Lab 6: Geographically Weighted Regression

Eric Robsky Huntley

05-18-2021

In this final lab, we will change our approach: rather than applying the spatial regression models that we've been working with—recall that these treat model results as spatially invariant—we'll instead be using *geographically weighted regression*. This is a form of exploratory data analysis that allows model results to be spatially variant: in other words, for each observation or area, a small-n model run is computed over only the observation's neighborhood. This is an alternative approach to addressing spatial autocorrelation: instead of accounting for it through lagged dependent variables or lagged error terms, we account for it by assuming that only adjacent observations factor in to the performance of a given model at a given location.

The result is many small-n model runs which, taken together, will tend to produce a better fit than a standard OLS regression. However, because each model run includes a small number of observations, GWR is a poor inferential statistic.

Let's Begin…

We'll be working with the eviction data that we've been familiarizing ourselves with over the course of the last several weeks. Let's read it in, and filter our those cases where `eviction_filing_rate` is greater than or equal to 95% (these, as we've seen previously are erroneous), and where tracts have zero population.

```r
library(sf)
```

```
## Linking to GEOS 3.8.1, GDAL 3.1.4, PROJ 6.3.1
```

```r
library(RColorBrewer)
library(leaflet)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(spdep)
```

```
## Loading required package: sp

## Loading required package: spData

## To access larger datasets in this package, install the spDataLarge
## package with: `install.packages('spDataLarge',
## repos='https://nowosad.github.io/drat/', type='source')`
```

```
library(spatialreg)
```

```
## Loading required package: Matrix

## Registered S3 methods overwritten by 'spatialreg':
##   method                   from
##   residuals.stsls          spdep
##   deviance.stsls           spdep
##   coef.stsls               spdep
##   print.stsls              spdep
##   summary.stsls            spdep
##   print.summary.stsls      spdep
##   residuals.gmsar          spdep
##   deviance.gmsar           spdep
##   coef.gmsar               spdep
##   fitted.gmsar             spdep
##   print.gmsar              spdep
##   summary.gmsar            spdep
##   print.summary.gmsar      spdep
##   print.lagmess            spdep
##   summary.lagmess          spdep
##   print.summary.lagmess    spdep
##   residuals.lagmess        spdep
##   deviance.lagmess         spdep
##   coef.lagmess             spdep
##   fitted.lagmess           spdep
##   logLik.lagmess           spdep
##   fitted.SFResult          spdep
##   print.SFResult           spdep
##   fitted.ME_res            spdep
##   print.ME_res             spdep
##   print.lagImpact          spdep
##   plot.lagImpact           spdep
##   summary.lagImpact        spdep
##   HPDinterval.lagImpact    spdep
##   print.summary.lagImpact  spdep
##   print.sarlm              spdep
##   summary.sarlm            spdep
##   residuals.sarlm          spdep
##   deviance.sarlm           spdep
##   coef.sarlm               spdep
##   vcov.sarlm               spdep
##   fitted.sarlm             spdep
##   logLik.sarlm             spdep
##   anova.sarlm              spdep
##   predict.sarlm            spdep
##   print.summary.sarlm      spdep
##   print.sarlm.pred         spdep
##   as.data.frame.sarlm.pred spdep
##   residuals.spautolm       spdep
##   deviance.spautolm        spdep
##   coef.spautolm            spdep
##   fitted.spautolm          spdep
##   print.spautolm           spdep
##   summary.spautolm         spdep
##   logLik.spautolm          spdep
##   print.summary.spautolm   spdep
##   print.WXImpact           spdep
##   summary.WXImpact         spdep
##   print.summary.WXImpact   spdep
##   predict.SLX              spdep

##
## Attaching package: 'spatialreg'

## The following objects are masked from 'package:spdep':
##
##     anova.sarlm, as_dgRMatrix_listw, as_dsCMatrix_I, as_dsCMatrix_IrW,
##     as_dsTMatrix_listw, as.spam.listw, bptest.sarlm, can.be.simmed,
##     cheb_setup, coef.gmsar, coef.sarlm, coef.spautolm, coef.stsls,
##     create_WX, deviance.gmsar, deviance.sarlm, deviance.spautolm,
##     deviance.stsls, do_ldet, eigen_pre_setup, eigen_setup, eigenw,
##     errorsarlm, fitted.gmsar, fitted.ME_res, fitted.sarlm,
##     fitted.SFResult, fitted.spautolm, get.ClusterOption,
##     get.coresOption, get.mcOption, get.VerboseOption,
##     get.ZeroPolicyOption, GMargminImage, GMerrorsar, griffith_sone,
##     gstsls, Hausman.test, HPDinterval.lagImpact, impacts, intImpacts,
##     Jacobian_W, jacobianSetup, l_max, lagmess, lagsarlm, lextrB,
##     lextrS, lextrW, lmSLX, logLik.sarlm, logLik.spautolm, LR.sarlm,
##     LR1.sarlm, LR1.spautolm, LU_prepermutate_setup, LU_setup,
##     Matrix_J_setup, Matrix_setup, mcdet_setup, MCMCsamp, ME, mom_calc,
##     mom_calc_int2, moments_setup, powerWeights, predict.sarlm,
##     predict.SLX, print.gmsar, print.ME_res, print.sarlm,
##     print.sarlm.pred, print.SFResult, print.spautolm, print.stsls,
##     print.summary.gmsar, print.summary.sarlm, print.summary.spautolm,
##     print.summary.stsls, residuals.gmsar, residuals.sarlm,
##     residuals.spautolm, residuals.stsls, sacsarlm, SE_classic_setup,
##     SE_interp_setup, SE_whichMin_setup, set.ClusterOption,
##     set.coresOption, set.mcOption, set.VerboseOption,
##     set.ZeroPolicyOption, similar.listw, spam_setup, spam_update_setup,
##     SpatialFiltering, spautolm, spBreg_err, spBreg_lag, spBreg_sac,
##     stsls, subgraph_eigenw, summary.gmsar, summary.sarlm,
##     summary.spautolm, summary.stsls, trW, vcov.sarlm, Wald1.sarlm
```

```
tracts <- st_read('eviction_by_tract.geojson') %>%
  st_transform(2249) %>%
  filter(eviction_filing_rate <= 95 & population > 0)
```

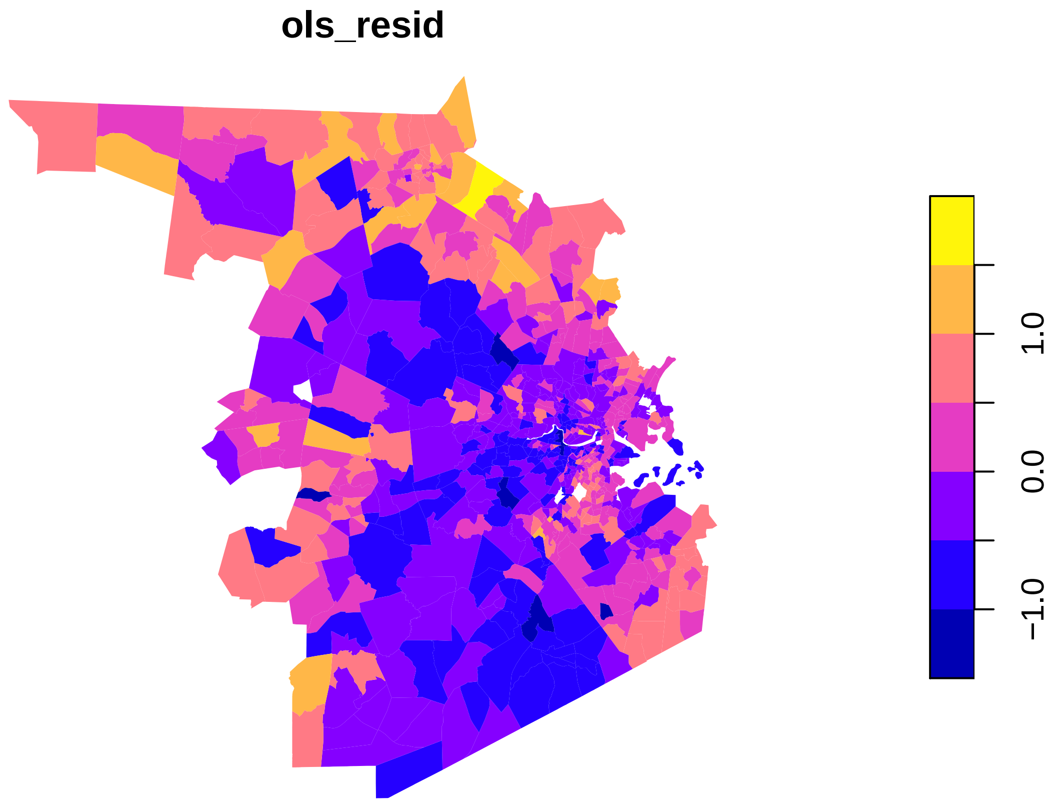
```
## Reading layer `eviction_by_tract' from data source `/Users/ehuntley/Dropbox (MIT)/11-GIS/ehuntley/11
## Simple feature collection with 650 features and 44 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -71.89872 ymin: 41.985 xmax: -70.90209 ymax: 42.73667
## Geodetic CRS:  WGS 84
```

As always, we begin by running an OLS model as a baseline. This step should be familiar to you at this point! We establish that we have spatial autocorrelated residuals (i.e., that their Moran's I).

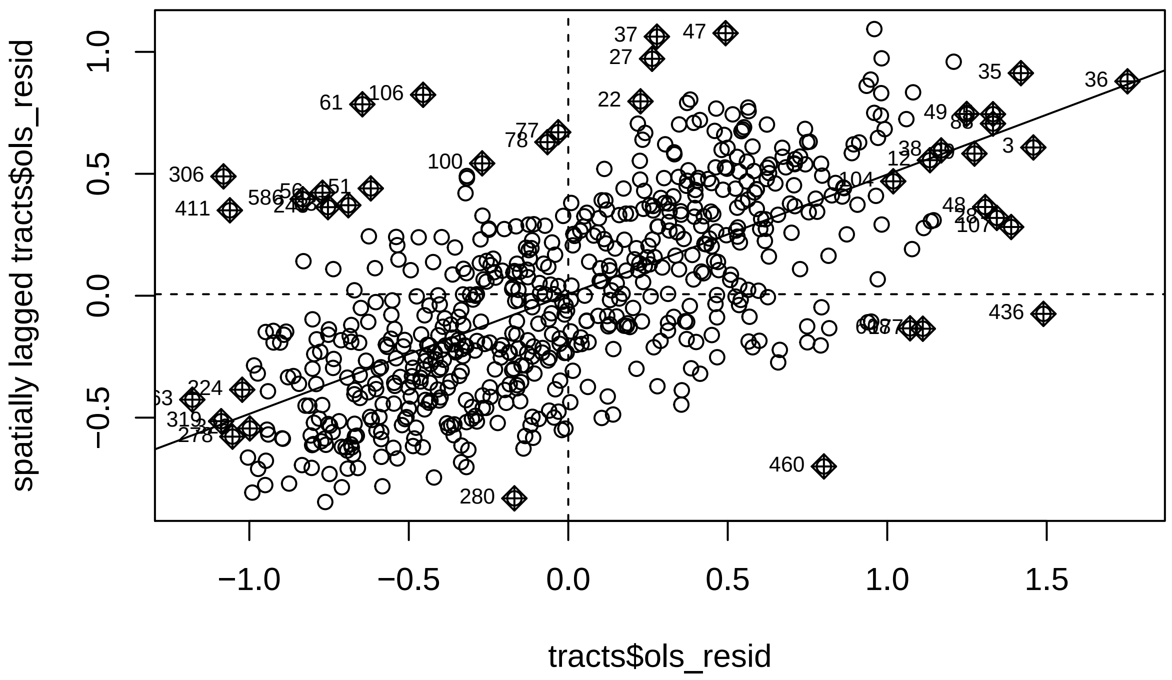
```
ols <- lm(
  formula = evic_filing_rate_log ~ pov_rate_log + pct_nonwhite_log,
  data = tracts
  )
summary(ols)
```

```
##
## Call:
## lm(formula = evic_filing_rate_log ~ pov_rate_log + pct_nonwhite_log,
##     data = tracts)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.17860 -0.42974 -0.04776  0.41141  1.75328
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -0.10474    0.09799  -1.069    0.286
## pov_rate_log      0.25389    0.02947   8.616  < 2e-16 ***
## pct_nonwhite_log  0.19105    0.03720   5.136 3.73e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5544 on 639 degrees of freedom
## Multiple R-squared:  0.2986, Adjusted R-squared:  0.2964
## F-statistic:   136 on 2 and 639 DF,  p-value: < 2.2e-16
```

```
tracts$ols_resid <- residuals(ols)
plot(tracts['ols_resid'],
     lwd=0.05,
     border=0
     )
```



```r
weights <- nb2listw(poly2nb(tracts, queen = FALSE), style="W")
moran.plot(tracts$ols_resid, weights)
```



```r
moran.test(tracts$ols_resid, weights)
```

```
##
##  Moran I test under randomisation
##
## data:  tracts$ols_resid
## weights: weights
##
## Moran I statistic standard deviate = 19.875, p-value < 2.2e-16
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##      0.4908145754     -0.0015600624      0.0006137159
```

```r
AIC(ols)
```

```
## [1] 1069.46
```

In previous weeks, we would have addressed this using a *spatial error model*—where we account for spatial autocorrelation as an error or an excluded variable by decomposing the error term—or a *spatial lag model*—where we assume that autocorrelation is a sign of spillover or proximity effects. We saw the results of this in Lab 5.

This weel, though, we will allow our model results to vary using geographically weighted regression. In R, the package supporting this method is `spgwr` It's customary here to do a cross-validation approach in order to select the bandwidth. The question we need to answer is: how big should each of our 'neighborhoods' be?

We'll answer this using leave-one-out cross validation. In this approach, a model is trained using all but one observation and we calculate the error—the difference between that excluded observation and the predicted value. These are then averaged over all observations for a given 'bandwidth', where 'bandwidth' simply means 'neighborhood size'.

```
library(spgwr)
```

```
## NOTE: This package does not constitute approval of GWR
## as a method of spatial analysis; see example(gwr)
```

```
bw <- gwr.sel(
  formula = evic_filing_rate_log ~ pov_rate_log + pct_nonwhite_log,
  data = as_Spatial(tracts)
)
```

```
## Bandwidth: 132998.2 CV score: 189.3325
## Bandwidth: 214980.7 CV score: 194.4291
## Bandwidth: 82330.17 CV score: 179.7075
## Bandwidth: 51015.63 CV score: 162.7602
## Bandwidth: 31662.17 CV score: 143.5275
## Bandwidth: 19701.08 CV score: 128.4862
## Bandwidth: 12308.72 CV score: 122.5506
## Bandwidth: 7739.992 CV score: 138.2297
## Bandwidth: 14870.95 CV score: 123.2056
## Bandwidth: 12461.61 CV score: 122.5032
## Bandwidth: 13045.17 CV score: 122.441
## Bandwidth: 13742.56 CV score: 122.5834
## Bandwidth: 12946.52 CV score: 122.439
## Bandwidth: 12957.74 CV score: 122.439
## Bandwidth: 12955.97 CV score: 122.439
## Bandwidth: 12955.94 CV score: 122.439
## Bandwidth: 12955.94 CV score: 122.439
## Bandwidth: 12955.94 CV score: 122.439
## Bandwidth: 12952.34 CV score: 122.439
## Bandwidth: 12954.57 CV score: 122.439
## Bandwidth: 12955.42 CV score: 122.439
## Bandwidth: 12955.74 CV score: 122.439
## Bandwidth: 12955.87 CV score: 122.439
## Bandwidth: 12955.91 CV score: 122.439
## Bandwidth: 12955.93 CV score: 122.439
## Bandwidth: 12955.94 CV score: 122.439
## Bandwidth: 12955.94 CV score: 122.439
## Bandwidth: 12955.94 CV score: 122.439
## Bandwidth: 12955.94 CV score: 122.439
## Bandwidth: 12955.94 CV score: 122.439
```

What we find is that a bandwidth of 12955.94 (13 kilometers) optimizes our CV score. We can then use this value in the `gwr()` function, which computes the results of our geographically weighted regression.

```
gwr_res <- gwr(
  formula = evic_filing_rate_log ~ pov_rate_log + pct_nonwhite_log,
  data = as_Spatial(tracts),
  bandwidth = bw,
  hatmatrix = TRUE
)
```

```
## Warning in proj4string(data): CRS object has comment, which is lost in output
```

These model results give us quartiles, minima and maxima for each of our coefficients: we see that, in some places, we have inverted signs.

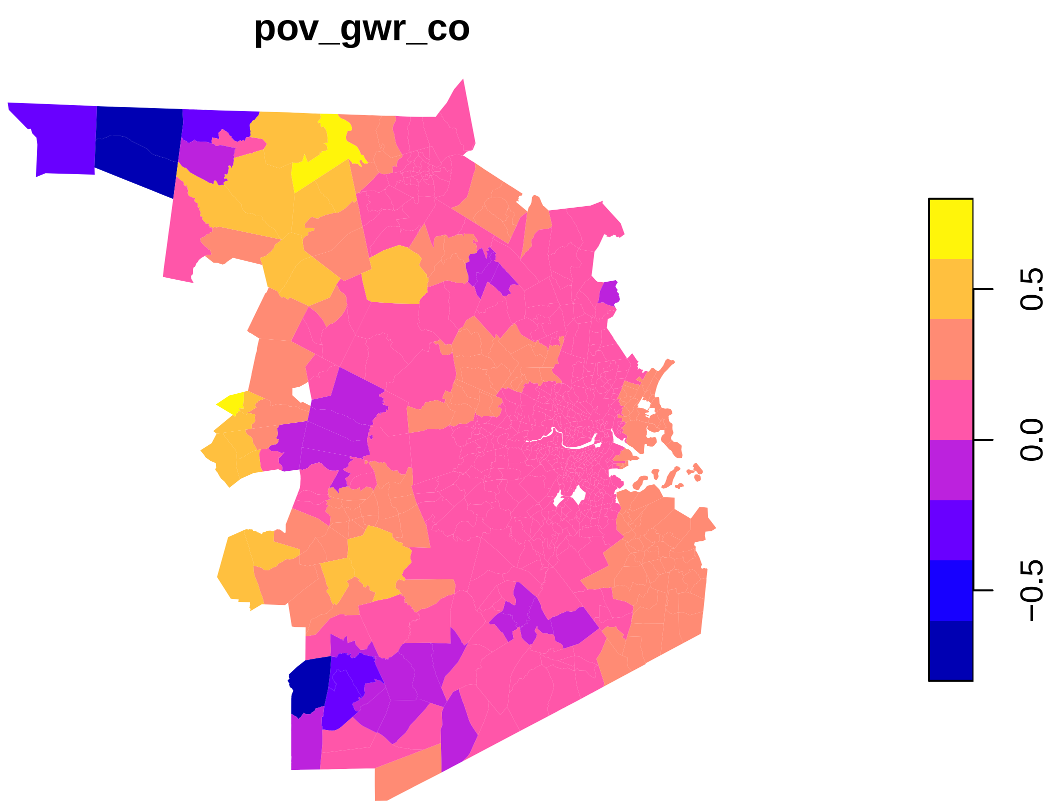
We also see substantial improvement in our R^2 value… but let's not get too excited. Keep in mind that we're generating these model runs on very small numbers of observations, so the model fit is both going to be better—precisely because of spatial autocorrelation—but it's going to be less predictive.

```
gwr_res
```

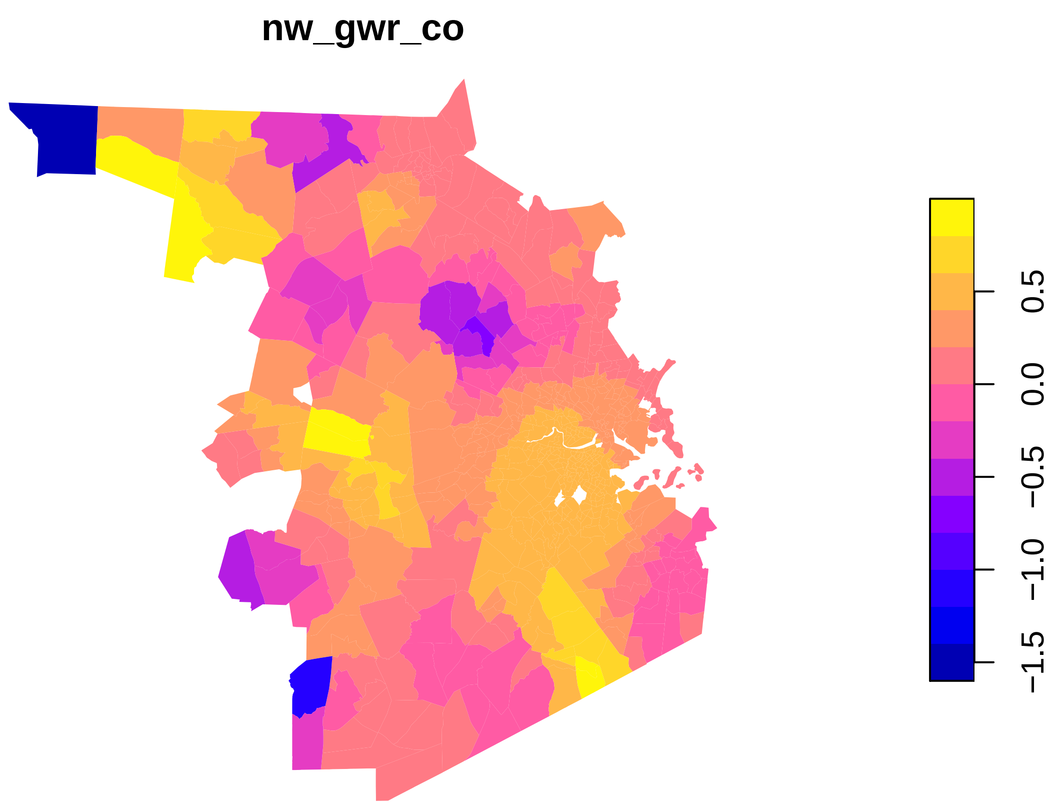
```
## Call:
## gwr(formula = evic_filing_rate_log ~ pov_rate_log + pct_nonwhite_log,
##     data = as_Spatial(tracts), bandwidth = bw, hatmatrix = TRUE)
## Kernel function: gwr.Gauss
## Fixed bandwidth: 12955.94
## Summary of GWR coefficient estimates at data points:
##                       Min.  1st Qu.   Median  3rd Qu.     Max.  Global
## X.Intercept.      -2.96645 -1.04635 -0.62835  0.24267  4.58011 -0.1047
## pov_rate_log      -0.75663  0.14911  0.17260  0.19878  0.76910  0.2539
## pct_nonwhite_log -1.51219  0.14607  0.35002  0.48536  0.93432  0.1911
## Number of data points: 642
## Effective number of parameters (residual: 2traceS - traceS'S): 110.3141
## Effective degrees of freedom (residual: 2traceS - traceS'S): 531.6859
## Sigma (residual: 2traceS - traceS'S): 0.4075331
## Effective number of parameters (model: traceS): 81.08222
## Effective degrees of freedom (model: traceS): 560.9178
## Sigma (model: traceS): 0.3967718
## Sigma (ML): 0.370871
## AICc (GWR p. 61, eq 2.33; p. 96, eq. 4.21): 736.8836
## AIC (GWR p. 96, eq. 4.22): 629.3986
## Residual sum of squares: 88.3041
## Quasi-global R2: 0.6846311
```

We can also map both our coefficients. . . .

```
tracts$pov_gwr_co <- gwr_res$SDF$pov_rate_log
plot(tracts['pov_gwr_co'],
  lwd=0.05,
  border=0
  )
```

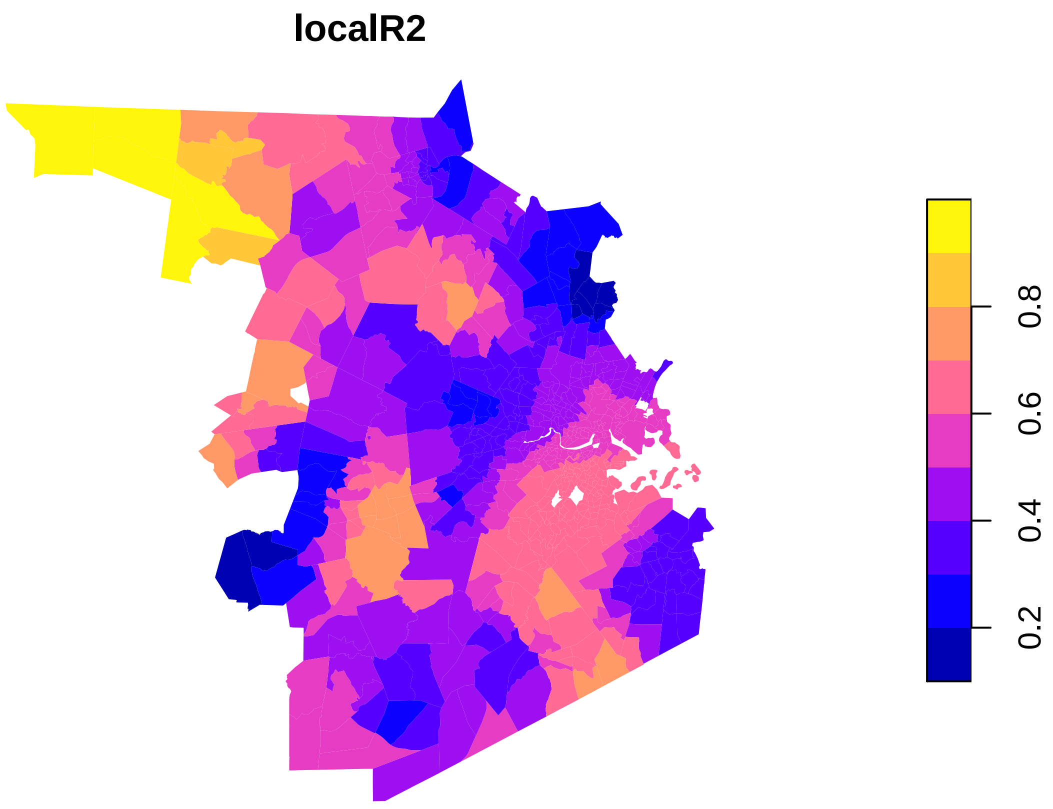


```r
tracts$nw_gwr_co <- gwr_res$SDF$pct_nonwhite_log
plot(tracts['nw_gwr_co'],
  lwd=0.05,
  border=0
  )
```



…and our local R^2 values…

```r
tracts$localR2 <- gwr_res$SDF$localR2
plot(tracts['localR2'],
     lwd=0.05,
     border=0
     )
```



We see, again, that we can begin to think spatially about our model fit—why might it be better in some places than others? Why might our independent variables fit better in some places than in others?

Using an Adaptive Kernel

Above we used a 'fixed kernel' - which is to say, we used `gwr.sel` to select an optimized bandwidth on the basis of a cross-validation method, but this bandwidth was the same for each observation.

When it comes to census geometries, though, this is dubious! Some census geometries are much, much smaller than others. This implies that a smaller bandwidth should be used for denser areas than for lower-density areas. So instead of using a constant distance, we can use a variable distance to select a constant number of observations and optimize that criteria. This looks like this...

```r
library(spgwr)

bw <- gwr.sel(
  formula = evic_filing_rate_log ~ pov_rate_log + pct_nonwhite_log,
  data = as_Spatial(tracts),
  adapt = TRUE
)
```

```
## Adaptive q: 0.381966 CV score: 174.9089
## Adaptive q: 0.618034 CV score: 181.0625
## Adaptive q: 0.236068 CV score: 169.8459
## Adaptive q: 0.145898 CV score: 162.7736
## Adaptive q: 0.09016994 CV score: 150.4255
## Adaptive q: 0.05572809 CV score: 140.8061
## Adaptive q: 0.03444185 CV score: 135.5698
## Adaptive q: 0.02128624 CV score: 132.2356
## Adaptive q: 0.01315562 CV score: 126.4222
## Adaptive q: 0.008130619 CV score: 119.9929
## Adaptive q: 0.005024999 CV score: 121.2821
## Adaptive q: 0.007573697 CV score: 119.5372
## Adaptive q: 0.007006723 CV score: 119.5185
## Adaptive q: 0.007266615 CV score: 119.4989
## Adaptive q: 0.007225925 CV score: 119.4985
## Adaptive q: 0.007185235 CV score: 119.4993
## Adaptive q: 0.007225925 CV score: 119.4985
```

We wind up selecting as optimal 0.007 (0.7%) of all tracts—the kernel will vary such that 0.7% of our 642 observations (~5) are included in each model run.

```r
gwr_res <- gwr(
  formula = evic_filing_rate_log ~ pov_rate_log + pct_nonwhite_log,
  data = as_Spatial(tracts),
  adapt = bw,
  hatmatrix = TRUE
)
```

```
## Warning in proj4string(data): CRS object has comment, which is lost in output
```

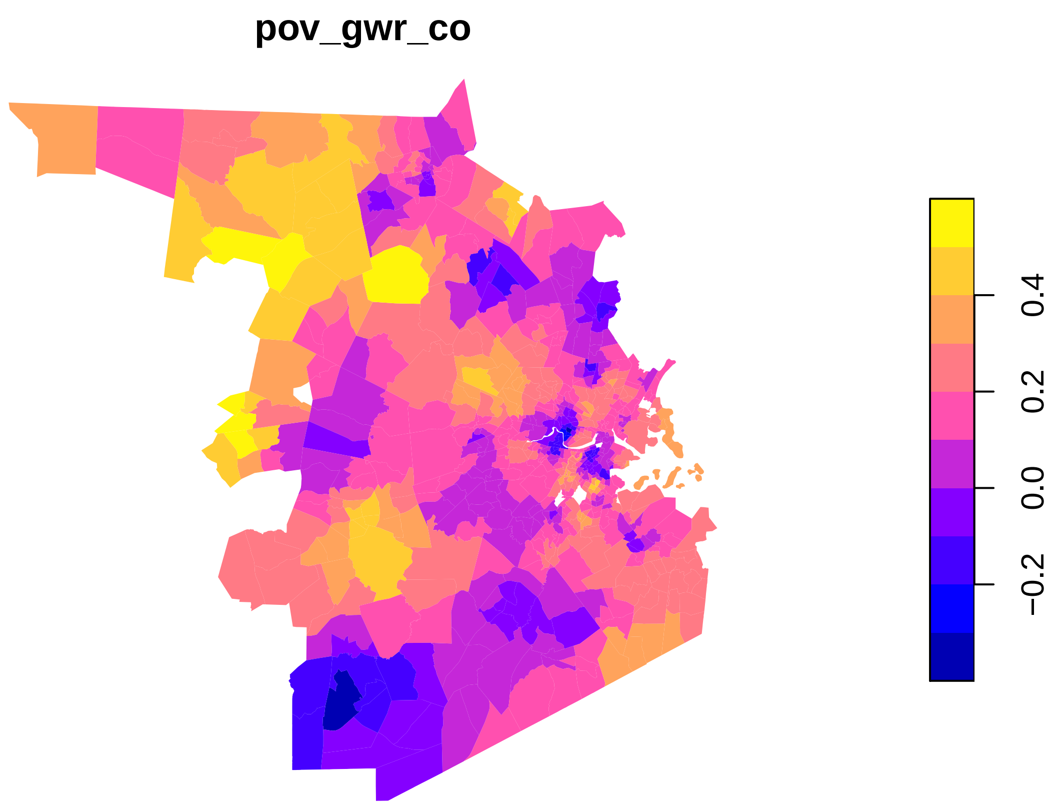
Once again, we see both inverted signs and a substantially improved R^2 (which, again, we should be skeptical of).

```r
gwr_res
```

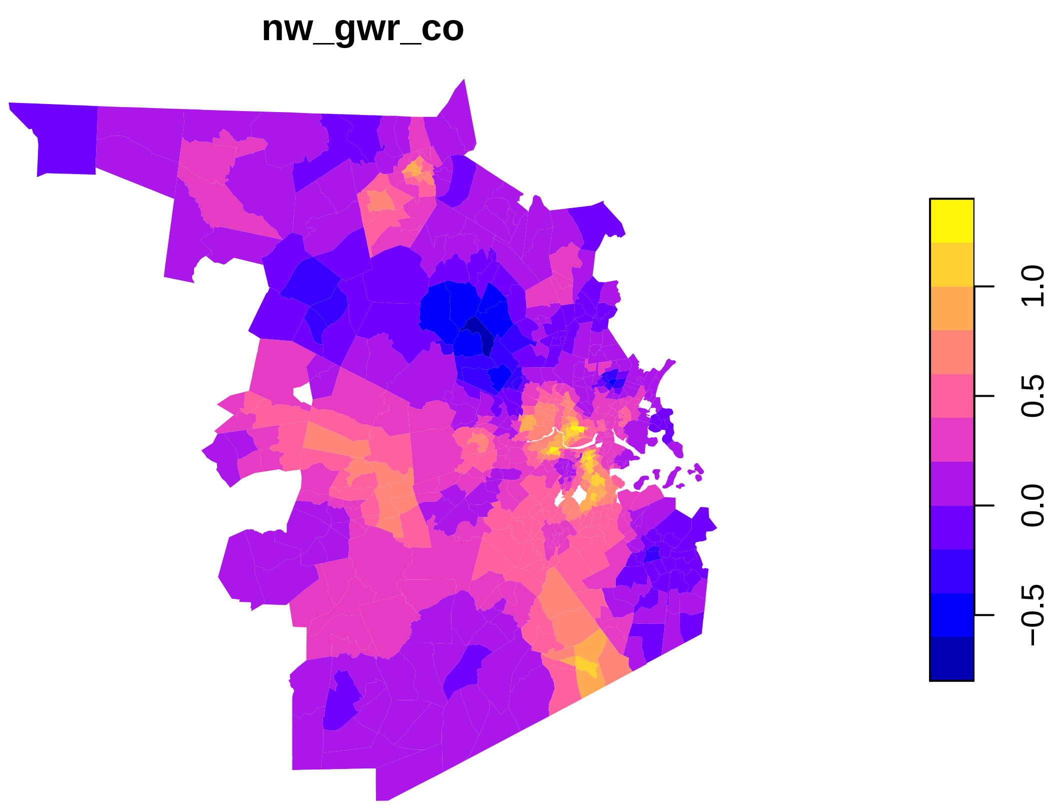
```
## Call:
## gwr(formula = evic_filing_rate_log ~ pov_rate_log + pct_nonwhite_log,
##     data = as_Spatial(tracts), adapt = bw, hatmatrix = TRUE)
## Kernel function: gwr.Gauss
## Adaptive quantile: 0.007225925 (about 4 of 642 data points)
## Summary of GWR coefficient estimates at data points:
##                        Min.   1st Qu.    Median   3rd Qu.      Max.  Global
## X.Intercept.     -4.205400 -1.132980 -0.363134  0.383192  2.299733 -0.1047
## pov_rate_log     -0.395399  0.053048  0.153555  0.230269  0.570952  0.2539
## pct_nonwhite_log -0.624656  0.102627  0.301932  0.539506  1.299618  0.1911
## Number of data points: 642
## Effective number of parameters (residual: 2traceS - traceS'S): 209.3885
## Effective degrees of freedom (residual: 2traceS - traceS'S): 432.6115
## Sigma (residual: 2traceS - traceS'S): 0.3805582
## Effective number of parameters (model: traceS): 151.5091
## Effective degrees of freedom (model: traceS): 490.4909
## Sigma (model: traceS): 0.3574001
## Sigma (ML): 0.3123939
## AICc (GWR p. 61, eq 2.33; p. 96, eq. 4.21): 728.8662
## AIC (GWR p. 96, eq. 4.22): 479.5046
## Residual sum of squares: 62.65276
## Quasi-global R2: 0.7762422
```

We can also map both our coefficients. . . .

```r
tracts$pov_gwr_co <- gwr_res$SDF$pov_rate_log
plot(tracts['pov_gwr_co'],
  lwd=0.05,
  border=0
  )
```

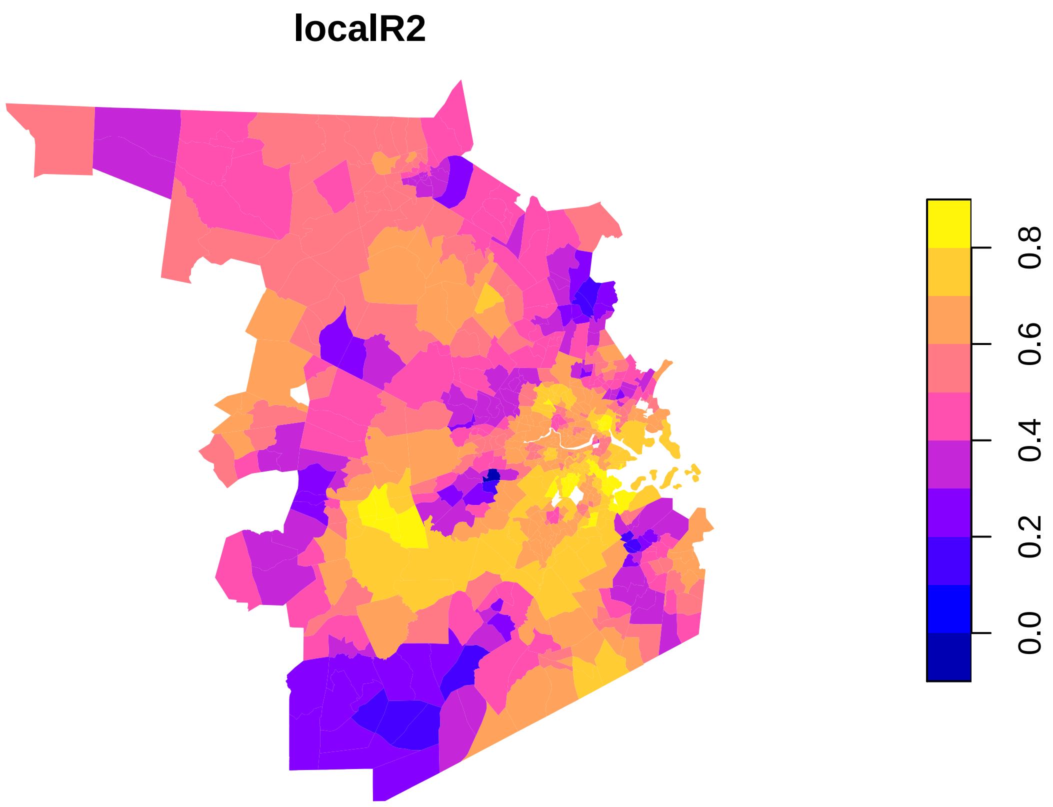


```
tracts$nw_gwr_co <- gwr_res$SDF$pct_nonwhite_log
plot(tracts['nw_gwr_co'],
  lwd=0.05,
  border=0
  )
```



. . . and our local R^2 values. . .

```
tracts$localR2 <- gwr_res$SDF$localR2
plot(tracts['localR2'],
     lwd=0.05,
     border=0
     )
```



This is