Estimating Spatial Probit Models in R

by Stefan Wilhelm and Miguel Godinho de Matos

Abstract In this article we present the Bayesian estimation of spatial probit models in R and provide an implementation in the package spatialprobit. We show that large probit models can be estimated with sparse matrix representations and Gibbs sampling of a truncated multivariate normal distribution with the precision matrix. We present three examples and point to ways to achieve further performance gains through parallelization of the Markov Chain Monte Carlo approach.

Introduction

The abundance of geolocation data and social network data has lead to a growing interest in spatial econometric methods, which model a contemporaneous dependence structure using neighboring observations (e.g. friends in social networks).

There are a variety of R packages for spatial models available, an incomplete list includes **spBayes** (Finley and Banerjee, 2013), **spatial** (Venables and Ripley, 2002), geostatistical packages called **geoR** (Diggle and Ribeiro, 2007) and **sgeostat** (Majure and Gebhardt, 2013), **spdep** (Bivand et al., 2013), **sphet** (Piras, 2010), **sna** (Butts, 2013) and **network** (Butts et al., March 1, 2012).

While all of these packages deal with linear spatial models, in this article we focus on a nonlinear model, the spatial probit model, and present the Bayesian estimation first proposed by LeSage (2000). As will be shown below, one crucial point we have been working on was the generation of random numbers of a truncated multivariate normal distribution in very high dimensions.

Together with these high dimensions and the size of problems in spatial models comes the need to work with sparse matrix representations rather than the usual dense `matrix()`. Popular R packages for dealing with sparse matrices are **Matrix** (Bates and Maechler, 2013) and **sparseM** (Koenker and Ng, 2012).

In the next section we first introduce spatial probit models and describe the Bayesian estimation procedure of the SAR probit model in detail. Subsequently we discuss some implementation issues in R, describe the `sarprobit` method in package **spatialprobit** (Wilhelm and de Matos, 2013), compare it against ML and GMM estimation in **McSpatial** (McMillen, 2012) and illustrate the estimation with an example from social networks. We also illustrate how to parallelize the Bayesian estimation with the **parallel** package.

Spatial probit models

The book of LeSage and Pace (2009) is a good starting point and reference for spatial econometric models in general and for limited dependent variable spatial models in particular (chapter 10, pp. 279).

Suppose we have the spatial autoregressive model (SAR model, spatial lag model)

$$\mathbf{z} = \rho \mathbf{W}\mathbf{z} + \mathbf{X}\boldsymbol{\beta} + \epsilon, \quad \epsilon \sim N\left(0, \sigma_\epsilon^2 \mathbf{I}_n\right) \tag{1}$$

for $\mathbf{z} = (z_1, \ldots, z_n)'$ with some fix matrix of covariates $\mathbf{X}$ $(n \times k)$ associated with the parameter vector $\boldsymbol{\beta}$ $(k \times 1)$. The matrix $\mathbf{W}$ $(n \times n)$ is called the spatial weight matrix and captures the dependence structure between neighboring observations such as friends or nearby locations. The term $\mathbf{W}\mathbf{z}$ is a linear combination of neighboring observations. The scalar ρ is the dependence parameter and will assumed $\text{abs}(\rho) < 1$. The $k+1$ model parameters to be estimated are the parameter vector $\boldsymbol{\beta}$ and the scalar ρ.

In a spatial probit model, $\mathbf{z}$ is regarded as a latent variable, which cannot be observed. Instead, the observables are only binary variables y_i (0, 1) as

$$y_i = \begin{cases} 1 & \text{if } z_i \geq 0, \\ 0 & \text{if } z_i < 0. \end{cases}$$

y_i can reflect any binary outcome such as survival, a buy/don't buy decision or a class variable in binary classification problems. For identification, σ_ϵ^2 is often set to $\sigma_\epsilon^2 = 1$.

The data generating process for $\mathbf{z}$ is

$$\begin{aligned} \mathbf{z} &= (\mathbf{I}_n - \rho\mathbf{W})^{-1}\mathbf{X}\boldsymbol{\beta} + (\mathbf{I}_n - \rho\mathbf{W})^{-1}\boldsymbol{\epsilon} \\ \boldsymbol{\epsilon} &\sim N(0, \mathbf{I}_n) \end{aligned}$$

Note that if $\rho = 0$ or $\mathbf{W} = \mathbf{I}_n$, the model reduces to an ordinary probit model, for which Wooldridge (2002, chapter 17) and Albert (2007, section 10.1) are good references. The ordinary probit model can be estimated in R by `glm()`.

Another popular spatial model is the spatial error model (SEM) which takes the form

$$\begin{aligned} &\mathbf{z} = \mathbf{X}\boldsymbol{\beta} + \mathbf{u}, \quad \mathbf{u} = \rho\mathbf{W}\mathbf{u} + \epsilon, \quad \epsilon \sim N\left(0, \sigma_\epsilon^2\mathbf{I}_n\right) \\ &\mathbf{z} = \mathbf{X}\boldsymbol{\beta} + (\mathbf{I}_n - \rho\mathbf{W})^{-1}\boldsymbol{\epsilon} \end{aligned} \tag{2}$$

where as before, only binary variables y_i (0,1) can be observed instead of z_i. Our following discussion focuses on the spatial lag probit model (1), but the estimation of the probit model with spatial errors is covered in **spatialprobit** as well.

Recent studies using spatial probit models include, among others, Marsh et al. (2000), Klier and McMillen (2008) and LeSage et al. (2011).

Bayesian estimation

Maximum Likelihood and GMM estimation of the spatial probit is implemented in the package **McSpatial** (McMillen, 2012) with the methods `spprobitml` and `spprobit`. Here we present the details of Bayesian estimation. Although GMM is massively more efficient computationally than Bayesian MCMC, as a instrumental-variables estimation it will work well only in very large samples. This point is also brought by Franzese et al. (2013), who compares different spatial probit estimation strategies.

The basic idea in Bayesian estimation is to sample from a posterior distribution of the model parameters $p(\mathbf{z}, \boldsymbol{\beta}, \rho|\mathbf{y})$ given the data $\mathbf{y}$ and some prior distributions $p(\mathbf{z})$, $p(\boldsymbol{\beta})$, $p(\rho)$. See for example Albert (2007) and the accompanying package **LearnBayes** for an introduction to Bayesian statistics in R (Albert, 2012). This sampling for the posterior distribution $p(\mathbf{z}, \boldsymbol{\beta}, \rho|\mathbf{y})$ can be realized by a Markov Chain Monte Carlo and Gibbs sampling scheme, where we sample from the following three conditional densities $p(\mathbf{z}|\boldsymbol{\beta}, \rho, \mathbf{y})$, $p(\boldsymbol{\beta}|\mathbf{z}, \rho, \mathbf{y})$ and $p(\rho|\mathbf{z}, \boldsymbol{\beta}, \mathbf{y})$:

1. Given the observed variables $\mathbf{y}$ and parameters $\boldsymbol{\beta}$ and ρ, we have $p(\mathbf{z}|\boldsymbol{\beta}, \rho, \mathbf{y})$ as a truncated multinormal distribution

$$\mathbf{z} \sim N\left((\mathbf{I}_n - \rho\mathbf{W})^{-1}\mathbf{X}\boldsymbol{\beta}, \left[(\mathbf{I}_n - \rho\mathbf{W})'(\mathbf{I}_n - \rho\mathbf{W})\right]^{-1}\right) \tag{3}$$

   subject to $z_i \geq 0$ for $y_i = 1$ and $z_i < 0$ for $y_i = 0$, which can be efficiently sampled from using the method `rtmvnorm.sparseMatrix()` in package **tmvtnorm** (Wilhelm and Manjunath, 2012), see the next section for details.

2. For a normal prior $\boldsymbol{\beta} \sim N(\mathbf{c}, \mathbf{T})$, we can sample $p(\boldsymbol{\beta}|\rho, \mathbf{z}, \mathbf{y})$ from a multivariate normal as

$$\begin{aligned} p\left(\boldsymbol{\beta}|\rho, \mathbf{z}, \mathbf{y}\right) &\propto N\left(\mathbf{c}^*, \mathbf{T}^*\right) \\ \mathbf{c}^* &= \left(\mathbf{X}'\mathbf{X} + \mathbf{T}^{-1}\right)^{-1}\left(\mathbf{X}'\mathbf{S}\mathbf{z} + \mathbf{T}^{-1}\mathbf{c}\right) \\ \mathbf{T}^* &= \left(\mathbf{X}'\mathbf{X} + \mathbf{T}^{-1}\right)^{-1} \\ \mathbf{S} &= \left(\mathbf{I}_n - \rho\mathbf{W}\right) \end{aligned} \tag{4}$$

   The standard way for sampling from this distribution is `rmvnorm()` from package **mvtnorm** (Genz et al., 2012).

3. The remaining conditional density $p(\rho|\boldsymbol{\beta}, \mathbf{z}, \mathbf{y})$ is

$$p\left(\rho|\boldsymbol{\beta}, \mathbf{z}, \mathbf{y}\right) \propto \left|\mathbf{I}_n - \rho\mathbf{W}\right| \exp\left(-\frac{1}{2}\left(\mathbf{S}\mathbf{z} - \mathbf{X}\boldsymbol{\beta}\right)'\left(\mathbf{S}\mathbf{z} - \mathbf{X}\boldsymbol{\beta}\right)\right) \tag{5}$$

   which can be sampled from using Metropolis-Hastings or some other sampling scheme (i.e. Importance Sampling). We implement a grid-based evaluation and numerical integration proposed by LeSage and Pace and then draw from the inverse distribution function (LeSage and Pace, 2009, p. 132).

Implementation issues

Since MCMC methods are widely considered to be (very) slow, we devote this section to the discussion of some implementation issues in R. Key factors to estimate large spatial probit models in R include the usage of sparse matrices and compiled Fortran code, and possibly also parallelization, which has been introduced to R 2.14.0 with the package **parallel**. We report estimation times and memory requirements for several medium-size and large-size problems and compare our times to those reported by LeSage and Pace (2009, p. 291). We show that our approach allows the estimation of large spatial probit models in R within reasonable time.

Drawing $p(\mathbf{z}|\boldsymbol{\beta}, \rho, \mathbf{y})$

In this section we describe the efforts made to generate samples from $p(\mathbf{z}|\boldsymbol{\beta}, \rho, \mathbf{y})$

$$\mathbf{z} \sim N\left(\left(\mathbf{I}_n - \rho\mathbf{W}\right)^{-1}\mathbf{X}\boldsymbol{\beta}, \left[\left(\mathbf{I}_n - \rho\mathbf{W}\right)'\left(\mathbf{I}_n - \rho\mathbf{W}\right)\right]^{-1}\right)$$

subject to $z_i \geq 0$ for $y_i = 1$ and $z_i < 0$ for $y_i = 0$.

The generation of samples from a truncated multivariate normal distribution in high dimensions is typically done with a Gibbs sampler rather than with other techniques such as rejection sampling (for a comparison, see Wilhelm and Manjunath (2010)). The Gibbs sampler draws from univariate conditional densities $f(z_i|\mathbf{z}_{-i}) = f(z_i|z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n)$.

The distribution of z_i conditional on $\mathbf{z}_{-i}$ is univariate truncated normal with variance

$$\begin{aligned} \boldsymbol{\Sigma}_{i.-i} &= \boldsymbol{\Sigma}_{ii} - \boldsymbol{\Sigma}_{i,-i}\boldsymbol{\Sigma}_{-i,-i}^{-1}\boldsymbol{\Sigma}_{-i,i} &(6)\\ &= \mathbf{H}_{ii}^{-1} &(7) \end{aligned}$$

and mean

$$\boldsymbol{\mu}_{i.-i} = \boldsymbol{\mu}_i + \boldsymbol{\Sigma}_{i,-i}\boldsymbol{\Sigma}^{-1}_{-i,-i}\left(\mathbf{z}_{-i} - \boldsymbol{\mu}_{-i}\right) \tag{8}$$

$$= \boldsymbol{\mu}_i - \mathbf{H}^{-1}_{ii}\mathbf{H}_{i,-i}\left(\mathbf{z}_{-i} - \boldsymbol{\mu}_{-i}\right) \tag{9}$$

and bounds $-\infty \leq z_i \leq 0$ for $y_i = 0$ and $0 \leq z_i \leq \infty$ for $y_i = 1$.

Package **tmvtnorm** has such a Gibbs sampler in place since version 0.9, based on the covariance matrix $\boldsymbol{\Sigma}$. Following the works of Geweke, the Gibbs sampler is easier to state in terms of the precision matrix $\mathbf{H}$ (Geweke, 1991, 2005, p. 171), simply because it requires fewer and easier operations in the above equations (7) and (9). These equations will further simplify in the case of a sparse precision matrix $\mathbf{H}$, in which most of the elements are zero. With sparse $\mathbf{H}$, operations involving $\mathbf{H}_{i,-i}$ need only to be executed for all non-zero elements rather than for all $n-1$ variables in $\mathbf{z}_{-i}$.

In our spatial probit model, the covariance matrix $\boldsymbol{\Sigma} = [(\mathbf{I}_n - \rho\mathbf{W})'(\mathbf{I}_n - \rho\mathbf{W})]^{-1}$ is a dense matrix, whereas the corresponding precision matrix $\mathbf{H} = \boldsymbol{\Sigma}^{-1} = (\mathbf{I}_n - \rho\mathbf{W})'(\mathbf{I}_n - \rho\mathbf{W})$ is sparse. Hence, using $\boldsymbol{\Sigma}$ for sampling $\mathbf{z}$ is inefficient. For this reason we reimplemented the Gibbs sampler with the precision matrix $\mathbf{H}$ in package **tmvtnorm** instead (Wilhelm and Manjunath, 2012).

Suppose one wants to draw N truncated multinormal samples in n dimensions, where the precision matrix $\mathbf{H}$ is sparse $(n \times n)$ with only $m < n$ entries different from zero per row on average (e.g. $m = 6$ nearest neighbors or average branching factor in a network). With growing dimension n, two types of problems arise with a usual dense matrix representation for $\mathbf{H}$:

1. **data storage problem:**
   Since matrix $\mathbf{H}$ is $n \times n$, the space required to store the dense matrix will be quadratic in n. One remedy is, of course, using some sparse matrix representation for $\mathbf{H}$ (e.g. packages **Matrix**, **sparseM** etc.), which actually holds only $n \cdot m$ elements instead of $n \cdot n$. The following code example shows the difference in object sizes for a dense vs. sparse identity matrix $\mathbf{I}_n$ for $n = 10000$ and $m = 1$.

```
> library(Matrix)
> I_n_dense <- diag(10000)
> print(object.size(I_n_dense), units = "Mb")

762.9 Mb

> rm(I_n_dense)
> I_n_sparse <- sparseMatrix(i = 1:10000, j = 1:10000, x = 1)
> print(object.size(I_n_sparse), units = "Mb")

0.2 Mb
```

2. **data access problem:**
   Even with a sparse matrix representation for $\mathbf{H}$, a naive strategy that tries to access an arbitrary matrix element `H[i,j]` like in triplet or hash table representations, will result in $N \cdot n \cdot n$ matrix accesses to $\mathbf{H}$. This number grows quadratically in n. For example, for $N = 30$ and $n = 10000$ it adds up to $30 \cdot 10000 \cdot 10000 = 3$ billion hash function calls, which is inefficient and too slow for most applications.
   Iterating only over the non-zero elements in row i, for example in term $H_{i,-i}$, reduces the number of accesses to $\mathbf{H}$ to only $N \cdot n \cdot m$ instead of $N \cdot n \cdot n$, which furthermore will only grow linearly in n for a fixed m. Suitable data structures to access all non-zero elements of the precision matrix $\mathbf{H}$ are linked lists of elements for each row i (list-of-lists) or, thanks to the symmetry of

H, the compressed-sparse-column/row representation of sparse matrices, directly available from the **Matrix** package.

How fast is the random number generation for $\mathbf{z}$ now? We performed a series of tests for varying sizes N and n and two different values of m. The results presented in table 1 show that the sampler with sparse $\mathbf{H}$ is indeed very fast and works fine in high dimensions. The time required scales with $N \cdot n$.

Table 1: Performance test results for the generation of truncated multivariate normal samples $\mathbf{z}$ with `rtmvnorm.sparseMatrix()` (**tmvtnorm**) for a varying number of samples N and dimension n. The precision matrix in each case is $\mathbf{H} = (\mathbf{I}_n - \rho\mathbf{W})'(\mathbf{I}_n - \rho\mathbf{W})$, the spatial weight matrix $\mathbf{W}$ contains m non-zero entries per row. Times are in seconds and measured on an Intel® Core™i7-2600 CPU @3.40 GHz.

		N					
	n	10^1	10^2	10^3	10^4	10^5	10^6
m=2	10^1				0.03	0.25	2.60
	10^2			0.03	0.25	2.75	
	10^3		0.03	0.25	2.84		
	10^4	0.03	0.25	2.80			
	10^5	0.26	2.79				
m=6	10^1				0.03	0.23	2.48
	10^2			0.01	0.23	2.53	
	10^3		0.03	0.23	2.59		
	10^4	0.02	0.24	2.57			
	10^5	0.25	2.68				

One more performance issue we discuss here is the burn-in size in the innermost Gibbs sampler generating $\mathbf{z}$ from $p(\mathbf{z}|\boldsymbol{\beta}, \rho, \mathbf{y})$. Depending on the start value, Gibbs samplers often require a certain burn-in phase until the sampler mixes well and draws from the desired target distribution.
In our MCMC setup, we only draw one sample of $\mathbf{z}$ from $p(\mathbf{z}|\boldsymbol{\beta}, \rho, \mathbf{y})$ in each MCMC iteration, but possibly in very high dimensions (e.g $N = 1$, $n = 10000$). With $burn.in = 20$ samples, we have to generate 21 draws in order to keep just one. In our situation, a large burn-in size will dramatically degrade the MCMC performance, so the number of burn-in samples for generating $\mathbf{z}$ has to be chosen carefully. LeSage and Pace (2009) discuss the role of the burn-in size and often use $burn.in = 10$, when the Gibbs sampler starts from zero. Alternatively, they also propose to use no burn-in phase at all (e.g. $burn.in = 0$), but then to set the start value of the sampler to the previous value of $\mathbf{z}$ instead of zero.

QR decomposition of $(\mathbf{I}_n - \rho\mathbf{W})$

The mean vector $\boldsymbol{\mu}$ of the truncated normal samples $\mathbf{z}$ in equation (3) takes the form

$$\boldsymbol{\mu} = (\mathbf{I}_n - \rho\mathbf{W})^{-1}\mathbf{X}\boldsymbol{\beta} \tag{10}$$

However, inverting the sparse matrix $\mathbf{S} = (\mathbf{I}_n - \rho\mathbf{W})$ will produce a dense matrix and will therefore offset all benefits from using sparse matrices (i.e. memory consumption and size of problems that can be solved). It is preferable to determine $\boldsymbol{\mu}$ by solving the equations $(\mathbf{I}_n - \rho\mathbf{W})\boldsymbol{\mu} = \mathbf{X}\boldsymbol{\beta}$ with a QR decomposition of $\mathbf{S} = (\mathbf{I}_n - \rho\mathbf{W})$. We point out that there is a significant performance difference between the usual code `mu <-qr.solve(S,X %*% beta)` and `mu <-solve(qr(S),X %*% beta)`. The latter will apply a QR decomposition for a sparse matrix $\mathbf{S}$ and

will use a method `qr()` from the package **Matrix**, whereas the first function from the **base** package will coerce **S** into a dense matrix. `solve(qr(S),X %*% beta)` takes only half the time required by `qr.solve(S,X %*% beta)`.

Drawing $p(\boldsymbol{\beta}|\mathbf{z},\rho,\mathbf{y})$

Drawing $p(\boldsymbol{\beta}|\mathbf{z},\rho,\mathbf{y})$ in equation (4) is a multivariate normal distribution, whose variance $\mathbf{T}^*$ does not depend on $\mathbf{z}$ and ρ. Therefore, we can efficiently vectorize and generate temporary draws $\beta_{tmp} \sim N(0,\mathbf{T}^*)$ for all MCMC iterations before running the chain and then just shift these temporary draws by $\mathbf{c}^*$ in every iteration to obtain a $\beta \sim N(\mathbf{c}^*,\mathbf{T}^*)$.

Determining log-determinants and drawing $p(\rho|\mathbf{z},\boldsymbol{\beta},\mathbf{y})$

The computation of log-determinants for $\ln|I_n - \rho W|$, whose evaluation is frequently needed for drawing $p(\rho|\mathbf{z},\boldsymbol{\beta},\mathbf{y})$, becomes a challenging task for large matrices. Bivand (2010) gives a survey of the different methods for calculating the Jacobian like the Pace and Barry (1997) grid evaluation of ρ in the interval $[-1,\ldots,+1]$ and spline approximations between grid points or the Chebyshev approximation of the log-determinant (Pace and LeSage, 2004). All of them are implemented in package **spdep**. For the estimation of the probit models we are using the existing facilities in **spdep** (`do_ldet()`).

Computation of marginal effects

In spatial lag models (and its probit variants), a change of an explanatory variable x_{ir} will affect both the same response variable z_i (direct effects) and possibly all other responses z_j ($j \neq i$; indirect effects or spatial spillovers). The resulting effects matrix $S_r(W)$ is $n \times n$ for $\mathbf{x}_r$ ($r = 1,\ldots,k$) and three summary measures of $S_r(W)$ will be computed: average direct effects ($M_r(D) = n^{-1} tr S_r(W)$), average total effects ($M_r(T) = n^{-1} 1_n' S_r(W) 1_n$) and the average indirect effects as the difference of total and direct effects ($M_r(I) = M_r(T) - M_r(D)$). In contrast to the SAR probit, there are no spatial spill-over effects in the SEM probit model (2). In the MATLAB spatial econometrics toolbox (LeSage, 2010), the computation of the average total effects requires an inversion of the $n \times n$ matrix $\mathbf{S} = (\mathbf{I}_n - \rho\mathbf{W})$ at every MCMC iteration. Using the same QR-decomposition of **S** as described above and solving the equation `solve(qr(S),rep(1,n))` speeds up the calculation of total effects by magnitudes.

Examples

Package spatialprobit

After describing the implementation issues to ensure that the estimation is fast enough and capable to handle large problems, we briefly describe the interface of the methods in package **spatialprobit** (Wilhelm and de Matos, 2013) and then turn to some examples.

The main estimation method for the SAR probit model `sar_probit_mcmc(y,X,W)` takes a vector of dependent variables `y`, a model matrix `X` and a spatial weight matrix `W`. The method `sarprobit(formula,W,data)` is a wrapper which allows a model formula and a data frame. Both methods require a spatial weight matrix `W` to be passed as an argument. Additionally, the number of MCMC start values, the number of burn-in iterations and a thinning parameter can be specified. The estimation `fit <-sarprobit(y ~ x,W,data)` returns an object of class `sarprobit`.

The model coefficients can be extracted via `coef(fit)`. `summary(fit)` returns a coefficient table with z-values, `impacts(fit)` gives the marginal effects and the `plot(fit)` method provides MCMC trace plots, posterior density plots as well as autocorrelation plots of the model parameters. `logLik(fit)` and `AIC(fit)` return the log likelihood and the AIC of the model for model comparison and testing.

Experiment from LeSage/Pace

We replicate the experiment from LeSage and Pace (2009, section 10.1.5) for $n = 400$ and $n = 1000$ random points in a plane and a spatial weight matrix with the 6 nearest neighbors. The spatial probit model parameters are $\sigma_\epsilon^2 = 1$, $\boldsymbol{\beta} = (0, 1, -1)'$ and $\rho = 0.75$. We generate data from this model with the following code.

```
> library(spatialprobit)
> set.seed(2)
> n <- 400
> beta <- c(0, 1, -1)
> rho <- 0.75
> X <- cbind(intercept = 1, x = rnorm(n), y = rnorm(n))
> I_n <- sparseMatrix(i = 1:n, j = 1:n, x = 1)
> nb <- knn2nb(knearneigh(cbind(x = rnorm(n), y = rnorm(n)), k = 6))
> listw <- nb2listw(nb, style = "W")
> W <- as(as_dgRMatrix_listw(listw), "CsparseMatrix")
> eps <- rnorm(n = n, mean = 0, sd = 1)
> z <- solve(qr(I_n - rho * W), X %*% beta + eps)
> y <- as.double(z >= 0)
```

We estimate the spatial probit model as

```
> sarprobit.fit1 <- sarprobit(y ~ X - 1, W, ndraw = 1000, burn.in = 200,
     thinning = 1, m = 10)
> summary(sarprobit.fit1)
> plot(sarprobit.fit1)
```

Table 2 shows the results of the SAR probit estimation for the same experiment as in LeSage and Pace (2009). The results shown there can be replicated either using the code above (for $n = 400$) or using the `LeSagePaceExperiment()` function in the package and setting `set.seed(2)` (for $n = 400$ and $n = 1000$)):

```
> set.seed(2)
> res1 <- LeSagePaceExperiment(n = 400, beta = c(0, 1, -1), rho = 0.75,
     ndraw = 1000, burn.in = 200, thinning = 1, m = 10)
> summary(res1)
> set.seed(2)
> res2 <- LeSagePaceExperiment(n = 1000, beta = c(0, 1, -1), rho = 0.75,
     ndraw = 1000, burn.in = 200, thinning = 1, m = 1)
> summary(res2)
```

The corresponding MLE and GMM estimates in table 2 can be replicated by creating the data as above (replacing `n <-1000` for $n = 1000$ accordingly) and

```
> library(McSpatial)
> wmat <- as.matrix(W)
> mle.fit1 <- spprobitml(y ~ X - 1, wmat = wmat)
> gmm.fit1 <- spprobit(y ~ X - 1, wmat = wmat)
```

Our Bayesian estimation yields similar results, but our implementation is much faster (factor 20-100) than the LeSage implementation, even on a single core. The **McSpatial** GMM estimation seems to be biased in this example. The evaluation of the performance relative to **McSpatial** MLE clearly needs to take into account

a number of factors: First, the choice of the number of MCMC iterations N and *burn.in* samples, as well as m do control the estimation time to a large degree. Second, the current ML implementation in **McSpatial** does not compute marginal effects. Finally, **McSpatial** works with dense matrices which is superior for smaller sample sizes, but will not scale for larger n.

Table 3 shows that a change of the Gibbs sampler burn-in size m for drawing $\mathbf{z}$ has little effect on the mean and the variance of the estimates. $m = 1$ means taking the previous value of $\mathbf{z}$, $m = 2, 5, 10$ start the chain from $\mathbf{z} = 0$. Clearly, choosing $m = 2$ might not be enough as burn-in phase.

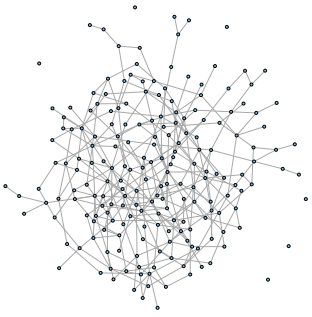
Random graph example

As a second example we present the estimation of the probit model based on a random undirected graph with $n = 200$ nodes and an average branching factor of 3. Of course this estimation procedure can be applied to real network structures such as from social networks. The package **igraph** can be used to create random graphs and to compute the adjacency matrix $\mathbf{A}$ as well as the spatial weight matrix $\mathbf{W}$ (Csardi and Nepusz, 2006).

```
> library(igraph)
> library(spatialprobit)
> set.seed(1)
> n <- 200
> branch <- 3
> probability <- branch/n
> grandom <- igraph::erdos.renyi.game(n = n, p.or.m = probability, type = "gnp",
    directed = F, loops = F)
> V(grandom)$name <- 1:n
> A <- igraph::get.adjacency(grandom, type = "both", sparse = TRUE)
> W <- A/ifelse(rowSums(A) == 0, 1, rowSums(A))
> plot(grandom, layout = layout.fruchterman.reingold,
    vertex.label.family = "sans", vertex.size = 2,
    vertex.label = "")
```

The figure 1 shows the resulting random graph.

Figure 1: Random graph with $n = 200$ nodes and average branching factor 3



Next we are going to estimate the spatial probit model with $N = 3000$ draws and compare it to the standard probit model which neglects the spatial dependencies in the network.

```
> set.seed(1.2345)
> x <- rnorm(n)
> X <- cbind(const = rep(1, n), x = x)
> p <- 0.3
> beta <- c(-1, 2)
> I_n <- sparseMatrix(i = 1:n, j = 1:n, x = 1)
> z <- solve(qr(I_n - p * W), X %*% beta + rnorm(n))
> y <- as.numeric(z >= 0)
> sarprobit.fit <- sarprobit(y ~ X - 1, W, ndraw = 3000,
     burn.in = 200, thinning = 1)
```

The true parameter in this model are $\boldsymbol{\beta} = (-1, 2)'$ and $\rho = 0.3$.

```
> summary(sarprobit.fit)

--------MCMC spatial autoregressive probit--------
Execution time  = 25.350 secs

N draws         =   3000, N omit (burn-in)=    200
N observations  =    200, K covariates    =      2
# of 0 Y values =    151, # of 1 Y values =     49
Min rho         = -1.000, Max rho         =  1.000
-------------------------------------------------

       Estimate Std. Dev  p-level t-value Pr(>|z|)
Xconst -1.25361  0.20035  0.00000   -6.26  2.3e-09 ***
Xx      2.05238  0.28529  0.00000    7.19  1.2e-11 ***
rho     0.24796  0.10571  0.00967    2.35     0.02 *
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1
```

The direct, indirect and total marginal effects are extracted using

```
> impacts(sarprobit.fit)

--------Marginal Effects--------

(a) Direct effects
   lower_005 posterior_mean upper_095
Xx     0.231          0.268       0.3

(b) Indirect effects
   lower_005 posterior_mean upper_095
Xx    -0.299         -0.266     -0.23

(c) Total effects
   lower_005 posterior_mean upper_095
Xx   0.00149        0.00179         0
```

The corresponding non-spatial probit model is estimated using the `glm()` function:

```
> glm1 <- glm(y ~ x, family = binomial("probit"))
> summary(glm1, digits = 4)

Call:
glm(formula = y ~ x, family = binomial("probit"))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2337  -0.3488  -0.0870  -0.0002   2.4107
```

Figure 2: MCMC trace plots for model parameters with horizontal lines marking the true parameters

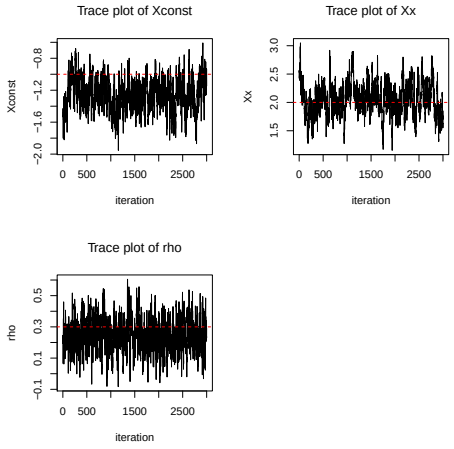
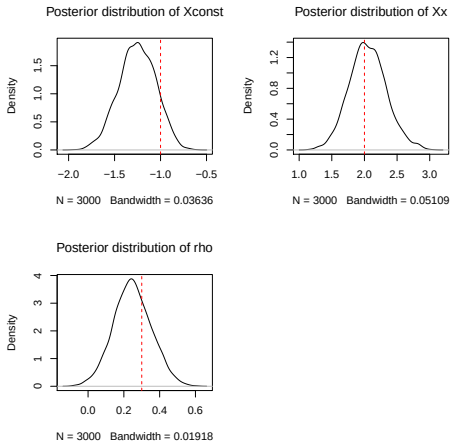


Figure 3: Posterior densities for model parameters with vertical markers for the true parameters




```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -1.491      0.208   -7.18  7.2e-13 ***
x              1.966      0.281    6.99  2.7e-12 ***
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 222.71  on 199  degrees of freedom
Residual deviance: 103.48  on 198  degrees of freedom
AIC: 107.5

Number of Fisher Scoring iterations: 7
```

Figures 2 and 3 show the trace plots and posterior densities as part of the MCMC estimation results. Table 4 compares the SAR probit and standard probit estimates.

Table 2: Bayesian SAR probit estimates for $n = 400$ and $n = 1000$ with $N = 1000$ draws and 200 burn-in samples. m is the burn-in size in the Gibbs sampler drawing $\mathbf{z}$. Timings in R include the computation of marginal effects and were measured using R 2.15.2 on an Intel® Core™i7-2600 CPU @3.40 GHz. Times marked with (†) are taken from LeSage and Pace (2009), table 10.1. To allow for a better comparison, these models were estimated anew (‡) using MATLAB R2007b and the spatial econometrics toolbox (LeSage, 2010) on the very same machine used for getting the R timings. The ML and GMM timings (§) do not involve the computation of marginal effects.

	LeSage (2009)		sarprobit		McSpatial ML		McSpatial GMM	
Estimates	Mean	Std dev	Mean	Std dev	Mean	Std dev	Mean	Std dev
	$n = 400, m = 10$				$n = 400$			
$\beta_1 = 0$	-0.1844	0.0686	0.0385	0.0562	0.0286	0.0302	-0.1042	0.0729
$\beta_2 = 1$	0.9654	0.1179	0.9824	0.1139	1.0813	0.1115	0.7690	0.0815
$\beta_3 = -1$	-0.8816	0.1142	-1.0014	0.1163	-0.9788	0.1040	-0.7544	0.0829
$\rho = 0.75$	0.6653	0.0564	0.7139	0.0427	0.7322	0.0372	1.2208	0.1424
Time (sec.)	1,276† / 623‡		11.5		1.4§		0.0§	
	$n = 1000, m = 1$				$n = 1000$			
$\beta_1 = 0$	0.05924	0.0438	-0.0859	0.0371	0.0010	0.0195	-0.1045	0.0421
$\beta_2 = 1$	0.96105	0.0729	0.9709	0.0709	1.0608	0.0713	0.7483	0.0494
$\beta_3 = -1$	-1.04398	0.0749	-0.9858	0.0755	-1.1014	0.0728	-0.7850	0.0528
$\rho = 0.75$	0.69476	0.0382	0.7590	0.0222	0.7227	0.0229	1.4254	0.0848
Time (sec.)	586† / 813‡		15.5		18.7§		0.0§	

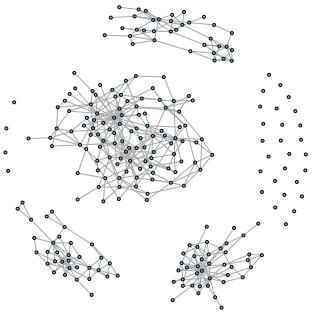
Table 3: Effects of the Gibbs sampler burn-in size m on SAR probit estimates for $n = 400$, $N = 1000$ draws and $burn.in = 200$

	m=1		m=2		m=5		m=10	
Estimates	Mean	Std dev	Mean	Std dev	Mean	Std dev	Mean	Std dev
$\beta_1 = 0$	0.0385	0.0571	0.0447	0.0585	0.0422	0.0571	0.0385	0.0562
$\beta_2 = 1$	1.0051	0.1146	0.8294	0.0960	0.9261	0.1146	0.9824	0.1139
$\beta_3 = -1$	-1.0264	0.1138	-0.8417	0.0989	-0.9446	0.1138	-1.0014	0.1163
$\rho = 0.75$	0.7226	0.0411	0.6427	0.0473	0.6922	0.0411	0.7139	0.0427
Time (sec.)	10.2		10.2		10.5		11.0	
Time (sec.) in LeSage (2009)	195		314		-		1270	

Table 4: SAR probit estimates vs. probit estimates for the random graph example

	SAR probit			Probit		
Estimates	Mean	Std dev	p-level	Mean	Std dev	p-level
$\beta_1 = -1$	-1.2536	0.2004	0.0000	-1.4905	0.2077	0.0000
$\beta_2 = 2$	2.0524	0.2853	0.0000	1.9656	0.2812	0.0000
$\rho = 0.3$	0.2480	0.1057	0.0097			
Time (sec.)	25.4					

Figure 4: Social network of physicians in 4 different cities based on "advisorship" relationship



Diffusion of innovation and information

A last example looks at the information flow in social networks. Coleman et al. (1957, 1966) have studied how innovations (i.e. new drugs) diffuse among physicians, until the new drug is widely adopted by all physicians. They interviewed 246 physicians in 4 different US cities and looked at the role of 3 interpersonal networks (friends, colleagues and discussion circles) in the diffusion process (November, 1953–February, 1955). Figure 4 illustrates one of the 3 social structures based on the question "Tho whom do you most often turn for advice and information?". The data set is available at `http://moreno.ss.uci.edu/data.html#ckm`, but also part of **spatialprobit** (`data(CKM)`). See also Burt (1987) and den Bulte and Lilien (2001) for further discussion of this data set. The dependent variable in the model is the month in which a doctor first prescribed the new drug. Explanatory variables are the social structure and individual variables.

We are estimating a standard probit model as well as a SAR probit model based on the "advisorship" network and trying to find determinants for all adopters of the new drug. We do not aim to fully reanalyze the Coleman data set, nor to provide a detailed discussion of the results. We rather want to illustrate the model estimation in R. We find a positive relationship for the social influence variable in the probit model, but social contagion effects as captured by $\rho\mathbf{W}$ in the more sound SAR probit model is rather small and insignificant. This result suggests that social influence is a factor in information diffusion, but the information flow might not be correctly described by a SAR model. Other drivers for adoption which are ignored here, such as marketing efforts or aggressive pricing of the new drug may play a role in the diffusion process too (den Bulte and Lilien, 2001).

```
> set.seed(12345)
> # load data set "CKM" and
> # spatial weight matrices "W1","W2","W3"
> data(CKM)
> # 0/1 variable for early adopter
> y <- as.numeric(CKM$adoption.date <= "February, 1955")
> # create social influence variable
> influence <- as.double(W1 %*% as.numeric(y))
> # Estimate Standard probit model
> glm.W1 <- glm(y ~ influence + city + med_sch_yr + meetings + jours +
  free_time + discuss + clubs +
  friends + community + patients + proximity + specialty,
  data=CKM, family=binomial("probit"))
```

Table 5: SAR probit and standard probit estimates for the Coleman data.

	Probit		SAR Probit	
Estimates	Mean	z value	Mean	z value
(Intercept)	−0.3659	−0.40	0.1309	0.16
influence	0.9042	2.67		
city	−0.0277	−0.26	−0.0371	−0.39
med_sch_yr	0.2937	2.63	0.3226	3.22
meetings	−0.1797	−1.55	−0.1813	−1.64
jours	0.1592	2.46	0.1368	2.08
free_time	0.3179	2.31	0.3253	2.58
discuss	−0.0010	−0.01	−0.0643	−0.57
clubs	−0.2366	−2.36	−0.2252	−2.45
friends	−0.0003	−0.00	−0.0185	−0.31
community	0.1974	1.65	0.2298	2.08
patients	−0.0402	−0.67	−0.0413	−0.72
proximity	−0.0284	−0.45	−0.0232	−0.40
specialty	−0.9693	−8.53	−1.0051	−8.11
ρ			0.1138	1.33

```
> summary(glm.W1, digits=3)
> # Estimate SAR probit model without influence variable
> sarprobit.fit.W1 <- sarprobit(y ~ 1 + city + med_sch_yr +
  meetings + jours + free_time + discuss + clubs +
  friends + community + patients + proximity + specialty,
  data=CKM, W=W1)
> summary(sarprobit.fit.W1, digits=3)
```

Table 5 presents the estimation results for the non-spatial probit and the SAR probit specification. The coefficient estimates are similar in magnitude and sign, but the estimate for ρ does not support the idea of spatial correlation in this data set.

Parallel estimation of models

The MCMC is, similar to the bootstrap, an embarrassingly parallel problem. It can be easily run in parallel on several cores. From version 2.14.0, R offers a unified way of doing parallelization with the **parallel** package. There are several different approaches available to achieve parallelization and not all approaches are available for all platforms. See for example the conceptual differences between the two main methods `mclapply` and `parLapply`, where the first will only work serially on Windows. Users are therefore encouraged to read the **parallel** package documentation for choosing the appropriate way.

Here, we only sketch how easy the SAR probit estimation can be done in parallel with 2 tasks:

```
> library(parallel)
> mc <- 2
> run1 <- function(...) sarprobit(y ~ X - 1,
     W, ndraw = 500, burn.in = 200, thinning = 1)
> system.time({
     set.seed(123, "LEcuyer")
     sarprobit.res <- do.call(c, mclapply(seq_len(mc), run1))
 })
> summary(sarprobit.res)
```

Due to the overhead in setting up the cluster, it is reasonable to expect another 50% performance gain when working with 2 CPUs.

Summary

In this article we presented the estimation of spatial probit models in R and pointed to the critical implementation issues. Our performance studies showed that even large problems with $n = 10,000$ or $n = 100,000$ observations can be handled within reasonable time. We provided an update of **tmvtnorm** and a new package **spatialprobit** on CRAN with the methods for estimating spatial probit models implemented (Wilhelm and de Matos, 2013). The package currently implements three limited dependent models: the spatial lag probit model (`sarprobit()`), the probit model with spatial errors (`semprobit()`) and the SAR Tobit model (`sartobit()`). The Bayesian approach can be further extended to other limited dependent spatial models, such as ordered probit or models with multiple spatial weights matrices. We are planning to include these in the package in near future.

Bibliography

J. Albert. *Bayesian Computation in R*. Springer, 2007. [p2]

J. Albert. *LearnBayes: Functions for Learning Bayesian Inference*, 2012. URL http://CRAN.R-project.org/package=LearnBayes. R package version 2.12. [p2]

D. Bates and M. Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2013. URL http://CRAN.R-project.org/package=Matrix. R package version 1.0-11. [p1]

R. Bivand. Computing the Jacobian in spatial models: an applied survey. Discussion paper / Norwegian School of Economics and Business Administration, Department of Economics ; 2010,20, August 2010. URL http://brage.bibsys.no/nhh/bitstream/URN:NBN:no-bibsys_brage_23930/1/dp2010-20.pdf. [p6]

R. Bivand, with contributions by Micah Altman, L. Anselin, R. Assunção, O. Berke, A. Bernat, G. Blanchet, E. Blankmeyer, M. Carvalho, B. Christensen, Y. Chun, C. Dormann, S. Dray, R. Halbersma, E. Krainski, P. Legendre, N. Lewin-Koh, H. Li, J. Ma, G. Millo, W. Mueller, H. Ono, P. Peres-Neto, G. Piras, M. Reder, M. Tiefelsdorf, and D. Yu. *spdep: Spatial dependence: weighting schemes, statistics and models*, 2013. URL http://CRAN.R-project.org/package=spdep. R package version 0.5-56. [p1]

R. S. Burt. Social contagion and innovation: Cohesion versus structural equivalence. *American Journal of Sociology*, 92(6):1287–1335, 1987. [p12]

C. T. Butts. *sna: Tools for Social Network Analysis*, 2013. URL http://CRAN.R-project.org/package=sna. R package version 2.2-2. [p1]

C. T. Butts, M. S. Handcock, and D. R. Hunter. *network: Classes for Relational Data*. Irvine, CA, March 1, 2012. URL http://statnet.org/. R package version 1.7-1.1. [p1]

J. Coleman, E. Katz, and H. Menzel. The diffusion of an innovation among physicians. *Sociometry*, 20:253–270, 1957. [p12]

J. S. Coleman, E. Katz, and H. Menzel. *Medical Innovation: A Diffusion Study*. New York: Bobbs Merrill, 1966. [p12]

G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. URL http://igraph.sf.net. [p8]

C. V. den Bulte and G. L. Lilien. Medical innovation revisited: Social contagion versus marketing effort. *American Journal of Sociology*, 106(5):1409–1435, 2001. [p12]

P. Diggle and P. Ribeiro. *Model Based Geostatistics*. Springer, New York, 2007. [p1]

A. O. Finley and S. Banerjee. *spBayes: Univariate and Multivariate Spatial Modeling*, 2013. URL http://CRAN.R-project.org/package=spBayes. R package version 0.3-4. [p1]

R. J. Franzese, J. C. Hays, and S. Cook. Spatial-, temporal-, and spatiotemporal-autoregressive probit models of interdependent binary outcomes: Estimation and interpretation. Prepared for the Spatial Models of Politics in Europe & Beyond, Texas A&M University, February 2013, February 2013. [p2]

A. Genz, F. Bretz, T. Miwa, X. Mi, F. Leisch, F. Scheipl, and T. Hothorn. mvtnorm: Multivariate normal and t distributions, 2012. URL http://CRAN.R-project.org/package=mvtnorm. R package version 0.9-9994. [p3]

J. F. Geweke. Effcient simulation from the multivariate normal and student-t distributions subject to linear constraints and the evaluation of constraint probabilities, 1991. URL http://www.biz.uiowa.edu/faculty/jgeweke/papers/paper47/paper47.pdf. [p4]

J. F. Geweke. *Contemporary Bayesian Econometrics and Statistics*. John Wiley and Sons, 2005. [p4]

T. Klier and D. P. McMillen. Clustering of auto supplier plants in the united states: Generalized method of moments spatial probit for large samples. *Journal of Business and Economic Statistics*, 26:460–471, 2008. [p2]

R. Koenker and P. Ng. *SparseM: Sparse Linear Algebra*, 2012. URL http://CRAN.R-project.org/package=SparseM. R package version 0.96. [p1]

J. LeSage and R. K. Pace. *Introduction to Spatial Econometrics*. CRC Press, 2009. [p1, 3, 5, 7, 11]

J. P. LeSage. Bayesian estimation of limited dependent variable spatial autoregressive models. *Geographical Analysis*, 32(1):19–35, 2000. [p1]

J. P. LeSage. Spatial econometrics toolbox for MATLAB, March 2010. URL http://www.spatial-econometrics.com/. [p6, 11]

J. P. LeSage, R. K. Pace, N. Lam, R. Campanella, and X. Liu. New orleans business recovery in the aftermath of hurricane Katrina. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 174:1007–1027, 2011. [p2]

J. J. Majure and A. Gebhardt. *sgeostat: An Object-oriented Framework for Geostatistical Modeling in S+*, 2013. URL http://CRAN.R-project.org/package=sgeostat. R package version 1.0-25; S original by James J. Majure Iowa State University and R port + extensions by Albrecht Gebhardt. [p1]

T. L. Marsh, R. C. Mittelhammer, and R. G. Huffaker. Probit with spatial correlation by field plot: Potato leafroll virus net necrosis in potatoes. *Journal of Agricultural, Biological, and Environmental Statistics*, 5:22–36, 2000. URL http://www.jstor.org/stable/1400629. [p2]

D. McMillen. *McSpatial: Nonparametric spatial data analysis*, 2012. URL http://CRAN.R-project.org/package=McSpatial. R package version 1.1.1. [p1, 2]

R. K. Pace and R. Barry. Quick computation of spatial autoregressive estimators. *Geographical Analysis*, 29:232–247, 1997. [p6]

R. K. Pace and J. LeSage. Chebyshev approximation of log-determinants of spatial weight matrices. *Computational Statistics & Data Analysis*, 45(2):179–196, 2004. [p6]

G. Piras. sphet: Spatial models with heteroskedastic innovations in R. *Journal of Statistical Software*, 35(1):1–21, 2010. URL http://www.jstatsoft.org/v35/i01/. [p1]

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, 2002. [p1]

S. Wilhelm and M. G. de Matos. *spatialprobit: Spatial Probit Models*, 2013. URL http://CRAN.R-project.org/package=spatialprobit. R package version 0.9-8. [p1, 6, 14]

S. Wilhelm and B. G. Manjunath. tmvtnorm: A package for the truncated multivariate normal distribution. *The R Journal*, 2(1):25–29, June 2010. URL http://journal.r-project.org/archive/2010-1/RJournal_2010-1_Wilhelm+Manjunath.pdf. [p3]

S. Wilhelm and B. G. Manjunath. *tmvtnorm: Truncated Multivariate Normal and Student t Distribution*, 2012. URL http://CRAN.R-project.org/package=tmvtnorm. R package version 1.4-7. [p2, 4]

J. M. Wooldridge. *Introductory Econometrics - A Modern Approach*. South Western College Publishing, 2002. [p2]

Stefan Wilhelm
Department of Finance, WWZ (Wirtschaftswissenschaftliches Zentrum)
University of Basel
Switzerland
wilhelm@financial.com

Miguel Godinho de Matos
Department of Engineering & Public Policy
Carnegie Mellon University
United States
miguelgodinhomatos@cmu.edu