Extending Predictions from Spatial Econometric Models on R

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

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

April 9, 2014

Abstract

This document presents a framework and some `R` code – `www.r-project.org` – to make predictions from spatial autoregressive models. In particular, it implements the predictors from LeSage and Pace (2004, 2008) and Kelejian and Prucha (2004) for a large number of autoregressive models from the `spdep` package (Bivand 2014). The status is actually under construction, comments are welcome.

TODO

- Code the variances and confidence intervals of predictors
- Allow different weight matrix between lags and errors
- Code the predictors for objects form `sphet` and `splm`

Contents

1 Major 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

1.1 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.

2 Statistical Framework

2.1 Spatial Econometric

From the more general from of the Cliff-Ord (1973, 1981) homoscedastic class of models with exogenous covariates,[1]

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

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. We limit ourselves to a same weight matrix in the outcome and error equations, but nothing precludes this restriction. The unknown parameters ρ, γ, λ and σ have to be estimated, as the vector u of residuals. Classically, we assume that $\text{diag}(W) = 0$, $\mid \rho \mid < 1$, $\mid \lambda \mid < 1$. *standardization of* W*?* Not the same notations than KP 2007.

By construction, $W_{ii} > 0$ to preclude an observation from directly predicting itself

The geo-statistical models usually involve specifying spatial dependence through the error process as opposed to the spatial lag of the y vector, and these "error models" take a simpler form than autoregressive models

This model is sufficiently general that the SARAR(1,1) model can be recovered with $\theta = 0$ (Kelejian2007) (also called SAC by Biva02, BPGR13), the spatial error model (SEM) can be recovered with $\rho = \theta = 0$, the spatial X model (SXM) with $\rho = 0$, the spatial autoregressive (SAR) model with $\theta = \lambda = 0$; and the spatial Durbin model (SDM) model can be recovered when $\lambda = 0$. Another useful non exclusive distinction is the error models (SEM, SDM and SARAR) and the lag models (SAR, SDM and SARAR)

[1] This model has different names in the literature: spatial autoregressive model with autoregressive disturbances (SARAR(1,1), Kelejian and Prucha, 1998) or Spatial Autoregressive Conditional (SAC, XX). We retain XX here.

2.2 Making Predictions

The bias of actual predictors come from the correlation between the spatially lagged dependent variable and the error term.

Since $W_{ii} = 0$, Wy does not use y_i to predict itself.

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.

3 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.

4 The sppred extension

4.1 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.

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,
                   zero.policy = NULL, condset = "X", avg = "DEF",
                   legacy= TRUE, power= NULL, order= 250,
                   tol= .Machine$double.eps^(3/5), ...) {
    ## USUAL VERIFICATIONS
    if (is.null(zero.policy))
        zero.policy <- get("zeroPolicy", envir = spdep:::.spdepOptions)
    stopifnot(is.logical(zero.policy))
    if (is.null(power)) power <- object$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$type, "lag"= "sar", "mixed"= "sdm",
                                   "sac"= "sac", "sacmixed"= "smc")
    }
    ## DATA SHAPING
    Wlg <- substr(names(object$coefficients), 1, 4)== "lag."
    B <- object$coefficients[ !Wlg]
    if (is.null(newdata)){
        nd  <- FALSE
        X   <- object$X[, !Wlg]
    } else {
        nd  <- TRUE
        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$aliased)]
    }
    ## WEIGHT MATRIX
    if (!nd) lsw <- eval(object$call$listw) else lsw <- listw
    ## THE PREDICTORS
    if (condset== "X") prd <- as.vector(X %*% B)
    if (condset== "XW")
        prd <- prd1(object, mod, nd, B, X, lsw)
    if (condset== "XW" && !mod %in% c("sem", "sxm") && avg == "INV")
        prd <- prd2(object, prd, mod, lsw, power, order, tol)
    if (condset== "XWe")
        prd <- prd3(object, B, X, listw, power, legacy, order, tol)
    if (condset== "XWy")
        prd <- prd4(object, B, X, listw, power, legacy, order, tol)
    if (condset== "XWc")
        prd <- prd5(object, B, X, listw, power, legacy, order, tol)
    class(prd) <- "sppred"
    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).

4.2 Predictors conditioned on X, W

4.2.1 without lagged endogenous

```
prd1 <- function(object, mod= mod, nd= nd, B= B, X= X, lsw= lsw){
    if (mod!= "sem" && nd){
        if (is.null(lsw) || !inherits(lsw, "listw"))
            stop("spatial weights list required")
    }
    if (mod %in% c("sxm", "sdm", "smc")){
        m <- ncol(X)
        K <- ifelse(colnames(object$X)[ 1] == "(Intercept)", 2, 1)
        WX <- matrix(nrow= nrow(X), ncol= m+ 1- K)
        for (k in K: m){
            wx <- lag.listw(lsw, X[, k])
            if (any(is.na(wx)))
                stop("NAs in lagged independent variable")
            WX[, k+ 1- K] <- wx
        }
        prdWX <- cbind(X, WX) %*% object$coefficients
    } else {
        prdWX <- X %*% B
    }
    as.vector(prdWX)
}
```

4.2.2 With lagged endogenous

```
prd2 <- function(object, prd= prd, mod= mod, lsw= lsw,
                 power= power, order= order, tol= tol){
    if (power){
        W <- as(as_dgRMatrix_listw(lsw), "CsparseMatrix")
        prdWXi <- c(as(powerWeights(W, rho= object$rho, X= prd,
                                    order= order, tol= tol), "matrix"))
    } else {
        prdWXi <- c(invIrW(lsw, object$rho) %*% prd)
    }
    as.vector(prdWXi)
}
```

4.3 Predictors conditioned on X, W, e

4.4 Predictors conditioned on X, W, y

4.5 Predictors conditioned by hand

5 How it works

5.1 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).

5.2 Specifying

5.3 General structure, usual checks, and IS predictions

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

5.4 The predictors 1 for OS predictions

6 Testing

6.1 Sample

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

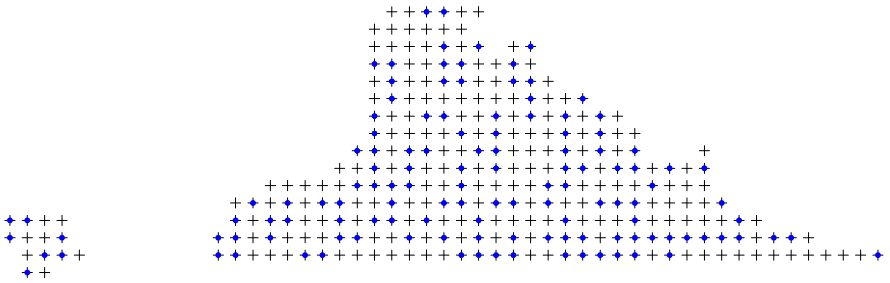

Figure 1: Calibration and exhaustive datasets

6.2 Estimating the spatial models

```
SEM <- errorsarlm(ARlog03~ PXLB03+ RTF003+ BdAlti, data= exsmp$Dat.cal,
                  exsmp$Wgt.cal, method= "eigen")
SXM <- errorsarlm(ARlog03~ PXLB03+ RTF003+ BdAlti, data= exsmp$Dat.cal,
                  exsmp$Wgt.cal, method= "eigen", etype= "emixed")
SAR <- lagsarlm(  ARlog03~ PXLB03+ RTF003+ BdAlti, data= exsmp$Dat.cal,
                  exsmp$Wgt.cal, method= "eigen")
SDM <- lagsarlm(  ARlog03~ PXLB03+ RTF003+ BdAlti, data= exsmp$Dat.cal,
                  exsmp$Wgt.cal, method= "eigen", type= "mixed")
SAC <- sacsarlm(  ARlog03~ PXLB03+ RTF003+ BdAlti, data= exsmp$Dat.cal,
                  exsmp$Wgt.cal, method= "eigen")
SMC <- sacsarlm(  ARlog03~ PXLB03+ RTF003+ 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
```

6.3 Testing the predictors

6.3.1 Conditioned on X

```
source("sppred.R")
SEMprdX <- sppred(SEM, newdata= exsmp$Dat.cal, listw= exsmp$Wgt.cal)
SXMprdX <- sppred(SXM)

SARprdX <- sppred(SAR)
SDMprdX <- sppred(SDM)
```

```
SACprdX <- sppred(SAC)
SMCprdX <- sppred(SMC)
sqrt(mean(I(SEMprdX- SAR$y)^2))
sqrt(mean(I(SXMprdX- SAR$y)^2))
sqrt(mean(I(SARprdX- SAR$y)^2))
sqrt(mean(I(SDMprdX- SAR$y)^2))
sqrt(mean(I(SACprdX- SAR$y)^2))
sqrt(mean(I(SMCprdX- SAR$y)^2))
```

6.3.2 Conditioned on X, W

```
source("sppred.R")
SEMprdX <- sppred(SEM)
SXMprdX <- sppred(SXM)
SARprdX <- sppred(SAR)
SDMprdX <- sppred(SDM)
SACprdX <- sppred(SAC)
SMCprdX <- sppred(SMC)
SEMprdXW <- sppred(SEM, condset= "XW")
SXMprdXW <- sppred(SXM, condset= "XW")
SXMprdXWi <- sppred(SXM, condset= "XW", avg= "INV")
SARprdXW <- sppred(SAR, condset= "XW")
SARprdXWi <- sppred(SAR, condset= "XW", avg= "INV")
SDMprdXW <- sppred(SDM, condset= "XW")
SDMprdXWi <- sppred(SDM, condset= "XW", avg= "INV")
SEMprdXW <- sppred(SEM, condset= "XW", avg= "INV")
SEMprdXW <- sppred(SEM, condset= "XW",
                   newdata= exsmp$Dat.cal, listw= exsmp$Wgt.cal)
SXMprdXW <- sppred(SXM, condset= "XW",
                   newdata= exsmp$Dat.cal, listw= exsmp$Wgt.cal)
summary(SEMprdX)
summary(SEMprdXW)
SXMprdX <- sppred(SXM, condset= "XW")
SXMprdXW <- sppred(SXM, newdata= exsmp$Dat.cal,
                   condset= "XW", listw= exsmp$Wgt.cal)
SARprdX <- sppred(SAR, condset= "X")
SARprdXW <- sppred(SAR, condset= "XW")
sqrt(mean(I(SEMprdX- SAR$y)^2))
sqrt(mean(I(SXMprdX- SAR$y)^2))
sqrt(mean(I(SARprdX- SAR$y)^2))
sqrt(mean(I(SARprdXW- SAR$y)^2))
sqrt(mean(I(SDMprdXWi- SAR$y)^2))
sqrt(mean(I(SACprdX- SAR$y)^2))
sqrt(mean(I(SMCprdX- SAR$y)^2))
```