Extending Predictions from Spatial Econometric Models on R

Jean-Sauveur AY
<jsay.site@gmail.com>

Julie LE GALLO
<jlegallo@univ-fcomte.fr>

April 12, 2014

Abstract

This document presents an integrative framework to make predictions from spatial autoregressive models (Cliff and Ord 1973, 1981; Anselin 1988). It also contains the corresponding `R` code to implement the predictors presented. The code is gathered 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 predictors for ex-sample predictions
- Code the predictors for `sphet` and `splm` objects

Contents

1 Theoretical Framework

1.1 Spatial Econometric Models

The particularity of spatial econometric models is the structural interdependence between statistical units. Associated predictions not only depend on the variables of the target units but also on the neighboring units. Hence, 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. The literature about predicting from spatial econometric models is not really unified (see XX for the most significant), and one can consider this note as an attempt, with the associated practical framework implemented in `R`.

To present analytically the available predictors, we start with the more general spatial autoregressive mixed conditional (SMC) specification of the Cliff-Ord (1973, 1981) heteroscedasticity class of models with exogenous covariates,[1]

$$\begin{aligned} y &= \rho W y + X\beta + WX\theta + u \\ u &= \lambda W u + \varepsilon \end{aligned}$$

with $u \sim \mathbf{N}(0, \sigma^2 \cdot I_N)$. The y is a $N \times 1$ vector continuous outcome, X is a $N \times K$ matrix of the K covariates, and W is a $N \times N$ 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 ρ, β, θ, λ and σ have to be estimated, as the vector ε of innovations. Classically, we assume that $\operatorname{diag}(W) = 0$ and $|\rho|, |\lambda| < 1$ but 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 can be recovered when $\lambda = 0$, 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 through the functions `errorsarlm`, `lagsarlm`, and `sacsarlm` of the `R` package `spdep` (Bivand, 2014).

The geo-statistical models (Cressie 1993) 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 Making Predictions

Basically, . 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. We consider β known to focus on prediction formula.

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

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

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

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

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.

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

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.

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

This simplification comes from…

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

yopla

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

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.

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) ; print(pt)
    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)
    WM <- listw2mat(lsw)[i, ]
    prdKP2 <- matrix(NA, ncol= 1, nrow= length(prdWXy))
    for (i in 1: length(prdKP2)){
        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 Estimating the models

```r
library(spdep) ; data(boston)
```

5.2 Predicting

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.