Spatial regression using the spmoran package: Boston housing price data examples

Daisuke Murakami

2020/01/10

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 analyze 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 additive mixed 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
## SE                     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:

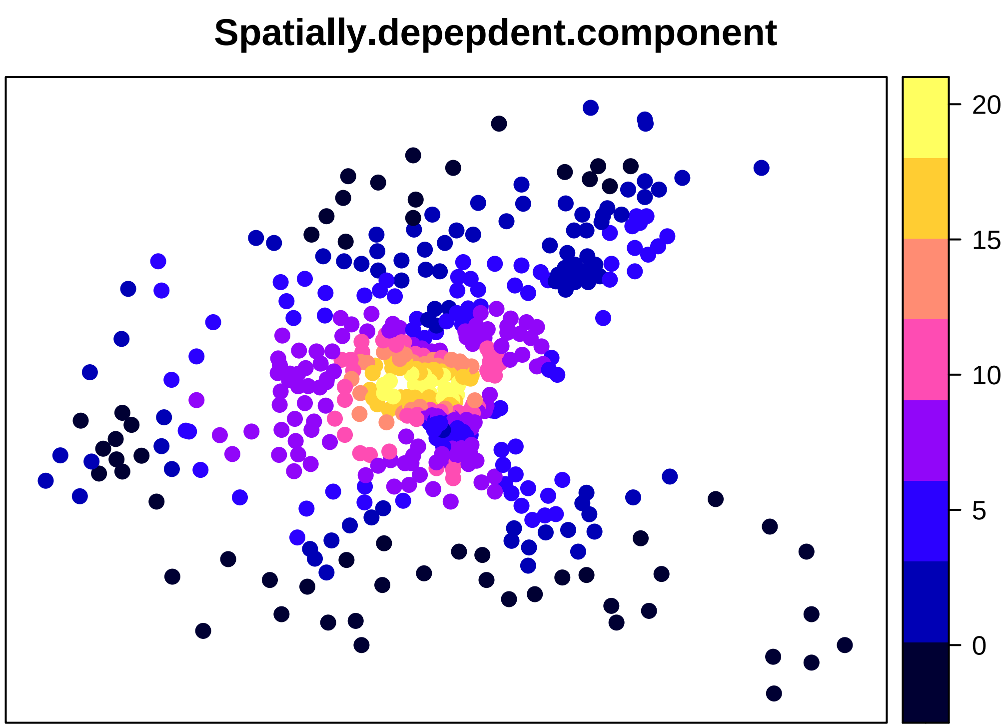
```r
res   <- resf(y = y, x = x, meig = meig)
res
```

```
## Call:
## resf(y = y, x = x, meig = meig)
##
## ----Coefficients------------------------------
##                   Estimate         SE    t_value      p_value
## (Intercept)   6.63220350 3.94484193  1.6812343 9.340107e-02
## CRIM         -0.19815203 0.03126666 -6.3374866 5.608678e-10
## ZN            0.01453736 0.01591772  0.9132814 3.615764e-01
## INDUS        -0.15560251 0.06842940 -2.2739131 2.343446e-02
## CHAS          0.51046251 0.92329946  0.5528678 5.806245e-01
## NOX         -31.26690020 5.02069123 -6.2276087 1.075126e-09
## RM            6.33993146 0.36671337 17.2885202 0.000000e+00
## AGE          -0.06351412 0.01526957 -4.1595218 3.810682e-05
##
## ----Variance parameter------------------------
##
## Spatial effects (residuals):
##                        (Intercept)
```

```
## random_SE              6.7424433
## Moran.I/max(Moran.I)   0.6648678
##
## ----Estimated probability distribution of y--------------
##                 Estimates
## skewness                0
## excess kurtosis         0
##
## ----Error statistics-------------------------
##                      stat
## resid_SE        4.3515211
## 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 )
```

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):

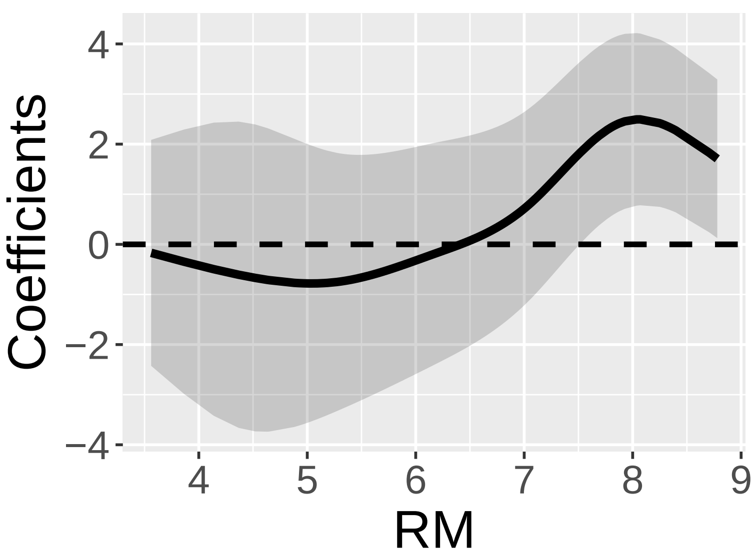
```
res   <- resf(y = y, x = x, meig = meig, nvc=TRUE)
res
```

```
## 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_SE                 3.6981527
## Moran.I/max(Moran.I)      0.4490228
##
## Non-spatial effects (coefficients on x):
```

```
##            CRIM ZN INDUS CHAS      NOX        RM AGE
## random_SE    0  0     0    0 1.850518 0.2459548   0
##
## ----Estimated probability distribution of y--------------
##                 Estimates
## skewness                0
## excess kurtosis         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 )
```

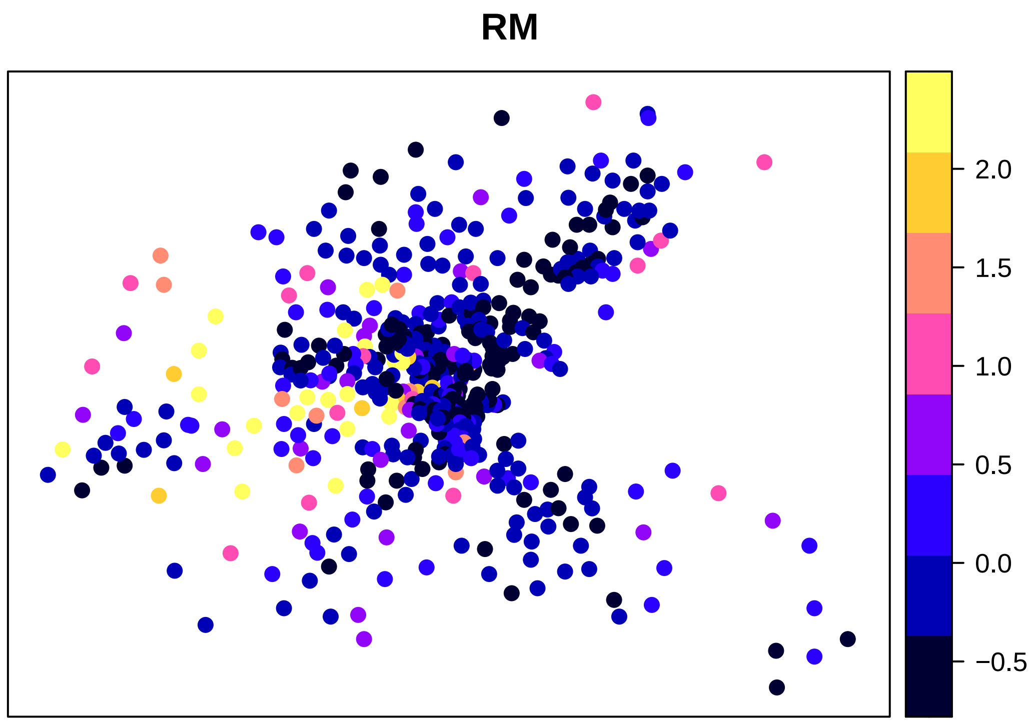
By default, this function selects constant or NVC through BIC minimization. "Non-spatially varying coefficients" in the "Variance parameter" section summarizes the estimated standard errors of the NVCs. Based on the result, coefficients on {NOX, RM} are NVCs, and coefficients on the others are constants. 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.

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



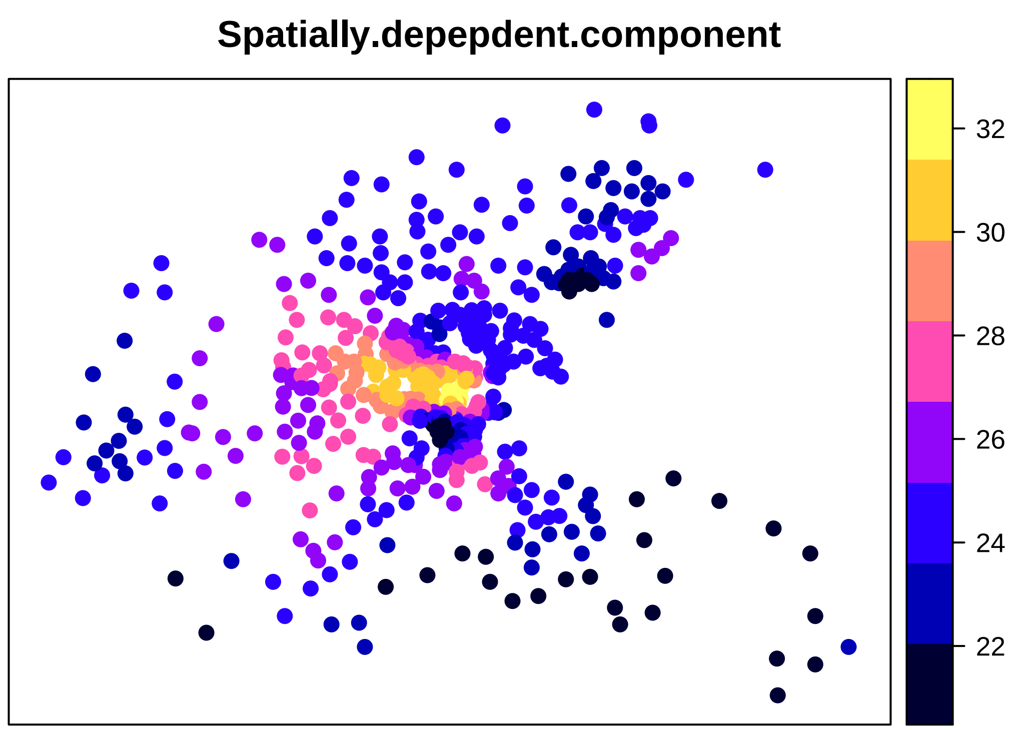
The NVC can also be spatially plotted as below:

```r
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:

```
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.

```
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"
```

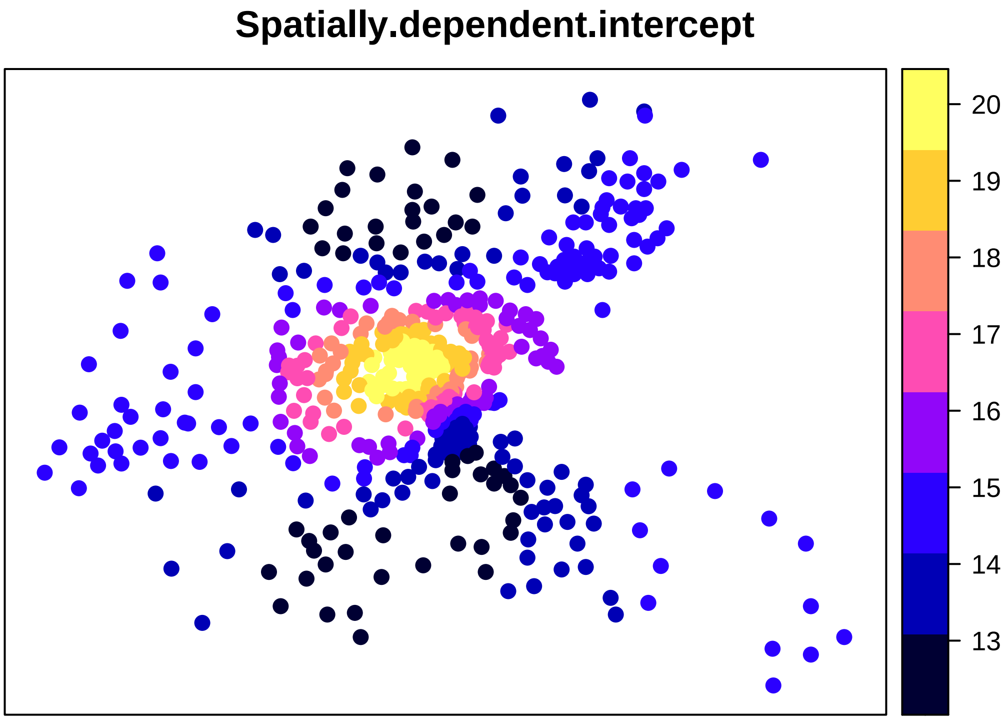
```
res
```

```
## Call:
## resf_vc(y = y, x = x, xconst = xconst, x_sel = FALSE, meig = meig)
```

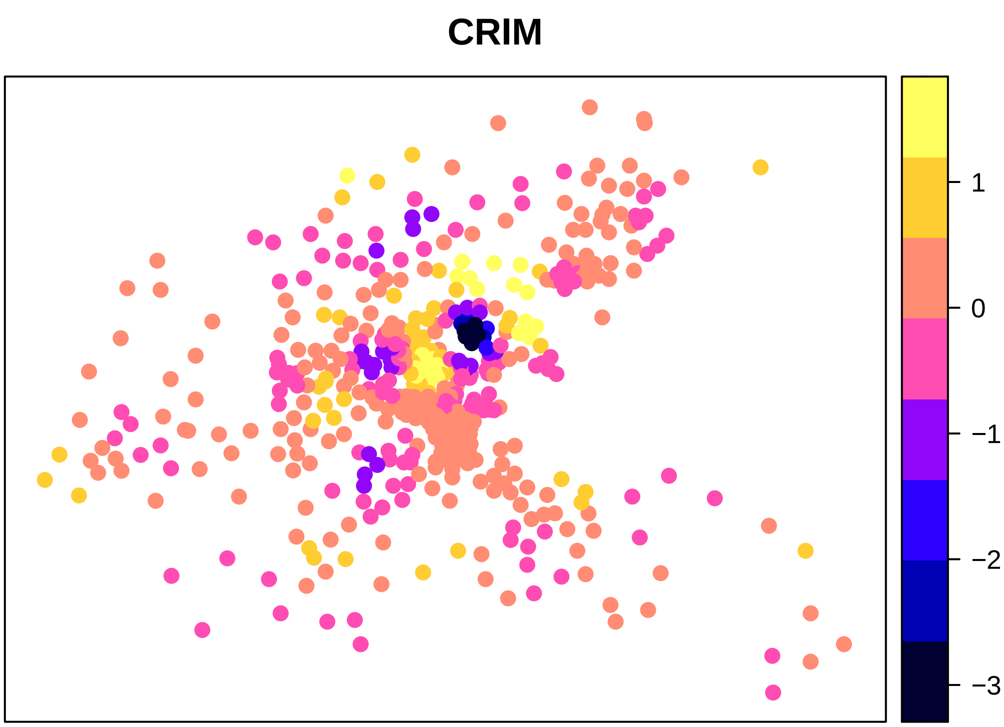
```
##
## ----Spatially varying coefficients on x (summary)----
##
## Coefficient estimates:
##   (Intercept)          CRIM                 AGE
##  Min.   :12.03   Min.   :-3.29294   Min.   :-0.14986
##  1st Qu.:13.99   1st Qu.:-0.19941   1st Qu.:-0.08377
##  Median :15.06   Median : 0.04993   Median :-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  -0.52573980 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_SE             3.9039832 1.59443322 0.05746111
## Moran.I/max(Moran.I)   0.6627375 0.04502003 0.06267778
##
## ----Estimated probability distribution of y--------------
##                 Estimates
## skewness                0
## excess kurtosis         0
##
## ----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 )
```

Estimated SVCs can be plotted as

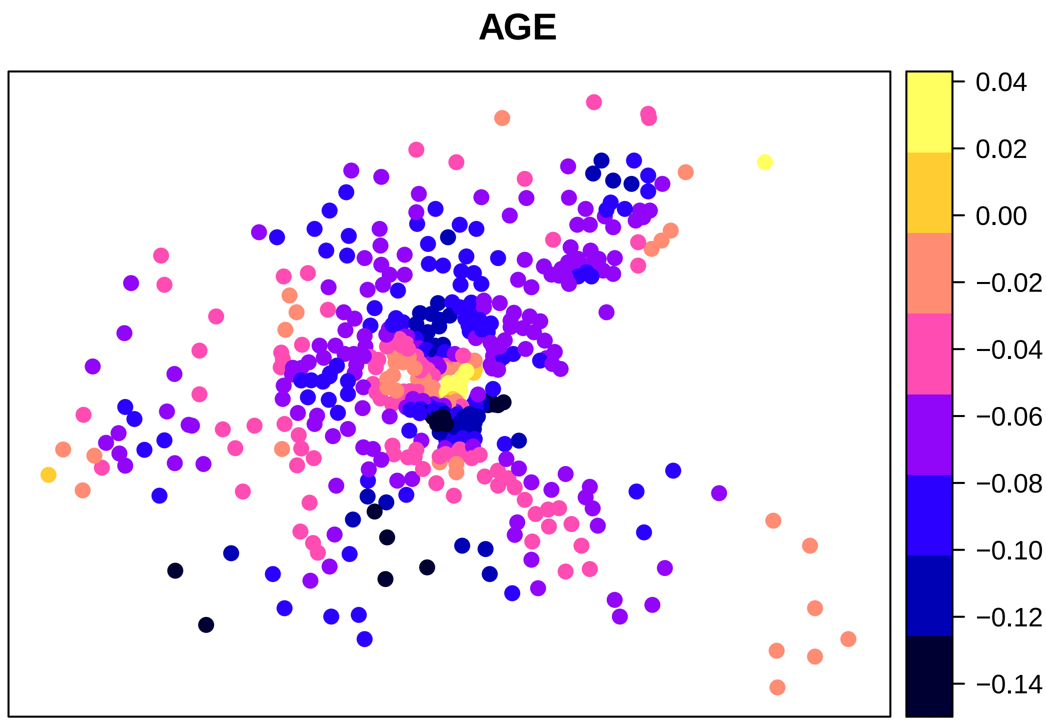
```
plot_s(res,0) # Spatially varying intercept
```




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




```
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:

```
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:

```
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:

```
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"
```

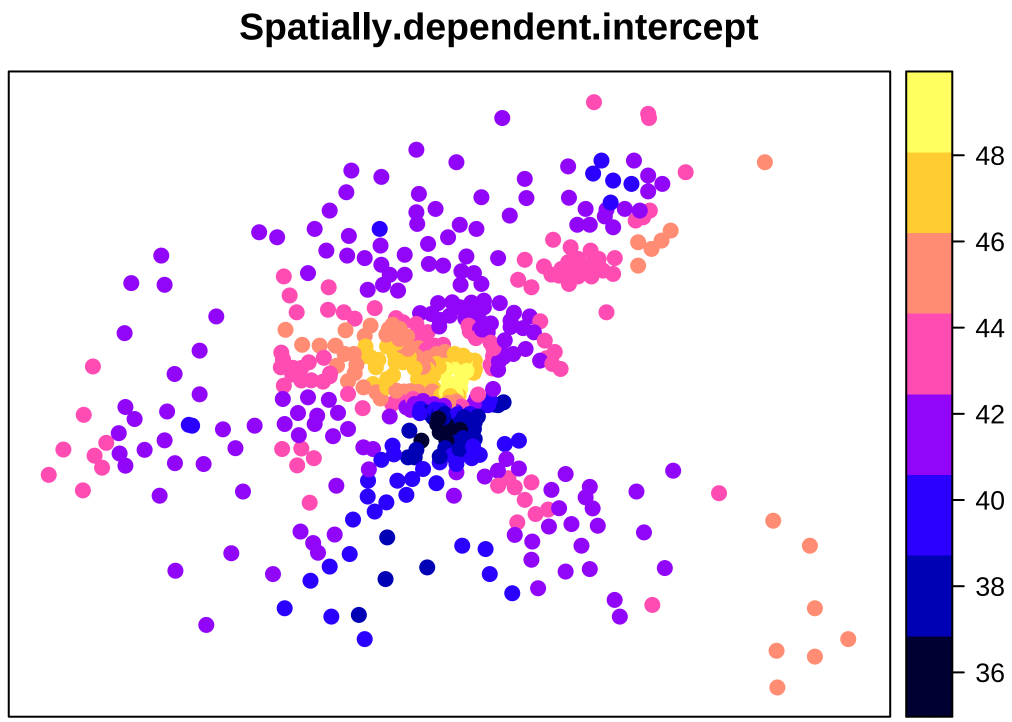
```
res
```

```
## Call:
## resf_vc(y = y, x = x, xconst = xconst, x_nvc = TRUE, xconst_nvc = TRUE,
##     meig = meig)
##
## ----Spatially and non-spatially varying coefficients on x (summary)----
##
## Coefficient estimates:
##   (Intercept)         CRIM               AGE
##  Min.   :34.97   Min.   :-2.1712   Min.   :-0.07496
##  1st Qu.:40.94   1st Qu.:-0.6141   1st Qu.:-0.07496
##  Median :42.28   Median :-0.4156   Median :-0.07496
##  Mean   :42.43   Mean   :-0.4288   Mean   :-0.07496
##  3rd Qu.:43.77   3rd Qu.:-0.2156   3rd Qu.:-0.07496
##  Max.   :49.94   Max.   : 0.5235   Max.   :-0.07496
##
## Statistical significance:
##                         Intercept CRIM AGE
## Not significant                 0  394   0
## Significant (10% level)         0   15   0
## Significant ( 5% level)         0   29   0
## Significant ( 1% level)       506   68 506
##
## ----Non-spatially varying coefficients on xconst (summary)----
##
## Coefficient estimates:
##        ZN               DIS               RAD               NOX
##  Min.   :0.02511   Min.   :-1.107   Min.   :0.6289   Min.   :-23.31
##  1st Qu.:0.02511   1st Qu.:-1.107   1st Qu.:0.6289   1st Qu.:-19.39
##  Median :0.02511   Median :-1.107   Median :0.6289   Median :-18.49
##  Mean   :0.02511   Mean   :-1.107   Mean   :0.6289   Mean   :-18.56
##  3rd Qu.:0.02511   3rd Qu.:-1.107   3rd Qu.:0.6289   3rd Qu.:-17.58
##  Max.   :0.02511   Max.   :-1.107   Max.   :0.6289   Max.   :-14.48
##       TAX                RM            PTRATIO              B
##  Min.   :-0.01512   Min.   :0.6017   Min.   :-0.6371   Min.   :0.01371
##  1st Qu.:-0.01512   1st Qu.:0.8399   1st Qu.:-0.6371   1st Qu.:0.01371
##  Median :-0.01512   Median :1.0419   Median :-0.6371   Median :0.01371
##  Mean   :-0.01512   Mean   :1.2079   Mean   :-0.6371   Mean   :0.01371
##  3rd Qu.:-0.01512   3rd Qu.:1.3036   3rd Qu.:-0.6371   3rd Qu.:0.01371
##  Max.   :-0.01512   Max.   :3.2998   Max.   :-0.6371   Max.   :0.01371
##
```

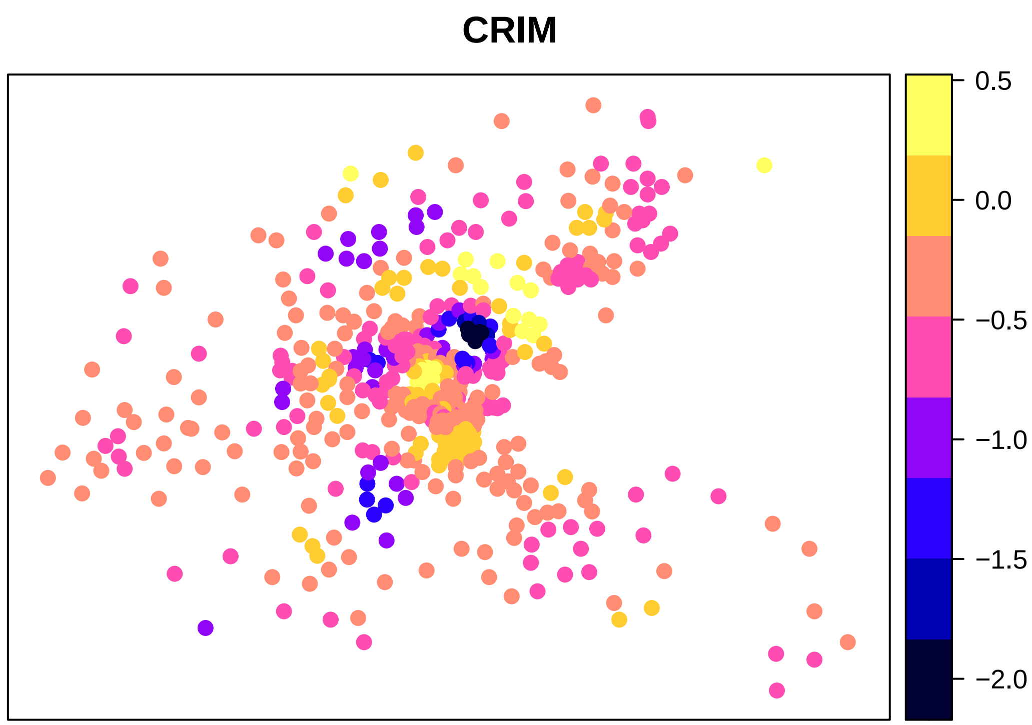
```
## Statistical significance:
##                          ZN DIS RAD NOX TAX  RM PTRATIO   B
## Not significant           0   0   0 185   0 414       0   0
## Significant (10% level) 506   0   0 217   0  27       0   0
## Significant ( 5% level)   0   0   0  40   0  23       0   0
## Significant ( 1% level)   0 506 506  64 506  42     506 506
##
## ----Variance parameters----------------------------------
##
## Spatial effects (coefficients on x):
##                        (Intercept)      CRIM AGE
## random_SE                4.0667763 1.0007384   0
## Moran.I/max(Moran.I)     0.3274953 0.0743859  NA
##
## Non-spatial effects (coefficients on x):
##                   CRIM AGE
## random_SE 0.03405477   0
##
## Non-spatial effects (coefficients on xconst):
##           ZN DIS RAD      NOX TAX       RM PTRATIO B
## random_SE  0   0   0 1.496402   0 0.200113       0 0
##
## ----Estimated probability distribution of y--------------
##                   Estimates
## skewness                  0
## excess kurtosis           0
##
## ----Error statistics-------------------------------------
##                        stat
## resid_SE          3.1945901
## adjR2(cond)       0.8767156
## rlogLik       -1447.2763228
## AIC            2932.5526456
## BIC            3012.8568423
##
## NULL model: lm( y ~ x + xconst )
##    (r)loglik: -1551.857 ( AIC: 3127.715,  BIC: 3178.433 )
```

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:

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

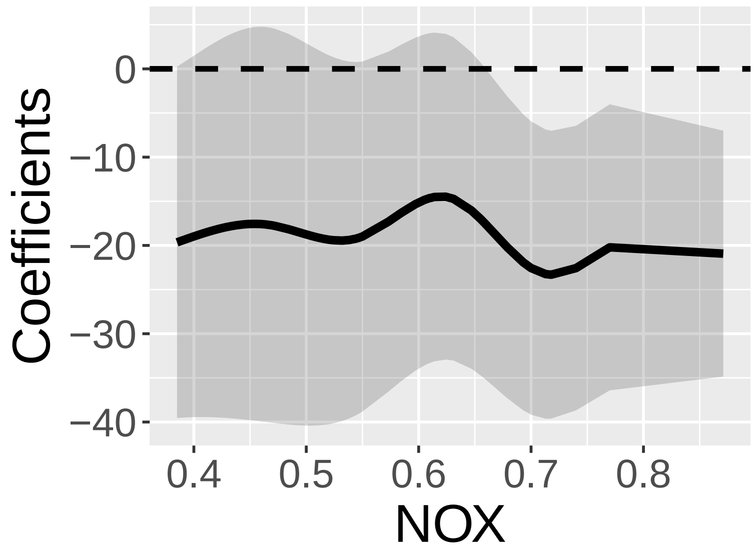



```
plot_s(res,1)             # SNVC on x[,1]
```

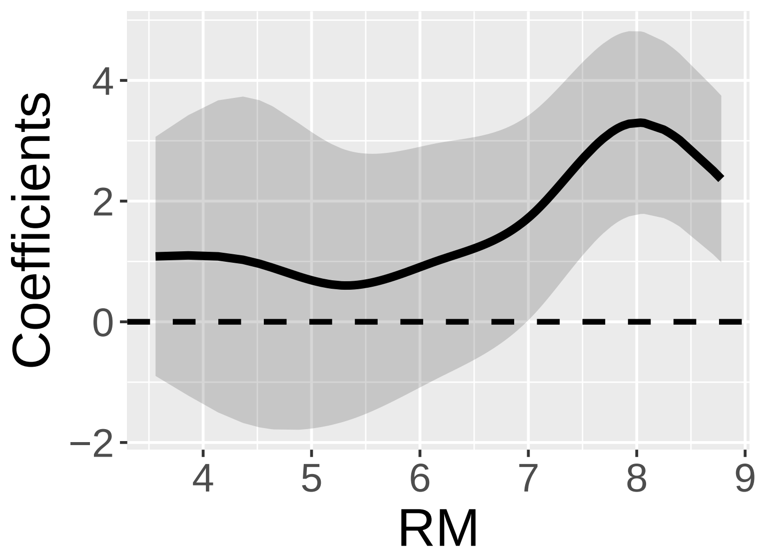


NVCs on xconst is plotted by specifying xtype="xconst" in the plot_n function, as below. The solid line denotes the estimated NVC, and the gray area denotes the 95% confidence interval:

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

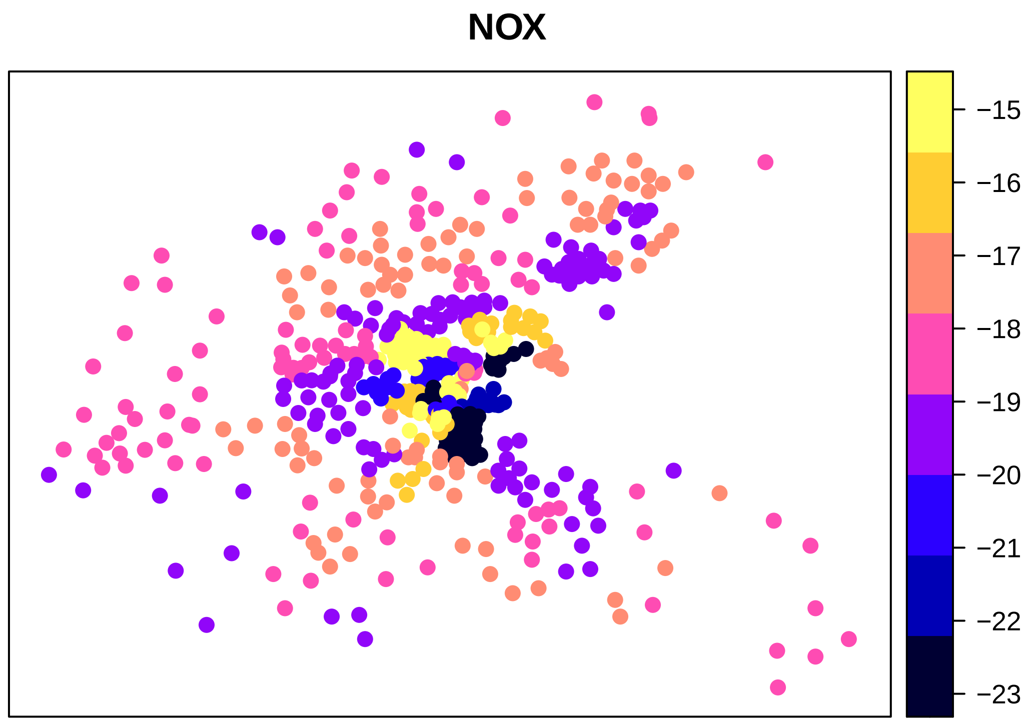



```
plot_n(res,6,xtype="xconst")#NVC on xconst[,6]
```

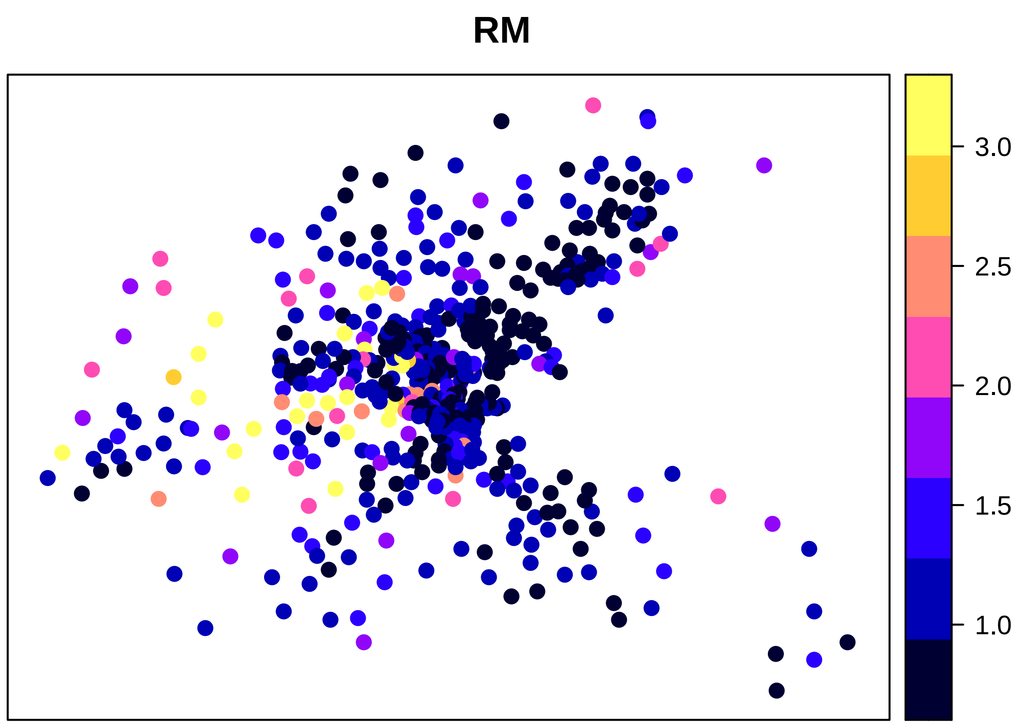


These NVCs can also be plotted spatially as follows:

```
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         -26.38632673 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_SE               5.074794
## Moran.I/max(Moran.I)    0.812936
##
## Group effects:
##           xgroup
## ramdom_SE 4.4404
##
## ----Estimated probability distribution of y--------------
##                 Estimates
## skewness                0
## excess kurtosis         0
##
## ----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 )
```

The estimated independent group effects are extracted as

```
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_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:

```
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:

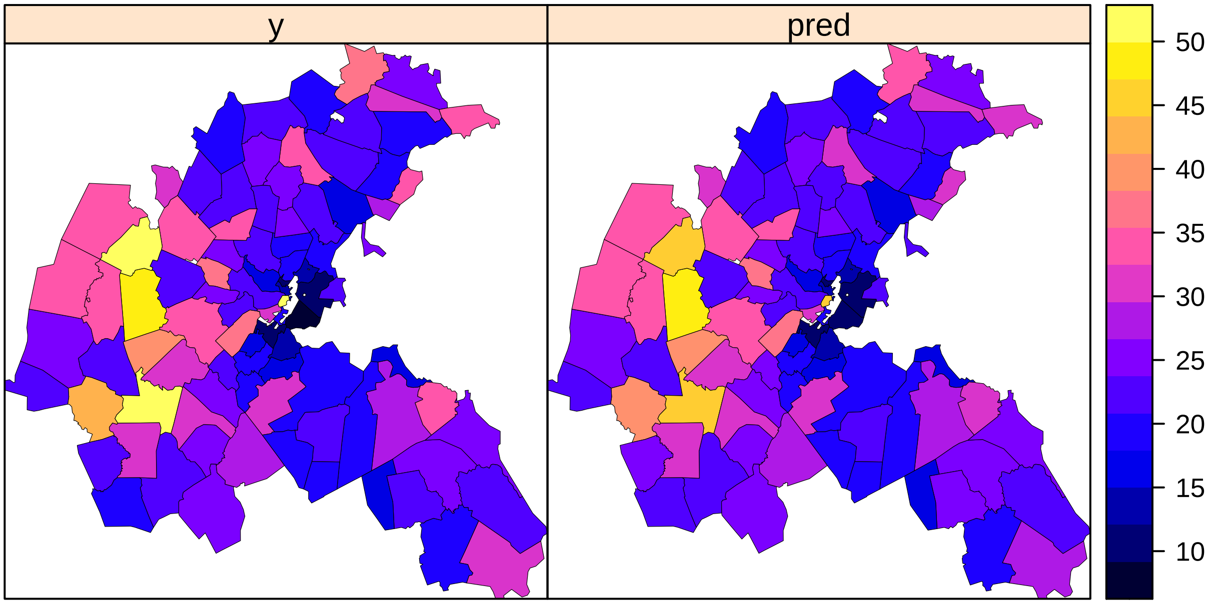
```
require(rgdal)
require(rgeos)
require(dplyr)
boston.tr   <- readOGR(system.file("shapes/boston_tracts.shp",package="spData")[1])
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "/Library/Frameworks/R.framework/Versions/4.0/Resources/library/spData/shapes/boston_tracts.s
## with 506 features
## It has 36 fields
```

```
b1          <- st_as_sf(boston.tr)
b1_dissolve <- b1 %>% group_by(TOWNNO) %>% summarize() #dissolve
boston.tr2  <- as_Spatial(b1_dissolve)
boston.tr2@data$id<- 1:(dim(boston.tr2)[1])
b2_dat      <- boston.tr2@data
b2_dat2     <- merge(b2_dat, adat,by.x="TOWNNO",by.y="Group.1",all.x=TRUE)
```

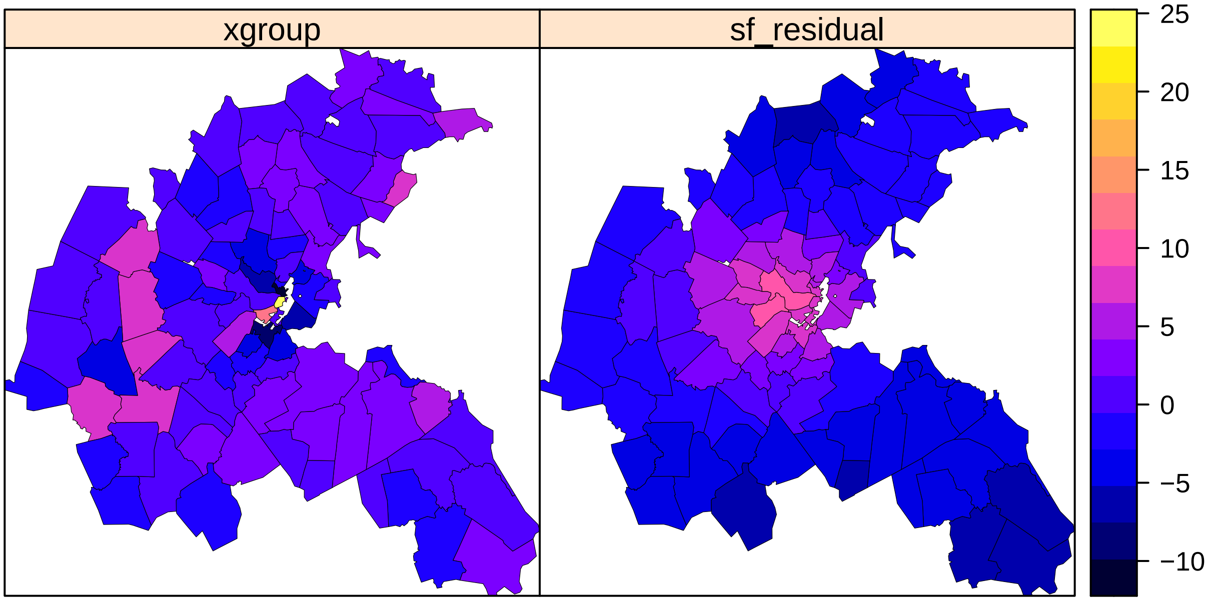
Here are the maps of our estimates. “y” denotes the observed mean prices, and “pred” denotes our predicted standard price. While they are similar, there are some differences in towns with high housing prices.

```
boston.tr2@data<- b2_dat2[order(b2_dat2$id),]
spplot(boston.tr2,c("y","pred"), lwd=0.3)
```

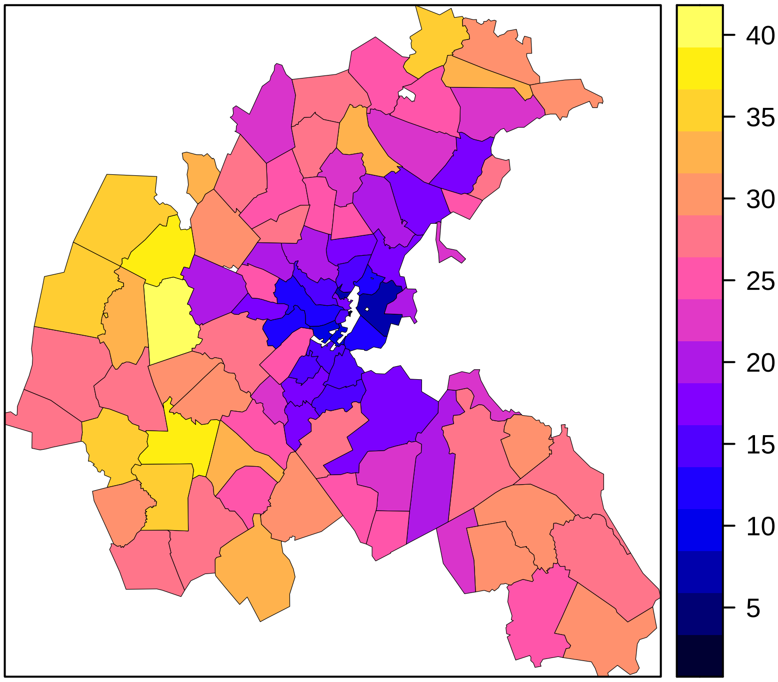


Here are the elements of the predicted values. The maps below show that each element explains different things to each other:

```
spplot(boston.tr2,c("xgroup","sf_residual"), lwd=0.3)
```




```
spplot(boston.tr2,"xb", lwd=0.3)
```



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

```
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: 3074.297"
## [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: 3040.896"
## [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: 3039.588"
## [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: 3039.572"
## [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: 3039.572"
```

```
pred_vc <- predict0_vc(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 23.67839 23.12533   -1.125536 1.678592
## 2       1 28.15000 27.81181 27.44629   -1.966846 2.332368
## 3       2 32.76667 32.28629 31.09675   -2.552106 3.741645
## 4       3 19.42857 19.25653 18.45742   -2.506070 3.305184
## 5       4 16.71364 16.68358 15.40519   -1.025996 2.304387
```

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")
```

```r
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 -86.82645 32.7926
## 2   5.2 -86.82645 32.7926
## 3   4.7 -86.82645 32.7926
## 4   3.9 -86.82645 32.7926
## 5   5.5 -86.82645 32.7926
```

Here are the first five rows of the data:

```r
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 -86.82645 32.7926
## 2   5.2 -86.82645 32.7926
## 3   4.7 -86.82645 32.7926
## 4   3.9 -86.82645 32.7926
## 5   5.5 -86.82645 32.7926
```


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.

```r
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:

```r
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:

```r
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

```r
pmod3
```

```
## Call:
## resf(y = y, x = x, xgroup = xgroup, meig = meig_p)
##
## ----Coefficients------------------------------
##                 Estimate          SE    t_value      p_value
## (Intercept)  2.266458952 0.157678635 14.3739128 0.0000000000
## log_pcap     0.007185026 0.023530593  0.3053483 0.7601856016
## log_pc       0.292350481 0.022207172 13.1646874 0.0000000000
## log_emp      0.732900408 0.024808722 29.5420464 0.0000000000
## unemp       -0.004356469 0.001066686 -4.0841178 0.0000490012
##
## ----Variance parameter------------------------
##
## Spatial effects (residuals):
##                      (Intercept)
## random_SE              0.1555241
## Moran.I/max(Moran.I)   0.3344001
##
## Group effects:
##                 state       year
## ramdom_SE 0.09492895 0.02433059
##
## ----Estimated probability distribution of y--------------
##                 Estimates
## skewness                0
## excess kurtosis         0
##
## ----Error statistics--------------------------
##                      stat
## resid_SE     3.381428e-02
## adjR2(cond)  9.988953e-01
## rlogLik      1.408381e+03
## AIC         -2.796762e+03
## BIC         -2.749718e+03
##
## NULL model: lm( y ~ x )
##    (r)loglik: 826.9817 ( AIC: -1641.963,  BIC: -1613.737 )
```

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    -0.07165160 0.01390024 -5.154702
## state_ARIZONA    -0.04404058 0.01667988 -2.640342
## state_ARKANSAS   -0.07256766 0.01471017 -4.933162
## state_CALIFORNIA  0.24012817 0.01967478 12.204875
## state_COLORADO   -0.11492607 0.01232067 -9.327905
```

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

```
##                 Estimate         SE    t_value
## year_1970 -0.006016459 0.01109130 -0.5424484
## year_1971  0.002901673 0.01056932  0.2745372
## year_1972  0.013281801 0.01041698  1.2750149
## year_1973  0.021949386 0.01028021  2.1351098
## year_1974 -0.009852614 0.00967949 -1.0178857
```

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

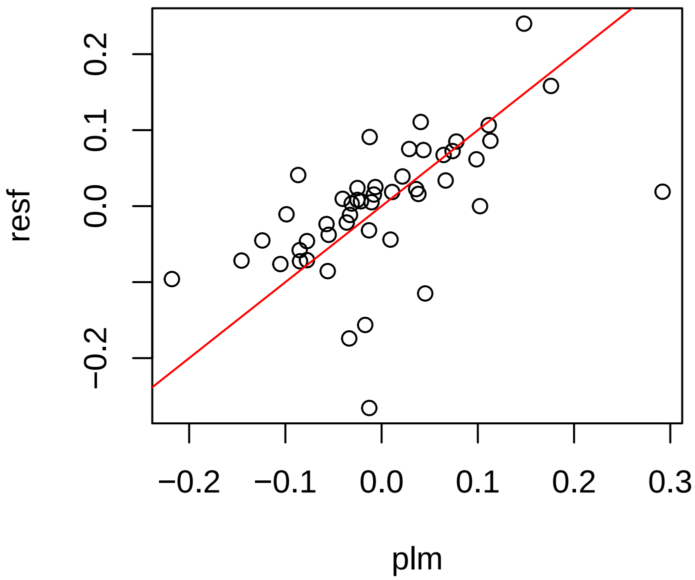
```r
pm0    <- plm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
              data = pdat, effect="twoways",model="random")
pm0
```

```
##
## 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
```

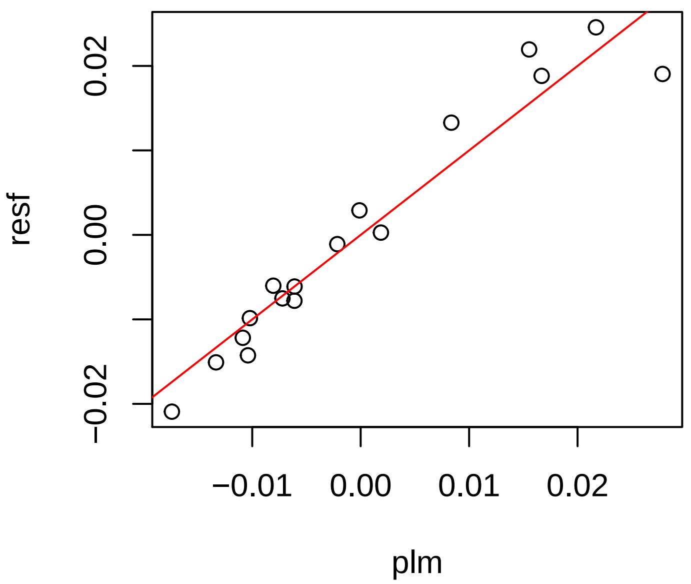
```r
s_g_plm<- ranef(pm0,"individual")
t_g_plm<- ranef(pm0,"time")
```

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.

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




```r
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 perform approximations, their spatial predictions might be less accurate relative to kriging. Instead, they are faster and available for very large samples.

The predict0 function is used for prediction based on the resf or besf function, while the predict0_vc function is used for resf_vc or besf_vc function (see Section 4 for besf and besf_vc functions).

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.

```r
require(spData)
data(house)
dat   <- data.frame(coordinates(house), house@data[,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):

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

coords0<- dat[-samp ,c("long","lat")]
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:

```r
start.time1<-proc.time()###### just 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()###### just 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).

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

```
##         pred       xb sf_residual
## 4  11.13933 10.77552   0.3638097
## 5  11.76268 11.34495   0.4177267
## 7  10.98430 10.57121   0.4130822
## 11 11.56381 11.18246   0.3813527
## 17 10.99135 10.53905   0.4522981
```

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

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

```r
start.time2<-proc.time()###### just 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: 13434.383"
```

```
## [1] "-------  Iteration 2  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 13069.94"
## [1] "-------  Iteration 3  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 13064.91"
## [1] "-------  Iteration 4  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 13064.742"
## [1] "-------  Iteration 5  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 13064.74"
## [1] "-------  Iteration 6  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 13064.737"
## [1] "-------  Iteration 7  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 13064.737"
```

```r
pred02   <- predict0_vc( mod = mod2, x0 = x0, meig0=meig0 )
end.time2<- proc.time()###### just for CP time evaluation
```

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

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

```
##        pred       xb  sf_residual
## 4  11.37177 11.35821  0.013559949
## 5  11.67948 11.69211 -0.012626023
## 7  11.02490 11.03355 -0.008640786
## 11 11.54755 11.52937  0.018178307
```

```
## 17 11.25625 11.27452 -0.018273726
```

```
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] 0.3417898
```

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

```
## elapsed
##   9.686
```

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

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

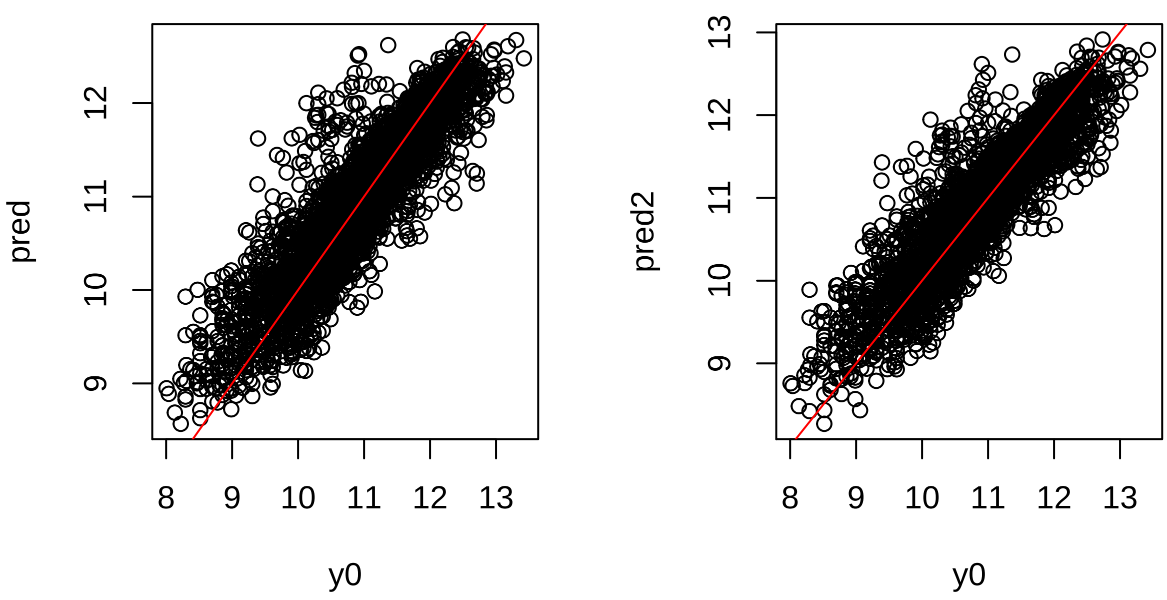
```
## [1] 0.3284294
```

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

```
## elapsed
##  87.103
```

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")
```



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:

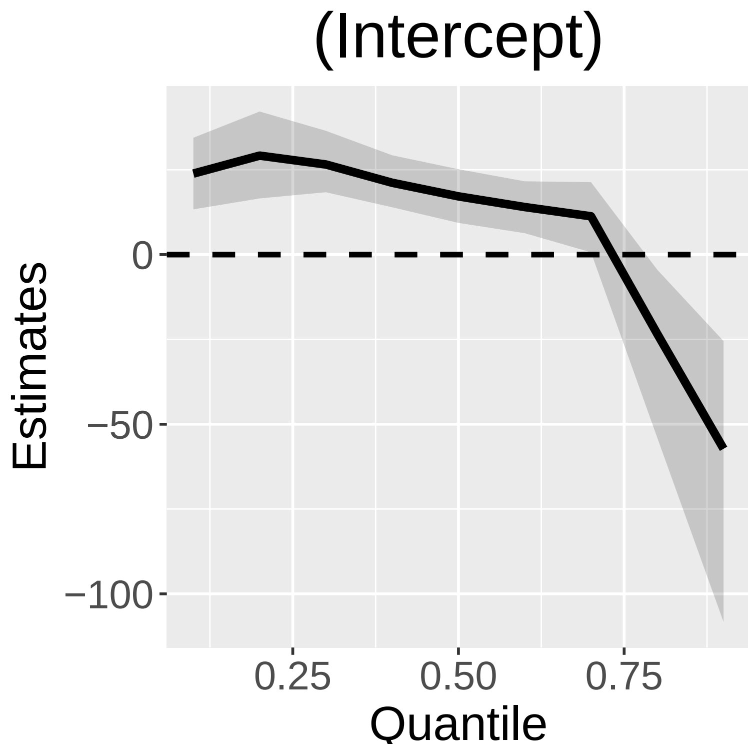
```
res
```

```
## 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
```

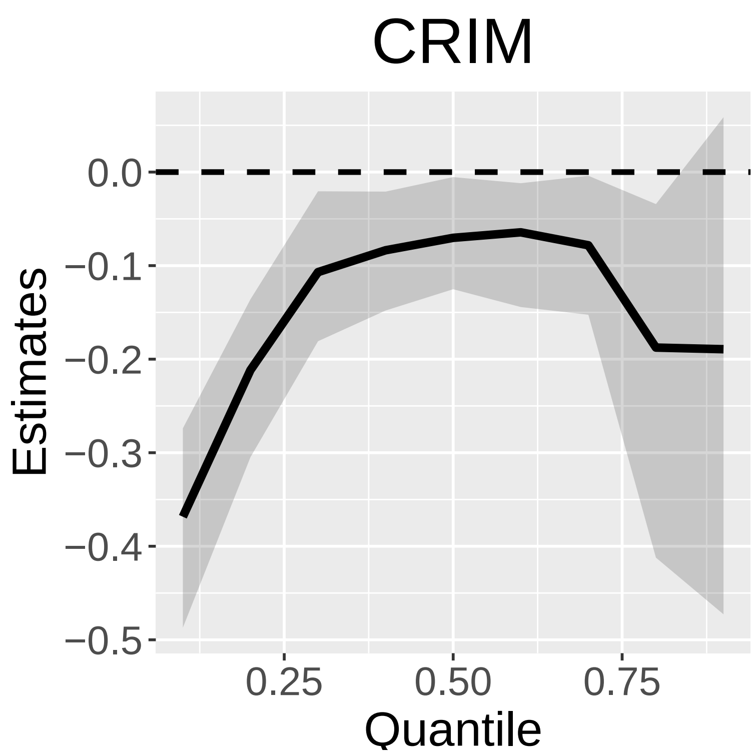
```
## ZN           -0.01169653  -0.01627637  -0.009652286  -0.01947512  -0.008198579
## INDUS         0.25009373   0.03992002  -0.111010420  -0.01521113  -0.096468769
## CHAS          0.98647836   1.28770409   0.438428954   0.26777796  -0.048278485
## NOX         -32.89857783 -23.60303480 -15.109338348 -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_SE                    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_SE                    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.

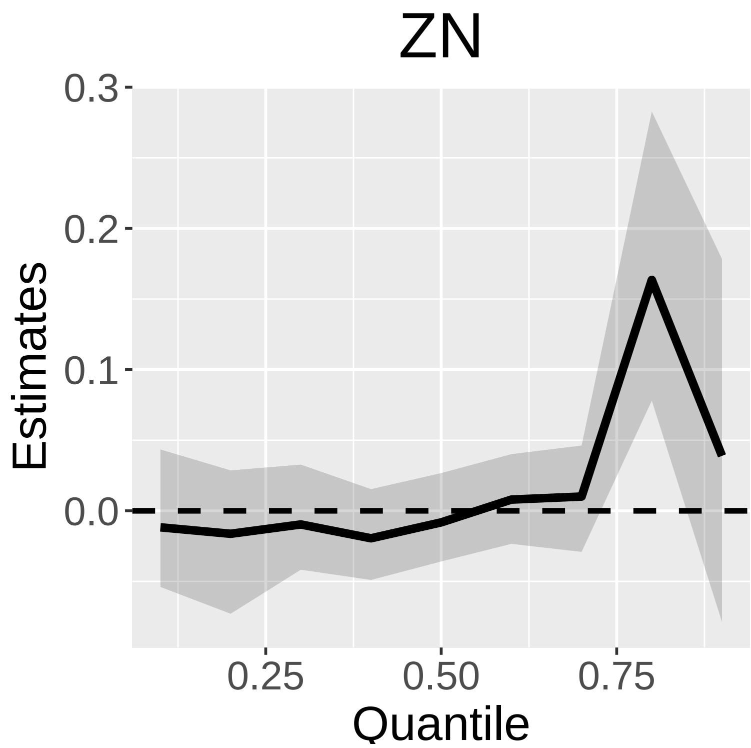
```
plot_qr( res, 1 )
```



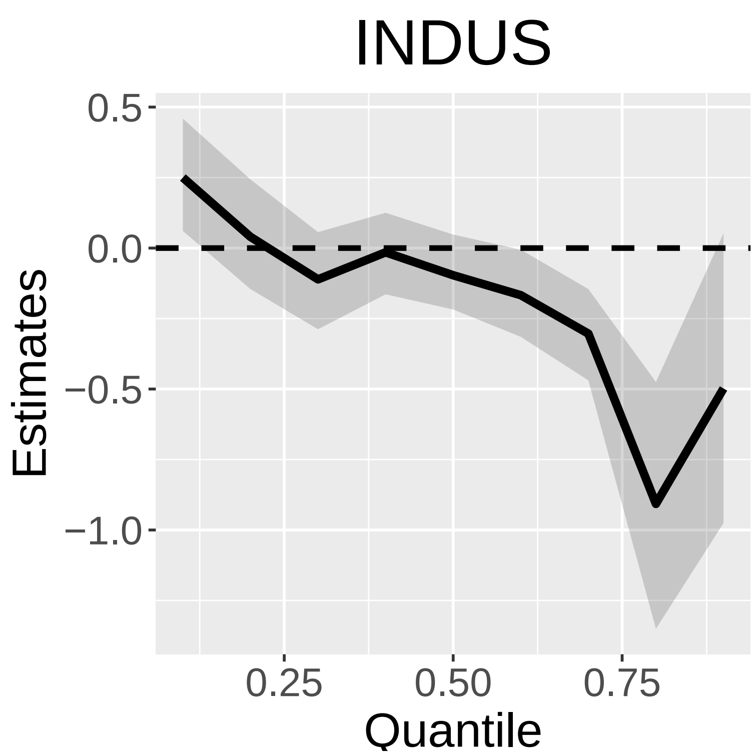

```r
plot_qr( res, 2 )
```



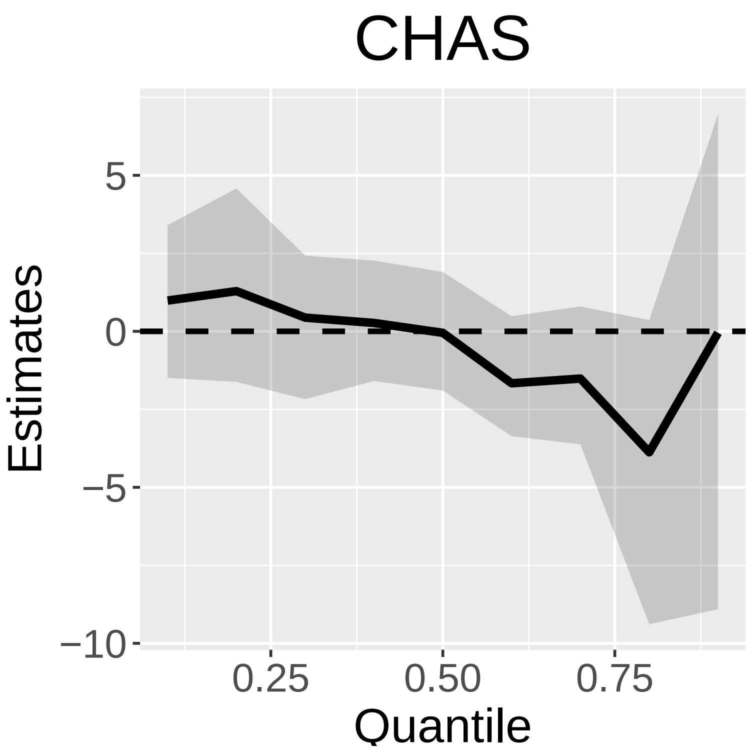

```r
plot_qr( res, 3 )
```




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

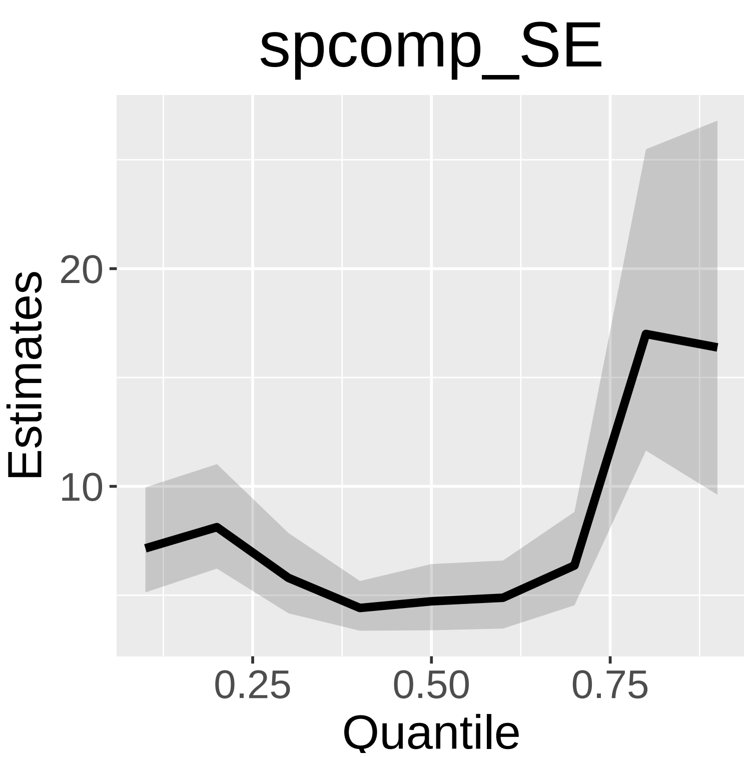



```
plot_qr( res, 5 )
```

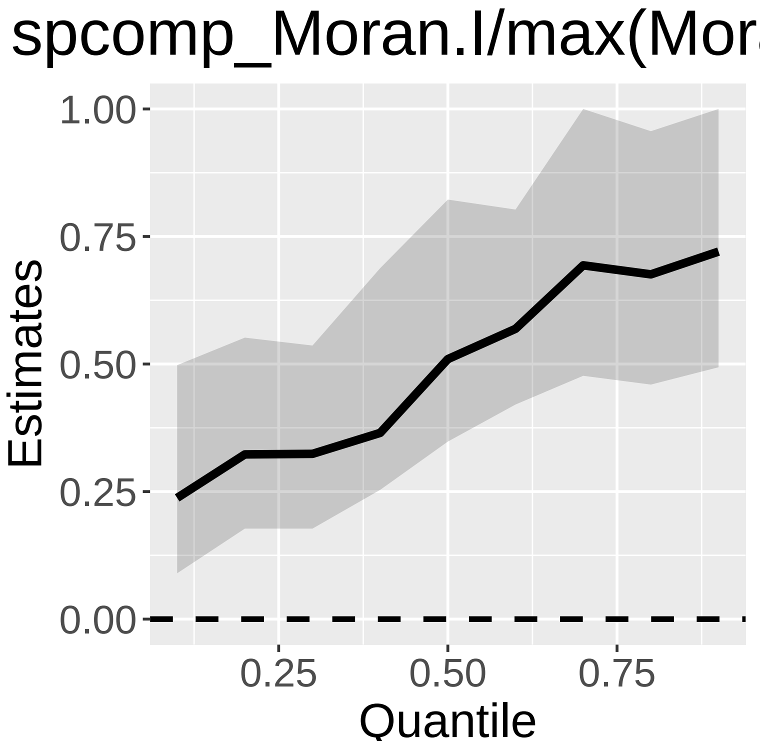


Standard errors and the scaled Moran coefficient (Moran.I/max(Moran.I)), which is a measure of spatial scale by quantile, are plotted if par = “s” is added. Here are the plots:

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



```r
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:

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

```
## OGR data source with driver: ESRI Shapefile
## Source: "/Library/Frameworks/R.framework/Versions/4.0/Resources/library/spData/shapes/boston_tracts.s
## with 506 features
## It has 36 fields
```

```r
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 -0.01506351 0.06551635
## sp_SE  7.54450065  6.53800441 8.48589356
```

```
##
## ----Effects estimates-------------------------
##
## Direct:
##          Estimates     CI_lower    CI_upper p_value
## CRIM  -0.107999852  -0.16109300 -0.03894377    0.00
## ZN     0.002603915  -0.02379086  0.02395807    0.86
## INDUS -0.098956945  -0.22340762  0.01806250    0.12
## CHAS   1.909981199  -0.07594212  3.77480062    0.06
## NOX   -5.119549463 -13.77106206  2.16599546    0.16
## RM     6.947486715   6.24965933  7.59531166    0.00
## AGE   -0.040836844  -0.06754693 -0.01819998    0.00
##
## Indirect:
##           Estimates      CI_lower     CI_upper p_value
## CRIM  -2.227815e-03 -0.0059384109 0.0008846955    0.18
## ZN     5.371341e-05 -0.0006535456 0.0008069910    0.86
## INDUS -2.041278e-03 -0.0079768670 0.0018664258    0.30
## CHAS   3.939898e-02 -0.0264500676 0.1388147474    0.22
## NOX   -1.056058e-01 -0.4628122401 0.0920116372    0.30
## RM     1.433123e-01 -0.0776551604 0.3605923023    0.18
## AGE   -8.423800e-04 -0.0025295003 0.0004555659    0.18
##
## ----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:

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

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_SE      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 Tips for 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:

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

```
## elapsed
##    0.23
```

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

```
## elapsed
##   0.089
```

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

```
## elapsed
##   0.039
```

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
##   5.564
```

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

```
## elapsed
##   1.962
```

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

```
## elapsed
##   1.006
```

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 For very large samples (e.g., millions of samples)

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)
dat   <- data.frame(coordinates(house),
                   house@data[,c("price","age","rooms","beds","syear")])
coords<- dat[ ,c("long","lat")]
y       <- log(dat[,"price"])
x      <- dat[,c("age","rooms","beds","syear")]
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) -59.12935709 2.592858061 -22.80470 4.118061e-115
## age          -0.74853906 0.013345062 -56.09109  0.000000e+00
## rooms         0.10929171 0.002964869  36.86224 1.862098e-297
## beds          0.05111374 0.005025957  10.16995  2.700425e-24
```

```
## syear         0.03494024 0.001299081  26.89612 2.437841e-159
##
## ----Variance parameter------------------------
##
## Spatial effects (residuals):
##                      (Intercept)
## random_SE             0.05150463
## Moran.I/max(Moran.I)  0.37937646
##
## ----Error statistics-------------------------
##                      stat
## resid_SE        0.3380269
## adjR2(cond)     0.8036598
## rlogLik     -9014.1842479
## AIC         18044.3684958
## BIC         18109.4949766
##
## Note: The AIC and BIC values are based on the restricted likelihood.
##       Use method ="ml" for comparison of models with different fixed effects (x)
```

```
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: 16857.433"
## [1] "-------  Iteration 2  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 16385.756"
## [1] "-------  Iteration 3  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 16383.058"
## [1] "-------  Iteration 4  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 16383.011"
## [1] "-------  Iteration 5  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
```

```
## [1] "5/5"
## [1] "BIC: 16383.011"
## [1] "-------  Iteration 6  -------"
## [1] "1/5"
## [1] "2/5"
## [1] "3/5"
## [1] "4/5"
## [1] "5/5"
## [1] "BIC: 16383.01"
```

```
res2
```

```
## Call:
## besf_vc(y = y, x = x, coords = coords)
##
## ----Spatially varying coefficients on x (summary)----
##
## Coefficient estimates:
##   (Intercept)          age               rooms               beds
##  Min.   :-61.54   Min.   :-2.4952   Min.   :0.002297   Min.   :-0.06546
##  1st Qu.:-60.80   1st Qu.:-0.9748   1st Qu.:0.081956   1st Qu.: 0.02733
##  Median :-60.62   Median :-0.6949   Median :0.096380   Median : 0.04257
##  Mean   :-60.62   Mean   :-0.7353   Mean   :0.101118   Mean   : 0.04560
##  3rd Qu.:-60.40   3rd Qu.:-0.4273   3rd Qu.:0.112445   3rd Qu.: 0.06112
##  Max.   :-59.88   Max.   : 0.7182   Max.   :0.226071   Max.   : 0.15729
##      syear
##  Min.   :0.03573
##  1st Qu.:0.03573
##  Median :0.03573
##  Mean   :0.03573
##  3rd Qu.:0.03573
##  Max.   :0.03573
##
## Statistical significance:
##                           Intercept   age rooms  beds syear
## Not significant                   0  3336   102 15351     0
## Significant (10% level)           0   837    44  2782     0
## Significant ( 5% level)           0  1755   198  3817     0
## Significant ( 1% level)       25357 19429 25013  3407 25357
##
## ----Variance parameters----------------------------------
##
## Spatial effects (coefficients on x):
##                      (Intercept)        age       rooms        beds syear
## random_SE             0.04158897 0.06313101 0.004654487 0.005667772     0
## Moran.I/max(Moran.I)  0.18096963 0.13861246 0.110117130 0.059731358    NA
##
## ----Error statistics-------------------------------------
##                      stat
## resid_SE        0.3211358
## adjR2(cond)     0.8227497
## rlogLik     -8120.5195421
## AIC         16269.0390842
## BIC         16383.0104257
##
```

```
## Note: The AIC and BIC values are based on the restricted 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.

I have evaluated the computational time for an SVC modeling using the besf_vc function using a Mac Pro (3.5 GHz, 12-Core Intel Xeon E5 processor with 64 GB memory). The besf_vc function took only 8.0 minutes to estimate the 7 SVCs from 1 million samples. I also confirmed that besf_vc took 70.3 minutes to estimate the same model from 10 million samples. besf and besf_vc are suitable for very large data analysis.

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. (2020) Balancing spatial and non-spatially variations in varying coefficient modeling: a remedy for spurious correlation. Arxiv, 2005:09981.
- 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., 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.