$seed(1L)
> mi <- esda$Moran(pol_pres15$I_turnout, nb_gal_ps, transformation="B",
+                  permutations=0L, two_tailed=FALSE)
> all.equal(c(mi$I, mi$EI, mi$VI_rand, mi$z_rand),
+           unname(do.call("c", tmtr)))
## [1] TRUE
```

Local tests for autocorrelation (Morans I)

Bivand and Wong (2018) discuss issues impacting the use of local indicators, such as local Moran's I and local Getis-Ord G. Some issues affect the calculation of the local indicators, others inference from their values. Because n statistics may be being calculated from the same number of observations, there are multiple comparison problems that need to be addressed. Although the apparent detection of hotspots from values of local indicators has been quite widely adopted, it remains fraught with difficulty because adjustment of the inferential basis to accommodate multiple comparisons is not often chosen, and as in the global case, mis-specification also remains a source of confusion. Further, interpreting local spatial autocorrelation in the presence of global spatial autocorrelation is challenging (Ord and Getis 2001; Tiefelsdorf 2002; Bivand, Müller, and Reder 2009).

```
> args(localmoran)
## function (x, listw, zero.policy = NULL, na.action = na.fail,
##   alternative = "greater", p.adjust.method = "none", mlvar = TRUE,
##     spChk = NULL, adjust.x = FALSE)
## NULL
```

Local tests for autocorrelation (Morans I)

In order to compare the results from `localmoran()` with the PySAL function in **esda**, we need first to re-run dividing by $n-1$ instead of n in parts of the calculation, by setting the argument `mlvar=FALSE`.

```
> locm_nml <- localmoran(pol_pres15$I_turnout, listw=lw_q_B,
+                        alternative="two.sided", mlvar=FALSE)
```

Once this is done, the local estimates of Moran's I agree within machine precision.

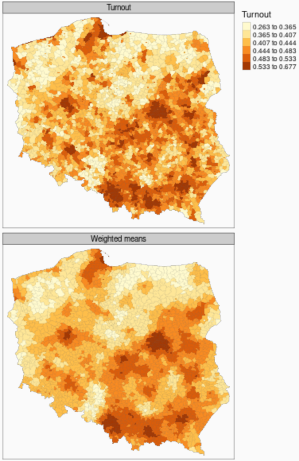
```
> np$random$seed(1L)
> loc_I_ps <- esda$Moran_Local(pol_pres15$I_turnout, nb_gal_ps,
+                              transformation="B", permutations=0L)
> all.equal(unname(locm_nml[,1]), c(loc_I_ps$Is))
## [1] TRUE
```

Local spatial heteroscedasticity (LOSH) statistic

Local spatial heteroscedasticity (LOSH) statistics were introduced fairly recently, and an implementation was contributed to **spdep** even more recently, so there is as yet little experience with the approach (Ord and Getis 2012). It has been extended to provide bootstrap p-values for the measures of heterogeneity (Xu, Mei, and Yan 2014). The **a=** argument takes a default value of 2, giving a Chi-squared interpretation to output.

```
> args(LOSH)
## function (x, listw, a = 2, var_hi = TRUE, zero.policy = NULL,
##     na.action = na.fail, spChk = NULL)
## NULL
> lh <- LOSH(pol_pres15$I_turnout, listw=lw_q_B)
```

It is also possible to map local spatially weighted mean values derived from the local measures, showing a smoothing effect



Spatial regression

Spatial regression with spatial weights

- Spatial autoregression models using spatial weights matrices were described in some detail using maximum likelihood estimation some time ago (Cliff and Ord 1973, 1981).
- A family of models were elaborated in spatial econometric terms extending earlier work, and in many cases using the simultaneous autoregressive framework and row standardization of spatial weights (Anselin 1988).
- The simultaneous and conditional autoregressive frameworks can be compared, and both can be supplemented using case weights to reflect the relative importance of different observations (Waller and Gotway 2004).

Boston hedonic house value air pollution data set

- Here we shall use the Boston housing data set, which has been restructured and furnished with census tract boundaries (Bivand 2017).
- The original data set used 506 census tracts and a hedonic model to try to estimate willingness to pay for clean air.
- The response was constructed from counts of ordinal answers to a 1970 census question about house value; the response is left and right censored in the census source.
- The key covariate was created from a calibrated meteorological model showing the annual nitrogen oxides (NOX) level for a smaller number of model output zones.
- The numbers of houses responding also varies by tract and model output zone.

Boston hedonic house value air pollution data set

We can start by reading in the 506 tract data set from **spData**, and creating a contiguity neighbour object and from that again a row standardized spatial weights object. If we examine the median house values, we find that they have been assigned as missing values, and that 17 tracts are affected.

```
> library(sf)
> suppressWarnings(boston_506 <- st_read(system.file(
+                                      "shapes/boston_tracts.shp",
+                                      package="spData")[1]))
## Reading layer `boston_tracts' from data source `/home/rsb/lib/r_libs/spDa
## Simple feature collection with 506 features and 36 fields
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: -71.52311 ymin: 42.00305 xmax: -
70.63823 ymax: 42.67307
## epsg (SRID):    4267
## proj4string:    +proj=longlat +datum=NAD27 +no_defs
> nb_q <- spdep::poly2nb(boston_506)
> lw_q <- spdep::nb2listw(nb_q, style="W")
> table(boston_506$censored)
##
##  left    no right
##     2   489    15
```

Boston hedonic house value air pollution data set

Next, we can subset to the remaining 489 tracts with non-censored house values, and the neighbour object to match. The neighbour object now has one observation with no neighbours.

```
> boston_489 <- boston_506[!is.na(boston_506$median),]
> nb_q_489 <- spdep::poly2nb(boston_489)
> lw_q_489 <- spdep::nb2listw(nb_q_489, style="W", zero.policy=TRUE)
```

Boston hedonic house value air pollution data set

The NOX_ID variable specifies the upper level aggregation, letting us aggregate the tracts to air pollution model output zones. We can create aggregate neighbour and row standardized spatial weights objects, and aggregate the NOX variable taking means, and the CHAS Charles River dummy variable for observations on the river.

```
> agg_96 <- list(as.character(boston_506$NOX_ID))
> boston_96 <- aggregate(boston_506[, "NOX_ID"], by=agg_96,
+                        unique)
> boston_96 <- st_cast(boston_96, "MULTIPOLYGON")
> nb_q_96 <- spdep::poly2nb(boston_96)
> lw_q_96 <- spdep::nb2listw(nb_q_96)
> boston_96$NOX <- aggregate(boston_506$NOX, agg_96, mean)$x
> boston_96$CHAS <- aggregate(as.integer(boston_506$CHAS)-1,
+                             agg_96, max)$x
```

Boston hedonic house value air pollution data set

The response is aggregated using the **`weightedMedian()`** function in **matrixStats**, and midpoint values for the house value classes. Counts of houses by value class were punched to check the published census values, which can be replicated using **`weightedMedian()`** at the tract level. Here we find two output zones with calculated weighted medians over the upper census question limit of USD 50,000, and remove them subsequently as they also are affected by not knowing the appropriate value to insert for the top class by value.

```
> nms <- names(boston_506)
> ccounts <- 23:31
> for (nm in nms[c(22, ccounts, 36)]) {
+   boston_96[[nm]] <- aggregate(boston_506[[nm]],
+                                 agg_96, sum)$x
+ }
> br2 <- c(3.50,  6.25,  8.75, 12.50, 17.50, 22.50,
+          30.00, 42.50, 60.00)*1000
> counts <- as.data.frame(boston_96)[, nms[ccounts]]
> f <- function(x) matrixStats::weightedMedian(x=br2, w=x,
+                                               interpolate=TRUE)
> boston_96$median <- apply(counts, 1, f)
> is.na(boston_96$median) <- boston_96$median > 50000
> summary(boston_96$median)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    9009   20417   23523   25263   30073   49496
##    NA's
##       2
```

Boston hedonic house value air pollution data set

Before subsetting, we aggregate the remaining covariates by weighted mean using the tract population counts punched from the census (Bivand 2017). We now have two data sets each at the lower, census tract level and the upper, air pollution model output zone level, one including the censored observations, the other excluding them.

```
> POP <- boston_506$POP
> f <- function(x) matrixStats::weightedMean(x[,1], x[,2])
> for (nm in nms[c(9:11, 14:19, 21, 33)]) {
+   s0 <- split(data.frame(boston_506[[nm]], POP), agg_96)
+   boston_96[[nm]] <- sapply(s0, f)
+ }
> boston_94 <- boston_96[!is.na(boston_96$median),]
> nb_q_94 <- spdep::subset.nb(nb_q_96, !is.na(boston_96$median))
> lw_q_94 <- spdep::nb2listw(nb_q_94, style="W")
```

Boston hedonic house value air pollution data set

In order to try out some of the variant models, we need to remove the no-neighbour observations from the tract level, and from the model output zone aggregated level, in two steps as reducing the tract level induces a no-neighbour outcome at the model output zone level.

```
> oo <- aggregate(boston_489[,"NOX_ID"], list(boston_489$NOX_ID),
+                 unique)
> boston_94a <- st_cast(oo, "MULTIPOLYGON")
> nb_q_94a <- spdep::poly2nb(boston_94a)
> oo <- which(spdep::card(nb_q_94a) == 0)
> NOX_ID_no_neighs <- boston_94a$NOX_ID[oo]
> oo <- is.na(match(boston_489$NOX_ID, NOX_ID_no_neighs))
> boston_487 <- boston_489[oo,]
> oo <- aggregate(boston_487[, "NOX_ID"], list(ids=boston_487$NOX_ID),
+                 unique)
> boston_93 <- st_cast(oo, "MULTIPOLYGON")
> row.names(boston_93) <- as.character(boston_93$NOX_ID)
> oo <- unique(as.character(boston_93$NOX_ID))
> nb_q_93 <- spdep::poly2nb(boston_93, row.names=oo)
```

Boston hedonic house value air pollution data set

The original model related the log of median house values by tract to the square of NOX values, including other covariates usually related to house value by tract, such as aggregate room counts, aggregate age, ethnicity, social status, distance to downtown and to the nearest radial road, a crime rate, and town-level variables reflecting land use (zoning, industry), taxation and education (Bivand 2017). This structure will be used here to exercise issues raised in fitting spatial regression models, including the presence of multiple levels.

```
> form <- formula(log(median) ~ CRIM + ZN + INDUS + CHAS +
+                 I((NOX*10)^2) + I(RM^2) + AGE + log(DIS) +
+                 log(RAD) + TAX + PTRATIO + I(BB/100) +
+                 log(I(LSTAT/100)))
```

Spatial regression

- In trying to model these spatial processes, we may choose to model the spatial autocorrelation in the residual with a spatial error model (SEM).
- If the processes in the covariates and the response match, we should find little difference between the coefficients of a least squares and a SEM, but very often they diverge, suggesting that a Hausman test for this condition should be employed (Pace and LeSage 2008).
- This may be related to earlier discussions of a spatial equivalent to the unit root and cointegration where spatial processes match (Fingleton 1999).

Spatial regression

- Work reviewed by Mur and Angulo (2006) on the Durbin model, including the spatially lagged covariates in the model, permits a shared spatial process to be viewed and tested for as a Common Factor (Burridge 1981; Bivand 1984).
- The inclusion of spatially lagged covariates lets us check whether the same spatial process is manifest in the response and the covariates (SEM), whether they are different processes, or whether no process is detected.
- A model with a spatial process in the response only is termed a spatial lag model (SLM, often SAR - spatial autoregressive), and with different processes in the response and covariates a spatial Durbin model (SDM) (LeSage and Pace 2009).

- If we extend this family with processes in the covariates and the residual, we get a spatial error Durbin model (SDEM).
- If it is chosen to admit a spatial process in the residuals in addition to a spatial process in the response, again two models are formed, a general nested model (GNM) nesting all the others, and a model without spatially lagged covariates (SAC, also known as SARAR).
- If neither the residuals nor the residual are modelled with spatial processes, spatially lagged covariates may be added to a linear model, as a spatially lagged X model (SLX) (Elhorst 2010; Bivand 2012; Halleck Vega and Elhorst 2015).

Spatial regression

- Although making predictions for new locations for which covariates are observed was raised as an issue some time ago, it has many years to make progress in reviewing the possibilities (Bivand 2002; Goulard, Laurent, and Thomas-Agnan 2017).
- The prediction method for SLM, SDM, SEM, SDEM, SAC and GNM models fitted with maximum likelihood were contributed as a Google Summer of Coding project by Martin Gubri.
- This work, and work on similar models with missing data (Suesse 2018) is also relevant for exploring censored median house values in the Boston data set.
- Work on prediction also exposed the importance of the reduced form of these models, in which the spatial process in the response interacts with the regression coefficients in the SLM, SDM, SAC and GNM models.

- The consequence of these interactions is that a unit change in a covariate will only impact the response as the value of the regression coefficient if the spatial coefficient of the lagged response is zero.
- Where it is non-zero, global spillovers, impacts, come into play, and these impacts should be reported rather than the regression coefficients (LeSage and Pace 2009; Elhorst 2010; Bivand 2012; Halleck Vega and Elhorst 2015).
- Local impacts may be reported for SDEM and SLX models, using linear combination to calculate standard errors for the total impacts of each covariate (sums of coefficients on the covariates and their spatial lags).

Spatial regression

- Current work in the **spatialreg** package is focused on refining the handling of spatially lagged covariates using a consistent `Durbin=` argument taking either a logical value or a formula giving the subset of covariates to add in spatially lagged form.
- There is a speculation that some covariates, for example some dummy variables, should not be added in spatially lagged form.
- This then extends to handling these included spatially lagged covariates appropriately in calculating impacts.
- This work applies to cross-sectional models fitted using MCMC or maximum likelihood, and will offer facilities to spatial panel models.

- It is worth mentioning the almost unexplored issues of functional form assumptions, for which flexible structures are useful, including spatial quantile regression presented in the **McSpatial** package (McMillen 2013).
- There are further issues with discrete response variables, covered by some functions in **McSpatial**, and in the **spatialprobit** and **ProbitSpatial** packages (Wilhelm and Matos 2013; Martinetti and Geniaux 2017); the MCMC implementations of the former are based on LeSage and Pace (2009).
- Finally, Wagner and Zeileis (2019) show how an SLM model may be used in the setting of recursive partitioning, with an implementation using `spatialreg::lagsarlm()` in the **lagsarlmtree** package.

Estimators

- The review of cross-sectional maximum likelihood and generalized method of moments (GMM) estimators in **spatialreg** and **sphet** for spatial econometrics style spatial regression models by Bivand and Piras (2015) is still largely valid.
- In the review, estimators in these R packages were compared with alternative implementations available in other programming languages elsewhere.
- The review did not cover Bayesian spatial econometrics style spatial regression.

Maximum likelihood

For models with single spatial coefficients (SEM and SDEM using `errorsarlm()`, SLM and SDM using `lagsarlm()`), the methods initially described by Ord (1975) are used. Both estimating functions take similar arguments, where the first two, `formula=` and `data=` are shared by most model estimating functions. The third argument is a `listw` spatial weights object, while `na.action=` behaves as in other model estimating functions if the spatial weights can reasonably be subsetted to avoid observations with missing values.

The `weights=` argument may be used to provide weights indicating the known degree of per-observation variability in the variance term - this is not available for `lagsarlm()`.

```
> suppressPackageStartupMessages(library(spatialreg))
> args(errorsarlm)
## function (formula, data = list(), listw, na.action, weights = NULL,
##     Durbin, etype, method = "eigen", quiet = NULL, zero.policy = NULL,
##     interval = NULL, tol.solve = .Machine$double.eps, trs = NULL,
##     control = list())
## NULL
```

The `Durbin=` argument replaces the earlier `type=` and `etype=` arguments, and if not given is taken as `FALSE`. If given, it may be `FALSE`, `TRUE` in which case all spatially lagged covariates are included, or a one-sided formula specifying which spatially lagged covariates should be included. The `method=` argument gives the method for calculating the log determinant term in the log likelihood function, and defaults to `"eigen"`, suitable for moderate sized data sets.

The `interval=` argument gives the bounds of the domain for the line search using `stats::optimize()` used for finding the spatial coefficient. The `control=` argument takes a list of control values to permit more careful adjustment of the running of the estimation function.

```
> args(lagsarlm)
## function (formula, data = list(), listw, na.action, Durbin, type,
##     method = "eigen", quiet = NULL, zero.policy = NULL, interval = NULL,
##     tol.solve = .Machine$double.eps, trs = NULL, control = list())
## NULL
```

The `sacsarlm()` function may take second spatial weights and interval arguments if the spatial weights used to model the two spatial processes in the SAC and GNM specifications differ. By default, the same spatial weights are used. By default, `stats::nlminb()` is used for numerical optimization, using a heuristic to choose starting values.

```
> args(sacsarlm)
## function (formula, data = list(), listw, listw2 = NULL, na.action,
##     Durbin, type, method = "eigen", quiet = NULL, zero.policy = NULL,
##     tol.solve = .Machine$double.eps, llprof = NULL, interval1 = NULL,
##     interval2 = NULL, trs1 = NULL, trs2 = NULL, control = list())
## NULL
```

Maximum likelihood

Standard methods for fitted models are provided, such as `summary()`. The `Nagelkerke=` argument permits the return of a value approximately corresponding to a coefficient of determination, although the summary method anyway provides the value of `stats::AIC()` because a `stats::logLik()` method is provided for `"sarlm"` objects. If the `"sarlm"` object is a SEM or SDEM, the Hausman test may be performed by setting `Hausman=TRUE` to see whether the regression coefficients are sufficiently like least squares coefficients, indicating absence of mis-specification from that source.

```
> args(summary.sarlm)
## function (object, correlation = FALSE, Nagelkerke = FALSE, Hausman = FALSE
##     adj.se = FALSE, ...)
## NULL
```

Hausman tests

As an example, we may fit SEM and SDEM to the 94 and 489 observation Boston data sets, and present the Hausman test results. Here we are using the `control=` list argument to pass through pre-computed eigenvalues for the default `"eigen"` method.

```
> eigs_489 <- eigenw(lw_q_489)
> SDEM_489 <- errorsarlm(form, data=boston_489, listw=lw_q_489,
+                        Durbin=TRUE, zero.policy=TRUE,
+                        control=list(pre_eig=eigs_489))
> SEM_489 <- errorsarlm(form, data=boston_489, listw=lw_q_489,
+                       zero.policy=TRUE,
+                       control=list(pre_eig=eigs_489))
> cbind(data.frame(model=c("SEM", "SDEM")),
+       rbind(broom::tidy(Hausman.test(SEM_489)),
+             broom::tidy(Hausman.test(SDEM_489))))[,1:4]
##   model statistic      p.value parameter
## 1   SEM   51.9862 2.827192e-06        14
## 2  SDEM   48.6551 6.481654e-03        27
```

```
> eigs_94 <- eigenw(lw_q_94)
> SDEM_94 <- errorsarlm(form, data=boston_94, listw=lw_q_94,
+                       Durbin=TRUE,
+                       control=list(pre_eig=eigs_94))
> SEM_94 <- errorsarlm(form, data=boston_94, listw=lw_q_94,
+                      control=list(pre_eig=eigs_94))
> cbind(data.frame(model=c("SEM", "SDEM")),
+       rbind(broom::tidy(Hausman.test(SEM_94)),
+             broom::tidy(Hausman.test(SDEM_94))))[, 1:4]
##   model statistic   p.value parameter
## 1   SEM 15.657083 0.3347612        14
## 2  SDEM  9.205152 0.9994394        27
```

LR tests

We can use `spatialreg::LR.sarlm()` to apply a likelihood ratio test between nested models, but here choose `lmtest::lrtest()`, which gives the same results, preferring models including spatially lagged covariates.

```
> suppressWarnings(broom::tidy(lmtest::lrtest(SEM_489, SDEM_489)))
## # A tibble: 2 x 5
##    X.Df LogLik    df statistic   p.value
##   <dbl>  <dbl> <dbl>     <dbl>     <dbl>
## 1    16   273.    NA       NA   NA
## 2    29   311.    13       74.4  1.23e-10
> suppressWarnings(broom::tidy(lmtest::lrtest(SEM_94, SDEM_94)))
## # A tibble: 2 x 5
##    X.Df LogLik    df statistic    p.value
##   <dbl>  <dbl> <dbl>     <dbl>      <dbl>
## 1    16   59.7    NA       NA   NA
## 2    29   81.3    13       43.2  0.0000421
```

The SLX model is fitted using least squares, and also returns a log likelihood value, letting us test whether we need a spatial process in the residuals.

```
> SLX_489 <- lmSLX(form, data=boston_489, listw=lw_q_489,
+                    zero.policy=TRUE)
> suppressWarnings(broom::tidy(lmtest::lrtest(SLX_489, SDEM_489)))
## # A tibble: 2 x 5
##    X.Df LogLik    df statistic   p.value
##   <dbl>  <dbl> <dbl>     <dbl>     <dbl>
## 1    28   231.    NA        NA  NA
## 2    29   311.     1      159.  1.55e-36
> SLX_94 <- lmSLX(form, data=boston_94, listw=lw_q_94)
> suppressWarnings(broom::tidy(lmtest::lrtest(SLX_94, SDEM_94)))
## # A tibble: 2 x 5
##    X.Df LogLik    df statistic p.value
##   <dbl>  <dbl> <dbl>     <dbl>   <dbl>
## 1    28   81.2    NA    NA       NA
## 2    29   81.3     1     0.216   0.642
```

LR tests with weights

This outcome is sustained also when we use the counts of house units by tract and output zones as weights:

```
> SLX_94w <- lmSLX(form, data=boston_94, listw=lw_q_94,
+                  weights=units)
> SDEM_94w <- errorsarlm(form, data=boston_94, listw=lw_q_94,
+                        Durbin=TRUE, weights=units,
+                        control=list(pre_eig=eigs_94))
> suppressWarnings(broom::tidy(lmtest::lrtest(SLX_94w, SDEM_94w)))
## # A tibble: 2 x 5
##    X.Df LogLik    df statistic p.value
##   <dbl>  <dbl> <dbl>     <dbl>   <dbl>
## 1    28   97.5    NA    NA      NA
## 2    29   98.0     1     0.917   0.338
```

Impacts

Since sampling is not required for inference for SLX and SDEM models, linear combination is used for models fitted using maximum likelihood; results are shown here for the air pollution variable only. The literature has not yet resolved the question of how to report model output, as each covariate is now represented by three impacts. Where spatially lagged covariates are included, two coefficients are replaced by three impacts.

```
> sum_imp_94_SDEM <- summary(impacts(SDEM_94))
> rbind(Impacts=sum_imp_94_SDEM$mat[5,],
+       SE=sum_imp_94_SDEM$semat[5,])
##                Direct      Indirect        Total
## Impacts -0.012764045 -0.018454532 -0.031218577
## SE       0.002354762  0.004717734  0.005303962
```

In the SLX and SDEM models, the direct impacts are the consequences for the response of changes in air pollution in the same observational entity, and the indirect (local) impacts are the consequences for the response of changes in air pollution in neighbouring observational entities. A recent question: how to correct standard errors for heterscedasticity before linear combination?

```
> sum_imp_94_SLX <- summary(impacts(SLX_94))
> rbind(Impacts=sum_imp_94_SLX$mat[5,],
+       SE=sum_imp_94_SLX$semat[5,])
##                Direct      Indirect        Total
## Impacts -0.012766473 -0.018743765 -0.031510238
## SE       0.002796966  0.005564633  0.006114058
```

- The Spatial Econometrics Library is part of the extensive Matlab code repository at https://www.spatial-econometrics.com/ and documented in LeSage and Pace (2009).
- The Google Summer of Coding project in 2011 by Abhirup Mallik mentored by Virgilio Gómez-Rubio yielded translations of some of the model fitting functions for SEM, SDEM, SLM, SDM, SAC and GNM from the Matlab code.
- These have now been added to **spatialreg** as `spBreg_err()`, `spBreg_lag()` and `spBreg_sac()` with `Durbin=` arguments to handle the inclusion of spatially lagged covariates.
- As yet, heteroskedastic disturbances are not accommodated.
- The functions return `"mcmc"` objects as specified in the **coda** package, permitting the use of tools from **coda** for handling model output.

Fitting the SDEM model for the tracts takes about an order of magnitude longer than using ML, but there is more work to do subsequently, and this difference scales more in the number of samples than covariates or observations. The impacts are extracted directly from the samples:

```
> system.time(SDEM_489B <- spBreg_err(form, data=boston_489,
+             listw=lw_q_489, Durbin=TRUE, zero.policy=TRUE))
##    user  system elapsed
##   3.466   0.018   3.548
> sum_imp_498_SDEM_B <- summary(impacts(SDEM_489B))
> rbind(Impacts=sum_imp_498_SDEM_B$mat[5,],
+       SE=sum_imp_498_SDEM_B$semat[5,])
##                Direct     Indirect       Total
## Impacts 0.0009563711 -0.009935701 -0.00897933
## SE      0.0016970515  0.002634999  0.00243357
```

In the MCMC case, the gridded log determinants (200 LU decompositions) and sampling takes most time; in the ML case using eigenvalues is taken by log determinant setup and optimization, and by dense matrix asymptotic standard errors:

```
> t(attr(SDEM_489B, "timings")[ , 3])
##      set_up SE_classic_set_up complete_setup
## [1,]  0.007             0.558          0.105
##      sampling finalise
## [1,]    2.769    0.109
> t(errorsarlm(form, data=boston_489, listw=lw_q_489, Durbin=TRUE,
+              zero.policy=TRUE)$timings[,2])
##      set_up eigen_set_up eigen_opt coefs
## [1,]  0.015        0.092     0.035 0.008
##      eigen_hcov eigen_se
## [1,]      0.109    0.251
```

Handling the log determinant term

- It has been known for over twenty years that the sparse matrix representation of spatial weights overcomes the difficulties of fitting models with larger numbers of observations using maximum likelihood and MCMC where the log determinant term comes into play (R. K. Pace and Barry 1997a, 1997c, 1997d, 1997b).
- During the development of these approaches in model fitting functions in **spatialreg**, use was first made of C code also used in the S-PLUS SpatialStats module (Kaluzny et al. 1998), then **SparseM** which used a compressed sparse row form very similar to `"nb"` and `"listw"` objects.
- This was followed by the use of **spam** and **Matrix** methods, both of which mainly use compressed sparse column representations. Details are provided in Bivand, Hauke and Kossowski (2013).
- We missed a method given by Martin (2005) which deserves implementation.

Handling the log determinant term

The domain of the spatial coefficient(s) is given by the `interval=` argument to model fitting functions, and returned in the fitted object; this case is trivial, because the upper bound is unity by definition, because of the use of row standardization. The interval is the inverse of the range of the eigenvalues of the weights matrix:

```
> SDEM_94$interval
## [1] -1.527257  1.000000
> 1/range(eigs_94)
## [1] -1.527257  1.000000
```

Finding the interval within which to search for the spatial coefficient is trivial for smaller data sets, but more complex for larger ones. It is possible to use heuristics implemented in `lextrW()` (Griffith, Bivand, and Chun 2015); or `RSpectra::eigs()` after coercion to a **Matrix** package compressed sparse column representation:

```
> 1/c(lextrW(lw_q_94))
##  lambda_n  lambda_1
## -1.527257  1.000000
> W <- as(lw_q_94, "CsparseMatrix")
> 1/Re(c(RSpectra::eigs(W, k=1, which="SR")$values,
+        RSpectra::eigs(W, k=1, which="LR")$values))
## [1] -1.527257  1.000000
```

Handling the log determinant term

The baseline log determinant term as given by Ord (1975) for a coefficient value proposed in sampling or during numerical optimization; this extract matches the `"eigen"` method (with or without `control=list(pre_eig=...)"`): Using sparse matrix functions from **Matrix**, the LU decomposition can be used for asymmetric matrices; this extract matches the `"LU"` method:

```
> coef <- 0.5
> sum(log(1 - coef * eigs_94))
## [1] -2.867292
> I <- Diagonal(nrow(boston_94))
> LU <- lu(I - coef * W)
> dU <- abs(diag(slot(LU, "U")))
> sum(log(dU))
## [1] -2.867292
```

Cholesky decomposition for symmetric matrices, with `similar.listw()` used to handle asymmetric weights that are similar to symmetric. The default value od `super` allows **Matrix** to choose between supernodal or simplicial decomposition; this extract matches the `"Matrix_J"` method; the `"Matrix"` and `"spam_update"` methods are to be preferred as they pre-compute the fill-reducing permutation of the decomposition since the weights do not change for different values of the coefficient.

```
> W <- as(similar.listw(lw_q_94), "CsparseMatrix")
> super <- as.logical(NA)
> cch <- Cholesky((I - coef * W), super=super)
> c(2 * determinant(cch, logarithm = TRUE)$modulus)
## [1] -2.867292
```

Handling the log determinant term

- Maximum likelihood model fitting functions in **spatialreg** and **splm** use `jacobianSetup()` to populate `env=` environment with intermediate objects needed to find log determinants during optimization; **HSAR** uses `mcdet_setup()` to set up Monte Carlo approximation terms.
- Passing environments to objective functions is efficient because they are passed by reference rather than value.
- As yet the Bayesian models are limited to control argument `ldet_method="SE_classic"` at present, using `"LU"` to generate a coarse grid of control argument `nrho=200L` log determinant values in the interval, spline interpolated to a finer grid of length control argument `interpn=2000L`, from which griddy Gibbs samples are drawn.
- It is hoped to add facilities to choose alternative methods in the future. This would offer possibilities to move beyond griddy Gibbs, but using gridded log determinant values seems reasonable at present.

Impacts

Impacts are calculated using model object class specific `impacts()` methods. In the **sphet** package, the impacts method for `"gstsls"` uses the **spatialreg** `impacts()` framework, as does the **splm** package for `"splm"` fitted model objects. `impacts()` methods require either a `tr=` argument a vector of traces of the power series of the weights object typically computed with `trW()` or a `listw=` argument; the `evalues=` argument is experimental, and takes the eigenvalues of the weights matrix.

```
> args(impacts.sarlm)
## function (obj, ..., tr = NULL, R = NULL, listw = NULL, evalues = NULL,
##     useHESS = NULL, tol = 1e-06, empirical = FALSE, Q = NULL)
## NULL
```

The summary method for the output of `impacts()` methods where inference from samples was requested by default uses the `summary()` method for `"mcmc"` objects defined in the **coda** package. It can instead report just matrices of standard errors, z-values and p-values by setting `zstats=` and `short=` to **TRUE**. Recently it was suggested that CI reporting may be easier to read.

```
> args(summary.lagImpact)
## function (object, ..., zstats = FALSE, short = FALSE, reportQ = NULL)
## NULL
```

Impacts

In contrast to local indirect impacts in SLX and SDEM models, global indirect impacts are found in models including the spatially lagged response. For purposes of exposition, let us fit an SLM. Traces of the first `m=` matrices of the power series in the spatial weights are pre-computed (LeSage and Pace 2009). The `type=` argument is `"mult"` by default.

```
> SLM_489 <- lagsarlm(form, data=boston_489, listw=lw_q_489,
+                     zero.policy=TRUE,
+                     control=list(pre_eig=eigs_489))
> W <- as(lw_q_489, "CsparseMatrix")
> tr_489 <- trW(W)
> str(tr_489)
##  num [1:30] 0 90.8 29.4 42.1 26.5 ...
##  - attr(*, "timings")= Named num [1:2] 0.194 0.198
##   ..- attr(*, "names")= chr [1:2] "user.self" "elapsed"
##  - attr(*, "type")= chr "mult"
##  - attr(*, "n")= int 489
```

In this case, the spatial process in the response is not strong, so the global indirect impacts (here for the air pollution variable) are weak.

```
> SLM_489_imp <- impacts(SLM_489, tr=tr_489, R=2000)
> SLM_489_imp_sum <- summary(SLM_489_imp, short=TRUE, zstats=TRUE)
> res <- rbind(Impacts=sapply(SLM_489_imp$res, "[", 5),
+              SE=SLM_489_imp_sum$semat[5,])
> colnames(res) <- c("Direct", "Indirect", "Total")
> res
##                Direct      Indirect        Total
## Impacts -0.005930731 -1.014747e-05 -0.005940879
## SE       0.001068486  9.864944e-05  0.001075766
```

Impacts

Of more interest is trying to reconstruct the direct and total impacts using dense matrix methods; the direct global impacts are the mean of the diagonal of the dense impacts matrix, and the total global impacts are the sum of all matrix elements divided by the number of observations. The direct impacts agree, but the total impacts differ slightly.

```
> coef_SLM_489 <- coef(SLM_489)
> IrW <- Diagonal(489) - coef_SLM_489[1] * W
> S_W <- solve(IrW)
> S_NOX_W <- S_W %*% (diag(489) * coef_SLM_489[7])
> c(Direct=mean(diag(S_NOX_W)), Total=sum(S_NOX_W)/489)
##       Direct        Total
## -0.005930731 -0.005940858
```

This bare-bones approach corresponds to using the `listw=` argument, and as expected gives the same output. The experimental `evalues=` approach which is known to be numerically exact by definition gives the same results as the matrix power series trace approach, so the sight difference may be attributed to the consequences of inverting the spatial process matrix.

```
> sapply(impacts(SLM_489, listw=lw_q_489), "[", 5)
##        direct      indirect         total
## -5.930731e-03 -1.012671e-05 -5.940858e-03
> sapply(impacts(SLM_489, evalues=eigs_489), "[", 5)
##        direct      indirect         total
## -5.930731e-03 -1.014747e-05 -5.940879e-03
```

Predictions and impacts

We'll use a `predict()` method for `"sarlm"` objects to double-check impacts, here for the pupil-teacher ratio (`PTRATIO`). The method was re-written by Martin Gubri based on Goulard, Laurent and Thomas-Agnan (2017). The `pred.type=` argument specifies the prediction strategy among those presented in the article. First we'll increment `PTRATIO` by one to show that, using least squares, the mean difference between predictions from the incremented new data and fitted values is equal to the regression coefficient.

In models including the spatially lagged response, and when the spatial coefficient in different from zero, this is not the case in general, and is why we need `impacts()` methods. The difference here is not great, but neither is it zero, and needs to be handled.

```
> nd_489 <- boston_489
> nd_489$PTRATIO <- nd_489$PTRATIO + 1
> OLS_489 <- lm(form, data=boston_489)
> fitted <- predict(OLS_489)
> nd_fitted <- predict(OLS_489, newdata=nd_489)
> all.equal(unname(coef(OLS_489)[12]), mean(nd_fitted - fitted))
## [1] TRUE
> fitted <- predict(SLM_489)
> nd_fitted <- predict(SLM_489, newdata=nd_489, listw=lw_q_489,
+                      pred.type="TS", zero.policy=TRUE)
> (tot <- mean(nd_fitted - fitted))
## [1] -0.03033088
> impacts(SLM_489, evalues=eigs_489)$total[11]
## [1] -0.03032871
> all.equal(unname(coef_SLM_489[13]), tot)
## [1] "Mean relative difference: 0.001783086"
```

Predictions

In the Boston tracts data set, 17 observations of median house values, the response, are censored. Using these as an example and comparing some `pred.type=` variants for the SDEM model and predicting out-of-sample, we can see that there are differences, suggesting that this is a fruitful area for study. There have been a number of alternative proposals for handling missing variables (Gómez-Rubio, Bivand, and Rue 2015; Suesse 2018). Here, we'll list the predictions for the censored tract observations using three different prediction types, taking the exponent to get back to the USD median house values.

```
> nd <- boston_506[is.na(boston_506$median),]
> t0 <- exp(predict(SDEM_489, newdata=nd, listw=lw_q,
+                   pred.type="TS", zero.policy=TRUE))
> suppressWarnings(t1  <- exp(predict(SDEM_489, newdata=nd,
+                                     listw=lw_q, pred.type="KP2",
+                                     zero.policy=TRUE)))
> suppressWarnings(t2  <- exp(predict(SDEM_489, newdata=nd,
+                                     listw=lw_q, pred.type="KP5",
+                                     zero.policy=TRUE)))
> head(data.frame(fit_TS=t0[,1], fit_KP2=c(t1), fit_KP5=c(t2),
+          censored=boston_506$censored[as.integer(attr(t0,
+                                                  "region.id"))]))
##       fit_TS   fit_KP2   fit_KP5 censored
## 13 23912.450 29477.240 28147.151    right
## 14 28125.588 27001.192 28516.159    right
## 15 30553.380 36184.007 32476.053    right
## 17 18518.448 19620.801 18878.244    right
## 43  9563.613  6816.721  7561.353     left
## 50  8371.200  7196.445  7383.139     left
```

Markov random field and multilevel models with spatial weights

There is a large literature in spatial epidemiology using CAR and ICAR models in spatially structured random effects. These extend to multilevel models, in which the spatially structured random effects may apply at different levels of the model (Bivand et al. 2017).

The **lme4** package lets us add an IID unstructured random effect at the model output zone level:

```
> library(lme4)
> MLM <- lmer(update(form, . ~ . + (1 | NOX_ID)), data=boston_487,
+              REML=FALSE)
> boston_93$MLM_re <- ranef(MLM)[[1]][,1]
```

Two packages, **hglm** and **HSAR**, offer SAR upper level spatially structured random effects, and require the specification of a sparse matrix mapping the upper level enities onto lower level entities, and sparse binary weights matrices:

```
> library(Matrix)
> suppressMessages(library(MatrixModels))
> Delta <- as(model.Matrix(~ -1 + as.factor(NOX_ID), data=boston_487,
+                          sparse=TRUE), "dgCMatrix")
> M <- as(spdep::nb2listw(nb_q_93, style="B"), "CsparseMatrix")
```

The extension of **hglm** to sparse spatial setting extended its facilities (Alam, Rönnegård, and Shen 2015), and also permits the modelling of discrete responses. First we fit an IID random effect:

```
> suppressPackageStartupMessages(library(hglm))
> y_hglm <- log(boston_487$median)
> X_hglm <- model.matrix(lm(form, data=boston_487))
> suppressWarnings(HGLM_iid <- hglm(y=y_hglm, X=X_hglm, Z=Delta))
```

followed by a SAR model at the upper level (corresponding to a spatial error (SEM) model), which reports the spatially structured random effect without fully converging, so coefficient standard errors are not available:

```
> suppressWarnings(HGLM_sar <- hglm(y=y_hglm, X=X_hglm, Z=Delta,
+                                   rand.family=SAR(D=M)))
> boston_93$HGLM_re <- unname(HGLM_iid$ranef)
> boston_93$HGLM_ss <- HGLM_sar$ranef[,1]
```

The **HSAR** package is restricted to the Gaussian response case, and fits an upper level SEM using MCMC; if `W=` is a lower level weights matrix, it will also fit a lower level SLM (Dong and Harris 2015; Dong et al. 2015):

```
> library(HSAR)
> suppressWarnings(HSAR <- hsar(form, data=boston_487, W=NULL,
+                                M=M, Delta=Delta, burnin=500,
+                                Nsim=2500, thinning=1))
> boston_93$HSAR_ss <- HSAR$Mus[1,]
```

Markov random field and multilevel models with spatial weights

The **R2BayesX** package provides flexible support for structured additive regression models, including spatial multilevel models. The models include an IID unstructured random effect at the upper level using the `"re"` specification (Umlauf et al. 2015); we choose the `"MCMC"`method:

```
> suppressPackageStartupMessages(library(R2BayesX))
> BX_iid <- bayesx(update(form, . ~ . + sx(NOX_ID, bs="re")),
+                  family="gaussian", data=boston_487, method="MCMC",
+                  iterations=12000, burnin=2000, step=2, seed=123)
> boston_93$BX_re <- BX_iid$effects["sx(NOX_ID):re"][[1]]$Mean
```

and the `"mrf"` (Markov random field) spatially structured random effect specification based on a graph derived from converting a suitable `"nb"` object for the upper level. The `"region.id"` attribute of the `"nb"` object needs to contain values corresponding the the indexing variable.

```
> RBX_gra <- nb2gra(nb_q_93)
> BX_mrf <- bayesx(update(form, . ~ . + sx(NOX_ID, bs="mrf",
+                                        map=RBX_gra)),
+                  family="gaussian", data=boston_487,
+                  method="MCMC", iterations=12000, burnin=2000,
+                  step=2, seed=123)
> boston_93$BX_ss <- BX_mrf$effects["sx(NOX_ID):mrf"][[1]]$Mean
```

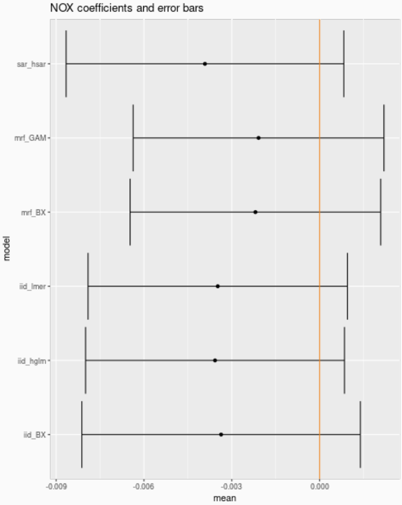
Markov random field and multilevel models with spatial weights

In a very similar way, `mgcv::gam()` can take an `"mrf"` term using a suitable `"nb"` object for the upper level. In this case the `"nb"` object needs to have the contents of the `"region.id"` attribute copied as the names of the neighbour list components, and the indexing variable needs to be a factor (Wood 2017) (the `"REML"` method of `bayesx()` gives the same result here):

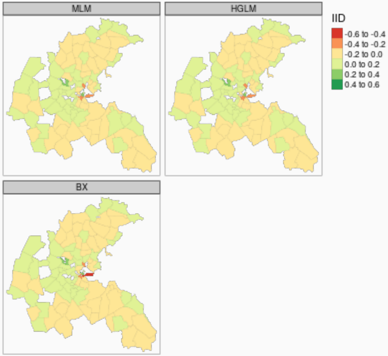
```
> library(mgcv)
> names(nb_q_93) <- attr(nb_q_93, "region.id")
> boston_487$NOX_ID <- as.factor(boston_487$NOX_ID)
> GAM_MRF <- gam(update(form, . ~ . + s(NOX_ID, bs="mrf",
+                                        xt=list(nb=nb_q_93))),
+                data=boston_487, method="REML")
> preds <- predict(GAM_MRF, type="terms", se=FALSE)[,14]
> boston_93$GAM_ss <- aggregate(preds, list(boston_487$NOX_ID),
+                               mean)$x
```

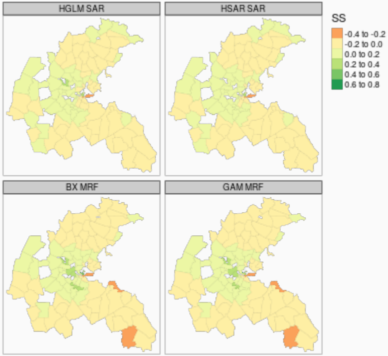
Markov random field and multilevel models with spatial weights

In the cases of `hglm()`, `bayesx()` and `gam()`, we could also model discrete responses without further major difficulty, and `bayesx()` and `gam()` also facilitate the generalization of functional form fitting for included covariates. Unfortunately, the coefficient estimates for the air pollution variable for these multilevel models are not helpful. All remain negative, but the inclusion of the model output zone level effects, be they IID or spatially structured, suggest that it hard to disentangle the influence of the scale of observation from that of covariates observed at that scale.



Markov random field and multilevel models with spatial weights





Conclusions

Conclusions i

- Progress with the **sf** and **stars** packages, including regular spatio-temporal data structures, continues; proxy access to data via APIs or cloud repositories and raster and vector tiles are of growing importance
- Work on **sf** and its use in **stars** is linked to using **Rcpp** to interface external C and C++ libraries; should this be extended to other spatial analysis packages?
- Changes in handling coordinate reference systems through PROJ are continuing, and will impact most work; web mapping depends on knowing the correct CRS of the data
- The legacy shapefile format for vector data should be discontinued as soon as possible, with binary SQLite-based GeoPackage (GPKG) a more robust choice than text-based GeoJSON or GML if PostGIS is not an option

Conclusions ii

- Visualization does not need to be web mapping; the **tmap** and **cartography** packages provide modern thematic mapping functionality; **ggplot2** also supports `"sf"` objects through `geom_sf()`
- As mentioned repeatedly, **spdep** has been split, and the stubs of modelling functions in **spdep** will shortly be defunct, avoiding confusing namespace clashes
- Should conditional permutation be added to tests for local spatial autocorrelation in **spdep**
- Comparative studies will continue with regard to **spatialreg** functionality, the next will cover Bayesian SAR and CAR models, and include the INLA `"slm"` latent model, as well as first prototypes for JAGS and STAN through R, for instance using **brms** (avoiding dense matrices is key)

Conclusions iii

- Other comparisons of spatial neighbours with graph methods, pagerank, and perhaps the Microsoft Space Partition Tree and Graph (SPTAG) algorithm are probably worth following up
- It is possible that models in **spatialreg** should move on from the `"listw"` object for spatial weights to only use sparse matrices defined in the Matrix package; timings need to be checked
- Possibly ML and MCMC fitting methods need to be unified across SLM/SEM/SAC and their Durbin variants to improve maintainability and to ease provision of unit testing
- The `"eigen"` and sparse matrix decomposition methods for computing the log determinant are adequate across the domain, but approximate methods weaken close to bounds, should we worry?

Conclusions iv

- Durbin formula valued arguments have been introduced, but have not yet been properly examined; their use in particular for dummy variables needs attention with consequences for impacts
- The Durbin interface needs to be systematised in **spatialreg** and exposed for use in other packages, especially **splm**
- Work on the **lagsarlmtree** package is needed to expose the SLM/SEM/SAC family also with Durbin terms and impacts
- An important reason for increasing attention on prediction is that it is fundamental for machine learning approaches, in which prediction for validation and test data sets drives model specification choice.

Conclusions v

- The choice of training and other data sets with dependent spatial data remains an open question for ML, and is certainly not as simple as with independent data.
- **pkgdown** documentation is in place for more packages (thanks to Angela Li): spdep and spatialreg

Aftermatter

R's sessionInfo() i

```
> sessionInfo()

## R version 3.6.0 (2019-04-26)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Fedora 29 (Workstation Edition)
##
## Matrix products: default
## BLAS:   /home/rsb/topics/R/R360-share/lib64/R/lib/libRblas.so
## LAPACK: /home/rsb/topics/R/R360-share/lib64/R/lib/libRlapack.so
##
## locale:
##  [1] LC_CTYPE=en_GB.UTF-8       LC_NUMERIC=C               LC_TIME=en_GB.UTF-8        LC_COLLATE=en_GB.UTF-8
##  [5] LC_MONETARY=en_GB.UTF-8    LC_MESSAGES=en_GB.UTF-8    LC_PAPER=en_GB.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C             LC_MEASUREMENT=en_GB.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] ggplot2_3.1.1      R2BayesX_1.1-1     mgcv_1.8-28        nlme_3.1-139       colorspace_1.4-1   BayesXsrc_3.0-1
##  [7] HSAR_0.4.2         hglm_2.2-1         hglm.data_1.0-1    MASS_7.3-51.4      MatrixModels_0.4-1 lme4_1.1-21
## [13] spatialreg_1.1-4   Matrix_1.2-17      tmap_2.2           reticulate_1.12    spdep_1.1-2        spData_0.3.0
## [19] stars_0.3-1        abind_1.4-5        sp_1.3-1           mapview_2.6.3      sf_0.7-4           osmdata_0.1.0
```

R's sessionInfo() ii

```
## [25] extrafont_0.17
##
## loaded via a namespace (and not attached):
##  [1] minqa_1.2.4        deldir_0.1-16      class_7.3-15       leaflet_2.0.2      rgdal_1.4-4        satellite_1.0.1
##  [7] base64enc_0.1-3    dichromat_2.0-0    RSpectra_0.14-0    fansi_0.4.0        lubridate_1.7.4    xml2_1.2.0
## [13] codetools_0.2-16   splines_3.6.0      knitr_1.22         jsonlite_1.6       nloptr_1.2.1       tmaptools_2.0-1
## [19] broom_0.5.2.9001   Rttf2pt1_1.3.7     png_0.1-7          rgeos_0.4-3        shiny_1.3.2        compiler_3.6.0
## [25] httr_1.4.0         backports_1.1.4    lazyeval_0.2.2     assertthat_0.2.1   cli_1.1.0          later_0.8.0
## [31] htmltools_0.3.6    tools_3.6.0        gtable_0.3.0       coda_0.19-2        glue_1.3.1         dplyr_0.8.0.1
## [37] gmodels_2.18.1     Rcpp_1.0.1         raster_2.8-19      gdata_2.18.0       extrafontdb_1.0    crosstalk_1.0.0
## [43] lmtest_0.9-37      lwgeom_0.1-7       xfun_0.6           stringr_1.4.0      ps_1.3.0           rvest_0.3.3
## [49] mime_0.6           gtools_3.8.1       XML_3.98-1.19      LearnBayes_2.15.1  zoo_1.8-5          scales_1.0.0
## [55] promises_1.0.1     parallel_3.6.0     expm_0.999-4       RColorBrewer_1.1-2 yaml_2.2.0         curl_3.3
## [61] stringi_1.4.3      highr_0.8          spDataLarge_0.3.1  e1071_1.7-1        boot_1.3-22        rlang_0.3.4
## [67] pkgconfig_2.0.2    matrixStats_0.54.0 evaluate_0.13      lattice_0.20-38    purrr_0.3.2        labeling_0.3
## [73] htmlwidgets_1.3    processx_3.3.0     tidyselect_0.2.5   plyr_1.8.4         magrittr_1.5       R6_2.4.0
## [79] generics_0.0.2     DBI_1.0.0          withr_2.1.2        pillar_1.3.1       units_0.6-3        tibble_2.1.1
## [85] crayon_1.3.4       KernSmooth_2.23-15 utf8_1.1.4         rmarkdown_1.12     grid_3.6.0         callr_3.2.0
## [91] digest_0.6.18      classInt_0.3-4     webshot_0.5.1      xtable_1.8-4       tidyr_0.8.3        httpuv_1.5.1
## [97] stats4_3.6.0       munsell_0.5.0      viridisLite_0.3.0
```

Alam, Moudud, Lars Rönnegård, and Xia Shen. 2015. "Fitting Conditional and Simultaneous Autoregressive Spatial Models in Hglm." *The R Journal* 7 (2): 5–18. https://doi.org/10.32614/RJ-2015-017.

Anselin, L. 1988. *Spatial Econometrics: Methods and Models*. Kluwer Academic Publishers.

Bivand, Roger S. 1984. "Regression Modeling with Spatial Dependence: an Application of Some Class Selection and Estimation Methods." *Geographical Analysis* 16: 25–37.

———. 2002. "Spatial Econometrics Functions in R: Classes and Methods." *Journal of Geographical Systems* 4: 405–21.

———. 2012. "After 'Raising the Bar': Applied Maximum Likelihood Estimation of Families of Models in Spatial Econometrics." *Estadística Española* 54: 71–88.

———. 2017. "Revisiting the Boston Data Set — Changing the Units of Observation Affects Estimated Willingness to Pay for Clean Air." *REGION* 4 (1): 109–27. https://doi.org/10.18335/region.v4i1.107.

Bivand, Roger S., Jan Hauke, and Tomasz Kossowski. 2013. "Computing the Jacobian in Gaussian Spatial Autoregressive Models: An Illustrated Comparison of Available Methods." *Geographical Analysis* 45 (2): 150–79. https://doi.org/10.1111/gean.12008.

Bivand, Roger S., Werner Müller, and Marcus Reder. 2009. "Power Calculations for Global and Local Moran's I." *Computational Statistics and Data Analysis* 53: 2859–72.

Bivand, Roger S., and Gianfranco Piras. 2015. "Comparing Implementations of Estimation Methods for Spatial Econometrics." *Journal of Statistical Software* 63 (1): 1–36. https://doi.org/10.18637/jss.v063.i18.

Bivand, Roger S., Zhe Sha, Liv Osland, and Ingrid Sandvig Thorsen. 2017. "A Comparison of Estimation Methods for Multilevel Models of Spatially Structured Data." *Spatial Statistics*. https://doi.org/10.1016/j.spasta.2017.01.002.

Bivand, Roger S., and David W. S. Wong. 2018. "Comparing Implementations of Global and Local Indicators of Spatial Association." *TEST* 27 (3): 716–48. https://doi.org/10.1007/s11749-018-0599-x.

Burridge, P. 1981. "Testing for a Common Factor in a Spatial Autoregression Model." *Environment and Planning A* 13: 795–800.

Cliff, A. D., and J. K. Ord. 1973. *Spatial Autocorrelation*. London: Pion.

———. 1981. *Spatial Processes*. London: Pion.

Dong, G., and R. Harris. 2015. "Spatial Autorgressive Models for Geographically Hierarchical Data Structures." *Geographical Analysis* 47 (2): 173–91.

Dong, G., R. Harris, K. Jones, and J. Yu. 2015. "Multilevel Modeling with Spatial Interaction Effects with Application to an Emerging Land Market in Beijing, China." *PLOS One* 10 (6). https://doi.org/doi:10.1371/journal.pone.0130761.

Elhorst, J. Paul. 2010. "Applied Spatial Econometrics: Raising the Bar." *Spatial Economic Analysis* 5: 9–28.

Fingleton, B. 1999. "Spurious spatial regression: Some Monte Carlo results with a spatial unit root and spatial cointegration." *Journal of Regional Science* 9: 1–19.

Goulard, Michel, Thibault Laurent, and Christine Thomas-Agnan. 2017. "About Predictions in Spatial Autoregressive Models: Optimal and Almost Optimal Strategies." *Spatial Economic Analysis* 12 (2-3). Routledge: 304–25. https://doi.org/10.1080/17421772.2017.1300679.

Gómez-Rubio, Virgilio, Roger S. Bivand, and Håvard Rue. 2015. "A New Latent Class to Fit Spatial Econometrics Models with Integrated Nested Laplace Approximations." *Procedia Environmental Sciences* 27: 116–18. https://doi.org/https://doi.org/10.1016/j.proenv.2015.07.119.

Griffith, Daniel, Roger S. Bivand, and Yongwan Chun. 2015. "Implementing Approximations to Extreme Eigenvalues and Eigenvalues of Irregular Surface Partitionings for Use in Sar and Car Models." *Procedia Environmental Sciences* 26: 119–22. https://doi.org/https://doi.org/10.1016/j.proenv.2015.05.013.

Halleck Vega, Solmaria, and J. Paul Elhorst. 2015. "The SLX Model." *Journal of Regional Science* 55 (3): 339–63. https://doi.org/10.1111/jors.12188.

Kaluzny, S.P., S.C. Vega, T.P. Cardoso, and A.A. Shelly. 1998. *S+SpatialStats*. New York, NY: Springer.

LeSage, James P., and Kelley R. Pace. 2009. *Introduction to Spatial Econometrics*. Boca Raton, FL: CRC Press.

Lovelace, Robin, Jakub Nowosad, and Jannes Muenchow. 2019. *Geocomputation with R*. Chapman; Hall/CRC. https://geocompr.robinlovelace.net/.

Martin, R. J. 2005. "Some Fast Methods for Fitting Some One-Parameter Spatial Models." *Journal of Mathematics and Statistics* 1: 326–36. https://www.thescipub.com/pdf/10.3844/jmssp.2005.326.336.

Martinetti, Davide, and Ghislain Geniaux. 2017. "Approximate Likelihood Estimation of Spatial Probit Models." *Regional Science and Urban Economics* 64: 30–45. https://doi.org/https://doi.org/10.1016/j.regsciurbeco.2017.02.002.

McMillen, D. P. 2013. *Quantile Regression for Spatial Data*. Heidelberg: Springer-Verlag.

Mur, Jesús, and Ana Angulo. 2006. "The Spatial Durbin Model and the Common Factor Tests." *Spatial Economic Analysis* 1 (2). Routledge: 207–26. https://doi.org/10.1080/17421770601009841.

Ord, J. K. 1975. “Estimation Methods for Models of Spatial Interaction.” *Journal of the American Statistical Association* 70 (349): 120–26.

Ord, J. K., and A. Getis. 2001. “Testing for Local Spatial Autocorrelation in the Presence of Global Autocorrelation.” *Journal of Regional Science* 41 (3): 411–32.

———. 2012. “Local Spatial Heteroscedasticity (LOSH).” *Annals of Regional Science* 48 (2): 529–39.

Pace, R. K., and R. P. Barry. 1997a. “Fast CARs.” *Journal of Statistical Computation and Simulation* 59 (2): 123–45.

———. 1997b. “Fast spatial estimation.” *Applied Economics Letters* 4 (5): 337–41.

———. 1997c. “Quick computation of spatial autoregressive estimators.” *Geographics Analysis* 29 (3): 232–47.

———. 1997d. "Sparse spatial autoregressions." *Statistics & Probability Letters* 33 (3): 291–97.

Pace, RK, and JP LeSage. 2008. "A Spatial Hausman Test." *Economics Letters* 101: 282–84.

Pebesma, Edzer. 2018. "Simple Features for R: Standardized Support for Spatial Vector Data." *The R Journal* 10 (1): 439–46. https://journal.r-project.org/archive/2018/RJ-2018-009/index.html.

Pebesma, Edzer, and Roger S. Bivand. 2020. *Spatial Data Science*. https://deploy-preview-20--keen-swartz-3146c4.netlify.com/.

Pebesma, Edzer, Thomas Mailund, and James Hiebert. 2016. "Measurement Units in R." *The R Journal* 8 (2): 486–94. https://doi.org/10.32614/RJ-2016-061.

Suesse, Thomas. 2018. "Marginal Maximum Likelihood Estimation of Sar Models with Missing Data." *Computational Statistics & Data Analysis* 120: 98–110. https://doi.org/https://doi.org/10.1016/j.csda.2017.11.004.

Tiefelsdorf, M. 2002. "The Saddlepoint Approximation of Moran's I and Local Moran's I_i Reference Distributions and Their Numerical Evaluation." *Geographical Analysis* 34: 187–206.

Ucar, Iñaki, Edzer Pebesma, and Arturo Azcorra. 2018. "Measurement Errors in R." *The R Journal* 10 (2): 549–57. https://journal.r-project.org/archive/2018/RJ-2018-075/index.html.

Umlauf, Nikolaus, Daniel Adler, Thomas Kneib, Stefan Lang, and Achim Zeileis. 2015. "Structured Additive Regression Models: An R Interface to BayesX." *Journal of Statistical Software* 63 (21): 1–46. http://www.jstatsoft.org/v63/i21/.

Wagner, Martin, and Achim Zeileis. 2019. "Heterogeneity and Spatial Dependence of Regional Growth in the EU: A Recursive Partitioning Approach." *German Economic Review* 20 (1): 67–82. https://doi.org/10.1111/geer.12146.

Waller, Lance A., and Carol A. Gotway. 2004. *Applied Spatial Statistics for Public Health Data*. Hoboken, NJ: John Wiley & Sons.

Wilhelm, Stefan, and Miguel Godinho de Matos. 2013. "Estimating Spatial Probit Models in R." *The R Journal* 5 (1): 130–43. https://doi.org/10.32614/RJ-2013-013.

Wood, S.N. 2017. *Generalized Additive Models: An Introduction with R*. 2nd ed. Chapman; Hall/CRC.

Xu, M., C. L. Mei, and N. Yan. 2014. "A Note on the Null Distribution of the Local Spatial Heteroscedasticity (LOSH) Statistic." *Annals of Regional Science* 52 (3): 697–710.