Extending Predictions from Spatial Econometric Models on R

Jean-Sauveur AY <jsay.site@gmail.com>
Raja CHAKIR <chakir@grignon.inra.fr>
Julie LE GALLO <jlegallo@univ-fcomte.fr>

April 17, 2014

Abstract

This document presents an integrative framework to make predictions from spatial autoregressive models (2; 3; 1). It also contains the corresponding **R** code to implement the predictors presented. The code is tangled into the `sppred` function that implements in particular the predictors from LeSage and Pace (2004, 2008) and Kelejian and Prucha (2004) for a large number of specifications of spatial autocorrelation from the `spdep` package (Bivand 2014). To use this code, save the file `sppred` in your working directory then submit `source("sppred.R")` to **R**. Some examples on the `boston` data are also available at section XX. The status of this work is actually under construction, comments are welcome.

`STILL TODO`

- Code the variances and confidence intervals of predictors
- Code the weight matrix for ex-sample predictions
- Code the predictors for `sphet` and `splm` objects

Contents

1 Theoretical Framework 3

2 Current function from `spdep` 5

3 The `sppred` extension 6

4 How it works 10

5 The `boston` example 11

1 Theoretical Framework

1.1 Spatial econometric models

The particularity of spatial econometric models is the *ex ante* specification of interdependence between statistical units, typically from their spatial proximity. Consequently, resulting predictions not only depend on the variables of the target units (i.e., the units for which we want to predict the outcome) but also depend on the variables of their neighbors. This implies that the sample pattern of the outcomes or the residuals contains additional information which may be used to modify the regression function so as to reduce the prediction variance (Goldberger 1962). (2)

The literature about predicting from spatial econometric models is not actually unified, due to different modeling frameworks and notations: see LeSage and Pace (2004, 2008) and Kelejian and Prucha (2004). A first contribution of this document is to unify the different predictors proposed in the literature into an integrative spatial econometric framework. A second contribution is to present the **R** function `sppred` that implements all the spatial predictors and presents some empirical evidence on their relative performance on a well-known dataset (`boston`, Bivand 2014).

We present analytically the available predictors from spatial econometric models estimated on a sample of N spatial units. We begin with the more general spatial autoregressive mixed conditional (SMC) specification of the Cliff-Ord (1973, 1981) class of homoscedastic models with exogenous covariates,[1]

$$\begin{aligned} y &= \alpha + \rho W y + X\beta + WX\theta + u \\ u &= \lambda W u + \varepsilon \quad \text{with} \quad \varepsilon \sim \mathbf{N}(0, \sigma^2 I_N) \end{aligned}$$

The term y is a $N \times 1$ vector of the continuous outcome of interest, X is a $N \times K$ matrix of the K non-constant covariates, and W is a $N \times N$ full-rank spatial weight matrix (see Anselin 1988). We limit ourselves to a same weight matrix in the outcome and error equations, but formally nothing requires this restriction. The unknown parameters (or vectors of parameters) α, ρ, β, θ, λ and σ grouped in Θ have to be estimated, as the vector ε of innovations. Classically, we assume that $\text{diag}(W) = 0$ and $|\rho|, |\lambda| < 1$. The spatial weight matrix W does not need to be symmetric.

This form is sufficiently general that the spatial autoregressive conditional (SAC) model can be recovered with $\theta = 0$, the spatial Durbin model (SDM) model with $\lambda = 0$, the spatial autoregressive (SAR) model with $\theta = \lambda = 0$, the spatial X model (SXM, also called Spatial Durban Error Model by LeSage and Pace, 2009) with $\rho = 0$ and the spatial error model (SEM) can be recovered with $\rho = \theta = 0$. All these models can be estimated by maximum likelihood through the functions `errorsarlm`, `lagsarlm`, and `sacsarlm` of the **R** package `spdep` (Bivand, 2014).

For future reference, note that the general SMC model can be reduced in $y = (I - \rho W)^{-1} X\beta + (I - \rho W)^{-1}(I - \lambda W)^{-1}\varepsilon$, with a distribution that involves all of the model parameters:

$$y \mid X, W, \Theta \sim \mathbf{N}(\mu, \sigma_\varepsilon^2 \Sigma), \tag{1}$$

[1] This terminology comes from LeSage and Pace (2009) and Bivand (2014). This model is also called spatial autoregressive model with autoregressive disturbances, SARAR(1,1), by Kelejian and Prucha (1998). The notations of spatial lag coefficients also depends on authors, we adopt here those of the **R** package `spdep` (Bivand, 2014).

with $\mu = (I - \rho W)^{-1} X\beta$, and $\Sigma = \left[(I - \rho W)(I - \lambda W)(I - \lambda W^\top)(I - \rho W^\top)\right]^{-1}$. According to the literature, we consider hereafter that the set of parameters Θ is known with certainty. This has the advantage of simplifying notations by neglecting a source of uncertainty that depends on the sample size. Moreover, all the parameters are estimated simultaneously and it makes no sens to consider Θ as partially known. Hence, the predictors presented are all conditioned by Θ, which is dropped from conditioning sets for the sake of clarity. So its not the true BLUP but an approximation.

Finally, note that geo-statistical models (Cressie 1990) usually involve specifying spatial dependence through the error process (like in SEM or SXM) as opposed to the spatial lag of the outcome vector (like in SAR or SDM). Predicting from these error models take a simpler form than autoregressive models and can be easily recovered from the theoretical framework presented here. The reverse is not true.

1.2 Implementing predictions

Basically, we consider M target spatial units for which we want to predict the values of the outcome y. These units may or may not be included in the dataset used for the estimation of the parameters. At this moment, it makes no differences and we distinguish them only in the research interest of obtaining predictions of y.[2] We discuss this point in more details at the next subsection XX.

The available information set is crucial here, as each predictor is formulated as a conditional expectation relative to it.

. in terms of conditional expectation put the focus of the available information used to construct the predictors which is important for the transparency of the code.

$$\mathbf{E}(y \mid X) = X\widehat{\beta} \tag{PRD.X}$$

$$\mathbf{E}(y \mid X, W) = X\widehat{\beta} + WX\widehat{\theta} \tag{PRD.WX}$$

$$\mathbf{E}(y \mid X, W) = (I_n - \rho W)^{-1}(X\widehat{\beta} + WX\widehat{\theta}) \tag{PRD.KP1}$$

This correspond to KP1 but also the exogenous predictor of LSP. It is the default predictor of the function `sppred`.

$$\mathbf{E}(y \mid X, W, Wy) = \rho Wy + X\widehat{\beta} + WX\widehat{\theta} + \lambda(Wy - X\beta - WX\theta) \tag{PRD4}$$

It is a bit strange but y cannot be recovered from W and Wy. KP analyses this predictor separately in KP4 and KP5 but such a distinction is not necessary. Since $\text{diag}(W) = 0$, Wy does not use individually the y_i to predict itself but needs them all to predict the entire vector.

The bias of actual predictors come from the correlation between the spatially lagged dependent variable and the error term. It is why the use of best linear unbiased predictor (Golberger) is of particular importance.

[2] In particular, it could be of interest to predict outcome values for units with known y for diagnostic, goodness-of-fit purposes or computations of marginal effects of covariates.

$$\mathbf{E}(y \mid X, W, y_{IS}) = (I_n - \rho W)^{-1} X\widehat{\beta} + WX\widehat{\theta} + \Omega_{21}\Omega_{11}^{-1}(y_{IS} - \mathbf{E}(y_{IS} \mid X, W)) \quad \text{(PRD5)}$$

The covariance can be simplified with precision matrix (Harville) which do not necessitate the inversion of matrix: $\Omega_{21}\Omega_{11}^{-1} = \Psi_{21}\Psi_{11}^{-1}$. LSP also present a leave-one-out (loo) specification of this predictor, that is based on the same conditioning set, except that we consider the predictions individually of each OS units indexed i.

$$\mathbf{E}(y_i \mid X, W, y_{IS}) = (I_n - \rho W)^{-1} X\widehat{\beta} + WX\widehat{\theta} + \Omega_{21}\Omega_{11}^{-1}(y_{IS} - \mathbf{E}(y_{IS} \mid X, W)) \quad \text{(prdLSP)}$$

This simplification comes from...

$$\mathbf{E}(y_i \mid X, W, y_{IS}) = \rho Wy + X\widehat{\beta} + WX\widehat{\theta} + \Omega_{21}\Omega_{11}^{-1}(y_{IS} - \mathbf{E}(y_{IS} \mid X, W)) \quad \text{(prdKP2)}$$

yopla

$$\mathbf{E}(y_i \mid X, W, y_{IS}) = \rho Wy + X\widehat{\beta} + WX\widehat{\theta} + \Omega_{21}\Omega_{11}^{-1}(y_{IS} - \mathbf{E}(y_{IS} \mid X, W)) \quad \text{(prdKP2)}$$

If we put aside the differences in the conditional mean of the exogenous part, this predictor `predKP2` is identical to `prdLSP`.

Can we still maintain the signal trend distinction? Does it the same as direct and indirect effects of covariates?

We develop a framework of prediction from models with interdependent observations.

We implement the KP1 predictors, also called exogenous by LeSage and Pace.

We have to explain the differences between in-sample, out-of-sample and ex-sample in a spatial context. Ex-sample is not necessary linked to temporal, it is also interesting to counterfactual simulations. The prediction in out-of-sample needs a certain spatial embedding between the two spatial samples, not having sampled neighbors does not mean no neighbors. But in a spatial segregative case, this corresponds to a ex-sample case.

1.3 A taxonomy of spatial predictions

In spatial econometrics, the usual distinction between in-sample (IS) and out-of-sample (OS) predictions has firstly to be refined. It is not simply inside our outside the calibration sample but it has also some relations (or not) with it.

2 Current function from `spdep`

Our code is an extension of the function `predict.sarlm()` actually the default function from the package `spdep` (Bivand).

```
library(spdep) ; predict.sarlm
```

predict-sarlm.R

The current function, accessible through previous link, implement different predictor according to the absence of the presence of newdata. For the in-sample predictions (`if(newdata== NULL)`), the predictors are computed as Eq. XX using BLUP. For the out of sample predictions (`if(newdata!= NULL)`), the predictors are computed as Eq. XX using biased and inefficient predictors. It produces inconsistencies by not implementing the same predictions if we put the data that are used to fit the model in the `newdata` argument (cf. XX example below). Another shortcoming of the current function is the class of objects from SEM and SXM: they are not vectors. Lastly, if we put `sacmixed` objects in the current function, they are not recognized as such and produce some errors about matrix dimension.

At the center of this distinction is the observability of the outcome variable y.

Some other particularities are present in the current function. The OS predictor for error models is KP1 but not directly for lag models. For that, we have to put `legacy== FALSE`. The signal is computed by difference for the lag models in out of sample.

3 The `sppred` extension

3.1 General Structure

Here is the general structure of the functions that call sub-functions that are defined below.

This function contents the usual verifications, with 2 more arguments: `cond.set` for the conditional set (see XX) and `mean` for the specification of the structural mean.

It is important that the same predictor is implemented when newdata are NULL or not, as when spatial matrix set.

The scan for the lagged WX is by the presence of "lag." at their name, it has to be changed.

```
sppred <- function(object, newdata = NULL, listw = NULL, yobs= object$y,
                   condset= "DEF", blup = NULL, loo = FALSE, power = NULL,
                   zero.policy = NULL, legacy = TRUE, order = 250,
                   tol= .Machine$double.eps^(3/5), ...) {
    require(spdep)
    ## USUAL VERIFICATIONS
    if (is.null(zero.policy))
        zero.policy <- get("zeroPolicy", envir = spdep:::.spdepOptions)
    stopifnot(is.logical(zero.policy))
    if (is.null(power)) power <- object$method != "eigen"
    stopifnot(is.logical(legacy)) ; stopifnot(is.logical(power))
    ## DETERMINING THE MODEL
    if (object$type== "error"){
        mod <- ifelse(object$etype== "error", "sem", "sxm")
    } else {
        mod <- switch(object$type, "lag"= "sar", "mixed"= "sdm",
                                   "sac"= "sac", "sacmixed"= "smc")
    }
```

```
    ## DATA SHAPING
    if (mod %in% c("sem", "sxm")) {lab= object$lambda ; rho= 0         }
    if (mod %in% c("sar", "sdm")) {lab= 0             ; rho= object$rho}
    if (mod %in% c("sac", "smc")) {lab= object$lambda ; rho= object$rho}
    Wlg <- substr(names(object$coefficients), 1, 4)== "lag."
    B <- object$coefficients[ !Wlg] ; Bl <- object$coefficients[ Wlg]
    if (is.null(newdata)){
        X   <- object$X[, !Wlg]
    } else {
        frm <- formula(object$call)
        mt  <- delete.response(terms(frm, data = newdata))
        mf  <- model.frame(mt, newdata)
        X   <- model.matrix(mt, mf)
        if (any(object$aliased)) X <- X[, -which(object$aliased)]
    }
    ## WEIGHT MATRIX, add an error message
    if (is.null(listw)) lsw <- eval(object$call$listw) else lsw <- listw
    ## PREDICTORS
    if (is.null(blup)){
        pt <- switch(condset, "X"= 1, "XW"= 2, "DEF"= 3, "XWy"= 4)
    } else {
        pt <- switch(blup, "LSP"= 5, "KP2"= 6, "KP3"= 7, "KPG"= 8)
    }
    prdX <- as.vector(X %*% B)
    if (pt> 1) prdWX   <- prdWX(prdX, X, Bl, mod, lsw)
    if (pt> 2 && pt!= 4) prdKP1  <- prdKP1(prdWX, rho, lsw, power, order, tol)
    if (pt> 3){
        prdWXy <- prdWX+
            rho* lag.listw(lsw, yobs)+ lab* lag.listw(lsw, yobs- prdWX)}
    if (pt==5) prdLSP <- prdLSP(prdKP1, rho, lab, lsw, yobs, loo)
    if (pt> 5 && !loo) stop("Set loo= TRUE for this blup predictor")
    if (pt==6){
        prdKP2 <- prdKP2(prdKP1, prdWXy,
                         rho, lab, lsw, yobs, power, order, tol)}
    if (pt==7){
        prdKP3 <- prdKP3(prdKP1, prdWXy,
                         rho, lab, lsw, yobs, power, order, tol)}
    if (pt==8) stop("not implemented")
    prd <- switch(pt, "1"= prdX  , "2"= prdWX , "3"= prdKP1, "4"= prdWXy,
                      "5"= prdLSP, "6"= prdKP2, "7"= prdKP3, "8"= prdKPG)
    class(prd) <- "sppred" ; as.vector(prd)
}
```

we choose to not use `object$tarX` and `object$tarY` for more transparencies. It is clear that we lost from that in terms of computation time. It is easy to predict by conditioning only on "X" because it is the same form for all the spatial models (see equation XX).

3.2 Predictors conditioned on X, W

3.2.1 exogenous predictor

```
prdWX <- function(prdX, X= X, Bl= Bl, mod= mod, lsw= lsw){
    if (!mod %in% c("sxm", "sdm", "smc")){
```

```
        prdWX <- prdX } else {
            K <- ifelse(colnames(X)[ 1] == "(Intercept)", 2, 1)
            m <- ncol(X) ; WX <- matrix(nrow= length(prdX), ncol= m+ 1- K)
            for (k in K: m){
                WX[, k+ 1- K] <- lag.listw(lsw, X[, k])
            }
            prdWX <- prdX+ (WX %*% Bl)
        }
    prdWX
}
```

3.2.2 endogenous predictor

```
prdKP1 <- function(prdWX, rho= rho, lsw= lsw,
                   power= power, order= order, tol= tol){
    if (power){
        W <- as(as_dgRMatrix_listw(lsw), "CsparseMatrix")
        prdKP1 <- c(as(powerWeights(W, rho= rho, X= as.matrix(prdWX),
                                    order= order, tol= tol), "matrix"))
    } else {
        prdKP1 <- c(invIrW(lsw, rho) %*% prdWX)
    }
    prdKP1
}
```

3.3 Predictors conditioned on X, W, y

3.3.1 biased predictors

The predictors equivalent to KP4 and KP5, we do not let the choice (because the omitted combination can be recovered from previous predictors) and we can eventually add a KP6 for SAC and SMC models. The computations are in the general.

3.3.2 BLUP LSP

It can make sens to distinguish one shot to one leave one.

```
prdLSP <- function(prdKP1, rho= rho, lab= lab,
                   lsw= lsw, yobs= yobs, loo= loo){
    ZL <- diag(length(prdKP1))- (lab* listw2mat(lsw))
    ZR <- diag(length(prdKP1))- (rho* listw2mat(lsw))
    Z  <- ZL %*% ZR ; P22 <- t(Z) %*% Z
    if (loo){
        prdLSP <- matrix(NA, ncol= 1, nrow= length(prdKP1))
        for (i in 1: length(prdKP1)){
            prdLSP[ i] <- prdKP1[ i]-
                (P22[-i, i] %*% (yobs[ -i]- prdKP1[ -i])/ P22[i, i])
```

```
        }
    } else {
        P11 <- P22
        prdLSP <- prdKP1+ ((solve(P22) %*% P11 %*% (yobs- prdKP1)))
    }
    prdLSP
}
```

3.3.3 BLUP KP2

```
prdKP2 <- function(prdKP1, prdWXy= prdWXy, rho= rho, lab= lab, lsw= lsw,
                   yobs= yobs, power= power, order= order, tol= tol){
    if (power){
        W <- as(as_dgRMatrix_listw(lsw), "CsparseMatrix")
        GL <- as(powerWeights(W, rho= lab, order= order, tol= tol,
                              X= diag(length(prdWXy))), "matrix")
        GR <- as(powerWeights(W, rho= rho, order= order, tol= tol,
                              X= diag(length(prdWXy))), "matrix")
    } else {
        GL <- invIrW(lsw, rho) ; GR <- invIrW(lsw, lab)
    }
    sum.u <- GL %*% t(GL) ; sum.y <- GR %*% sum.u %*% t(GR)
    prdKP2 <- matrix(NA, ncol= 1, nrow= length(prdWXy))
    for (i in 1: length(prdKP2)){
        WM <- listw2mat(lsw)[i, ]
        rg <- (sum.u[i, ] %*% GR %*% WM)/ (WM %*% sum.y %*% WM)
        prdKP2[ i] <- prdWXy[ i]+ (rg %*% WM %*% (yobs- prdKP1))
    }
    prdKP2
}
```

3.3.4 BLUP KP3

```
prdKP3 <- function(prdKP1, prdWXy= prdWXy, rho= rho, lab= lab, lsw= lsw,
                   yobs= yobs, power= power, order= order, tol= tol){
    if (power){
        W <- as(as_dgRMatrix_listw(lsw), "CsparseMatrix")
        GL <- as(powerWeights(W, rho= lab, order= order, tol= tol,
                              X= diag(length(prdWXy))), "matrix")
        GR <- as(powerWeights(W, rho= rho, order= order, tol= tol,
                              X= diag(length(prdWXy))), "matrix")
    } else {
        GL <- invIrW(lsw, lab) ; GR <- invIrW(lsw, rho)
    }
    sum.u <- GL %*% t(GL) ; sum.y <- GR %*% sum.u %*% t(GR)
    prdKP3 <- matrix(NA, ncol= 1, nrow= length(prdWXy))
    for (i in 1: length(prdKP3)){
        rg <- sum.u[i, ] %*% GR[, -i] %*% solve(sum.y[-i, -i])
        prdKP3[ i] <- prdWXy[ i]+ (rg %*% (yobs[i ]- prdKP1[-i ]))
```

```
    }
    prdKP3
}
```

3.4 Predictors conditioned by hand

Try with

> rg ¡- sum.u[i,] %*% GR %*% solve(sum.y) prdKP3[i] ¡- prdWXy[i]+ (rg %*% (yobs-prdKP1))

4 How it works

4.1 Choosing a type of predictor

Our new `R` function for spatial predictions – called `sppred` for the moment – admits a first additional argument `predictor` that specify the computed predictor. Knowing that predictors corresponding to larger information sets are more complex, flexibility is needed to let the user makes its own trade-off between simplicity and prediction efficiency. The following table define the available predictors.

Table 1: The available values for the new `predictor` argument

`predictor`	label	equation (see XX)
″1″	minimum information	(XX)
″2″	heuristic BLUP	(XX)
″3″	BLUP	(XX)
″4″	heuristic data	(XX)

The `predictor` 4 is currently the default for IS prediction in `predict.sarlm` (it corresponds to the predictor KP4 for lag models and KP5 for error models).

4.2 Specifying

4.3 General structure, usual checks, and IS predictions

Here the code, for the inverse integrating directly the code from powerWeigths?

4.4 The predictors 1 for OS predictions

5 The `boston` example

5.1 Subset the dataset

and plot nb matrix

```
library(spdep) ; data(boston) ; library(rgdal)
bst.sp <- readOGR(system.file("etc/shapes", package= "spdep"),
                  "boston_tracts", verbose= FALSE)
bst.nb <- poly2nb(bst.sp) ; bst.lw <- nb2listw(bst.nb)
set.seed(85)
tst.sp <- bst.sp[cal <- unique(round(runif(250)* 500)), ]
tst.nb <- poly2nb(tst.sp) ; ins.sp <- subset(tst.sp, card(tst.nb)!= 0)
ins.nb <- poly2nb(ins.sp) ; ins.lw <- nb2listw(ins.nb)
tst.sp <- subset(bst.sp,
                 !bst.sp$poltract %in% levels(factor(ins.sp$poltract)))
tst.nb <- poly2nb(tst.sp) ; ots.sp <- subset(tst.sp, card(tst.nb)!= 0)
ots.nb <- poly2nb(ots.sp) ; ots.lw <- nb2listw(ots.nb)
bst <- data.frame(A= rep(row.names(bst.sp), card(bst.nb)), B= unlist(bst.nb))
ins <- data.frame(A= rep(row.names(ins.sp), card(ins.nb)), B= unlist(ins.nb))
ots <- data.frame(A= rep(row.names(ots.sp), card(ots.nb)), B= unlist(ots.nb))
ins$ins <- 1 ; ots$ots <- 1
yop <- data.frame(merge(bst, ins, by= c("A", "B"), all.x= TRUE),
                  "ots"= merge(bst, ots, by= c("A", "B"), all.x= TRUE)[, 3])
par(mar= c(0, 0, 0, 0)) ; plot(bst.sp)
plot(ins.sp, col= "grey30", add= TRUE)
plot(ots.sp, col= "grey70", add= TRUE)
plot(bst.nb, coordinates(bst.sp), add= TRUE, col= "red")
plot(ins.nb, coordinates(ins.sp), add= TRUE, col= "blue")
plot(ots.nb, coordinates(ots.sp), add= TRUE, col= "green")
```

5.2 Estimating the models

```
eqn.bst <- log(CMEDV)~ CRIM    + ZN + INDUS   + CHAS    + I(NOX^2)+
                       I(RM^2) + AGE+ log(DIS)+ log(RAD)+ TAX     +
                       PTRATIO + B  + log(LSTAT)
sem <- errorsarlm(eqn.bst, data= ins.sp, ins.lw, method= "Matrix")
sxm <- errorsarlm(eqn.bst, data= ins.sp,
                  ins.lw, etype= "emixed"   , method= "Matrix")
sar <- lagsarlm(  eqn.bst, data= ins.sp, ins.lw, method= "Matrix")
sdm <- lagsarlm(  eqn.bst, data= ins.sp,
                  ins.lw,  type= "mixed"    , method= "Matrix")
sac <- sacsarlm(  eqn.bst, data= ins.sp, ins.lw, method= "Matrix")
smc <- sacsarlm(  eqn.bst, data= ins.sp,
                  ins.lw,  type= "sacmixed" , method= "Matrix")
AIC(sem, sxm, sar, sdm, sac, smc)[, 2]
```

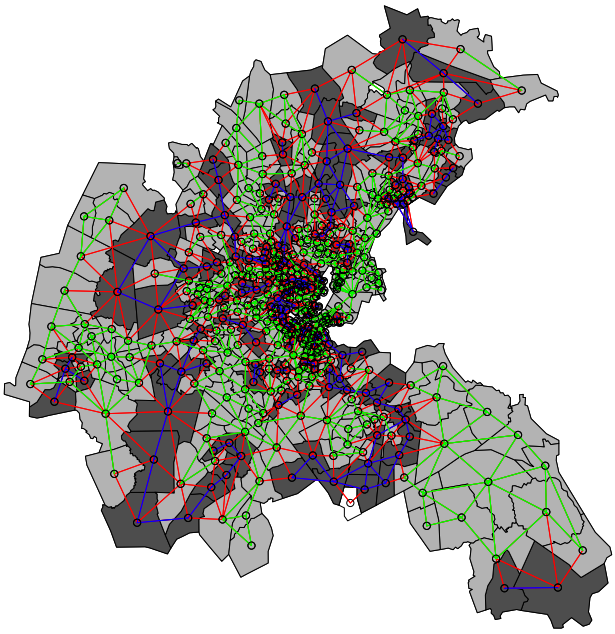

Figure 1: Neighboring relationships for the different Boston subsamples

```
[1] -150.6177 -137.7367 -143.6867 -138.6026 -148.6190 -155.1596
```

The SXM is the more parsimonious in terms of AC

5.3 Predicting

5.3.1 Inefficient predictors

```
source("sppred.R")
rmse <- function(prd, mod) sqrt(mean(I(prd- mod$y)^2))
semKP1 <- sppred(sem) ; sxmKP1 <- sppred(sxm) ; sarKP1 <- sppred(sar)
sdmKP1 <- sppred(sdm) ; sacKP1 <- sppred(sac) ; smcKP1 <- sppred(smc)
rmse(semKP1, sem) ; rmse(sxmKP1, sxm) ; rmse(sarKP1, sar)
rmse(sdmKP1, sdm) ; rmse(sacKP1, sac) ; rmse(smcKP1, smc)

semX <- sppred(sem, condset= "X")
sxmX <- sppred(sxm, condset= "X")
sarX <- sppred(sar, condset= "X")
sdmX <- sppred(sdm, condset= "X")
sacX <- sppred(sac, condset= "X")
smcX <- sppred(smc, condset= "X")
rmse(semX, sem) ; rmse(sxmX, sxm) ; rmse(sarX, sar)
rmse(sdmX, sdm) ; rmse(sacX, sac) ; rmse(smcX, smc)

semXW <- sppred(sem, condset= "XW")
sxmXW <- sppred(sxm, condset= "XW")
sarXW <- sppred(sar, condset= "XW")
sdmXW <- sppred(sdm, condset= "XW")
```

```
sacXW <- sppred(sac, condset= "XW")
smcXW <- sppred(smc, condset= "XW")
rmse(semXW, sem) ; rmse(sxmXW, sxm) ; rmse(sarXW, sar)
rmse(sdmXW, sdm) ; rmse(sacXW, sac) ; rmse(smcXW, smc)

semXWy <- sppred(sem, condset= "XWy")
sxmXWy <- sppred(sxm, condset= "XWy")
sarXWy <- sppred(sar, condset= "XWy")
sdmXWy <- sppred(sdm, condset= "XWy")
sacXWy <- sppred(sac, condset= "XWy")
smcXWy <- sppred(smc, condset= "XWy")
rmse(semXWy, sem) ; rmse(sxmXWy, sxm) ; rmse(sarXWy, sar)
rmse(sdmXWy, sdm) ; rmse(sacXWy, sac) ; rmse(smcXWy, smc)
```

5.3.2 BLUP predictors

```
source("sppred.R")
rmse <- function(prd, mod) sqrt(mean(I(prd- mod$y)^2))

semLSP <- sppred(sem, blup= "LSP", loo= TRUE)
sxmLSP <- sppred(sxm, blup= "LSP", loo= TRUE)
sarLSP <- sppred(sar, blup= "LSP", loo= TRUE)
sdmLSP <- sppred(sdm, blup= "LSP", loo= TRUE)
sacLSP <- sppred(sac, blup= "LSP", loo= TRUE)
smcLSP <- sppred(smc, blup= "LSP", loo= TRUE)
rmse(semLSP, sem) ; rmse(sxmLSP, sxm) ; rmse(sarLSP, sar)
rmse(sdmLSP, sdm) ; rmse(sacLSP, sac) ; rmse(smcLSP, smc)


semKP2 <- sppred(sem, blup= "KP2", loo= TRUE)
sxmKP2 <- sppred(sxm, blup= "KP2", loo= TRUE)
sarKP2 <- sppred(sar, blup= "KP2", loo= TRUE)
sdmKP2 <- sppred(sdm, blup= "KP2", loo= TRUE)
sacKP2 <- sppred(sac, blup= "KP2", loo= TRUE)
smcKP2 <- sppred(smc, blup= "KP2", loo= TRUE)
rmse(semKP2, sem) ; rmse(sxmKP2, sxm) ; rmse(sarKP2, sar)
rmse(sdmKP2, sdm) ; rmse(sacKP2, sac) ; rmse(smcKP2, smc)

semKP3 <- sppred(sem, blup= "KP3", loo= TRUE, power= TRUE)
sxmKP3 <- sppred(sxm, blup= "KP3", loo= TRUE, power= TRUE)
sarKP3 <- sppred(sar, blup= "KP3", loo= TRUE, power= TRUE)
sdmKP3 <- sppred(sdm, blup= "KP3", loo= TRUE, power= TRUE)
sacKP3 <- sppred(sac, blup= "KP3", loo= TRUE, power= TRUE)
smcKP3 <- sppred(smc, blup= "KP3", loo= TRUE, power= TRUE)
rmse(semKP3, sem) ; rmse(sxmKP3, sxm) ; rmse(sarKP3, sar)
rmse(sdmKP3, sdm) ; rmse(sacKP3, sac) ; rmse(smcKP3, smc)

yop <- sdmKP3- sdmXWy
yap <- sdmLSP- sdmKP1
```

5.4 Sample

```
load("Data/exsmp.Rda") ; library(spdep)
plot(exsmp$Dat.all)
plot(exsmp$Dat.cal, col= "blue", pch= 20, add= TRUE)
```

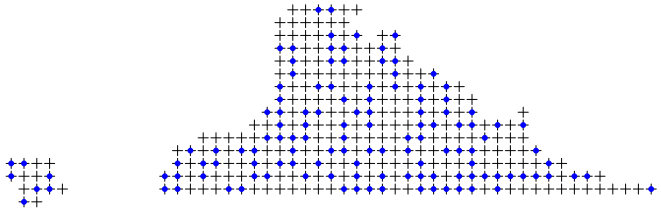

Figure 2: Calibration and exhaustive datasets

5.5 Estimating the spatial models

```
SEM <- errorsarlm(ARlog03~ PXLB03+ RTFO03+ BdAlti, data= exsmp$Dat.cal,
                  exsmp$Wgt.cal, method= "eigen")
SXM <- errorsarlm(ARlog03~ PXLB03+ RTFO03+ BdAlti, data= exsmp$Dat.cal,
                  exsmp$Wgt.cal, method= "eigen", etype= "emixed")
SAR <- lagsarlm(  ARlog03~ PXLB03+ RTFO03+ BdAlti, data= exsmp$Dat.cal,
                  exsmp$Wgt.cal, method= "eigen")
SDM <- lagsarlm(  ARlog03~ PXLB03+ RTFO03+ BdAlti, data= exsmp$Dat.cal,
                  exsmp$Wgt.cal, method= "eigen", type= "mixed")
SAC <- sacsarlm(  ARlog03~ PXLB03+ RTFO03+ BdAlti, data= exsmp$Dat.cal,
                  exsmp$Wgt.cal, method= "eigen")
SMC <- sacsarlm(  ARlog03~ PXLB03+ RTFO03+ BdAlti, data= exsmp$Dat.cal,
                  exsmp$Wgt.cal, method= "eigen", type= "sacmixed")
library(plyr)
t(ldply(list(SEM, SXM, SAR, SDM, SAC, SMC), AIC))
```

```
       [,1]     [,2]     [,3]     [,4]     [,5]    [,6]
V1 445.7127 433.3333 435.5886 434.1438 436.3016 435.197
```

5.6 Constructing spatial weight matrix

```
str(exsmp$Dat.cal)
```

```
attr(exsmp$Wgt.cal, "region.id")
attr(exsmp$Wgt.all, "region.id")
attr(exsmp$Dat.cal@coords, "dimnames")
attr(exsmp$Dat.cal@coords, "dimnames")
mrgOS <- coordinates(exsmp$Dat.all)
mrgIS <- coordinates(exsmp$Dat.cal)
yop <- merge(mrgOS)
```

5.7 Testing the predictors

5.7.1 Conditioned on X

```
source("sppred.R")
SEMprdX <- sppred(SEM, condset= "X")
sqrt(mean(I(SEMprdX- SEM$y)^2))
SXMprdX <- sppred(SXM, condset= "X")
sqrt(mean(I(SXMprdX- SXM$y)^2))
SARprdX <- sppred(SAR, condset= "X")
sqrt(mean(I(SARprdX- SAR$y)^2))
SDMprdX <- sppred(SDM, condset= "X")
sqrt(mean(I(SDMprdX- SDM$y)^2))
SACprdX <- sppred(SAC, condset= "X")
sqrt(mean(I(SACprdX- SAC$y)^2))
SMCprdX <- sppred(SMC, condset= "X")
sqrt(mean(I(SMCprdX- SMC$y)^2))
SMCprdX <- sppred(SMC, newdata= exsmp$Dat.cal, condset= "X", power= T)
sqrt(mean(I(SMCprdX- SMC$y)^2))

## A ESSAYER AVEC L'AUTRE BASE SPATIALE
```

5.7.2 Conditioned on X, W

```
source("sppred.R")
SEMprdWX <- sppred(SEM, condset= "XW")
sqrt(mean(I(SEMprdWX- SEM$y)^2))
SXMprdWX <- sppred(SXM, condset= "XW")
sqrt(mean(I(SXMprdWX- SXM$y)^2))
SARprdWX <- sppred(SAR, condset= "XW")
sqrt(mean(I(SARprdWX- SAR$y)^2))
SDMprdWX <- sppred(SDM, condset= "XW")
sqrt(mean(I(SDMprdWX- SAR$y)^2))
SACprdWX <- sppred(SAC, condset= "XW")
sqrt(mean(I(SACprdWX- SAR$y)^2))
SMCprdWX <- sppred(SMC, condset= "XW")
sqrt(mean(I(SMCprdWX- SAR$y)^2))
SXMprdXW <- sppred(SXM, newdata= exsmp$Dat.cal,
                   condset= "XW", listw= exsmp$Wgt.cal)
sqrt(mean(I(SXMprdWX- SXM$y)^2))
```

5.7.3 KP1 predictors

```
source("sppred.R")
SEMprdKP1 <- sppred(SEM)
sqrt(mean(I(SEMprdKP1- SEM$y)^2))
SXMprdKP1 <- sppred(SXM)
sqrt(mean(I(SXMprdKP1- SXM$y)^2))
SARprdKP1 <- sppred(SAR)
sqrt(mean(I(SARprdKP1- SAR$y)^2))
SDMprdKP1 <- sppred(SDM)
sqrt(mean(I(SDMprdKP1- SAR$y)^2))
SACprdKP1 <- sppred(SAC)
sqrt(mean(I(SACprdKP1- SAR$y)^2))
SMCprdKP1 <- sppred(SMC)
sqrt(mean(I(SMCprdKP1- SAR$y)^2))

yop <- sppred(SXM, condset= "XW")
WWW <- as(as_dgRMatrix_listw(exsmp$Wgt.cal), "CsparseMatrix")
pp <- c(as(powerWeights(WWW, rho= 0, X= as.matrix(yop)), "matrix"))

SXMprdKP1 <- sppred(SXM, newdata= exsmp$Dat.cal)
sqrt(mean(I(SXMprdKP1- SXM$y)^2))
SXMprdKP1 <- sppred(SXM, power= TRUE)
SXMprdKP1 <- sppred(SXM, newdata= exsmp$Dat.cal, listw= exsmp$Wgt.cal)
SXMprdKP1 <- sppred(SXM, newdata= exsmp$Dat.cal, listw= exsmp$Wgt.cal,
                    power= TRUE)
sqrt(mean(I(SXMprdKP1- SXM$y)^2))
```

5.7.4 Biased predictor

```
source("sppred.R")
SEMprdXWy <- sppred(SEM, condset= "XWy")
sqrt(mean(I(SEMprdX- SEM$y)^2))
sqrt(mean(I(SEMprdWX- SEM$y)^2))
sqrt(mean(I(SEMprdKP1- SEM$y)^2))
sqrt(mean(I(SEMprdXWy- SEM$y)^2))

SXMprdXWy <- sppred(SXM, condset= "XWy")
sqrt(mean(I(SXMprdX- SXM$y)^2))
sqrt(mean(I(SXMprdXW- SXM$y)^2))
sqrt(mean(I(SXMprdKP1- SXM$y)^2))
sqrt(mean(I(SXMprdXWy- SXM$y)^2))

SARprdXWy <- sppred(SAR, condset= "XWy")
sqrt(mean(I(SARprdX- SAR$y)^2))
sqrt(mean(I(SARprdWX- SAR$y)^2))
sqrt(mean(I(SARprdKP1- SAR$y)^2))
sqrt(mean(I(SARprdXWy- SAR$y)^2))

SDMprdXWy <- sppred(SDM, condset= "XWy")
sqrt(mean(I(SDMprdX- SDM$y)^2))
sqrt(mean(I(SDMprdWX- SDM$y)^2))
```

```
sqrt(mean(I(SDMprdKP1- SDM$y)^2))
sqrt(mean(I(SDMprdXWy- SDM$y)^2))

SACprdXWy <- sppred(SAC, condset= "XWy")
sqrt(mean(I(SACprdX- SAC$y)^2))
sqrt(mean(I(SACprdWX- SAC$y)^2))
sqrt(mean(I(SACprdKP1- SAC$y)^2))
sqrt(mean(I(SACprdXWy- SAC$y)^2))

SMCprdXWy <- sppred(SMC, condset= "XWy")
sqrt(mean(I(SMCprdX- SMC$y)^2))
sqrt(mean(I(SMCprdWX- SMC$y)^2))
sqrt(mean(I(SMCprdKP1- SMC$y)^2))
sqrt(mean(I(SMCprdXWy- SMC$y)^2))

SDMprdX <- sppred(SDM, condset= "XWy", power= TRUE)
SDMprdX <- sppred(SDM, condset= "XWy", newdata= exsmp$Dat.cal)
sqrt(mean(I(SDMprdXWy- SDM$y)^2))

## ON PEUT COMMENCER A METTRE DU yobs choisit
```

5.7.5 LSP predictor

```
source("sppred.R")
SEMprdKP1 <- sppred(SEM, blup= "LSP")
sqrt(mean(I(SEMprdKP1- SEM$y)^2))

SEMprdKP1 <- sppred(SEM, blup= "LSP", loo= TRUE)
sqrt(mean(I(SEMprdKP1- SEM$y)^2))


SXMprdXWy <- sppred(SXM, blup= "LSP", loo= TRUE)
sqrt(mean(I(SXMprdXWy- SXM$y)^2))

SARprdXWy <- sppred(SAR, condset= "XWy")
sqrt(mean(I(SARprdXWy- SAR$y)^2))
SDMprdXWy <- sppred(SDM, condset= "XWy")
sqrt(mean(I(SDMprdXWy- SDM$y)^2))
SACprdXWy <- sppred(SAC, condset= "XWy")
sqrt(mean(I(SACprdXWy- SAC$y)^2))
SMCprdXWy <- sppred(SMC, condset= "XWy")
sqrt(mean(I(SMCprdXWy- SMC$y)^2))

SDMprdX <- sppred(SDM, condset= "XWy", power= TRUE)
SDMprdX <- sppred(SDM, condset= "XWy", newdata= exsmp$Dat.cal)
sqrt(mean(I(SDMprdXWy- SDM$y)^2))

## ON PEUT COMMENCER A METTRE DU yobs choisit
```

5.7.6 KP2 predictor

```
source("sppred.R")
SEMprdKP2 <- sppred(SEM, blup= "KP2", loo= TRUE)
sqrt(mean(I(SEMprdKP2- SEM$y)^2))
SXMprdKP2 <- sppred(SXM, blup= "KP2", loo= TRUE)
sqrt(mean(I(SXMprdKP2- SXM$y)^2))
SARprdKP2 <- sppred(SAR, blup= "KP2", loo= TRUE)
sqrt(mean(I(SARprdKP2- SAR$y)^2))
SDMprdKP2 <- sppred(SDM, blup= "KP2", loo= TRUE)
sqrt(mean(I(SDMprdKP2- SDM$y)^2))
SACprdKP2 <- sppred(SAC, blup= "KP2", loo= TRUE)
sqrt(mean(I(SACprdKP2- SAC$y)^2))
SMCprdKP2 <- sppred(SMC, blup= "KP2", loo= TRUE)
sqrt(mean(I(SMCprdKP2- SMC$y)^2))

SMCprdKP2 <- sppred(SMC, condset= "X", blup= "KP2", loo= TRUE)
sqrt(mean(I(SMCprdKP2- SMC$y)^2))

## ON PEUT COMMENCER A METTRE DU yobs choisit
```

5.7.7 KP3 predictor

```
source("sppred.R")
SEMprdKP3 <- sppred(SEM, blup= "KP3", loo= TRUE)
sqrt(mean(I(SEMprdKP3- SEM$y)^2))
yop <- sppred(SEM, blup= "LSP", loo= TRUE)
sqrt(mean(I(yop- SEM$y)^2))

system.time(SXMprdKP3 <- sppred(SXM, blup= "KP3", loo= TRUE, power= FALSE))
sqrt(mean(I(SXMprdKP3- SXM$y)^2))
system.time(yop <- sppred(SXM, blup= "LSP", loo= TRUE, power= TRUE))
sqrt(mean(I(yop- SXM$y)^2))


system.time(SARprdKP3 <- sppred(SAR, blup= "KP3", loo= TRUE))
sqrt(mean(I(SARprdKP3- SAR$y)^2))
system.time(yop <- sppred(SAR, blup= "LSP", loo= TRUE, power= TRUE))
sqrt(mean(I(yop- SAR$y)^2))

a <- SARprdKP3- sppred(SAR, condset= "XWy")
b <- yop- sppred(SAR)
plot(a, b)


SDMprdKP3 <- sppred(SDM, blup= "KP3", loo= TRUE)
sqrt(mean(I(SDMprdKP3- SDM$y)^2))
SACprdKP3 <- sppred(SAC, blup= "KP3", loo= TRUE)
sqrt(mean(I(SACprdKP3- SAC$y)^2))
SMCprdKP3 <- sppred(SMC, blup= "KP3", loo= TRUE, power= TRUE)
sqrt(mean(I(SMCprdKP3- SMC$y)^2))
yop <- sppred(SMC, blup= "LSP", loo= TRUE)
sqrt(mean(I(yop- SMC$y)^2))
```

```
SMCprdKP3 <- sppred(SMC, condset= "X",
                    newdata= exsmp$Dat.cal[ 1: 10,],
                    blup= "KP3", loo= TRUE)
sqrt(mean(I(SMCprdKP3- SMC$y)^2))

yop <- sppred(SMC, blup= "LSP", loo= TRUE)
sqrt(mean(I(SMCprdKP3- SMC$y)^2))

plot(yop, SMCprdKP3)

## ON PEUT COMMENCER A METTRE DU yobs choisit
```

6 Summary

6.1 Changes relative to `predict.sarlm`

- Implement predictions for SARAR and Mixed SARAR models from respectively `sac` and `sacmixed` classes.
- Compute BLUP and almost BLUP spatial predictors
- About the in-sample / out of sample structure (`newdata`)
- About the distinction between trend and signal
- The simplification of the in-sample predictions

6.2 About the intercept

We change the scan of the intercept, in particular in presence of WX in the regression. If W is row standardized, we have to drop the intercept to avoid collinearity. The initial function add the constant at the end of the computations, we only drop the intercept in the presence of WX.

References

[1] **Anselin, L.** (1988). *Spatial Econometrics: Methods and Models*, 4. Springer, Boston.

[2] **Cliff, A. D. and J. K. Ord** (1973). *Spatial Autocorrelation*, 5. Pion, London.

[3] **Cliff, A. D. and J. K. Ord** (1981). *Spatial Processes, Models & Applications*. Pion, London.