Gaussian spatial regression using the spmoran package: case study examples

Daisuke Murakami

2023/11/01

Contents

1 Introduction

This package provides functions for estimating Gaussian and non-Gaussian spatial regression models and extensions, including spatially and non-spatially varying coefficient models, models with group effects, spatial unconditional quantile regression models, and low rank spatial econometric models. All these models are estimated computationally efficiently.

An approximate Gaussian process (GP or kriging model), which is interpretable in terms of the Moran coefficient (MC), is used for modeling the spatial process. The approximate GP is defined by a linear combination of the Moran eigenvectors (MEs) corresponding to positive eigenvalue, which are known to explain positive spatial dependence. The resulting spatial process describes positively dependent map patterns (i.e., MC > 0), which are dominant in regional science (Griffith, 2003). Below, the spmoran package is used to analyse the Boston housing dataset.

The sample codes used below are available from <https://github.com/dmuraka/spmoran>. While this vignette mainly focuses on Gaussian regression modeling, another vignette focusing on non-Gaussian regression modeling and count regression modeling is also available from the same GitHub page (and Murakami 2021).

```
library(spmoran)
```

2 Gaussian spatial regression models

2.1 Basic models

This section considers the following model:

$$y_i = \sum_{k=1}^{K} x_{i,k}\beta_k + f_{MC}(s_i) + \epsilon_i, \qquad \epsilon_i \sim N(0, \sigma^2),$$

which decomposes the explained variable y_i observed at the i-th sample site into trend $\sum_{k=1}^{K} x_{i,k}\beta_{i,k}$, spatial process $f_{MC}(s_i)$ depending on location s_i, and noise ϵ_i. The spatial process is required to eliminate residual spatial dependence and estimate/infer regression coefficients β_k appropriately. ESF and RE-ESF define $f_{MC}(s_i)$ using the MC-based spatial process to efficiently eliminate residual spatial dependence. These processes are defined by the weighted sum of the Moran eigenvectors (MEs), which are spatial basis functions (distinct map pattern variables; see Griffith, 2003).

2.1.1 Eigenvector spatial filtering (ESF)

ESF specifies $f_{MC}(s_i)$ using an MC-based deterministic spatial process (see Griffith, 2003). Below is a code estimating the linear ESF model. In the code, the meigen function extracts the MEs, and the esf function estimates the model.

```
require(spdep)
data(boston)
y       <- boston.c[, "CMEDV" ]
x       <- boston.c[,c("CRIM","ZN","INDUS", "CHAS", "NOX","RM", "AGE")]
coords<- boston.c[,c("LON","LAT")]

#########Distance-based ESF
meig    <- meigen(coords=coords)
res   <- esf(y=y,x=x,meig=meig, vif=10)
res
```

```
## Call:
## esf(y = y, x = x, vif = 10, meig = meig)
##
## ----Coefficients------------------------------
##                 Estimate         SE    t_value      p_value
## (Intercept)  11.34040959 3.91692274  2.8952344 3.968277e-03
## CRIM         -0.20942091 0.03048530 -6.8695702 2.089395e-11
## ZN            0.02322000 0.01384823  1.6767492 9.426799e-02
## INDUS        -0.15063613 0.06823776 -2.2075188 2.776856e-02
## CHAS          0.15172838 0.93842988  0.1616832 8.716260e-01
## NOX         -38.02167637 4.79403898 -7.9310320 1.651338e-14
## RM            6.33316024 0.36887955 17.1686403 1.842211e-51
## AGE          -0.07820247 0.01564970 -4.9970593 8.274067e-07
##
## ----Spatial effects (residuals)----------------
##                       Estimate
## SD                   6.8540461
## Moran.I/max(Moran.I) 0.6701035
##
## ----Error statistics--------------------------
##                  stat
## resid_SE     4.476459
## adjR2        0.762328
## logLik   -1453.376154
## AIC       2996.752308
## BIC       3186.946458
```

While the meigen function is slow for large samples, it can be substituted with the meigen_f function performing a fast eigen-approximation. Here is a fast ESF code for large samples:

```r
meig_f<- meigen_f(coords)
res   <- esf(y=y, x=x, meig=meig_f,vif=10, fn="all")
```

2.1.2 Random effects ESF (RE-ESF)

RE-ESF specifies $f_{MC}(s_i)$ using an MC-based spatial random process, again to eliminate residual spatial dependence (see Murakami and Griffith, 2015). Here is a sample example:

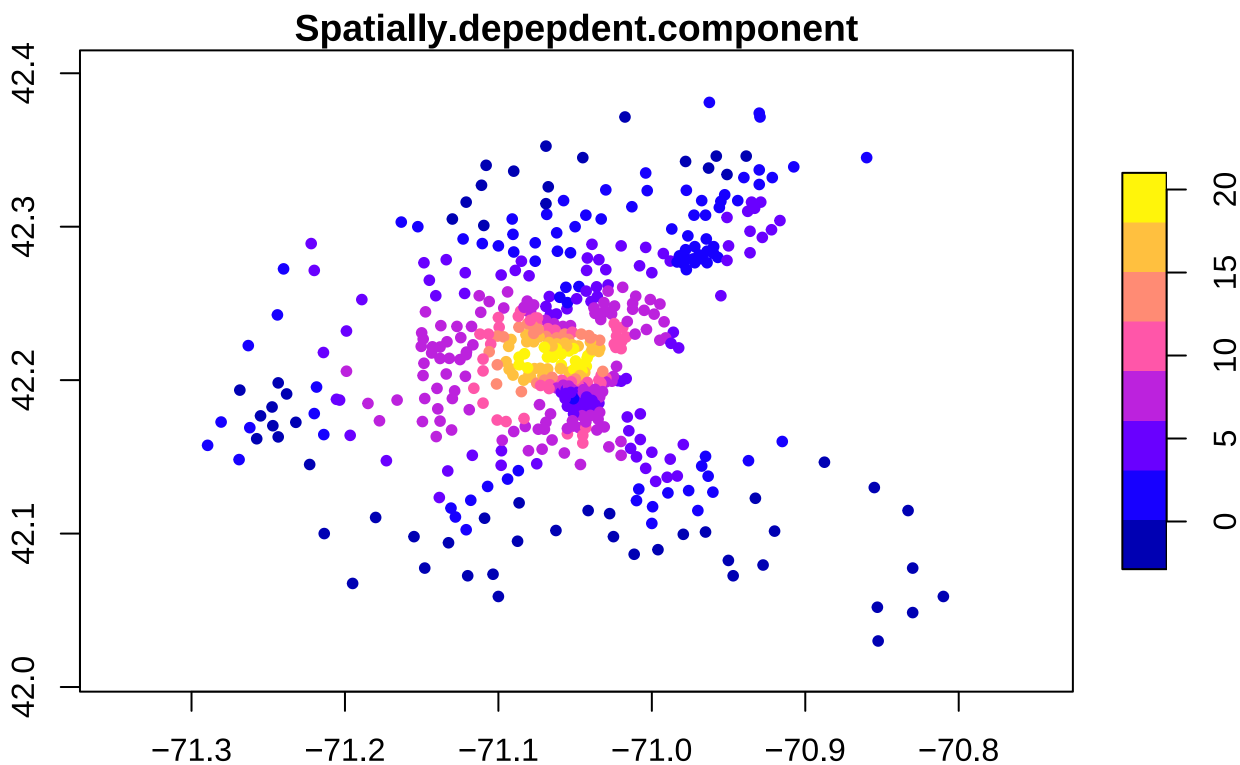
```r
res   <- resf(y = y, x = x, meig = meig)
res
```

```
## Call:
## resf(y = y, x = x, meig = meig)
##
## ----Coefficients------------------------------
##                 Estimate         SE    t_value      p_value
## (Intercept)   6.63220082 3.94484181  1.6812337 9.340120e-02
## CRIM         -0.19815202 0.03126666 -6.3374866 5.608682e-10
## ZN            0.01453736 0.01591772  0.9132813 3.615765e-01
## INDUS        -0.15560251 0.06842940 -2.2739131 2.343446e-02
## CHAS          0.51046267 0.92329947  0.5528679 5.806243e-01
## NOX         -31.26689684 5.02069107 -6.2276082 1.075128e-09
## RM            6.33993153 0.36671337 17.2885204 0.000000e+00
## AGE          -0.06351411 0.01526957 -4.1595215 3.810686e-05
##
```

```
## ----Variance parameter------------------------
##
## Spatial effects (residuals):
##                       (Intercept)
## random_SD               6.7424411
## Moran.I/max(Moran.I)    0.6648678
##
## ----Error statistics-------------------------
##                       stat
## resid_SE        4.3515212
## adjR2(cond)     0.7735912
## rlogLik     -1540.3812428
## AIC          3102.7624855
## BIC          3149.2543889
##
## NULL model: lm( y ~ x )
##    (r)loglik: -1612.825 ( AIC: 3243.65,  BIC: 3281.689 )
##
## Note: AIC and BIC are based on the restricted/marginal likelihood.
##       Use method="ml" for comparison of models with different fixed effects (x)
```

The residual spatial process $f_{MC}(s_i)$ is plotted as follows:

```
plot_s(res)
```



For large data, the meigen_f function is available again:

```
meig_f<- meigen_f(coords)
res   <- resf(y = y, x = x, meig = meig_f)
```

The meigen_f function is available for all the regression models explained below.

2.2 Extended models

2.2.1 Models with non-spatially varying coefficients (coefficients varying wrt covariate value)

Influence from covariates can vary depending on covariate value. For example, distance to railway station might have a strong impact on housing price if the distance is small, while it might be weak if the distance is large. To capture such an effect, the resf function estimates coefficients varying with respect to covariate value. I call such coefficients non-spatially varying coefficients (NVCs). If nvc=TRUE, the resf function estimates the following model considering NSVs and residual spatial dependence:

$$y_i = \sum_{k=1}^{K} x_{i,k}\beta_{i,k} + f_{MC}(s_i) + \epsilon_i, \qquad \beta_{i,k} = b_k + f(x_{i,k}), \qquad \epsilon_i \sim N(0, \sigma^2),$$

where $f(x_{i,k})$ is a smooth function of $x_{i,k}$ capturing the non-spatial influence. Here is a code estimating a spatial NVC model (with selection of constant or NVC):

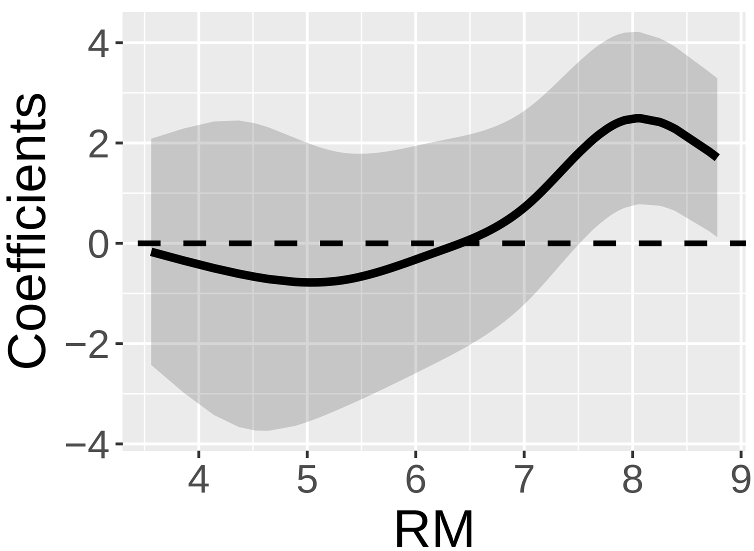
```
res   <- resf(y = y, x = x, meig = meig, nvc=TRUE)
res
```

```
## Call:
## resf(y = y, x = x, nvc = TRUE, meig = meig)
##
## ----Non-spatially varying coefficients on x (summary) ----
##
## Coefficients:
##    Intercept          CRIM                ZN               INDUS
##  Min.    :25.41   Min.   :-0.1822   Min.   :0.02042   Min.   :-0.2119
##  1st Qu.:25.41   1st Qu.:-0.1822   1st Qu.:0.02042   1st Qu.:-0.2119
##  Median :25.41   Median :-0.1822   Median :0.02042   Median :-0.2119
##  Mean   :25.41   Mean   :-0.1822   Mean   :0.02042   Mean   :-0.2119
##  3rd Qu.:25.41   3rd Qu.:-0.1822   3rd Qu.:0.02042   3rd Qu.:-0.2119
##  Max.   :25.41   Max.   :-0.1822   Max.   :0.02042   Max.   :-0.2119
##       CHAS             NOX                RM                 AGE
##  Min.   :1.375   Min.   :-0.463   Min.   :-0.78043   Min.   :-0.06742
##  1st Qu.:1.375   1st Qu.: 6.083   1st Qu.:-0.40834   1st Qu.:-0.06742
##  Median :1.375   Median : 7.792   Median :-0.16098   Median :-0.06742
##  Mean   :1.375   Mean   : 7.074   Mean   : 0.03975   Mean   :-0.06742
##  3rd Qu.:1.375   3rd Qu.: 8.654   3rd Qu.: 0.19417   3rd Qu.:-0.06742
##  Max.   :1.375   Max.   :11.517   Max.   : 2.49406   Max.   :-0.06742
##
## Statistical significance:
##                            Intercept CRIM  ZN INDUS CHAS NOX  RM AGE
## Not significant                    0    0 506     0    0 506 472   0
## Significant (10% level)            0    0   0     0  506   0   7   0
## Significant ( 5% level)            0    0   0     0    0   0  10   0
## Significant ( 1% level)          506  506   0   506    0   0  17 506
##
## ----Variance parameter------------------------
##
## Spatial effects (residuals):
##                       (Intercept)
## random_SD               3.6981527
## Moran.I/max(Moran.I)    0.4490228
##
## Non-spatial effects (coefficients on x):
```

```
##           CRIM ZN INDUS CHAS      NOX        RM AGE
## random_SD    0  0     0    0 1.850518 0.2459548   0
##
## ----Error statistics-------------------------
##                      stat
## resid_SE        3.7949128
## adjR2(cond)     0.8271073
## rlogLik     -1478.6128728
## AIC          2983.2257457
## BIC          3038.1707224
##
## NULL model: lm( y ~ x )
##    (r)loglik: -1612.825 ( AIC: 3243.65,  BIC: 3281.689 )
##
## Note: AIC and BIC are based on the restricted/marginal likelihood.
##       Use method="ml" for comparison of models with different fixed effects (x)
```

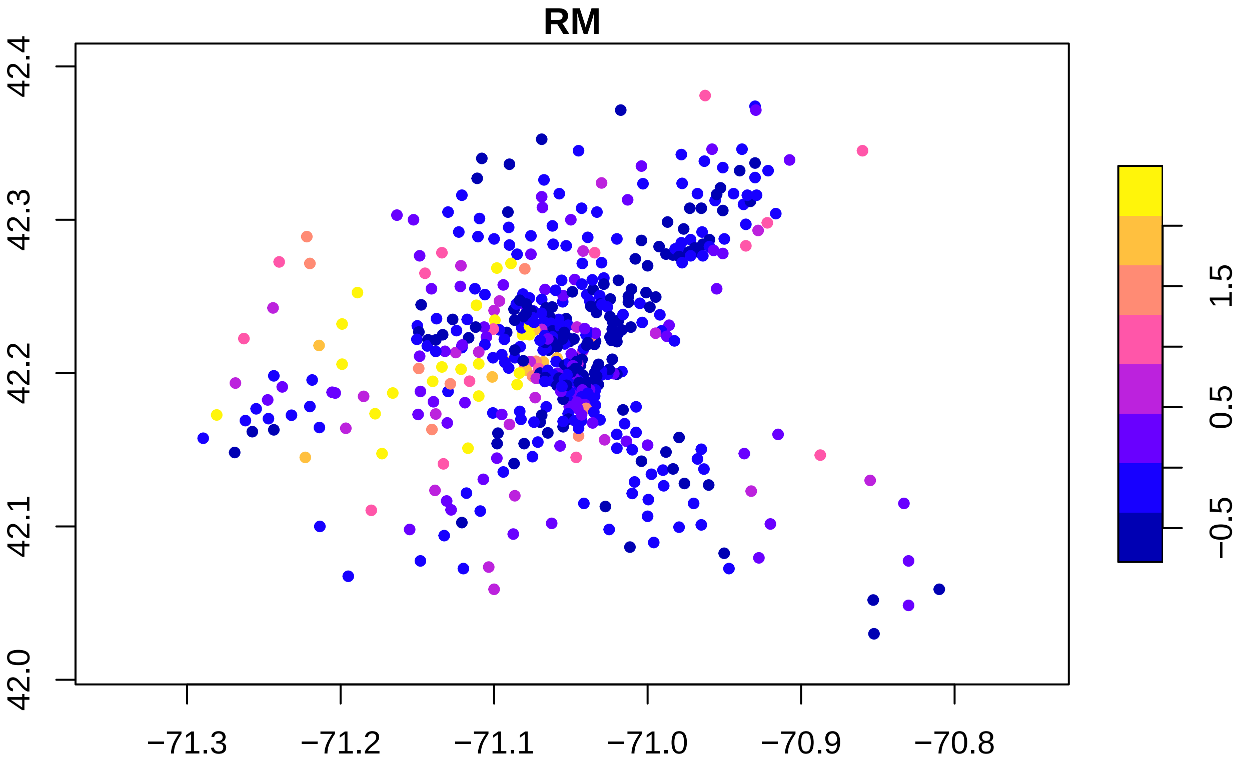
By default, this function selects constant or NVC through BIC minimization. "Non-spatially varying coefficients" in the "Variance parameter" section summarizes the estimated standard errors of the NVCs. Based on the result, coefficients on {NOX, RM} are NVCs, and coefficients on the others are constants. The NVC on RM, which is the 6-th covariate, is plotted as below. The solid line in the panel denotes the estimated NVC, and the gray area denotes the 95% confidence interval. This plot shows that RM is positively statistically significant only if RM is large.

```
plot_n(res,6)
```



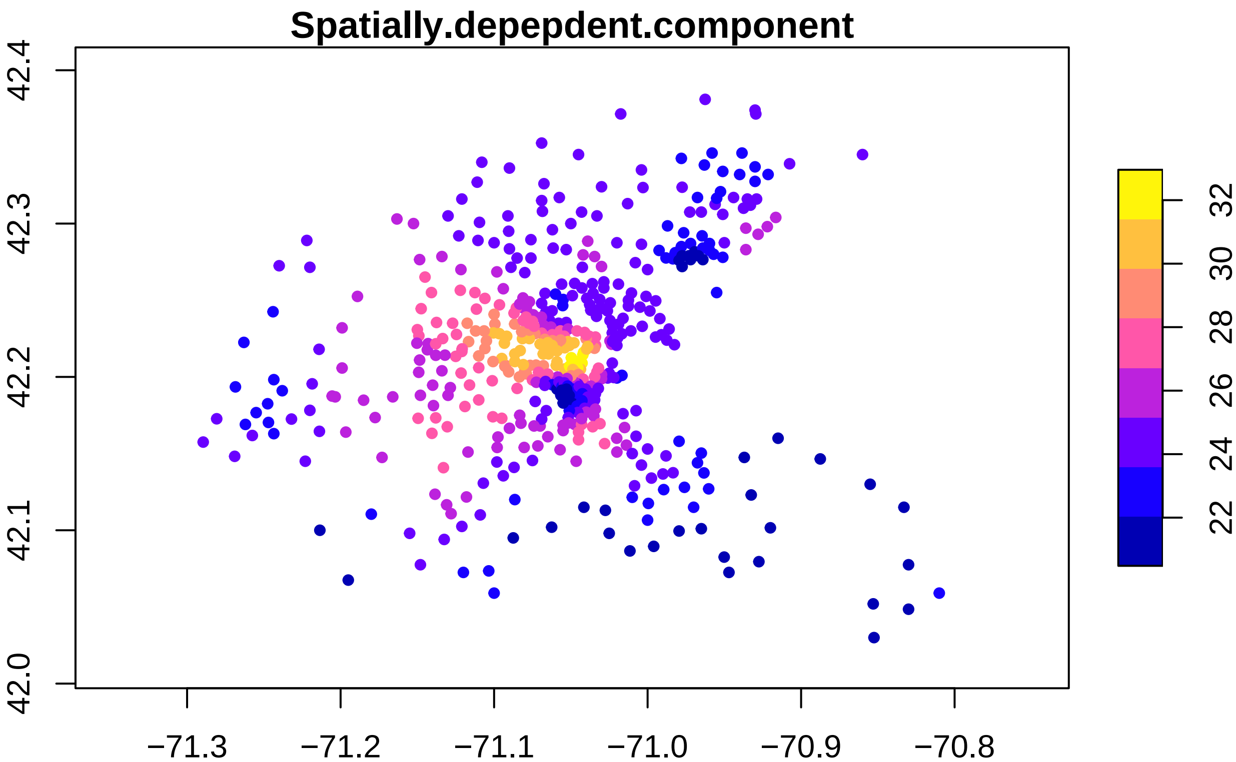
The NVC can also be spatially plotted as below:

```
plot_s(res,6)
```



On the other hand, the residual spatial process $f_{MC}(s_i)$ is plotted as

```
plot_s(res)
```



Sometimes, the user may wish to assume NVCs only on the first three covariates and constant coefficients on the others. The following code estimates such a model:

```r
res   <- resf(y = y, x = x, meig = meig, nvc=TRUE, nvc_sel=1:3)
```

2.2.2 Models with spatially varying coefficients

This package implements an ME-based spatially varying coefficient (M-SVC) model (Murakami et al., 2017), which is formulated as

$$y_i = \sum_{k=1}^{K} x_{i,k}\beta_{i,k} + f_{MC}(s_i) + \epsilon_i, \qquad \beta_{i,k} = b_k + f_{MC,k}(s_i), \qquad \epsilon_i \sim N(0, \sigma^2),$$

This model defines the k-th coefficient at site i by $\beta_{i,k}$= [constant mean b_k] + [spatially varying component $f_{MC,k}(s_i)$]. Geographically weighted regression (GWR) is known as another SVC estimation approach. Major advantages of the M-SVC modeling approach over GWR are as follows:

- The M-SVC model estimates the spatial scale (or MC value) of each SVC, while the classical GWR assumes a common scale across SVCs.
- The M-SVC model can assume SVCs on some covariates and constant coefficients on the others. This is achieved by simply assuming $\beta_{i,k} = b_k$
- This model is faster and available for very large samples. In addition, the model is free from memory limitations if the besf_vc function is used (see Section 4).
- Model selection (i.e., constant coefficient or SVC) is implemented without losing its computational efficiency.

Here is a sample code estimating an SVC model without coefficient type selection. In the code, x specifies covariates assuming SVCs, while xconst specifies covariates assuming constant coefficients. If x_sel = FALSE, the types of coefficients on x are fixed.

```r
y         <- boston.c[, "CMEDV"]
x       <- boston.c[,c("CRIM", "AGE")]
xconst  <- boston.c[,c("ZN","DIS","RAD","NOX",  "TAX","RM", "PTRATIO", "B")]
coords  <- boston.c[,c("LON","LAT")]
meig      <- meigen(coords=coords)
res     <- resf_vc(y=y,x=x,xconst=xconst,meig=meig, x_sel = FALSE )
```

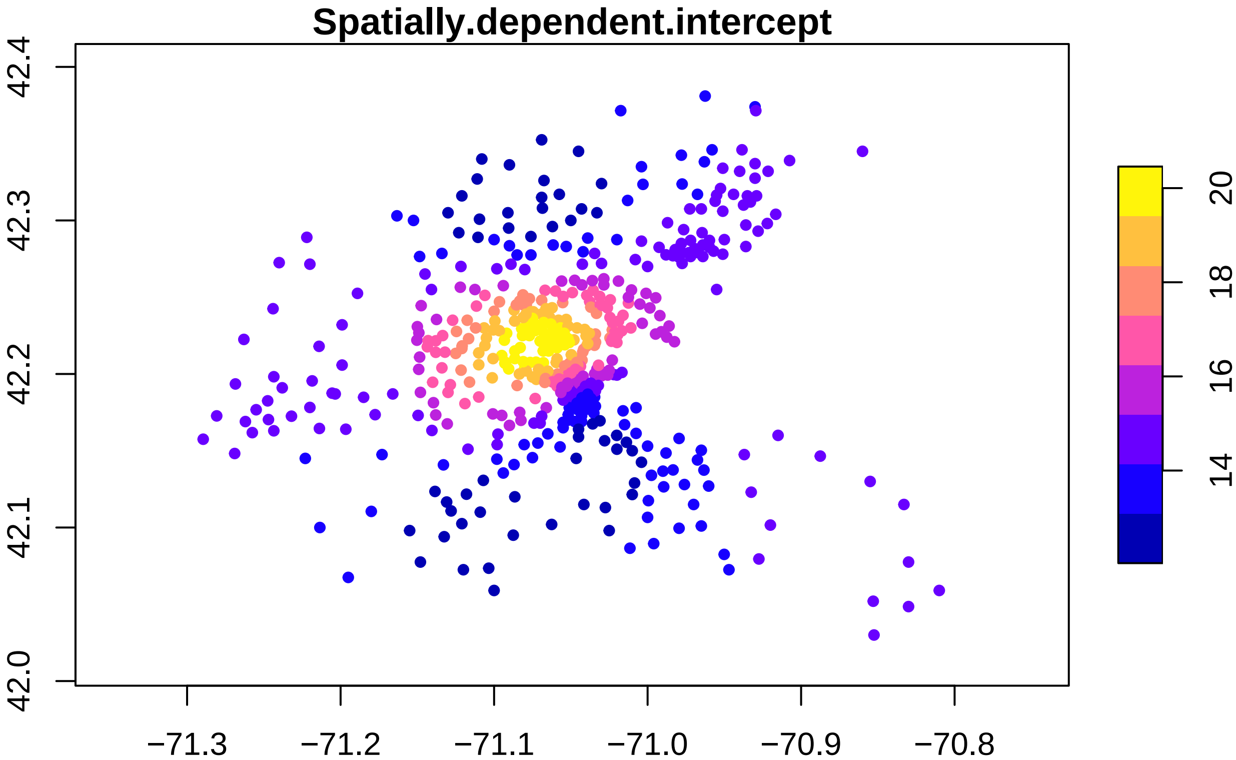
```
## [1] "-------  Iteration 1  -------"
## [1] "1/3"
## [1] "2/3"
## [1] "3/3"
## [1] "BIC: 3120.605"
## [1] "-------  Iteration 2  -------"
## [1] "1/3"
## [1] "2/3"
## [1] "3/3"
## [1] "BIC: 3114.252"
## [1] "-------  Iteration 3  -------"
## [1] "1/3"
## [1] "2/3"
## [1] "3/3"
## [1] "BIC: 3114.139"
## [1] "-------  Iteration 4  -------"
## [1] "1/3"
## [1] "2/3"
## [1] "3/3"
## [1] "BIC: 3114.138"
```

```r
res
```

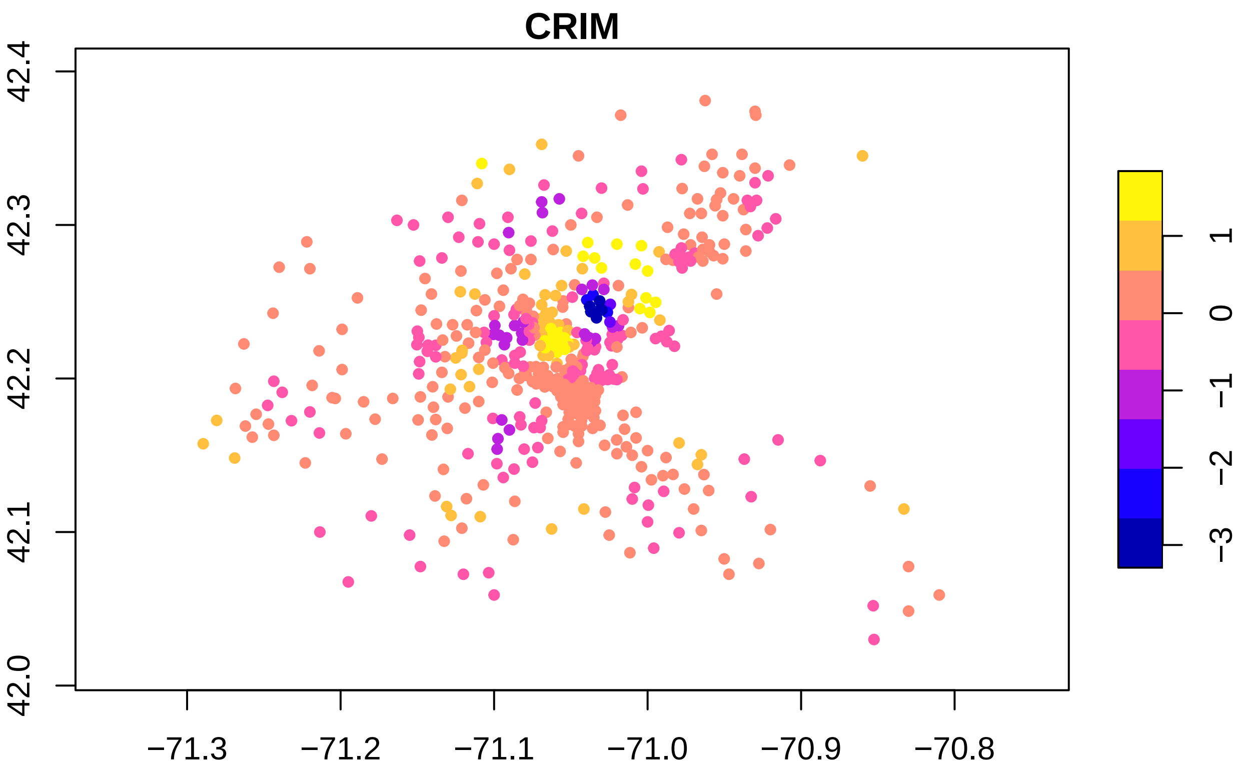
```
## Call:
## resf_vc(y = y, x = x, xconst = xconst, x_sel = FALSE, meig = meig)
##
## ----Spatially varying coefficients on x (summary)----
##
## Coefficient estimates:
##   (Intercept)         CRIM               AGE
##  Min.   :12.03   Min.   :-3.29294   Min.   :-0.14986
##  1st Qu.:13.99   1st Qu.:-0.19941   1st Qu.:-0.08377
##  Median :15.06   Median : 0.04993   Median :-0.06780
##  Mean   :15.70   Mean   : 0.05902   Mean   :-0.06582
##  3rd Qu.:17.31   3rd Qu.: 0.36587   3rd Qu.:-0.04710
##  Max.   :20.46   Max.   : 1.83866   Max.   : 0.04298
##
## Statistical significance:
##                         Intercept CRIM AGE
## Not significant                 0  416 147
## Significant (10% level)         0   27  40
## Significant ( 5% level)       190   17  99
## Significant ( 1% level)       316   46 220
##
## ----Constant coefficients on xconst----------------------------
##            Estimate          SE   t_value      p_value
## ZN        0.03202068 0.013219003  2.422322 1.582817e-02
## DIS      -1.47514930 0.334360238 -4.411856 1.292875e-05
## RAD       0.36064288 0.090818317  3.971037 8.368693e-05
## NOX     -36.21088316 5.134427150 -7.052565 6.925571e-12
## TAX      -0.01242296 0.003502523 -3.546862 4.320840e-04
## RM        6.49212566 0.326197980 19.902409 0.000000e+00
## PTRATIO  -0.52573979 0.151594626 -3.468064 5.762765e-04
## B         0.02091202 0.003094117  6.758638 4.477529e-11
##
## ----Variance parameters----------------------------------
##
## Spatial effects (coefficients on x):
##                      (Intercept)       CRIM        AGE
## random_SD              3.9039832 1.59443322 0.05746111
## Moran.I/max(Moran.I)   0.6627375 0.04502003 0.06267778
##
## ----Error statistics-------------------------------------
##                      stat
## resid_SE        3.6706778
## adjR2(cond)     0.8375658
## rlogLik     -1501.0302460
## AIC          3038.0604921
## BIC          3114.1381521
##
## NULL model: lm( y ~ x + xconst )
##    (r)loglik: -1551.857 ( AIC: 3127.715,  BIC: 3178.433 )
##
## Note: AIC and BIC are based on the restricted/marginal likelihood.
##       Use method="ml" for comparison of models with different fixed effects (x and xconst)
```

Estimated SVCs can be plotted as

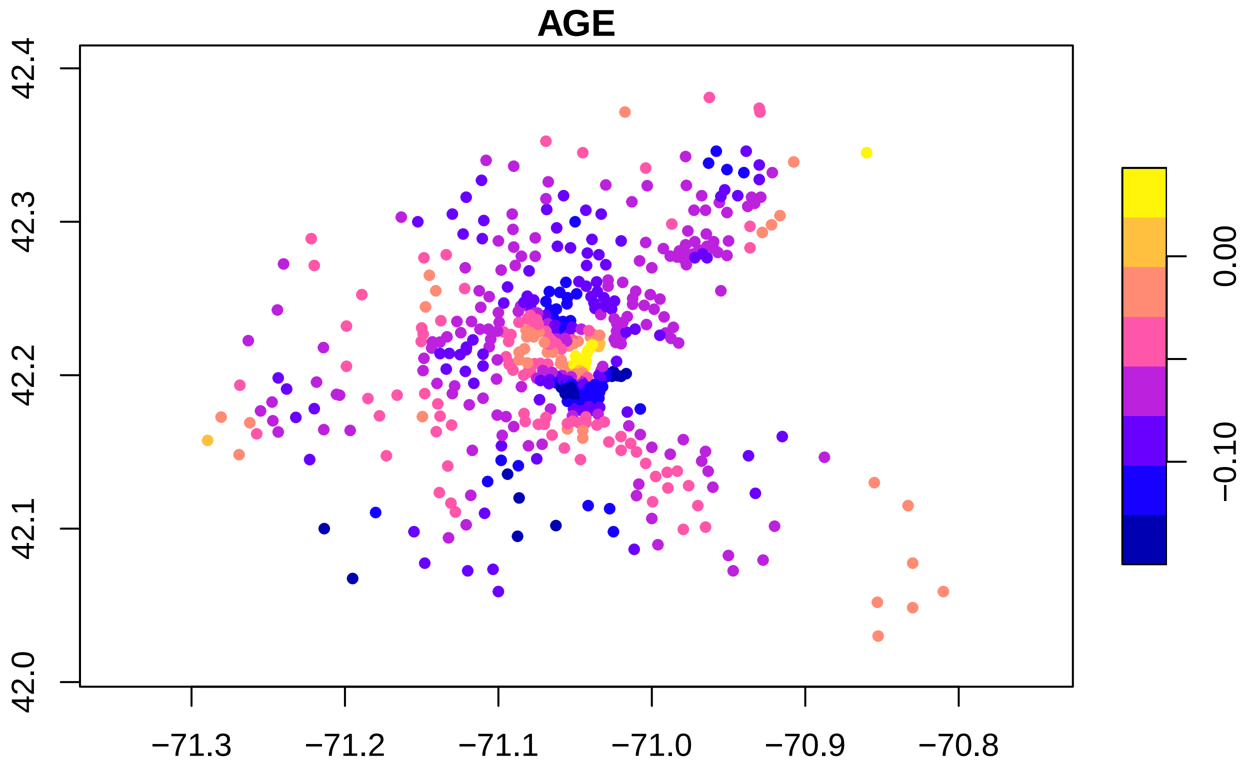
```
plot_s(res,0) # Spatially varying intercept
```




```
plot_s(res,1) # 1st SVC
```



```r
plot_s(res,2) # 2nd SVC
```



On the other hand, by default, the resf_vc function selects constant or SVCs through a BIC minimization (i.e., x_sel=TRUE by default). Here is a code:

```r
res      <- resf_vc(y=y,x=x,xconst=xconst,meig=meig )
```

2.2.3 Models with spatially and non-spatially varying coefficients

The spatially and non-spatially varying coefficient (SNVC) model is defined as

$$y_i = \sum_{k=1}^{K} x_{i,k}\beta_{i,k} + f_{MC}(s_i) + \epsilon_i, \qquad \beta_{i,k} = b_k + f_{MC,k}(s_i) + f(x_{i,k}), \qquad \epsilon_i \sim N(0, \sigma^2),$$

This model defines the k-th coefficient as $\beta_{i,k}$= [constant mean b_k] + [spatially varying component $f_{MC,k}(s_i)$] + [non-spatially varying component $f(x_{i,k})$]. Murakami and Griffith (2020) showed that, unlike SVC models that tend to be unstable owing to spurious correlation among SVCs (see Wheeler and Tiefelsdorf, 2005), this SNVC model is stable and quite robust against spurious correlations. Therefore, I recommend using the SNVC model, even if the purpose of the analysis is estimating SVCs.

An SNVC model is estimated by specifying x_nvc = TRUE in the resf_vc function as follows:

```r
res    <- resf_vc(y=y,x=x,xconst=xconst,meig=meig, x_nvc =TRUE)
```

This model assumes SNVC on x and constant coefficients on xconst. By default, the coefficient type (SNVC, SVC, NVC, or constant) on x is selected.

It is also possible to assume SNVCs on x and NVCs on xcnost by specifying xconst_nvc = TRUE as follows:

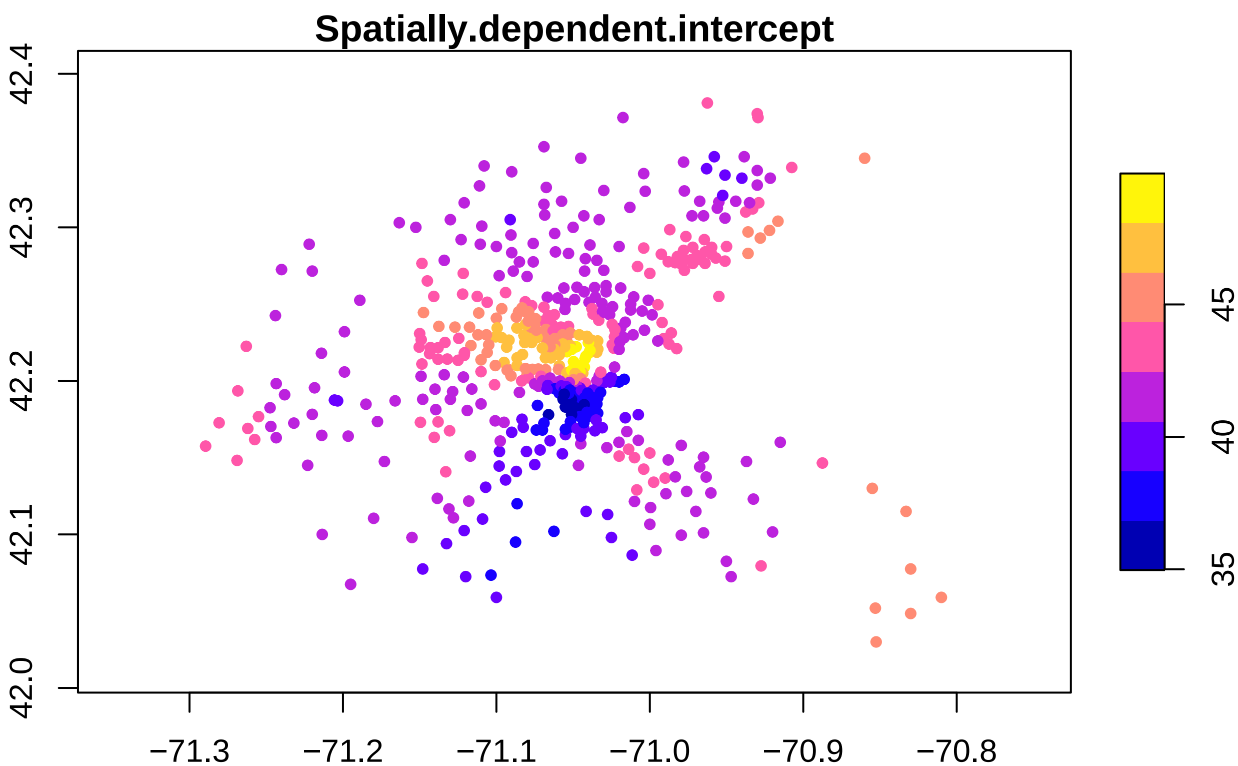
```r
res    <- resf_vc(y=y,x=x,xconst=xconst,meig=meig, x_nvc =TRUE, xconst_nvc=TRUE)
```

```
## [1] "-------  Iteration 1  -------"
## [1] "1/13"
## [1] "2/13"
## [1] "3/13"
## [1] "4/13"
## [1] "5/13"
## [1] "7/13"
## [1] "8/13"
## [1] "9/13"
## [1] "10/13"
## [1] "11/13"
## [1] "12/13"
## [1] "13/13"
## [1] "BIC: 3023.362"
## [1] "-------  Iteration 2  -------"
## [1] "1/13"
## [1] "2/13"
## [1] "3/13"
## [1] "4/13"
## [1] "5/13"
## [1] "7/13"
## [1] "8/13"
## [1] "9/13"
## [1] "10/13"
## [1] "11/13"
## [1] "12/13"
## [1] "13/13"
## [1] "BIC: 3013.007"
## [1] "-------  Iteration 3  -------"
## [1] "1/13"
## [1] "2/13"
## [1] "3/13"
## [1] "4/13"
## [1] "5/13"
## [1] "7/13"
## [1] "8/13"
## [1] "9/13"
## [1] "10/13"
## [1] "11/13"
## [1] "12/13"
## [1] "13/13"
## [1] "BIC: 3012.859"
## [1] "-------  Iteration 4  -------"
## [1] "1/13"
## [1] "2/13"
## [1] "3/13"
## [1] "4/13"
## [1] "5/13"
## [1] "7/13"
## [1] "8/13"
## [1] "9/13"
## [1] "10/13"
## [1] "11/13"
## [1] "12/13"
```

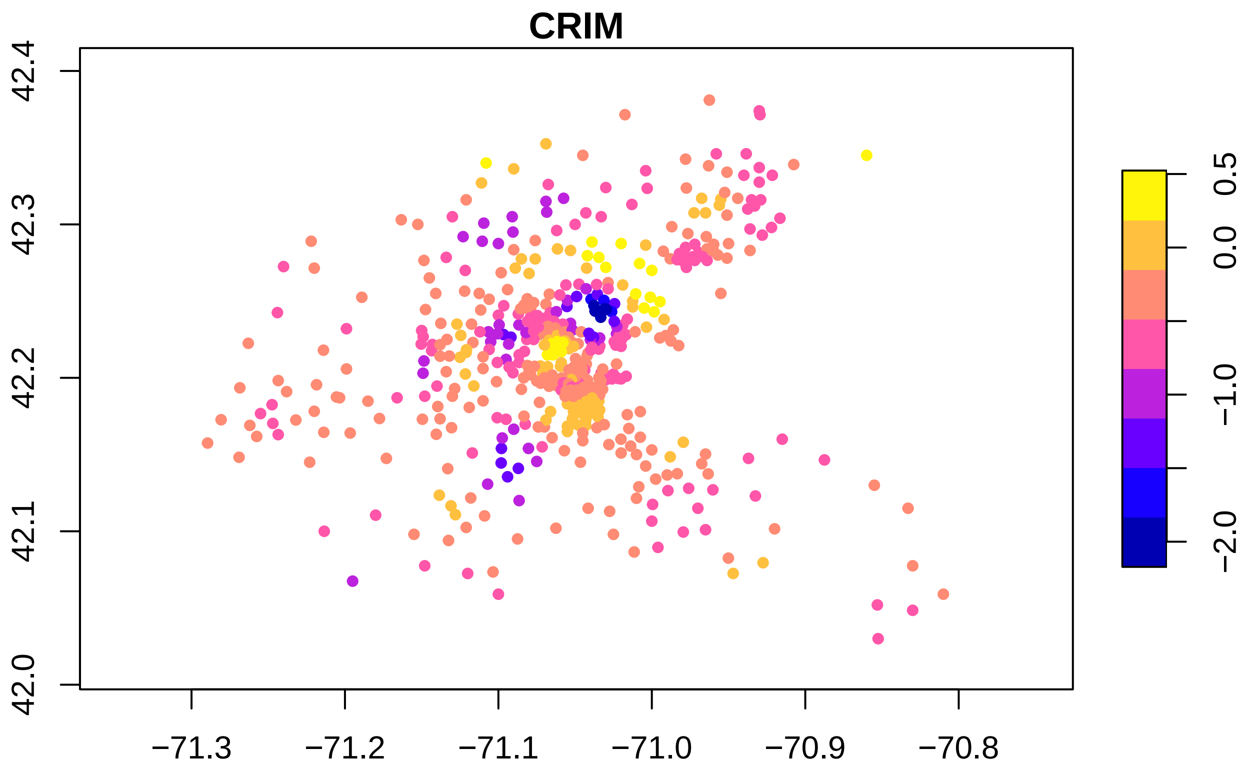
```
## [1] "13/13"
## [1] "BIC: 3012.857"
## [1] "-------  Iteration 5  -------"
## [1] "1/13"
## [1] "2/13"
## [1] "3/13"
## [1] "4/13"
## [1] "5/13"
## [1] "7/13"
## [1] "8/13"
## [1] "9/13"
## [1] "10/13"
## [1] "11/13"
## [1] "12/13"
## [1] "13/13"
## [1] "BIC: 3012.857"
```

By default, the coefficient type (SNVC, SVC, NVC, or constant) on x and those (NVC or const) on xconst are selected. The estimated SNVCs are plotted as follows:

```r
plot_s(res,0)           # Spatially varying intercept
```

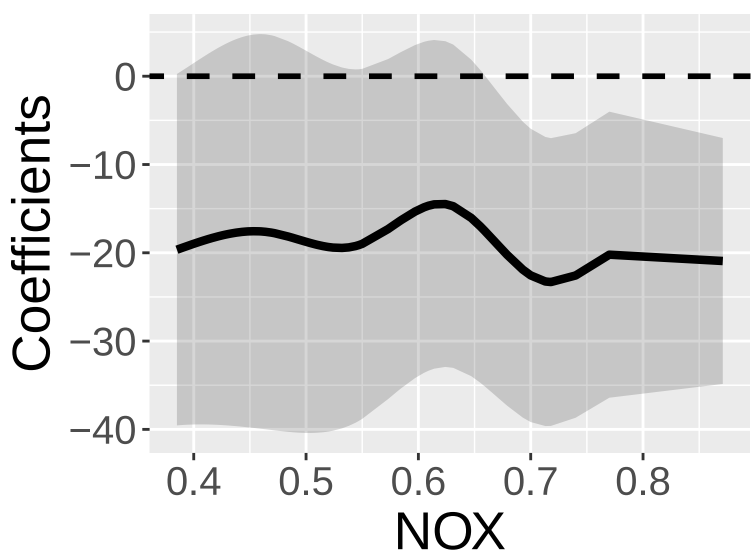



```r
plot_s(res,1)           # SNVC on x[,1]
```

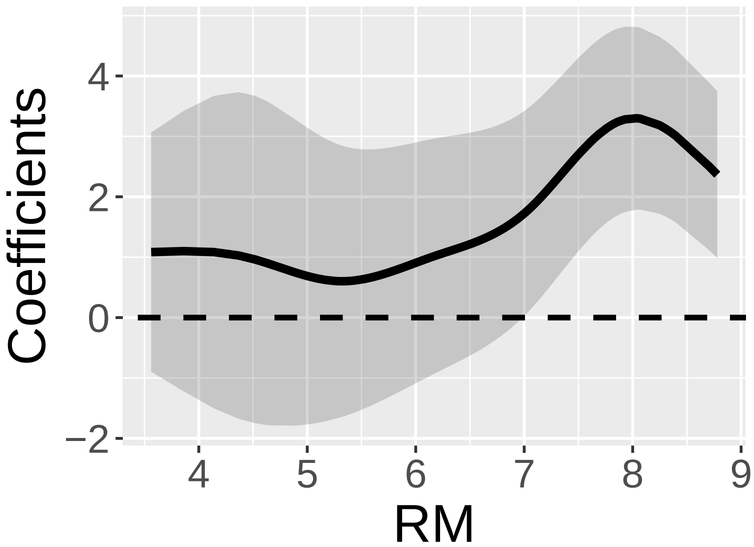


NVCs on xconst is plotted by specifying xtype="xconst" in the plot_n function, as below. The solid line denotes the estimated NVC, and the gray area denotes the 95% confidence interval:

```
plot_n(res,4,xtype="xconst")#NVC on xconst[,4]
```

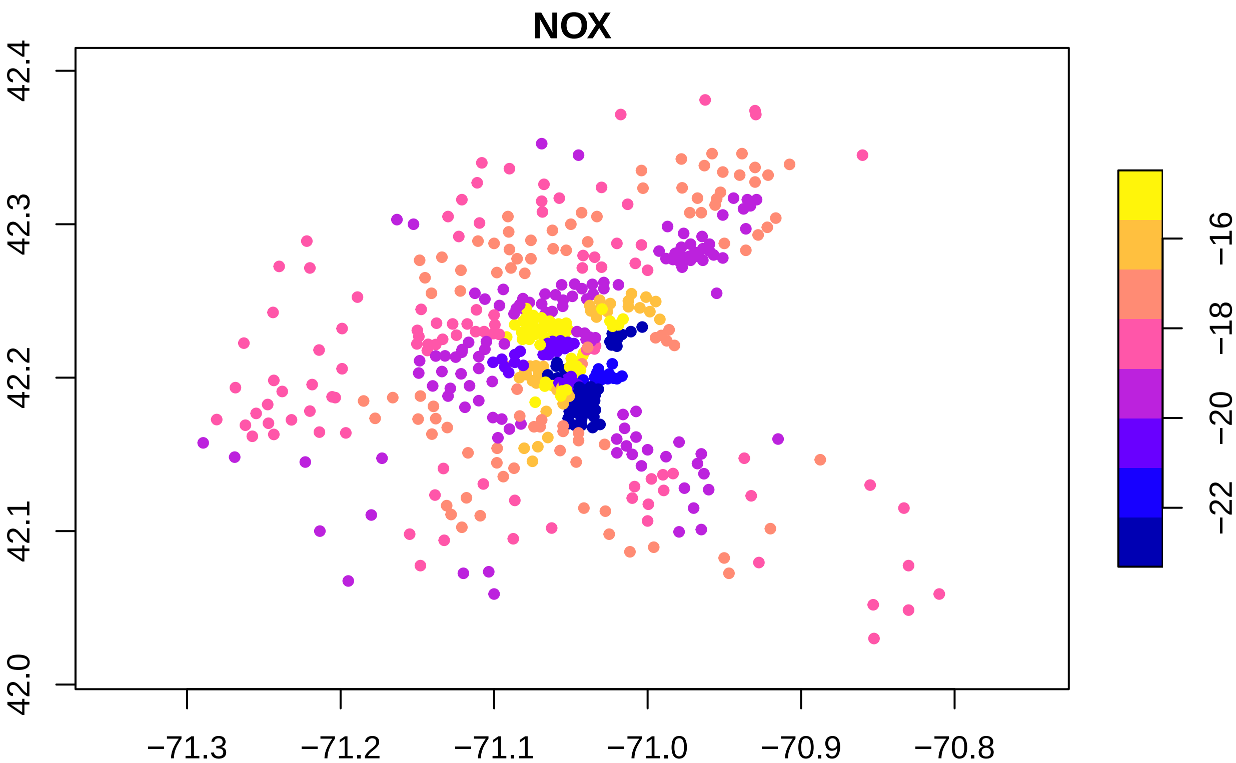



```
plot_n(res,6,xtype="xconst")#NVC on xconst[,6]
```

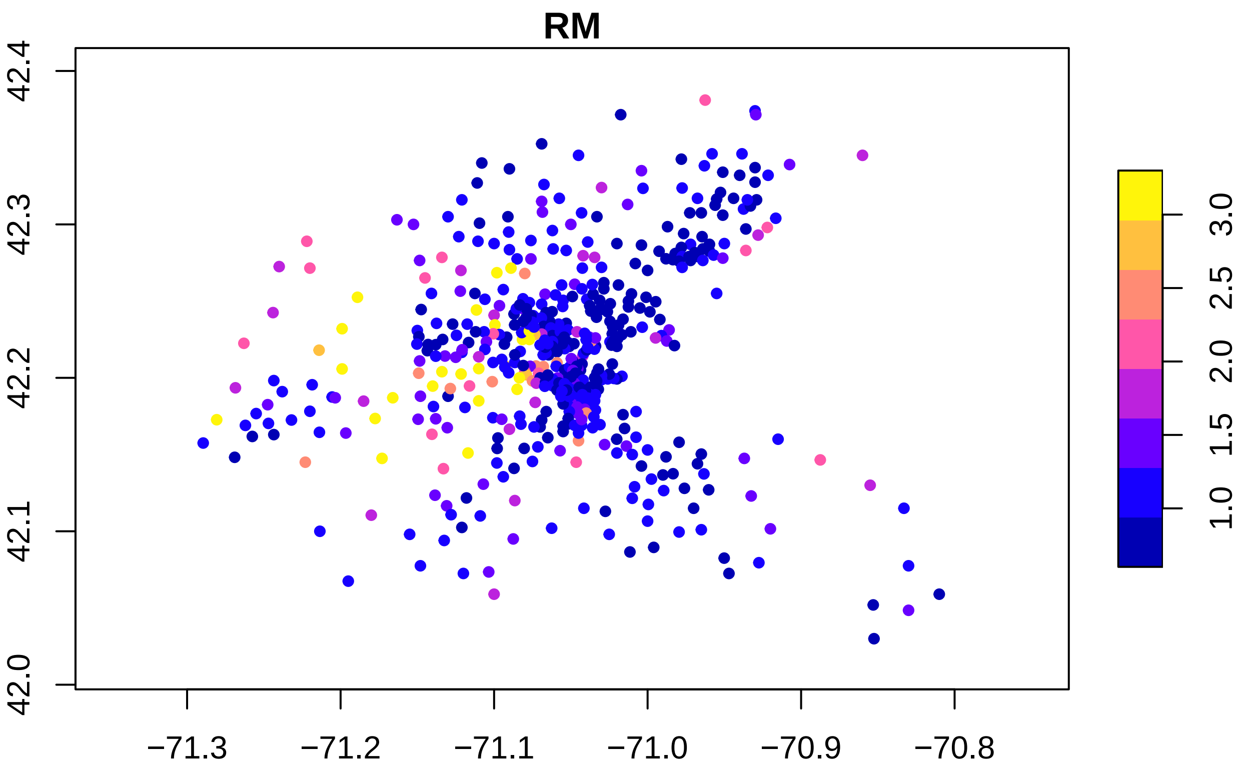


These NVCs can also be plotted spatially as follows:

```
plot_s(res,4,xtype="xconst")#Spatial plot of NVC on xconst[,4]
```




```
plot_s(res,6,xtype="xconst")#Spatial plot of NVC on xconst[,6]
```



2.2.4 Models with group effects

2.2.4.1 Outline Two group effects are available in this package:

1. Spatially dependent group effects. Spatial dependence among groups is modeled instead of modeling spatial dependence among individuals.
2. Spatially independent group effects assuming independence across groups (usual group effects)

They are estimated in the resf and resf_vc functions. When considering both these effects, the resf function estimates the following model (if no NVC is assumed):

$$y_i = \sum_{k=1}^{K} x_{i,k}\beta_k + f_{MC}(g_{I(0)}) + \sum_{h=1}^{H} \gamma(g_{I(h)}) + \epsilon_i, \qquad \epsilon_i \sim N(0, \sigma^2),$$

where $g_{I(0)}, g_{I(1)}, \ldots, g_{I(H)}$ represent group variables. $f_{MC}(g_{I(0)})$ denotes spatially dependent group effects, while $\gamma(g_{I(h)})$ denotes spatially independent group effects for the h-th group variable. On the other hand, the resf_vc function can estimate the following model considering these two effects (again, no NVC is assumed):

$$y_i = \sum_{k=1}^{K} x_{i,k}\beta_{i,k} + f_{MC}(g_{I(0)}) + \sum_{h=1}^{H} \gamma(g_{I(h)}) + \epsilon_i, \qquad \beta_{i,k} = b_k + f_{MC,k}(g_{i(0)}), \qquad \epsilon_i \sim N(0, \sigma^2),$$

Below, multilevel modeling, small area estimation, and panel data analysis are demonstrated.

2.2.4.2 Multilevel model Data often have a multilevel structure. For example, the school achievement of individual students changes depending on the class and school. A condominium unit price depends, not only on unit attributes, but also on building attributes. Multilevel modeling is required to explicitly consider the multilevel structure behind data and perform spatial regressions.

This section demonstrates the modeling considering the two group effects using the resf function. The data used are the Boston housing datasets that consist of 506 samples in 92 towns, which are regarded as groups.

To model spatially dependent group effects, Moran eigenvectors are defined by groups. This is done by specifying s_id in the meigen function using a group variable, which is the town name (TOWNNO), in this case, as follows:

```
xgroup<- boston.c[,"TOWNNO"]
coords<- boston.c[,c("LON","LAT")]
meig_g<- meigen(coords=coords, s_id=xgroup)
```

When additionally estimating spatially independent group effects, the user needs to specify xgroup in the resf function by one or more group variables, as follows:

```
x       <- boston.c[,c("CRIM","ZN","INDUS", "CHAS", "NOX","RM", "AGE")]
res    <- resf(y = y, x = x, meig = meig_g, xgroup = xgroup)
res
```

```
## Call:
## resf(y = y, x = x, xgroup = xgroup, meig = meig_g)
##
## ----Coefficients------------------------------
##                  Estimate         SE    t_value      p_value
## (Intercept)  -0.81545943 3.23135854 -0.2523581 8.008871e-01
## CRIM         -0.04596392 0.02505503 -1.8345188 6.728064e-02
## ZN            0.03285021 0.02313784  1.4197611 1.564153e-01
## INDUS         0.03549188 0.11980486  0.2962474 7.671869e-01
## CHAS         -0.62561231 0.72381491 -0.8643264 3.878995e-01
## NOX         -26.38632671 3.88238119 -6.7964286 3.668488e-11
## RM            6.30273567 0.29409796 21.4307357 0.000000e+00
## AGE          -0.06730232 0.01048068 -6.4215611 3.637544e-10
##
## ----Variance parameter------------------------
##
## Spatial effects (residuals):
##                       (Intercept)
## random_SD                5.074794
## Moran.I/max(Moran.I)     0.812936
##
## Group effects:
##           xgroup
## ramdom_SD 4.4404
##
## ----Error statistics--------------------------
##                       stat
## resid_SE         3.2429178
## adjR2(cond)      0.8740022
## rlogLik      -1465.8457138
## AIC           2955.6914276
## BIC           3006.4098677
##
## NULL model: lm( y ~ x )
##    (r)loglik: -1612.825 ( AIC: 3243.65,  BIC: 3281.689 )
##
## Note: AIC and BIC are based on the restricted/marginal likelihood.
##       Use method="ml" for comparison of models with different fixed effects (x)
```

The estimated independent group effects are extracted as

```r
res$b_g[[1]][1:5,]# Estimates in the first 5 groups
```

```
##           Estimate       SE   t_value
## xgroup_0 2.165726 2.061093 1.0507657
## xgroup_1 3.747633 1.783543 2.1012294
## xgroup_2 6.544205 1.659184 3.9442318
## xgroup_3 2.431558 1.431325 1.6988163
## xgroup_4 1.036033 1.181672 0.8767521
```

2.2.4.3 Small area estimation Small area estimation (SAE; Ghosh and Rao, 1994) is a statistical technique estimating parameters for small areas such as districts and municipality. SAE is useful for obtaining reliable small area statistics from noisy data. The resf and resf_vc functions are available for SEA (see as explained in Murakami 2020 for further detail).

The Boston housing datasets consist of 506 samples in 92 towns. This section estimates the standard housing price in the I-th towns by assuming the following model:

$$y_I = \hat{y}_I + \epsilon_I, \qquad \epsilon_I \sim N(0, \frac{\sigma^2}{N_I})$$

where $\hat{y}_I = \sum_{i=1}^{N_I} \frac{\hat{y}_i}{N_I}$. This model decomposes the observed mean house price y_I in the I-th town into the standard price $\hat{y}_I$ and noise ϵ_I, which reduces as the number of samples in the I-th town increases. The standard price is defined by an aggregate of the predictors $\hat{y}_i$ by individuals.

The above equation suggests that, if $\hat{y}_i$ is predicted using the resf or resf_vc function and aggregated into the towns, we can estimate the standard house price. Here is a sample code for the individual level prediction:

```r
r_res <-resf(y=y, x=x, meig=meig_g, xgroup=xgroup)
pred  <-predict0(r_res, x0=x, meig0=meig_g, xgroup0=xgroup)
pred$pred[1:5,]
```

```
##       pred       xb sf_residual   xgroup
## 1 23.70932 22.71407   -1.170482 2.165726
## 2 24.57615 22.21874   -1.390220 3.747633
## 3 30.58942 28.23201   -1.390220 3.747633
## 4 33.24998 28.19959   -1.493814 6.544205
## 5 33.62206 28.57167   -1.493814 6.544205
```

As shown above, the predict0 function returns predicted values (pred), predicted trends (xb), predicted residual spatial components (sf_residuals), and predicted group effects (xgroup). Then, these individual-level variables are aggregated into towns. Here is a code:

```r
adat  <- aggregate(data.frame(y, pred$pred),by=list(xgroup),mean)
adat[1:5,]
```

```
##   Group.1        y     pred       xb sf_residual   xgroup
## 1       0 24.00000 23.70932 22.71407   -1.170482 2.165726
## 2       1 28.15000 27.58279 25.22537   -1.390220 3.747633
## 3       2 32.76667 31.89132 26.84093   -1.493814 6.544205
## 4       3 19.42857 19.36679 18.51187   -1.576641 2.431558
## 5       4 16.71364 16.72781 17.10793   -1.416151 1.036033
```

The outputs are the predicted standard price (pred), trend (xb), spatially dependent group effects (sf_residual), and spatially independent group effects (xgroup) by town.

To map the result, spatial polygons for the towns are loaded and combined with our estimates:

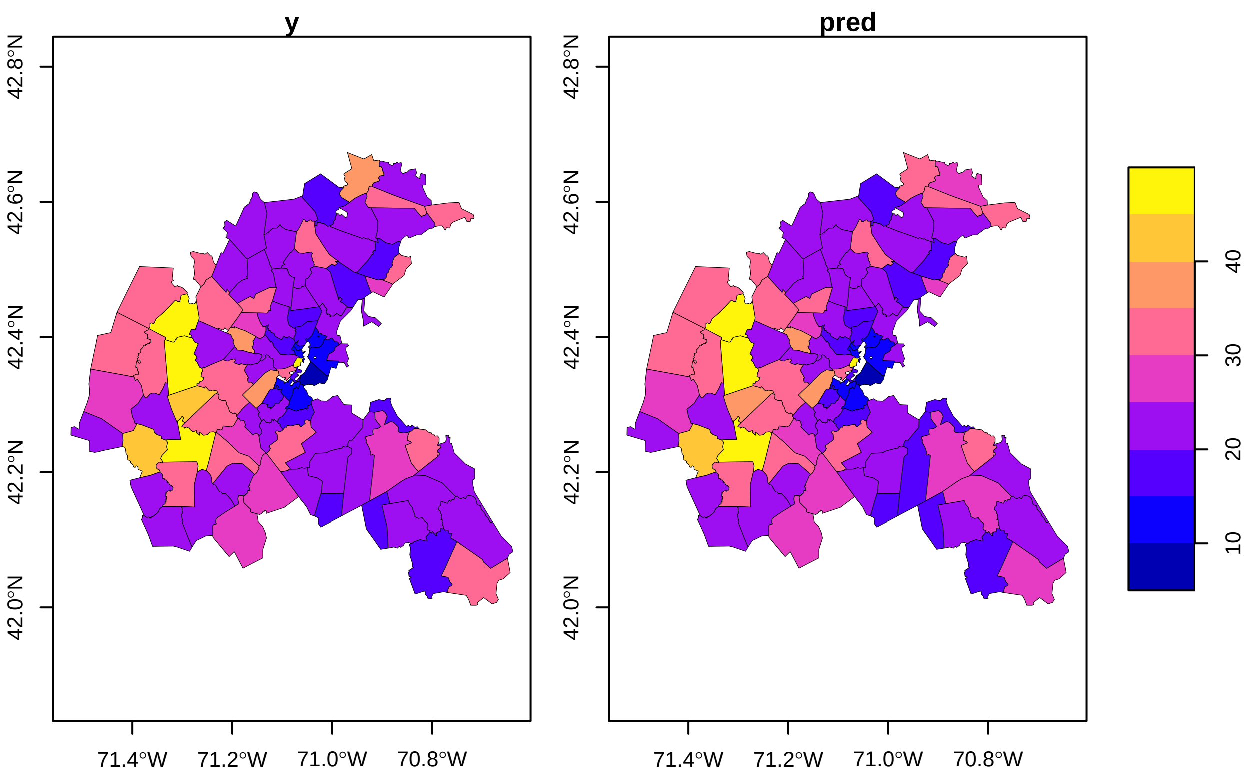
```r
require(dplyr)
b1          <- st_read(system.file("shapes/boston_tracts.shp",package="spData")[1])
```

```
## Reading layer `boston_tracts' from data source
##   `/Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/spData/shapes/boston_tracts
##   using driver `ESRI Shapefile'
## Simple feature collection with 506 features and 36 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: -71.52311 ymin: 42.00305 xmax: -70.63823 ymax: 42.67307
## Geodetic CRS:  NAD27
```

```r
boston.tr2   <- b1 %>% group_by(TOWNNO) %>% summarize() #dissolve
boston.tr2$id<- 1:(dim(boston.tr2)[1])
boston.tr3   <- merge(boston.tr2, adat,by.x="TOWNNO",by.y="Group.1",all.x=TRUE)
```

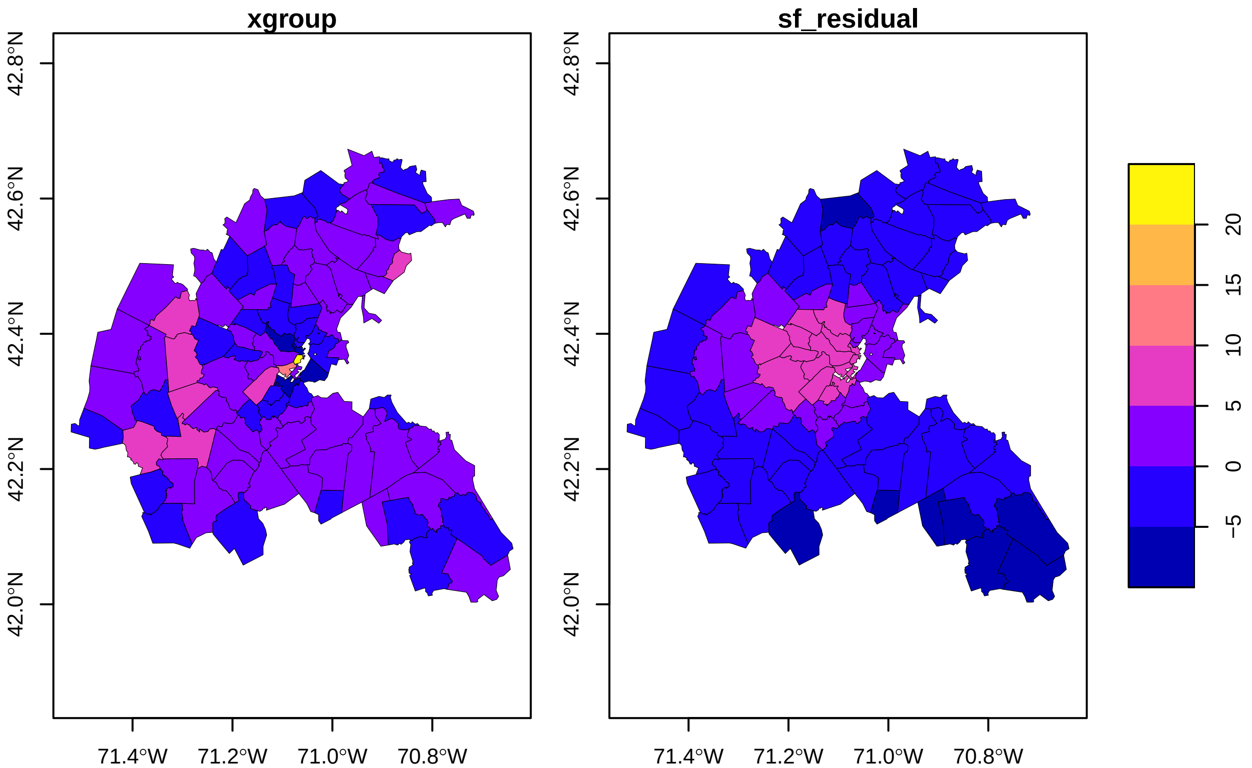
Here are the maps of our estimates. "y" denotes the observed mean prices, and "pred" denotes our predicted standard price. While they are similar, there are some differences in towns with high housing prices.

```r
boston.tr4  <- boston.tr3[order(boston.tr3$id),]
plot(boston.tr4[,c("y","pred")], lwd=0.3,axes=TRUE,key.pos=4)
```

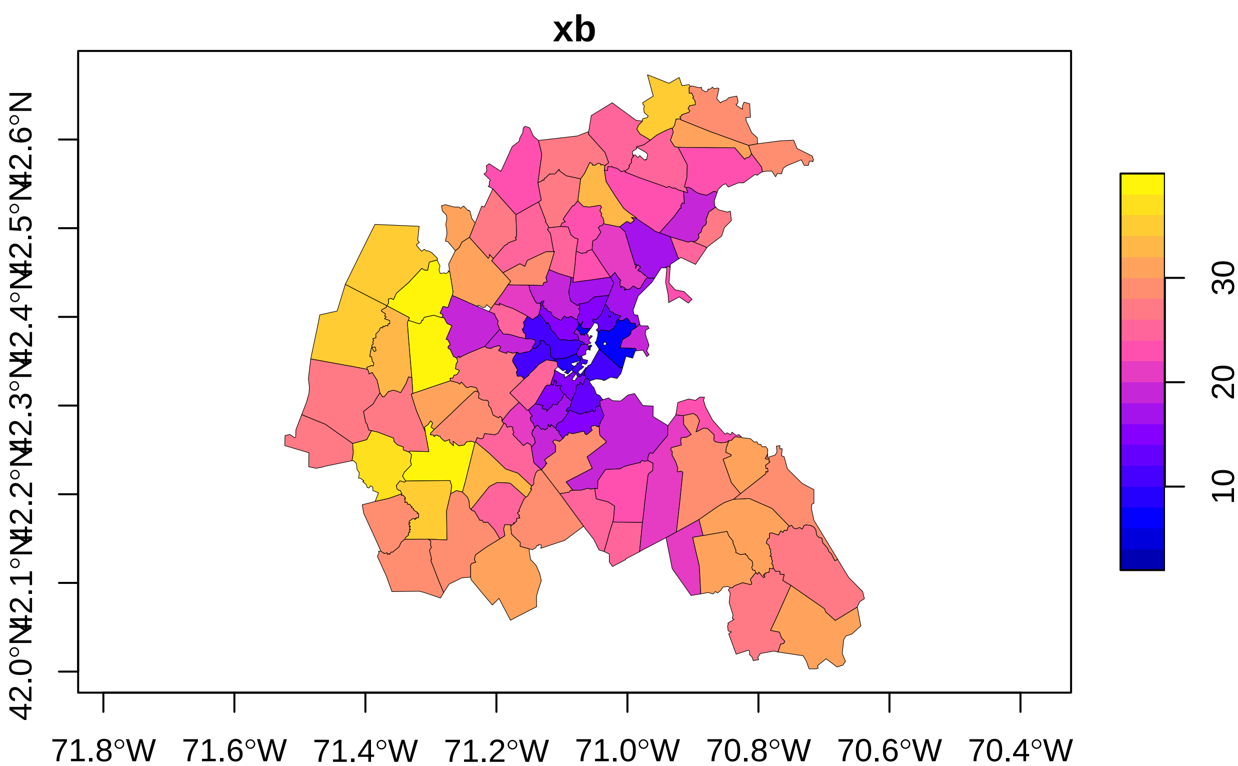


Here are the elements of the predicted values. The maps below show that each element explains different things to each other:

```r
plot(boston.tr4[,c("xgroup","sf_residual")], lwd=0.3,axes=TRUE,key.pos=4)
```




```r
plot(boston.tr4[,"xb"], lwd=0.3,axes=TRUE,key.pos=4,nbreaks=20)
```



Note that the resf_vc function is also available for SVC model-based SAE. Here is a sample code:

```r
rv_res  <- resf_vc(y=y, x=x, meig=meig_g, xgroup=xgroup, x_sel=FALSE)
```

```
## [1] "-------  Iteration 1  -------"
## [1] "1/9"
## [1] "2/9"
## [1] "3/9"
## [1] "4/9"
## [1] "5/9"
## [1] "6/9"
## [1] "7/9"
## [1] "8/9"
## [1] "9/9"
## [1] "BIC: 3123.556"
## [1] "-------  Iteration 2  -------"
## [1] "1/9"
## [1] "2/9"
## [1] "3/9"
## [1] "4/9"
## [1] "5/9"
## [1] "6/9"
## [1] "7/9"
## [1] "8/9"
## [1] "9/9"
## [1] "BIC: 3124.588"
## [1] "-------  Iteration 3  -------"
## [1] "1/9"
## [1] "2/9"
## [1] "3/9"
## [1] "4/9"
## [1] "5/9"
## [1] "6/9"
## [1] "7/9"
## [1] "8/9"
## [1] "9/9"
## [1] "BIC: 3041.564"
## [1] "-------  Iteration 4  -------"
## [1] "1/9"
## [1] "2/9"
## [1] "3/9"
## [1] "4/9"
## [1] "5/9"
## [1] "6/9"
## [1] "7/9"
## [1] "8/9"
## [1] "9/9"
## [1] "BIC: 3039.611"
## [1] "-------  Iteration 5  -------"
## [1] "1/9"
## [1] "2/9"
## [1] "3/9"
## [1] "4/9"
## [1] "5/9"
## [1] "6/9"
## [1] "7/9"
```

```
## [1] "8/9"
## [1] "9/9"
## [1] "BIC: 3039.573"
## [1] "-------  Iteration 6  -------"
## [1] "1/9"
## [1] "2/9"
## [1] "3/9"
## [1] "4/9"
## [1] "5/9"
## [1] "6/9"
## [1] "7/9"
## [1] "8/9"
## [1] "9/9"
## [1] "BIC: 3039.572"
```

```
pred_vc <- predict0(rv_res, x0=x, meig0=meig_g, xgroup0=xgroup)
adat_vc <- aggregate(data.frame(y, pred_vc$pred), by=list(xgroup), mean)
adat_vc[1:5,]
```

```
##   Group.1        y     pred       xb sf_residual   xgroup
## 1       0 24.00000 23.67486 23.10559   -1.124267 1.693539
## 2       1 28.15000 27.81232 27.43213   -1.964242 2.344436
## 3       2 32.76667 32.28842 31.08102   -2.548396 3.755800
## 4       3 19.42857 19.25780 18.44941   -2.502559 3.310950
## 5       4 16.71364 16.68373 15.39676   -1.025117 2.312090
```

2.2.4.4 Longitudinal/panel data analysis

The resf and resf_vc functions are also available for longitudinal or panel data analysis with/without S(N)VC (see Yu et al., 2020). Although this section takes resf as an example, resf_vc function-based panel analysis is implemented in the same way.

To illustrate this, we use a panel data of 48 US states from 1970 to 1986, which is published in the plm package (Croissant and Millo, 2008). Because our approach uses spatial coordinates by default, we added center spatial coordinates (px and py) to the panel data. Here is the code:

```
require(plm)
require(spData)

data(Produc, package = "plm")
data(us_states)
us_states2 <- data.frame(us_states$GEOID,us_states$NAME,st_coordinates(st_centroid(us_states)))
names(us_states2)[3:4]<- c("px","py")
us_states3 <- us_states2[order(us_states2[,1]),][-8,]
us_states3$state<- unique(Produc[,1])
pdat0      <- na.omit(merge(Produc,us_states3[,c(3:5)],by="state",all.x=TRUE,sort=FALSE))
pdat       <- pdat0[order(pdat0$state,pdat0$year),]
pdat[1:5,]
```

```
##     state year region     pcap     hwy   water    util       pc   gsp    emp
## 1 ALABAMA 1970      6 15032.67 7325.80 1655.68 6051.20 35793.80 28418 1010.5
## 2 ALABAMA 1971      6 15501.94 7525.94 1721.02 6254.98 37299.91 29375 1021.9
## 3 ALABAMA 1972      6 15972.41 7765.42 1764.75 6442.23 38670.30 31303 1072.3
## 4 ALABAMA 1973      6 16406.26 7907.66 1742.41 6756.19 40084.01 33430 1135.5
## 5 ALABAMA 1974      6 16762.67 8025.52 1734.85 7002.29 42057.31 33749 1169.8
##   unemp        px       py
## 1   4.7 -86.82797 32.78034
```

```
## 2   5.2 -86.82797 32.78034
## 3   4.7 -86.82797 32.78034
## 4   3.9 -86.82797 32.78034
## 5   5.5 -86.82797 32.78034
```

Here are the first five rows of the data:

```
pdat[1:5,]
```

```
##     state year region     pcap     hwy   water    util       pc   gsp    emp
## 1 ALABAMA 1970      6 15032.67 7325.80 1655.68 6051.20 35793.80 28418 1010.5
## 2 ALABAMA 1971      6 15501.94 7525.94 1721.02 6254.98 37299.91 29375 1021.9
## 3 ALABAMA 1972      6 15972.41 7765.42 1764.75 6442.23 38670.30 31303 1072.3
## 4 ALABAMA 1973      6 16406.26 7907.66 1742.41 6756.19 40084.01 33430 1135.5
## 5 ALABAMA 1974      6 16762.67 8025.52 1734.85 7002.29 42057.31 33749 1169.8
##   unemp        px       py
## 1   4.7 -86.82797 32.78034
## 2   5.2 -86.82797 32.78034
## 3   4.7 -86.82797 32.78034
## 4   3.9 -86.82797 32.78034
## 5   5.5 -86.82797 32.78034
```

Following a vignette of the plm package, this section uses logged gross state product as explained variables (y) and logged public capital stock (log_pcap), logged private capital stock (log_pc), logged labor input measured by the employment in non-agricultural payrolls (log_emp), and unemployment rate (unemp) as covariables.

```
y     <- log(pdat$gsp)
x     <- data.frame(log_pcap=log(pdat$pcap), log_pc=log(pdat$pc),
                    log_emp=log(pdat$emp), unemp=pdat$unemp)
```

Because spatial coordinates are defined by states, Moran eigenvectors must be extracted by state by specifying s_id in the meigen function, as follows:

```
coords<- pdat[,c("px", "py")]
s_id  <- pdat$state
meig_p<- meigen(coords,s_id=s_id)# Moran eigenvectors by states
```

Currently, the following spatial panel models are available: pooling model (no group effects); individual random effects model (state-level group effects); time random effects model (year-level group effects); and two-way random effects model (state and year-level group effects). All these models consider residual spatial dependence. Here are the codes implementing these models:

```
pmod0 <- resf(y=y,x=x,meig=meig_p) # pooling model

xgroup<- pdat[,c("state")]
pmod1 <- resf(y=y,x=x,meig=meig_p,xgroup=xgroup)# individual model

xgroup<- pdat[,c("year")]
pmod2 <- resf(y=y,x=x,meig=meig_p,xgroup=xgroup)# time model

xgroup<- pdat[,c("state","year")]
pmod3 <- resf(y=y,x=x,meig=meig_p,xgroup=xgroup)# two-way model
```

Among these models, the two-way model indicates the smallest BIC. The output is summarized as

```
pmod3
```

```
## Call:
```

```
## resf(y = y, x = x, xgroup = xgroup, meig = meig_p)
##
## ----Coefficients------------------------------
##                  Estimate          SE    t_value      p_value
## (Intercept)  2.267117239 0.157658762 14.3799000 0.000000e+00
## log_pcap     0.007169432 0.023527602  0.3047243 7.606606e-01
## log_pc       0.292327635 0.022204710 13.1651181 0.000000e+00
## log_emp      0.732863243 0.024803751 29.5464677 0.000000e+00
## unemp       -0.004357492 0.001066662 -4.0851680 4.878493e-05
##
## ----Variance parameter------------------------
##
## Spatial effects (residuals):
##                       (Intercept)
## random_SD               0.1554450
## Moran.I/max(Moran.I)    0.3332442
##
## Group effects:
##                 state       year
## ramdom_SD 0.09486666 0.02434585
##
## ----Error statistics--------------------------
##                      stat
## resid_SE        0.0338136
## adjR2(cond)     0.9988953
## rlogLik      1408.4119302
## AIC         -2796.8238604
## BIC         -2749.7797169
##
## NULL model: lm( y ~ x )
##    (r)loglik: 826.9817 ( AIC: -1641.963,  BIC: -1613.737 )
##
## Note: AIC and BIC are based on the restricted/marginal likelihood.
##       Use method="ml" for comparison of models with different fixed effects (x)
```

The estimated group effects are displayed as follows:

```r
s_g   <- pmod3$b_g[[1]]
s_g[1:5,]# State-level group effects
```

```
##                      Estimate         SE   t_value
## state_ALABAMA    -0.07201926 0.01388808 -5.185690
## state_ARIZONA    -0.04386270 0.01661108 -2.640569
## state_ARKANSAS   -0.07240066 0.01469584 -4.926610
## state_CALIFORNIA  0.23935032 0.01976082 12.112367
## state_COLORADO   -0.11569510 0.01232985 -9.383333
```

```r
t_g   <- pmod3$b_g[[2]]
t_g[1:5,]# Year-level group effects
```

```
##               Estimate         SE    t_value
## year_1970 -0.006035684 0.01108551 -0.5444659
## year_1971  0.002885623 0.01056380  0.2731613
## year_1972  0.013268803 0.01041171  1.2744120
## year_1973  0.021939416 0.01027511  2.1352003
## year_1974 -0.009861168 0.00967466 -1.0192781
```

For validation, the same panel model (but without spatial dependence) is estimated using the plm function:

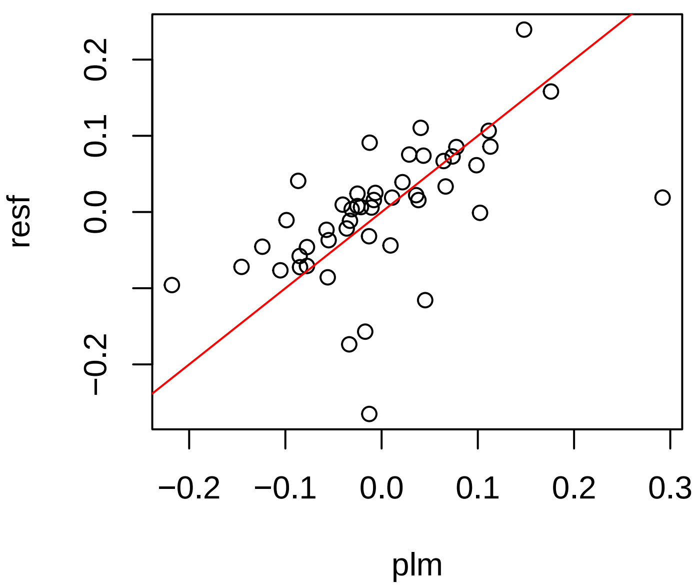
```
pm0    <- plm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
              data = pdat, effect="twoways",model="random")
pm0
```

```
##
## Model Formula: log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp
##
## Coefficients:
## (Intercept)   log(pcap)     log(pc)    log(emp)       unemp
##   2.3634993   0.0178529   0.2655895   0.7448989  -0.0045755
```

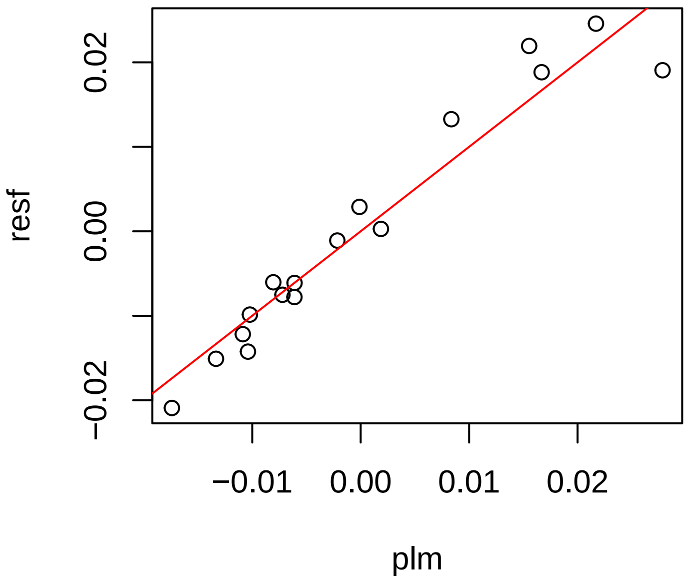
```
s_g_plm<- ranef(pm0,"individual")
t_g_plm<- ranef(pm0,"time")
```

The coefficient estimates are similar. The plots below compare estimated group effects. Estimated state-level effects have differences because our models consider residual spatial dependence, while plm does not (by default). Time effects are quite similar.

```
plot(s_g_plm,s_g[,1],xlab="plm",ylab="resf")
abline(0,1,col="red")
```




```
plot(t_g_plm,t_g[,1],xlab="plm",ylab="resf")
abline(0,1,col="red")
```



2.3 Spatial prediction

This package provides functions for ESF/RE-ESF-based spatial interpolation minimizing the expected prediction error (just like kriging). RE-ESF approximates a Gaussian process or the kriging model, which has actively been used for spatial prediction, and ESF is a special case (Murakami and Griffith, 2015). Because ESF and RE-ESF impose approximations, they are faster for very large samples.

In this tutorial, the Lucas housing price data with sample size being 25,357 is used. In the prediction, "price" is used as the explained variable, and "age," "rooms," "beds," and "year" are used as covariates.

```
require(spData)
data(house)
dat0  <- st_as_sf(house)
dat   <- data.frame(st_coordinates(dat0), dat0[,c("price","age","rooms","beds","syear")])
```

A total of 20,000 randomly selected samples are used for model estimation, and the other 5,357 samples are used for accuracy evaluation. The code below creates the data for observation sites (coords, y, x) and for unobserved sites (coords0, y0, x0):

```
samp    <- sample(dim(dat)[1], 20000)
coords<- dat[samp ,c("X","Y")]
y       <- log(dat[samp,"price"])
x     <- dat[samp,c("age","rooms","beds","syear")]

coords0<- dat[-samp ,c("X","Y")]
y0    <- log(dat[-samp,"price"]) # for valudation
x0    <- dat[-samp,c("age","rooms","beds","syear")]
```

The prediction is done in two steps: (1) evaluation of Moran eigenvectors at prediction sites using the meigen0 function; (2) prediction using the predict0 function. Below is a sample code based on the resf function:

```
start.time1<-proc.time()###### For CP time evaluation
meig     <- meigen_f(coords)
meig0    <- meigen0( meig=meig, coords0=coords0 )
mod      <- resf( y = y, x = x, meig = meig )
pred0      <- predict0( mod = mod, x0 = x0, meig0=meig0 )
```

```
end.time1<- proc.time()###### For CP time evaluation
```

Note that the first and last lines are just for computing time evaluation. NVCs are considered if adding NVC=TRUE in the resf function. The meigen_f function is used for fast computation.

The outputs shown below include predicted values (pred), predicted trend (xb), and predicted residual spatial component (sf_residuals).

```
pred0$pred[1:5,]
```

```
##        pred       xb sf_residual
## 5  11.75937 11.34271   0.4166665
## 15 11.33792 10.89578   0.4421374
## 19 11.56131 11.10802   0.4532928
## 20 11.55627 11.11665   0.4396160
## 22 11.71223 11.25905   0.4531816
```

```
pred     <- pred0$pred[,1]
```

On the other hand, here is a code for a spatial prediction based on an S(N)VC model:

```
start.time2<-proc.time()###### For CP time evaluation
meig     <- meigen_f(coords)
meig0    <- meigen0( meig=meig, coords0=coords0 )
mod2       <- resf_vc( y = y, x = x, meig = meig )
```

```
## [1] "-------  Iteration 1  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 13692.564"
## [1] "-------  Iteration 2  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 13307.394"
## [1] "-------  Iteration 3  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 13303.956"
## [1] "-------  Iteration 4  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 13303.884"
## [1] "-------  Iteration 5  -------"
## [1] "1/5"
```

```
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 13303.883"
## [1] "-------  Iteration 6  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 13303.883"
```

```
pred02   <- predict0( mod = mod2, x0 = x0, meig0=meig0 )
end.time2<- proc.time()###### For CP time evaluation
```

NVCs are considered by adding NVC=TRUE in the resf_vc function. Here are the output variables:

```
pred02$pred[1:5,]
```

```
##        pred       xb  sf_residual
## 5  11.64812 11.65038 -0.002258632
## 15 11.33052 11.32866  0.001862047
## 19 11.31361 11.29374  0.019867102
## 20 11.68258 11.65186  0.030721153
## 22 11.63507 11.61165  0.023419053
```

```
pred2    <- pred02$pred[,1]
```

The root mean squared prediction error (RMSPE) and the computational time of the spatial regression model (resf) are as follows:

```
sqrt(sum((pred-y0)^2)/length(y0))#rmse
```

```
## [1] 0.3328418
```

```
(end.time1 - start.time1)[3]#computational time (second)
```

```
## elapsed
##  12.382
```

while those of the SVC model (resf_vc) are as follows:

```
sqrt(sum((pred2-y0)^2)/length(y0))#rmse
```

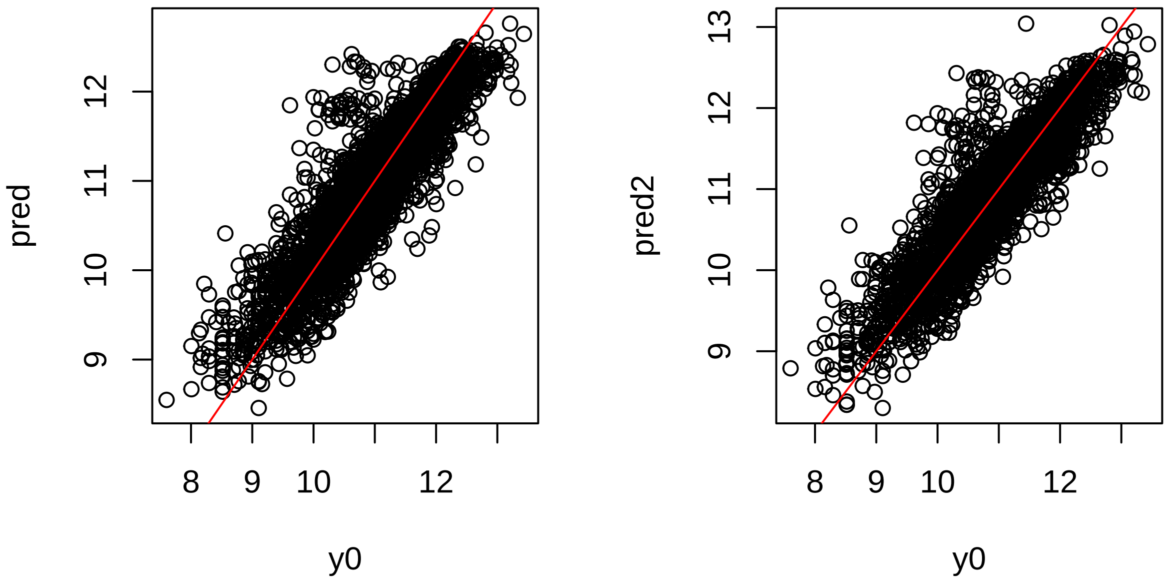
```
## [1] 0.3230838
```

```
(end.time2 - start.time2)[3]#computational time (second)
```

```
## elapsed
##  84.521
```

The results suggest that both models are available for large samples. It is also demonstrated that while the spatial regression model is faster than the SVC model, the SVC model is slightly more accurate. The actual values (y0) and predicted values (pred/pred2) are compared below:

```
par(mfrow=c(1,2))
plot(y0,pred);abline(0,1,col="red")
plot(y0,pred2);abline(0,1,col="red")
```



(RE-)ESF considers a limited number of eigenvectors, which limits the model flexibility. Because of that, (RE-)ESF suffers from a degeneracy/over-smoothing problem that decreases modeling accuracy for large samples. The addlearn_local function is useful to address this problem. This function estimates an improved SVC model by aggregating/averaging the pre-estimated SVC model (i.e., mod2) with local SVC models that are estimated by k-means-based spatial clusters each of which contains roughly 600 samples (see Murakami et al., 2023). Unlike the resf_vc and/or besf_vc function, the improved SVC model considers not only (global) eigenvectors but also local eigenvectors; the resulting model accurately captures local patterns even from very large samples.

Here is a sample code for the model aggregation and prediction after the model aggregation:

```
start.time3<-proc.time()  ###### For CP time evaluation
mod3       <- addlearn_local(mod=mod2, meig0 = meig0, x0 = x0)
```

```
## [1] "-------- Synthesizing 38 local sub-models ---------"
## [1] "1/38"
## [1] "2/38"
## [1] "3/38"
## [1] "4/38"
## [1] "5/38"
## [1] "6/38"
## [1] "7/38"
## [1] "8/38"
## [1] "9/38"
## [1] "10/38"
## [1] "11/38"
## [1] "12/38"
## [1] "13/38"
## [1] "14/38"
## [1] "15/38"
## [1] "16/38"
## [1] "17/38"
## [1] "18/38"
## [1] "19/38"
## [1] "20/38"
## [1] "21/38"
## [1] "22/38"
```

```
## [1] "23/38"
## [1] "24/38"
## [1] "25/38"
## [1] "26/38"
## [1] "27/38"
## [1] "28/38"
## [1] "29/38"
## [1] "30/38"
## [1] "31/38"
## [1] "32/38"
## [1] "33/38"
## [1] "34/38"
## [1] "35/38"
## [1] "36/38"
## [1] "37/38"
## [1] "38/38"
```

```
pred3      <- mod3$pred0[,1]
end.time3  <- proc.time() ###### For CP time evaluation
```

The resulting RMSE is confirmed to be smaller than mod2, which is before the model aggregation:

```
sqrt(sum((pred3-y0)^2)/length(y0))#rmse
```

```
## [1] 0.2987441
```

```
(end.time3 - start.time3)[3]#computational time (second)
```

```
## elapsed
## 182.344
```

While the addlearn_local function requires an additional computation time, it can be paralleled by specifying parallel = TRUE. The accuracy difference between the models with/without the model aggregation/averaging increases as the sample size increases. This addlearn_local function is especially recommended for larger samples.

The addlearn_local function is also useful to improve SVC coefficient estimation accuracy as demonstrated in Murakami et al. (2023). See Section 6.3 for further detail.

3 Non-Gaussian spatial regression models

This package is now available for modeling a wide variety of non-Gaussian data including count data. Unlike the conventional generalized linear model (GLM), the implemented model estimates the most likely data distribution (i.e., probability density/mass function) without explicitly specifying the data distribution (see Murakami et al., 2021). See Murakami (2021) or vignette_spmoran(nongaussian).pdf, which is another vignette in the same GitHub page https://github.com/dmuraka/spmoran for details on how to implement it.

4 Spatially filtered unconditional quantile regression

While the usual (conditional) quantile regression (CQR) estimates the influence of x_k on the τ-th conditional quantile of y, $q_\tau(y|x_k)$, the unconditional quantile regression estimates the influence of x_k on the "unconditional" quantile of y, $q_\tau(y)$ (Firpo et al., 2009).

Suppose that y and x_k represent land price and accessibility, respectively. UQR estimates the influence of accessibility on land price by quantile; it is interpretable and useful for hedonic land price analysis, for example.

By contrast, this interpretation does not hold for CQR because it estimates the influence of accessibility on conditional land prices (land price conditional on explanatory variables). Higher conditional land price does not mean higher land price; rather, it means overprice relative to the price expected by the explanatory variables. Therefore, CQR has difficulty in its interpretation, in some cases, including hedonic land price modeling.

The spatial filter UQR (SF-UQR) model (Murakami and Seya, 2019), which is implemented in this package, is formulated as

$$q_\tau(y_i) = \sum_{k=1}^{K} x_{i,k}\beta_{k,\tau} + f_{MC,\tau}(s_i) + \epsilon_{i,\tau}, \qquad \epsilon_{i,\tau} \sim N(0, \sigma_\tau^2),$$

This model is a UQR considering spatial dependence.

The resf_qr function implements this model. Below is a sample code. If boot=TRUE in resf_qr, a semiparametric bootstrapping is performed to estimate the standard errors of the regression coefficients. By default, this function estimates models at 0.1, 0.2,. . . , 0.9 quantiles.

```
y       <- boston.c[, "CMEDV" ]
x       <- boston.c[,c("CRIM","ZN","INDUS", "CHAS", "NOX","RM", "AGE")]
coords<- boston.c[,c("LON","LAT")]
meig    <- meigen(coords=coords)
res   <- resf_qr(y=y,x=x,meig=meig, boot=TRUE)
```

```
## [1] "------- Complete: tau=0.1 -------"
## [1] "------- Complete: tau=0.2 -------"
## [1] "------- Complete: tau=0.3 -------"
## [1] "------- Complete: tau=0.4 -------"
## [1] "------- Complete: tau=0.5 -------"
## [1] "------- Complete: tau=0.6 -------"
## [1] "------- Complete: tau=0.7 -------"
## [1] "------- Complete: tau=0.8 -------"
## [1] "------- Complete: tau=0.9 -------"
```

Here is a summary of the estimation result:

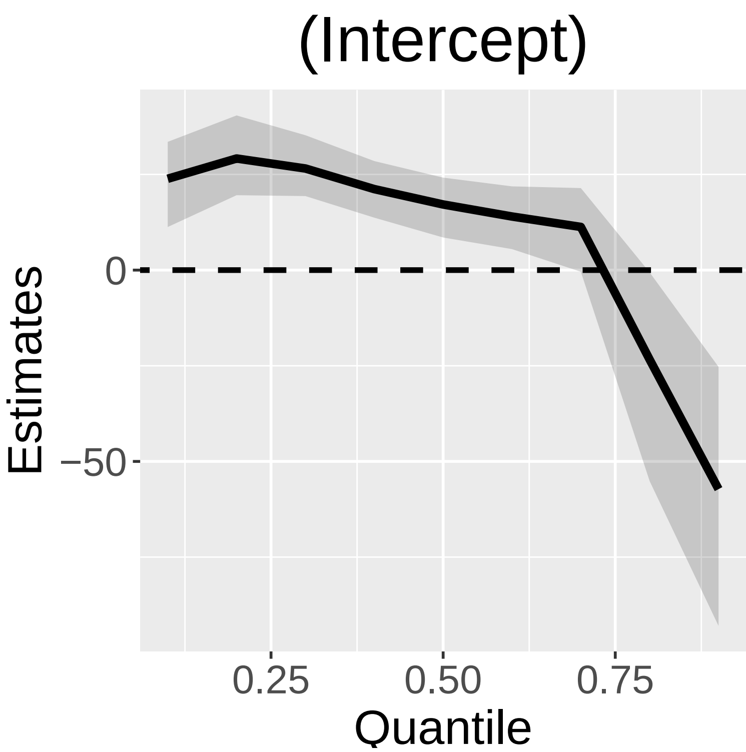
```
res
```

```
## Call:
## resf_qr(y = y, x = x, meig = meig, boot = TRUE)
##
## ----Coefficients------------------------------
##                  tau=0.1      tau=0.2       tau=0.3      tau=0.4       tau=0.5
## (Intercept)  23.86841970  29.16185736  26.550125353  21.16263694  17.151053980
## CRIM         -0.36845124  -0.21172051  -0.106949379  -0.08357496  -0.070290258
## ZN           -0.01169653  -0.01627637  -0.009652286  -0.01947512  -0.008198579
## INDUS         0.25009373   0.03992002  -0.111010420  -0.01521113  -0.096468769
## CHAS          0.98647836   1.28770409   0.438428954   0.26777796  -0.048278485
## NOX         -32.89857783 -23.60303480 -15.109338348 -12.70090129 -11.263158727
## RM            0.71728433   0.49201634   1.169115918   2.21382993   3.004059676
## AGE           0.01977978  -0.05087471  -0.082548477  -0.11192561  -0.105681036
##                   tau=0.6      tau=0.7     tau=0.8      tau=0.9
## (Intercept)  13.999671526  11.28433168 -23.3939330 -57.24239068
## CRIM         -0.064412593  -0.07823561  -0.1876252  -0.18934294
## ZN            0.007962903   0.01009742   0.1635369   0.03890142
## INDUS        -0.167039581  -0.30344029  -0.9074079  -0.49797629
## CHAS         -1.665298913  -1.51518801  -3.8773852  -0.04635798
## NOX         -11.405913169 -20.36309658 -39.1980207 -41.26421537
## RM            3.730680883   5.25253569  13.7698457  19.62200618
```

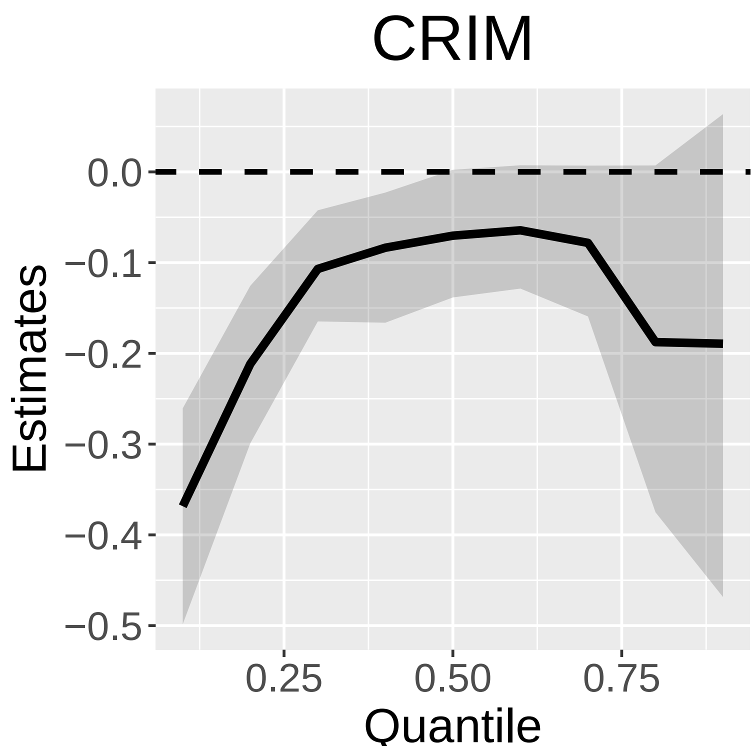
```
## AGE          -0.092068861  -0.07567382  -0.0587608  -0.03904752
##
## ----Spatial effects (residuals)---------------
##                               tau=0.1   tau=0.2   tau=0.3   tau=0.4   tau=0.5
## spcomp_SD                   7.1522586 8.1254770 5.7952363 4.4135132 4.7198329
## spcomp_Moran.I/max(Moran.I) 0.2375865 0.3228553 0.3239407 0.3650454 0.5096847
##                               tau=0.6   tau=0.7    tau=0.8    tau=0.9
## spcomp_SD                   4.8818059 6.3633073 16.9989855 16.3826940
## spcomp_Moran.I/max(Moran.I) 0.5690447 0.6935049  0.6757823  0.7203891
##
## ----Error statistics-------------------------
##                     tau=0.1   tau=0.2  tau=0.3   tau=0.4   tau=0.5   tau=0.6
## resid_SE          6.4395412 6.2086846 5.169030 4.7999618 4.5977255 4.8160068
## quasi_adjR2(cond) 0.6007294 0.6828421 0.666506 0.6183801 0.6229795 0.6121279
##                     tau=0.7    tau=0.8    tau=0.9
## resid_SE          5.6288391 12.2961444 18.6716254
## quasi_adjR2(cond) 0.6153019  0.6741455  0.4582676
```

The estimated coefficients can be visualized using the plot_qr function, as below. The numbers 1 to 5 specify which coefficients are plotted (1: intercept). In each panel, solid lines are estimated coefficients, and gray areas are their 95% confidence intervals.

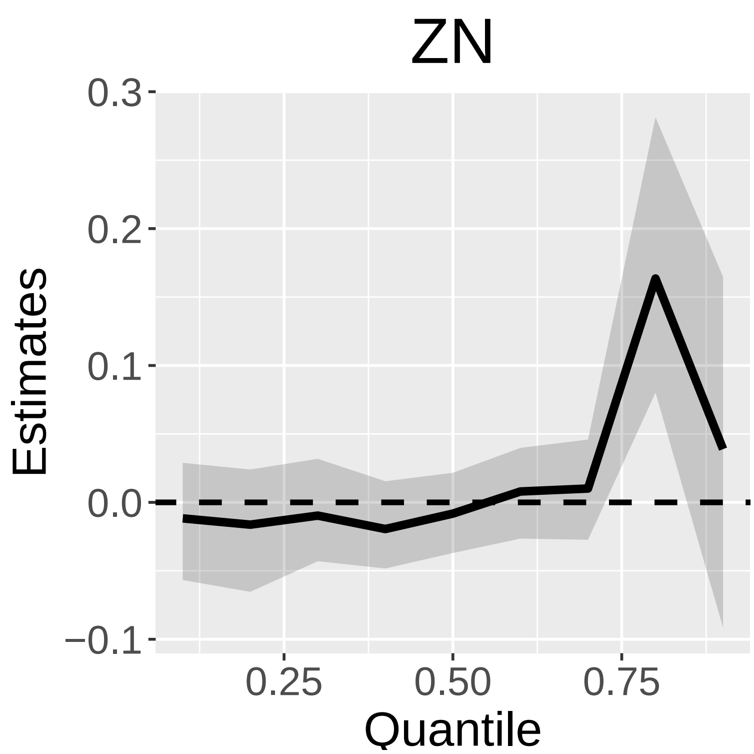
```r
plot_qr( res, 1 )
```



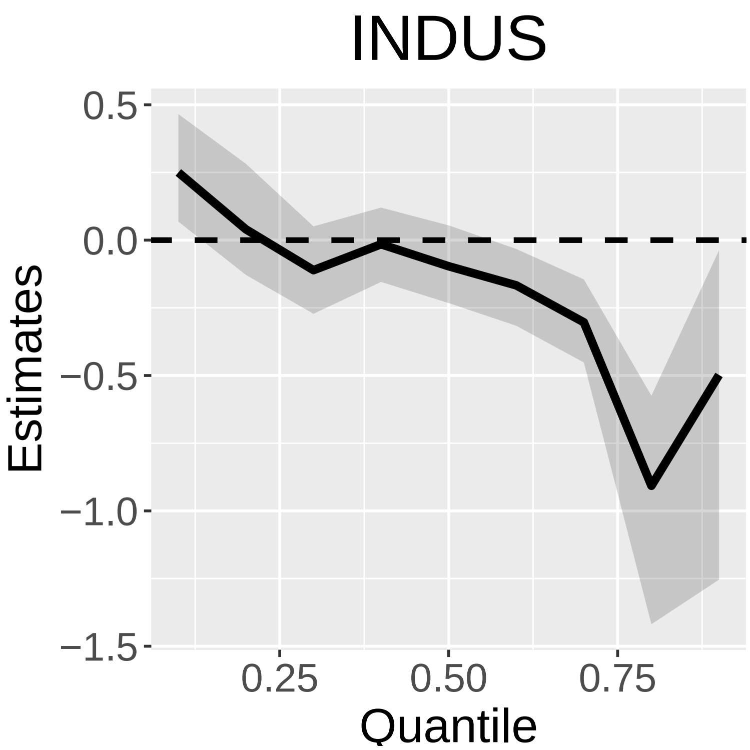

```r
plot_qr( res, 2 )
```



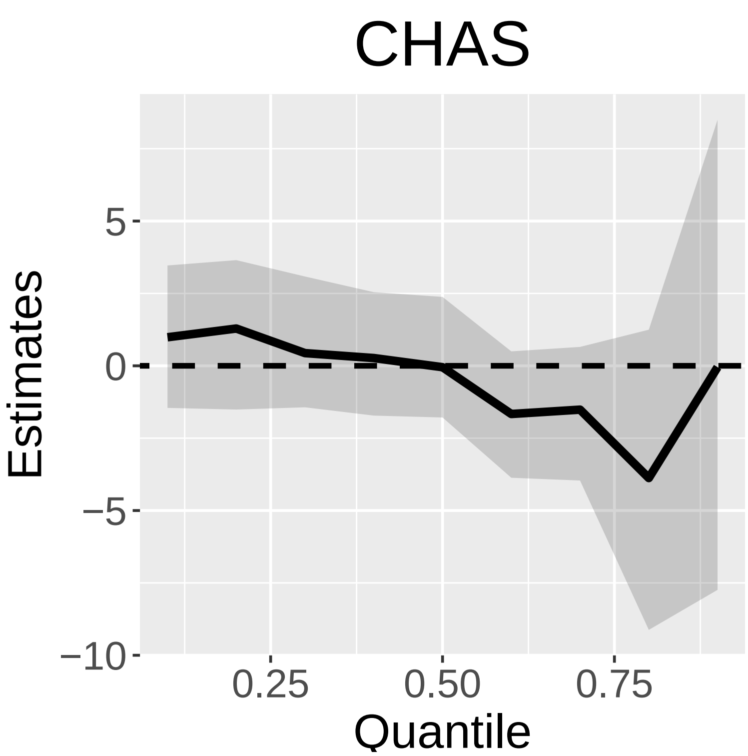

```
plot_qr( res, 3 )
```




```
plot_qr( res, 4 )
```

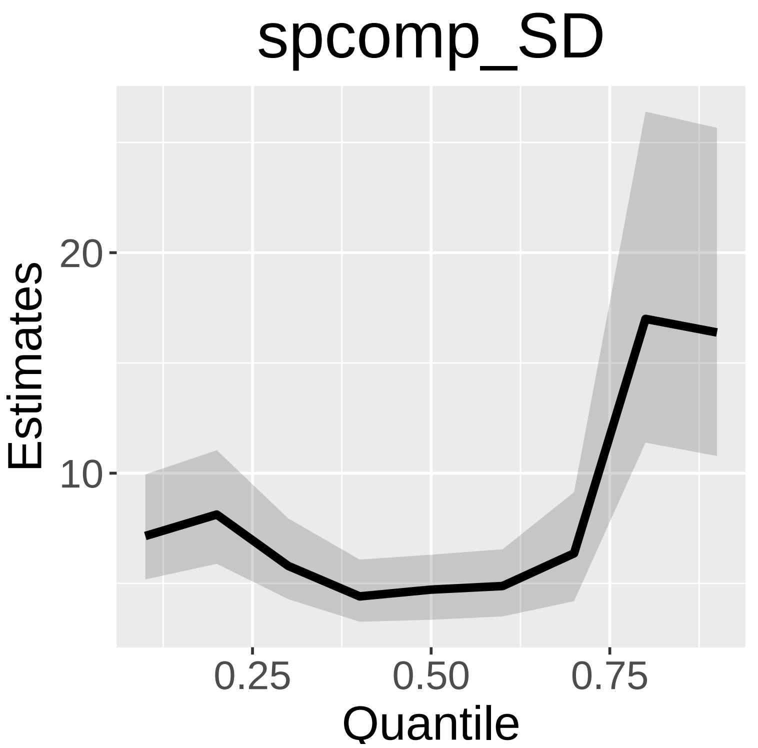



```
plot_qr( res, 5 )
```

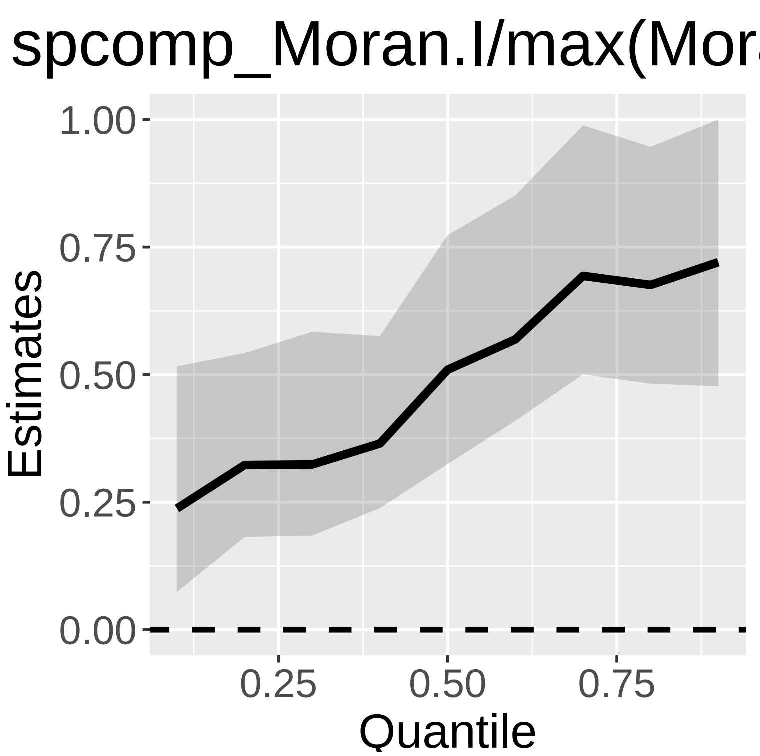


Standard errors and the scaled Moran coefficient (Moran.I/max(Moran.I)), which is a measure of spatial scale by quantile, are plotted if par = "s" is added. Here are the plots:

```
plot_qr( res, par = "s" , 1 )
```




```
plot_qr( res, par = "s" , 2 )
```



5 Low rank spatial econometric models

While ESF/RE-ESF and their extensions approximate Gaussian processes, this section explains low rank spatial econometric models approximating spatial econometric models (see Murakami et al., 2018).

5.1 Spatial weight matrix and their eigenvectors

The low rank models use eigenvectors and eigenvalues of a spatial connectivity matrix, which is called a spatial weight matrix or W matrix in spatial econometrics. The weigen function is available for the eigen-decomposition. Here is a code extracting the eigenvectors and eigenvalues from spatial polygons:

```
data( boston )
poly    <- st_read( system.file( "shapes/boston_tracts.shp", package = "spData" )[ 1 ] )
```

```
## Reading layer `boston_tracts' from data source
##   `/Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library/spData/shapes/boston_tracts
##   using driver `ESRI Shapefile'
## Simple feature collection with 506 features and 36 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: -71.52311 ymin: 42.00305 xmax: -70.63823 ymax: 42.67307
## Geodetic CRS:  NAD27
```

```
weig    <- weigen( poly )           #### Rook adjacency-based W
```

By default, the weigen function returns a Rook adjacency-based W matrix. Other than that, knn-based W, Delaunay triangulation-based W, and user-specified W are also available.

5.2 Models

5.2.1 Low rank spatial lag model

The low rank spatial lag model (LSLM) approximates the following model:

$$y_i = \beta_0 + z_i + \epsilon_i \qquad \epsilon_i \sim N(0, \sigma^2) z_i = \rho \sum_{i \neq j}^{N} w_{i,j} z_j + \sum_{k \neq 1}^{K} x_{i,k}\beta_k + u_i \qquad u_i \sim N(0, \tau^2)$$

where z_i is defined by the classical spatial lag model (SLM; see LeSage and Pace, 2009) with parameters ρ and τ^2. Just like the original SLM, ρ takes a value between 1 and $1/\lambda_N (< 0)$. Larger positive ρ means stronger positive dependence. τ^2 represents the variance of the SLM-based spatial process (i.e., z_i), while σ^2 represents the variance of the data noise ϵ_i. Because of the additional noise term, the LSLM estimates are different from the original SLM, in particular if data is noisy.

The LSLM is implemented using the lslm function. Here is a sample code:

```
y       <- boston.c[, "CMEDV" ]
x       <- boston.c[,c("CRIM","ZN","INDUS", "CHAS", "NOX","RM", "AGE")]
coords<- boston.c[,c("LON","LAT")]
res    <- lslm( y = y, x = x, weig = weig, boot = TRUE )
```

```
## [1] "------- Complete:20/200 -------"
## [1] "------- Complete:40/200 -------"
## [1] "------- Complete:60/200 -------"
## [1] "------- Complete:80/200 -------"
```

```
## [1] "------- Complete:100/200 -------"
## [1] "------- Complete:120/200 -------"
## [1] "------- Complete:140/200 -------"
## [1] "------- Complete:160/200 -------"
## [1] "------- Complete:180/200 -------"
## [1] "------- Complete:200/200 -------"
```

If boot=TRUE, a nonparametric bootstrapping is performed to estimate the 95% confidence intervals for the τ^2 and ρ parameters and the direct and indirect effects, which quantify spill-over effects. Default is FALSE. Here is the output in which {s_rho, sp_SE} are parameters $\{\rho, \tau^2\}$:

```
res
```

```
## Call:
## lslm(y = y, x = x, weig = weig, boot = TRUE)
##
## ----Coefficients------------------------------
##                 Estimate         SE    t_value      p_value
## (Intercept) -14.719039676 2.82212543 -5.2155866 2.748705e-07
## CRIM         -0.107615211 0.02851293 -3.7742599 1.809488e-04
## ZN            0.002594642 0.01276738  0.2032243 8.390474e-01
## INDUS        -0.098604511 0.06191541 -1.5925681 1.119273e-01
## CHAS          1.903178819 0.89128954  2.1353093 3.325050e-02
## NOX          -5.101316236 3.84673642 -1.3261414 1.854349e-01
## RM            6.922743307 0.33388005 20.7342228 0.000000e+00
## AGE          -0.040691404 0.01262483 -3.2231248 1.355874e-03
##
## ----Spatial effects (lag)---------------------
##         Estimates    CI_lower   CI_upper
## sp_rho 0.02709059 -0.02410232 0.06470767
## sp_SD  7.54450065  6.66530460 8.52291238
##
## ----Effects estimates-------------------------
##
## Direct:
##          Estimates     CI_lower    CI_upper p_value
## CRIM  -0.107999852  -0.16262815 -0.04692787    0.00
## ZN     0.002603915  -0.02508718  0.03045106    0.72
## INDUS -0.098956945  -0.20151886  0.02216299    0.11
## CHAS   1.909981199   0.12255230  3.79594045    0.04
## NOX   -5.119549463 -13.31129838  1.62051646    0.16
## RM     6.947486715   6.35466169  7.54757357    0.00
## AGE   -0.040836844  -0.06356934 -0.01640287    0.00
##
## Indirect:
##           Estimates      CI_lower     CI_upper p_value
## CRIM  -2.227815e-03 -0.0066794429 0.0018777769    0.25
## ZN     5.371341e-05 -0.0006886667 0.0008165909    0.77
## INDUS -2.041278e-03 -0.0076433236 0.0029188945    0.34
## CHAS   3.939898e-02 -0.0337321718 0.1268128739    0.25
## NOX   -1.056058e-01 -0.4264644997 0.0994646749    0.41
## RM     1.433123e-01 -0.1204256289 0.3462488226    0.25
## AGE   -8.423800e-04 -0.0024092543 0.0006560118    0.25
##
## ----Error statistics--------------------------
##                      stat
```

```
## resid_SE        3.9555161
## adjR2(cond)     0.8129243
## rlogLik     -1561.3219098
## AIC          3144.6438195
## BIC          3191.1357229
##
## Note: The AIC and BIC values are based on the restricted likelihood.
##       Use method ="ml" for comparison of models with different fixed effects (x)
```

5.2.2 Low rank spatial error model

The low rank spatial error model (LSEM) approximates the following model:

$$y_i = \beta_0 + z_i + \epsilon_i \qquad \epsilon_i \sim N(0, \sigma^2) z_i = \sum_{k \neq 1}^{K} x_{i,k}\beta_k + e_i \qquad e_i = \lambda \sum_{i \neq j}^{N} w_{i,j} e_j + u_i \qquad u_i \sim N(0, \tau^2)$$

where z_i is defined by the classical spatial error model (SLM) with parameters λ and τ^2. Just like the original SEM, λ takes a larger positive value in the presence of stronger positive dependence. τ^2 represents the variance of the SEM-based spatial process (i.e., z_i). As with LSLM, the LSEM estimates can be different from the original SEM if data is noisy.

The lsem function estimates LSEM, as follows:

```r
data(boston)
res   <- lsem( y = y, x = x, weig = weig )
res
```

```
## Call:
## lsem(y = y, x = x, weig = weig)
##
## ----Coefficients------------------------------
##                 Estimate         SE    t_value      p_value
## (Intercept) -15.535928399 2.82054020 -5.5081393 6.082512e-08
## CRIM         -0.093112127 0.02911351 -3.1982447 1.479351e-03
## ZN            0.002300116 0.01292558  0.1779507 8.588411e-01
## INDUS        -0.063433279 0.06176206 -1.0270591 3.049394e-01
## CHAS          1.335521734 0.88216035  1.5139217 1.307414e-01
## NOX          -5.717186159 3.86329642 -1.4798725 1.396007e-01
## RM            7.052094665 0.33425292 21.0980796 0.000000e+00
## AGE          -0.037131943 0.01253448 -2.9623833 3.212894e-03
##
## ----Spatial effects (residuals)---------------
##            Estimates
## sp_lambda  0.885701
## sp_SD      2.926975
##
## ----Error statistics--------------------------
##                     stat
## resid_SE        4.0001174
## adjR2(cond)     0.8086816
## rlogLik     -1544.3307054
## AIC          3110.6614108
## BIC          3157.1533142
##
## Note: The AIC and BIC values are based on the restricted likelihood.
```

```
##       Use method ="ml" for comparison of models with different fixed effects (x)
```

{s_lambda, sp_SE} are parameters $\{\lambda, \tau^2\}$.

6 Modeling large samples

6.1 Eigen-decomposition

The meigen function implements an eigen-decomposition that is slow for large samples. For fast eigen-approximation, the meigen_f function is available. By default, this function approximates 200 eigenvectors; 200 is based on simulation results in Murakami and Griffith (2019a). The computation is further accelerated by reducing the number of eigenvectors. It is achieved by specifying enum by a number smaller than 200. While the meigen function took 243.8 seconds for 5,000 samples, the meigen_f took less than 1 second, as demonstrated below:

```
coords_test     <- cbind( rnorm( 5000 ), rnorm( 5000 ) )
system.time( meig_test200   <- meigen_f( coords = coords_test ))[3]
```

```
## elapsed
##   0.242
```

```
system.time( meig_test100   <- meigen_f( coords = coords_test, enum=100 ))[3]
```

```
## elapsed
##    0.08
```

```
system.time( meig_test50    <- meigen_f( coords = coords_test, enum=50 ))[3]
```

```
## elapsed
##   0.035
```

On the other hand, the weigen function implements the ARPACK routine for fast eigen-decomposition by default. The computational times with 5,000 samples and enum = 200 (default), 100, and 50 are as follows:

```
system.time( weig_test200   <- weigen( coords_test ))[3]
```

```
## elapsed
##   5.881
```

```
system.time( weig_test100   <- weigen( coords_test, enum=100 ))[3]
```

```
## elapsed
##    1.89
```

```
system.time( weig_test50    <- weigen( coords_test, enum=50 ))[3]
```

```
## elapsed
##   1.119
```

6.2 Parameter estimation

The basic ESF model is estimated computationally efficiently by specifying fn = “all” in the esf function. This setting is acceptable for large samples (Murakami and Griffith, 2019a). The resf and resf_vc functions estimate all the models explained above using a fast estimation algorithm developed in Murakami and Griffith (2019b). They are available for large samples (e.g., 100,000 samples). Although the SF-UQR model requires a bootstrapping to estimate confidential intervals for the coefficients, the computational cost for the iteration does not depend on sample size. Therefore, it is available for large samples too.

6.3 Sub-model aggregation for improved scalability in terms of accuracy

The spatial regressions implemented in this package rely on a low rank approximation (i.e., approximation that considers only a limited number of eigen-pairs). For large samples (e.g., n > 5,000), this approximation can lead to an degeneracy/over-smoothing of SVCs that decreases modeling accuracy. To address this problem, the addlearn_local function additionally learns local patterns in the SVCs by aggregating/averating a model pre-estimated by the resf_vc or besf_vc function with local SVC models, which are defined by k-means-based spatial clusters each of which contains roughly 600 samples (see Murakami et al., 2023). The last line below is a sample example for the additional learning:

```
data(house)
dat0    <- st_as_sf(house)
dat0    <- dat0[dat0$yrbuilt>1950,]
dat     <- data.frame(st_coordinates(dat0),dat0[,c("price","age","rooms","beds","syear")])
coords  <- dat[ ,c("X","Y")];names(coords)<-c("px","py")
y         <- log(dat[,"price"])
x       <- dat[,c("age","rooms","beds","syear")]
meig      <- meigen_f(coords=coords)
res     <- resf_vc(y=y,x=x,meig=meig )
```

```
## [1] "-------  Iteration 1  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 6743.771"
## [1] "-------  Iteration 2  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 6645.661"
## [1] "-------  Iteration 3  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 6635.905"
## [1] "-------  Iteration 4  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 6635.009"
## [1] "-------  Iteration 5  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 6634.937"
```

```
## [1] "-------  Iteration 6  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 6634.931"
## [1] "-------  Iteration 7  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 6634.931"
## [1] "-------  Iteration 8  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 6634.931"
```

```r
res2    <- addlearn_local(res)
```

```
## [1] "-------- Synthesizing 23 local sub-models ---------"
## [1] "1/23"
## [1] "2/23"
## [1] "3/23"
## [1] "4/23"
## [1] "5/23"
## [1] "6/23"
## [1] "7/23"
## [1] "8/23"
## [1] "9/23"
## [1] "10/23"
## [1] "11/23"
## [1] "12/23"
## [1] "13/23"
## [1] "14/23"
## [1] "15/23"
## [1] "16/23"
## [1] "17/23"
## [1] "18/23"
## [1] "19/23"
## [1] "20/23"
## [1] "21/23"
## [1] "22/23"
## [1] "23/23"
```

```r
res2
```

```
## Call:
## addlearn_local(mod = res)
##
## ----Spatially varying coefficients on x (summary)----
##
```

```
## Coefficient estimates:
##   (Intercept)           age                rooms                 beds       
##  Min.   :-86.69   Min.   :-2.9976   Min.   :-0.07639   Min.   :0.02109  
##  1st Qu.:-66.89   1st Qu.:-0.8152   1st Qu.: 0.07229   1st Qu.:0.02109  
##  Median :-63.83   Median :-0.5864   Median : 0.08217   Median :0.02109  
##  Mean   :-63.61   Mean   :-0.4337   Mean   : 0.08640   Mean   :0.02109  
##  3rd Qu.:-57.04   3rd Qu.:-0.2461   3rd Qu.: 0.09565   3rd Qu.:0.02109  
##  Max.   :-41.41   Max.   : 9.3051   Max.   : 0.21700   Max.   :0.02109  
##      syear        
##  Min.   :0.03738  
##  1st Qu.:0.03738  
##  Median :0.03738  
##  Mean   :0.03738  
##  3rd Qu.:0.03738  
##  Max.   :0.03738  
## 
## Statistical significance:
##                         Intercept  age rooms  beds syear
## Not significant                 0 3483   166     0     0
## Significant (10% level)         0  702    40     0     0
## Significant ( 5% level)         0 1351   106 12299     0
## Significant ( 1% level)     12299 6763 11987     0 12299
## 
## ----Variance parameters----------------------------------
## 
## Spatial effects (Local sub-models; Average):
##                      (Intercept)       age     rooms beds syear
## random_SD              0.2338508 0.8680511 0.0225417    0     0
## Moran.I/max(Moran.I)   0.5130785 0.2895505 0.2400077   NA    NA
## 
## Spatial effects (Global sub-model):
##                      (Intercept)        age       rooms beds syear
## random_SD             0.07617129 0.16924348 0.007359602    0     0
## Moran.I/max(Moran.I)  0.15560575 0.07378315 0.089974494   NA    NA
## 
## ----Error statistics-------------------------------------
##                      stat
## resid_SE        0.2584178
## adjR2(cond)     0.7725793
## rlogLik      1723.8131954
## AIC         -2955.6263909
## BIC         -1130.9771745
## 
## NULL model: lm( y ~ x )
##    (r)loglik: -6266.884 ( AIC: 12545.77,  BIC: 12590.27 )
```

The addlearn_local function can be paralleled by specifying parallel= TRUE. The smaller error of res2 over res confirms its better accuracy:

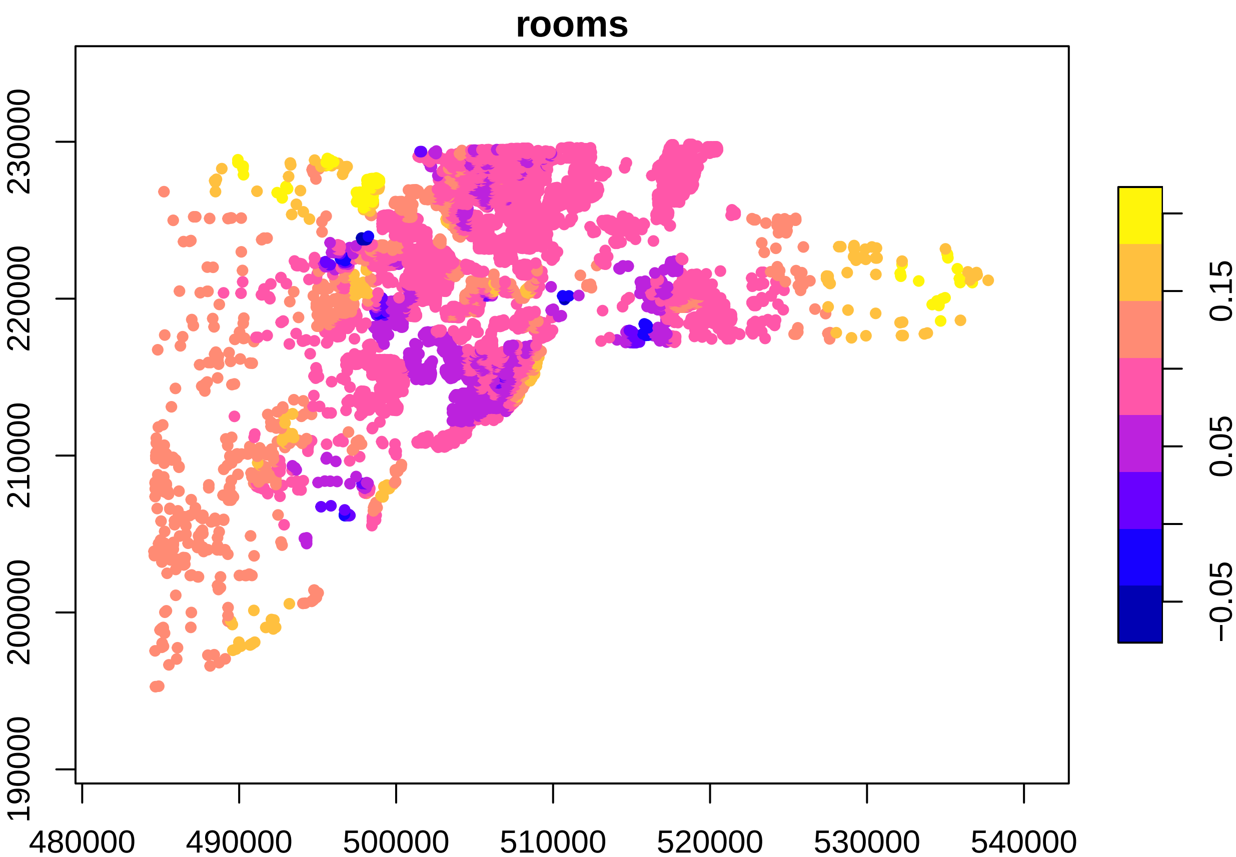
```
res$e # Before the adjustment
```

```
##                      stat
## resid_SE        0.2952775
## adjR2(cond)     0.7087546
## rlogLik     -3260.9616918
```

```
## AIC          6545.9233837
## BIC          6634.9306625
```

The plot_s function is available to quickly visualize the estimated SVCs:

```
plot_s(res2,2)### coefficients on rooms
```



The addlearn_local function is also useful to improve predictive accuracy for large samples. See Section 2.3 for further detail.

6.4 For very large samples (e.g., n > 100,000)

A computational limitation is the memory consumption of the meigen and meigen_f functions to store Moran eigenvectors. Because of the limitation, the resf and resf_vc functions are not available for very large samples (e.g., millions of samples). To overcome this limitation, the besf and besf_vc functions perform the same calculation as resf and resf_vc but without saving the eigenvectors in the memory. Besides, for fast computation, these functions perform a parallel model estimation (see Murakami and Griffith, 2019c).

Here is an example implementing a spatial regression model using the besf function and an SVC model using the besf_vc function:

```
data(house)
dat0  <- st_as_sf(house)
dat0  <- dat0[dat0$yrbuilt>1950,]
dat   <- data.frame(st_coordinates(dat0),dat0[,c("price","age","rooms","beds","syear")])
coords<- dat[ ,c("X","Y")]
y       <- log(dat[,"price"])
x     <- dat[,c("age","rooms","beds","syear")]
```

```r
res1    <- besf(y=y, x=x, coords=coords)
res1
```

```
## Call:
## besf(y = y, x = x, coords = coords)
##
## ----Coefficients------------------------------
##                  Estimate          SE    t_value       p_value
## (Intercept) -60.13333439 3.467927984 -17.339845  2.353700e-67
## age          -0.45090492 0.032570928 -13.843785  1.387061e-43
## rooms         0.10875315 0.003827260  28.415404 1.304740e-177
## beds          0.01254432 0.006900199   1.817965  6.906941e-02
## syear         0.03556862 0.001737363  20.472765  3.766353e-93
##
## ----Variance parameter------------------------
##
## Spatial effects (residuals):
##                      (Intercept)
## random_SD             0.05444638
## Moran.I/max(Moran.I)  0.20613975
##
## ----Error statistics--------------------------
##                       stat
## resid_SE         0.3122995
## adjR2(cond)      0.6743136
## rlogLik      -3574.5722938
## AIC           7165.1445877
## BIC           7224.4827736
##
## Note: The AIC and BIC values are based on the restricted likelihood.
##       Use method ="ml" for comparison of models with different fixed effects (x)
```

```r
res2    <- besf_vc(y=y, x=x, coords=coords)
```

```
## [1] "-------  Iteration 1  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 6939.739"
## [1] "-------  Iteration 2  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 6820.868"
## [1] "-------  Iteration 3  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 6805.303"
```

```
## [1] "-------  Iteration 4  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 6804.724"
## [1] "-------  Iteration 5  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 6804.71"
## [1] "-------  Iteration 6  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 6804.71"
## [1] "-------  Iteration 7  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 6804.71"
```

```
res2
```

```
## Call:
## besf_vc(y = y, x = x, coords = coords)
##
## ----Spatially varying coefficients on x (summary)----
##
## Coefficient estimates:
##   (Intercept)          age               rooms              beds
##  Min.   :-63.91   Min.   :-3.9866   Min.   :0.02532   Min.   :0.01546
##  1st Qu.:-62.64   1st Qu.:-0.9850   1st Qu.:0.08310   1st Qu.:0.01546
##  Median :-62.42   Median :-0.6284   Median :0.09384   Median :0.01546
##  Mean   :-62.45   Mean   :-0.5371   Mean   :0.09928   Mean   :0.01546
##  3rd Qu.:-62.18   3rd Qu.:-0.2037   3rd Qu.:0.10708   3rd Qu.:0.01546
##  Max.   :-61.74   Max.   : 2.5398   Max.   :0.24926   Max.   :0.01546
##      syear
##  Min.   :0.03676
##  1st Qu.:0.03676
##  Median :0.03676
##  Mean   :0.03676
##  3rd Qu.:0.03676
##  Max.   :0.03676
##
## Statistical significance:
##                         Intercept  age rooms  beds syear
## Not significant                 0 4680   120     0     0
```

```
## Significant (10% level)         0  819    67     0     0
## Significant ( 5% level)         0 1606   189 12299     0
## Significant ( 1% level)     12299 5194 11923     0 12299
##
## ----Variance parameters----------------------------------
##
## Spatial effects (coefficients on x):
##                        (Intercept)       age       rooms beds syear
## random_SD               0.07782681 0.1420487 0.007001344    0     0
## Moran.I/max(Moran.I)    0.19878930 0.1061436 0.146056275   NA    NA
##
## ----Error statistics-------------------------------------
##                        stat
## resid_SE        0.2993518
## adjR2(cond)     0.7006619
## rlogLik     -3345.8513997
## AIC          6715.7027994
## BIC          6804.7100783
##
## Note: AIC and BIC are based on the restricted/marginal likelihood.
##       Use method="ml" for comparison of models with different fixed effects (x and xconst)
```

Roughly speaking, these functions are faster than the resf and resf_vc functions if the sample size is more than 100,000.

As with the resf_vc function, the besf_vc function can suffer from the degeneracy/over-smoothing problem. The addlearn_local function is useful to address this problem and improves SVC modeling accuracy:

```
res2b <- addlearn_local(res2)
```

```
## [1] "-------- Synthesizing 23 local sub-models ---------"
```

```
res2b
```

```
## Call:
## addlearn_local(mod = res2)
##
## ----Spatially varying coefficients on x (summary)----
##
## Coefficient estimates:
##   (Intercept)          age                rooms                 beds
##  Min.   :-86.88   Min.   :-3.6598   Min.   :-0.09271   Min.   :0.02086
##  1st Qu.:-72.19   1st Qu.:-0.8151   1st Qu.: 0.07143   1st Qu.:0.02086
##  Median :-64.53   Median :-0.5854   Median : 0.08247   Median :0.02086
##  Mean   :-64.06   Mean   :-0.4175   Mean   : 0.08590   Mean   :0.02086
##  3rd Qu.:-56.25   3rd Qu.:-0.2520   3rd Qu.: 0.09282   3rd Qu.:0.02086
##  Max.   :-37.85   Max.   :15.0354   Max.   : 0.21605   Max.   :0.02086
##      syear
##  Min.   :0.0376
##  1st Qu.:0.0376
##  Median :0.0376
##  Mean   :0.0376
##  3rd Qu.:0.0376
##  Max.   :0.0376
##
## Statistical significance:
##                          Intercept   age rooms  beds syear
```

```
## Not significant                0 2994    84     0     0
## Significant (10% level)        0  611    56     0     0
## Significant ( 5% level)        0 1213   178 12299     0
## Significant ( 1% level)    12299 7481 11981     0 12299
##
## ----Variance parameters----------------------------------
##
## Spatial effects (Local sub-models; Average):
##                      (Intercept)       age      rooms beds syear
## random_SD              0.2140326 0.7912069 0.02102652    0     0
## Moran.I/max(Moran.I)   0.5461748 0.2940237 0.19024357   NA    NA
##
## Spatial effects (Global sub-model):
##                      (Intercept)       age       rooms beds syear
## random_SD             0.07782681 0.1420487 0.007001344    0     0
## Moran.I/max(Moran.I)  0.19878930 0.1061436 0.146056275   NA    NA
##
## ----Error statistics-------------------------------------
##                      stat
## resid_SE        0.2623947
## adjR2(cond)     0.7655258
## rlogLik      1685.8130296
## AIC         -2879.6260592
## BIC         -1054.9768428
##
## NULL model: lm( y ~ x )
##    (r)loglik: -6266.884 ( AIC: 12545.77,  BIC: 12590.27 )
```

The function is paralleled by default when besf_vc is assumed.

7 Reference

- Chan, A. B., and Vasconcelos, N. (2011) Counting people with low-level features and Bayesian regression. IEEE Transactions on Image Processing, 21(4), 2160-2177.
- Croissant, Y., and Millo, G. (2008) Panel data econometrics in R: The plm package. Journal of statistical software, 27(2), 1-43.
- Firpo, S., Fortin, N.M., and Lemieux, T. (2009) Unconditional quantile regressions. Econometrica, 77 (3), 953-973.
- Griffith, D.A. (2003) Spatial autocorrelation and spatial filtering: gaining understanding through theory and scientific visualization. Springer Science & Business Media.
- Ghosh, M., and Rao, J. N.K. (1994) Small area estimation: an appraisal. Statistical science, 9 (1), 55-76.
- LeSage, J.P. and Pace, R.K. (2009) Introduction to Spatial Econometrics. CRC Press.
- Murakami, D. and Griffith, D.A. (2015) Random effects specifications in eigenvector spatial filtering: a simulation study. Journal of Geographical Systems, 17 (4), 311-331.
- Murakami, D. and Griffith, D.A. (2019a) Eigenvector spatial filtering for large data sets: fixed and random effects approaches. Geographical Analysis, 51 (1), 23-49.
- Murakami, D. and Griffith, D.A. (2019b) Spatially varying coefficient modeling for large datasets: Eliminating N from spatial regressions. Spatial Statistics, 30, 39-64.
- Murakami, D. and Griffith, D.A. (2019c) A memory-free spatial additive mixed modeling for big spatial data. Japan Journal of Statistics and Data Science, doi: 10.1007/s42081-019-00063-x.
- Murakami, D., Griffith, D.A. (2021) Balancing spatial and non-spatially variations in varying coefficient modeling: a remedy for spurious correlation. Geographical Analysis, DOI:10.1111/gean.12310.

- Murakami, D. (2021) Transformation-based generalized spatial regression using the spmoran package: Case study examples, ArXiv.
- Murakami, D., Kajita, M., Kajita, S., Matsui, T. (2021) Compositionally warped additive modeling for a wide variety of non-Gaussian spatial data. Arxiv.
- Murakami, D. and Seya, H. (2019) Spatially filtered unconditional quantile regression. Environmetrics, 30 (5), e2556.
- Murakami, D., Seya, H., and Griffith, D.A. (2018) Low rank spatial econometric models. Arxiv, 1810.02956.
- Murakami, D., Sugasawa, S., T., Seya, H., and Griffith, D.A. (2023) Sub-model aggregation-based scalable eigenvector spatial filtering: application to spatially varying coefficient modeling. Arxiv.
- Murakami, D., Yoshida, T., Seya, H., Griffith, D.A., and Yamagata, Y. (2017) A Moran coefficient-based mixed effects approach to investigate spatially varying relationships. Spatial Statistics, 19, 68-89.
- Rios, G., Tobar, F. (2019). Compositionally-warped Gaussian processes. Neural Networks, 118, 235-246.
- Wheeler, D., and Tiefelsdorf, M. (2005) Multicollinearity and correlation among local regression coefficients in geographically weighted regression. Journal of Geographical Systems, 7(2), 161-187.
- Yu, D., Murakami, D., Zhang, Y., Wu, X., Li, D., Wang, X., and Li, G. (2020) Investigating high-speed rail construction's support to county level regional development in China: An eigenvector based spatial filtering panel data analysis. Transportation Research Part B: Methodological, 133, 21-37.