

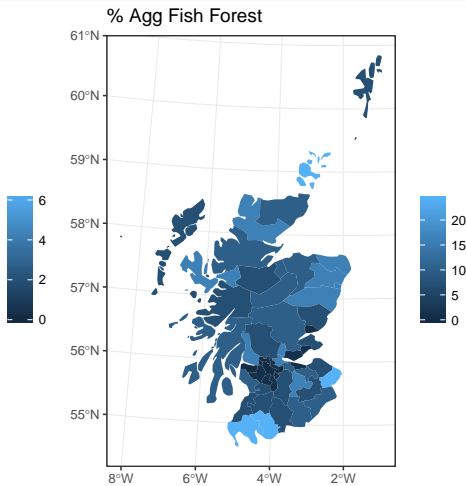
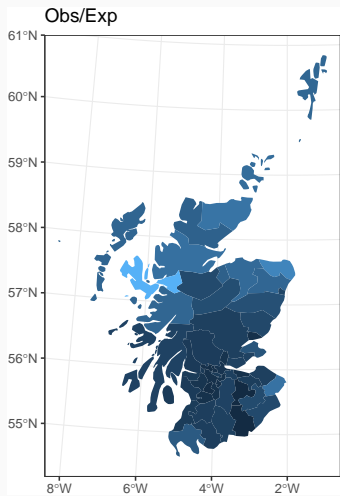
Lecture 20

Spatial Random Effects Models + Point Reference Spatial Data

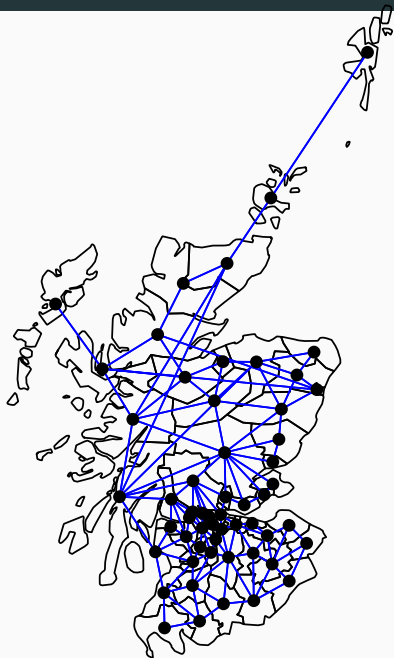
Colin Rundel

04/03/2017

Spatial Random Effects Models



Neighborhood / weight matrix



Moran's I

```
morans_I = function(y, w)
{
  n = length(y)
  y_bar = mean(y)
  num = sum(w * (y-y_bar) %*% t(y-y_bar))
  denom = sum( (y-y_bar)^2 )
  (n/sum(w)) * (num/denom)
}

morans_I(y = lip_cancer$Observed, w = diag(rowSums(W)) %*% W)
## [1] 0.2258758

morans_I(y = lip_cancer$Observed / lip_cancer$Expected,
         w = diag(rowSums(W)) %*% W)
## [1] 0.5323161

ape::Moran.I(lip_cancer$Observed / lip_cancer$Expected,
             weight = diag(rowSums(W)) %*% W) %>% str()
## List of 4
## $ observed: num 0.666
## $ expected: num -0.0182
## $ sd      : num 0.0784
## $ p.value : num 0
```

A hierarchical model for lip cancer

We have observed counts of lip cancer for 56 districts in Scotland. Let y_i represent the number of lip cancer for district i .

$$y_i \sim \text{Poisson}(\lambda_i)$$

$$\log(\lambda_i) = \log(E_i) + x_i\beta + \omega_i$$

$$\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \sigma^2(\mathbf{D} - \phi\mathbf{W})^{-1})$$

where E_i is the expected counts for each region (and serves as an offset).

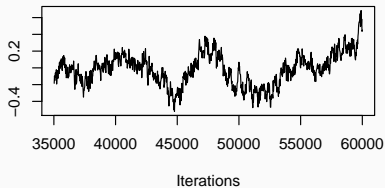
Data prep & JAGS model

```
D = diag(rowSums(W))
X = model.matrix(~scale(lip_cancer$pcaff))
log_offset = log(lip_cancer$Expected)
y = lip_cancer$Observed

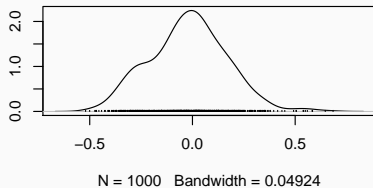
## model{
##   for(i in 1:length(y)) {
##     y[i] ~ dpois(lambda[i])
##     y_pred[i] ~ dpois(lambda[i])
##     log(lambda[i]) <- log_offset[i] + X[i,] %*% beta + omega[i]
##   }
##
##   for(i in 1:2) {
##     beta[i] ~ dnorm(0,1)
##   }
##
##   omega ~ dnorm(rep(0,length(y)), tau * (D - phi*W))
##   sigma2 <- 1/tau
##   tau ~ dgamma(2, 2)
##   phi ~ dunif(0,0.99)
## }
```


Model Results

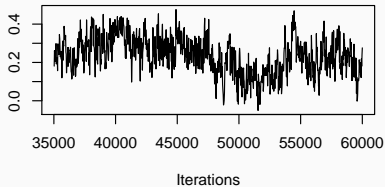
Trace of beta[1]



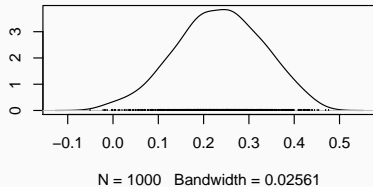
Density of beta[1]



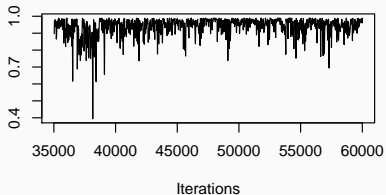
Trace of beta[2]



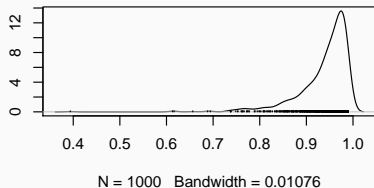
Density of beta[2]



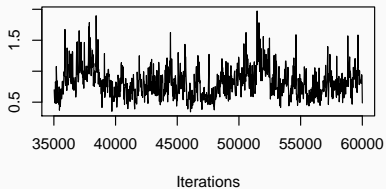
Trace of phi



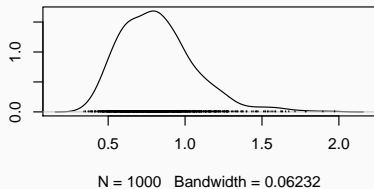
Density of phi



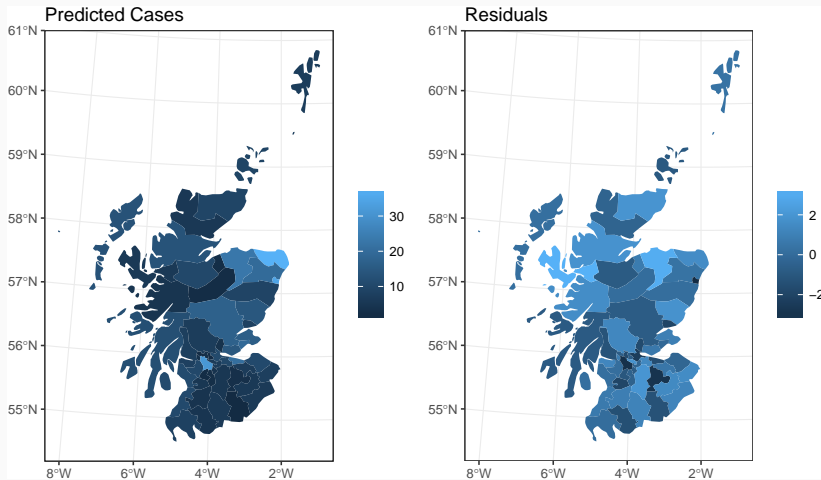
Trace of sigma2



Density of sigma2



Predictions & Residuals



Residuals + RMSE + Moran's I

```
#RMSE
```

```
lip_cancer_pred$resid %>% .^2 %>% mean() %>% sqrt()  
## [1] 1.498675
```

```
#Moran's I
```

```
morans_I(y = lip_cancer_pred$resid, w = diag(rowSums(W)) %**% W)  
## [1] 0.05661104
```

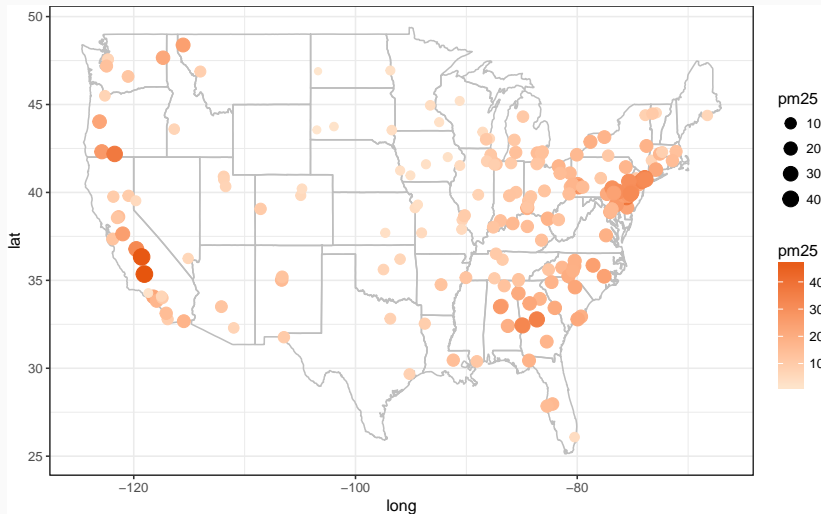
```
ape::Moran.I(lip_cancer_pred$resid,  
             weight = diag(rowSums(W)) %**% W) %>% str()
```

```
## List of 4  
## $ observed: num 0.0642  
## $ expected: num -0.0182  
## $ sd      : num 0.0803  
## $ p.value : num 0.305
```

Point Referenced Data

Example - PM2.5 from CSN

The Chemical Speciation Network are a series of air quality monitors run by EPA (221 locations in 2007). We'll look at a subset of the data from Nov 11th, 2007 (n=191) for just PM2.5.



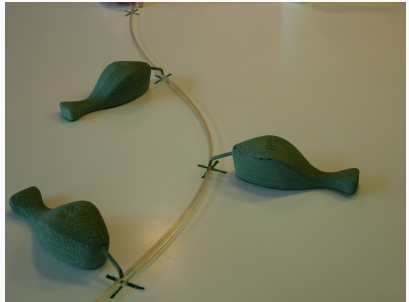
csn

A tibble: 191 × 5

| ## | site | longitude | latitude | date | pm25 |
|-------|----------|------------|----------|------------|----------|
| ## | <int> | <dbl> | <dbl> | <dtm> | <dbl> |
| ## 1 | 10730023 | -86.81500 | 33.55306 | 2007-11-14 | 19.43555 |
| ## 2 | 10732003 | -86.92417 | 33.49972 | 2007-11-14 | 26.40000 |
| ## 3 | 10890014 | -86.58637 | 34.68767 | 2007-11-14 | 13.40000 |
| ## 4 | 11011002 | -86.25637 | 32.40712 | 2007-11-14 | 19.70000 |
| ## 5 | 11130001 | -84.99917 | 32.47639 | 2007-11-14 | 22.60000 |
| ## 6 | 40139997 | -112.09577 | 33.50383 | 2007-11-14 | 12.30000 |
| ## 7 | 40191028 | -110.98230 | 32.29515 | 2007-11-14 | 7.20000 |
| ## 8 | 51190007 | -92.28130 | 34.75619 | 2007-11-14 | 12.70000 |
| ## 9 | 60070002 | -121.84222 | 39.75750 | 2007-11-14 | 10.00000 |
| ## 10 | 60190008 | -119.77222 | 36.78139 | 2007-11-14 | 32.26205 |

... with 181 more rows

Aside - Splines



Splines in 1d - Smoothing Splines

These are a mathematical analogue to the drafting splines represented using a penalized regression model.

Splines in 1d - Smoothing Splines

These are a mathematical analogue to the drafting splines represented using a penalized regression model.

We want to find a function $f(x)$ that best fits our observed data $y = y_1, \dots, y_n$ while being as *smooth* as possible.

$$\arg \min_{f(x)} \sum_{i=1}^n (y_i - f(x_i))^2 + \lambda \int f''(x)^2 dx$$

Interestingly, this minimization problem has an exact solution which is given by a mixture of weighted natural cubic splines (cubic splines that are linear in the tails) with knots at the observed data locations (x s).

Splines in 2d - Thin Plate Splines

Now imagine we have observed data of the form (x_i, y_i, z_i) where we wish to predict z_i given x_i and y_i for all i . We can naturally extend the smoothing spline model in two dimensions,

$$\arg \min_{f(x,y)} \sum_{i=1}^n (z_i - f(x_i, y_i))^2 + \lambda \int \int \left(\frac{\partial^2 f}{\partial x^2} + 2 \frac{\partial^2 f}{\partial x \partial y} + \frac{\partial^2 f}{\partial y^2} \right) dx dy$$

The solution to this equation has a natural representation using a weighted sum of *radial basis functions* with knots at the observed data locations

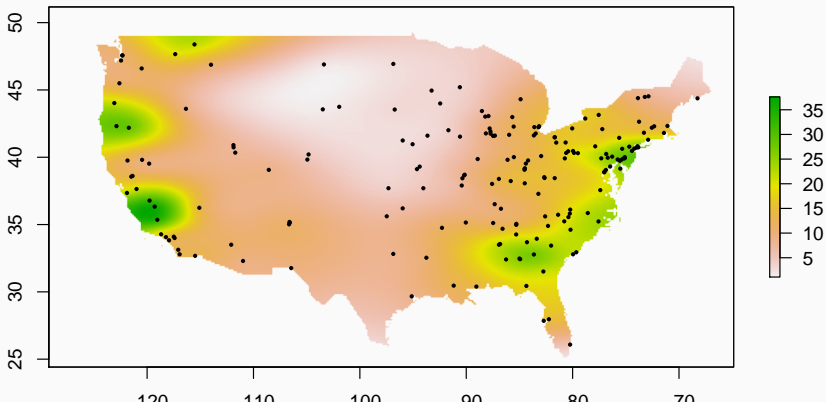
$$f(x, y) = \sum_{i=1}^n w_i d(x_i, y_i)^2 \log d(x_i, y_i).$$

Fitting a TPS

```
library(fields)
coords = select(csn, longitude, latitude) %>% as.matrix()
tps = Tps(x = coords, Y=csn$pm25)

pm25_pred = r
pm25_pred[cells] = predict(tps, pred_coords)

plot(pm25_pred)
points(coords, pch=16, cex=0.5)
```

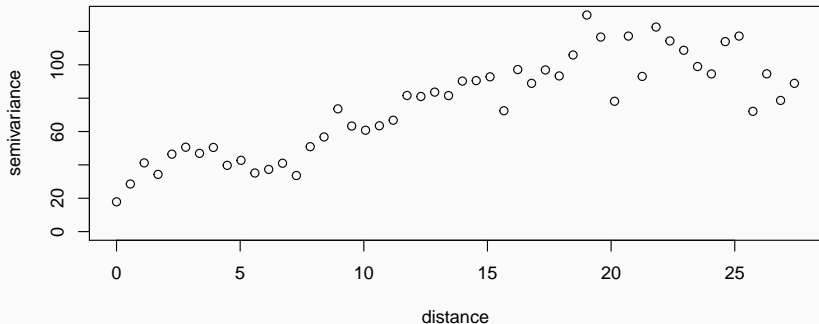


Gaussian Process Models / Kriging

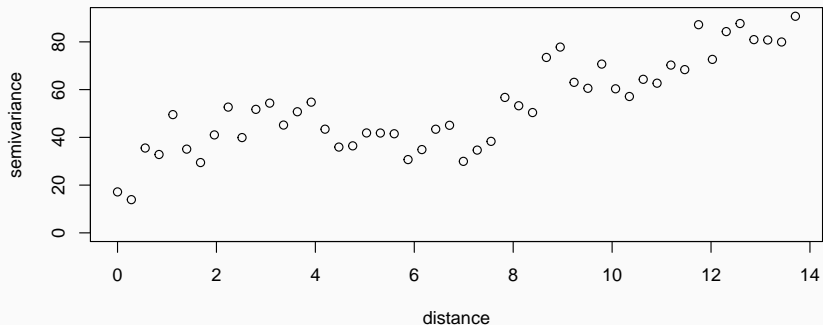
Variogram

```
library(geoR)
coords = csn %>% select(latitude, longitude) %>% as.matrix()
d = dist(coords) %>% as.matrix()

variog(coords = coords, data = csn$pm25, messages = FALSE,
        uvec = seq(0, max(d)/2, length.out=50)) %>% plot()
```

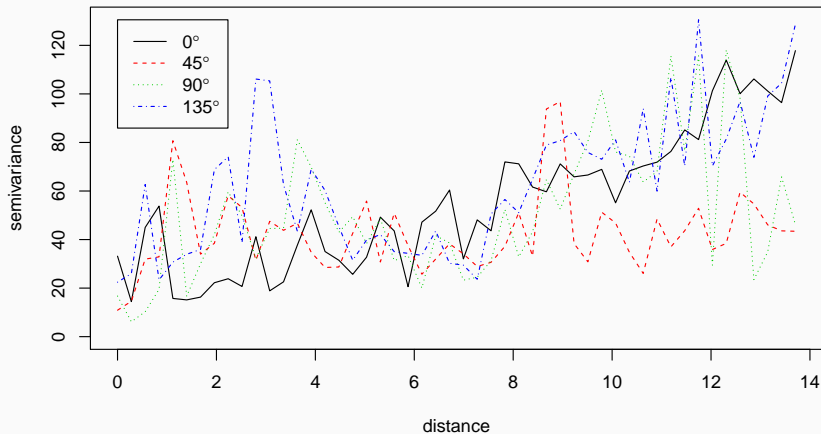


```
variog(coords = coords, data = csn$pm25, messages = FALSE,  
        uvec = seq(0, max(d)/4, length.out=50)) %>% plot()
```



Isotropy / Anisotropy

```
v4 = variog4(coords = coords, data = csn$pm25, messages = FALSE,  
            uvec = seq(0, max(d)/4, length.out = 50))  
plot(v4)
```



If we assume that our data is *stationary* and *isotropic* then we can use a Gaussian Process model to fit the data. We will assume an exponential covariance structure.

$$y \sim \mathcal{N}(\mu \mathbf{1}, \Sigma)$$
$$\{\Sigma\}_{ij} = \sigma^2 \exp(-r \|s_i - s_j\|) + \sigma_n^2 \mathbf{1}_{i=j}$$

If we assume that our data is *stationary* and *isotropic* then we can use a Gaussian Process model to fit the data. We will assume an exponential covariance structure.

$$y \sim \mathcal{N}(\mu \mathbf{1}, \Sigma)$$
$$\{\Sigma\}_{ij} = \sigma^2 \exp(-r \|s_i - s_j\|) + \sigma_n^2 \mathbf{1}_{i=j}$$

we can also view this as a spatial random effects model where

$$y(\mathbf{s}) = \mu(\mathbf{s}) + w(\mathbf{s}) + \epsilon(\mathbf{s})$$
$$w(\mathbf{s}) \sim \mathcal{N}(0, \Sigma')$$
$$\epsilon(s_i) \sim \mathcal{N}(0, \sigma_n^2)$$
$$\{\Sigma'\}_{ij} = \sigma^2 \exp(-r \|s_i - s_j\|)$$

Fitting with spBayes

```
library(spBayes)

n = nrow(csn)
n_samp = 20000
coords = select(csn, longitude, latitude) %>% as.matrix()
max_range = max(dist(coords)) / 4

starting = list(phi = 3/3, sigma.sq = 33, tau.sq = 17)
tuning = list("phi"=0.1, "sigma.sq"=0.1, "tau.sq"=0.1)
priors = list(
  beta.Norm = list(0, 1000),
  phi.Unif = c(3/max_range, 6),
  sigma.sq.IG = c(2, 2),
  tau.sq.IG = c(2, 2)
)
```

```
m = spLM(pm25 ~ 1, data = csn, coords = coords, starting = starting, priors = priors,  
cov.model = "exponential", n.samples = n_samp, tuning = tuning,  
n.report = n_samp/2)
```

```
## -----  
## General model description  
## -----  
## Model fit with 191 observations.  
##  
## Number of covariates 1 (including intercept if specified).  
##  
## Using the exponential spatial correlation model.  
##  
## Number of MCMC samples 20000.  
##  
## Priors and hyperpriors:  
## beta normal:  
## mu: 0.000  
## cov:  
## 1000.000  
##  
## sigma.sq IG hyperpriors shape=2.00000 and scale=2.00000  
## tau.sq IG hyperpriors shape=2.00000 and scale=2.00000  
## phi Unif hyperpriors a=0.21888 and b=6.00000
```

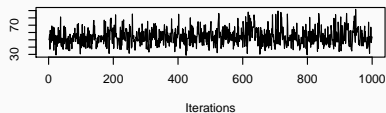
```
## -----  
## Sampling  
## -----  
## Sampled: 10000 of 20000, 50.00%  
## Report interval Metrop. Acceptance rate: 33.25%  
## Overall Metrop. Acceptance rate: 33.25%  
## -----  
## Sampled: 20000 of 20000, 100.00%  
## Report interval Metrop. Acceptance rate: 33.52%
```

```
m = spRecover(m, start=n_samp/2+1, thin = (n_samp/2)/1000)
## -----
##           Recovering beta and w
## -----
## Sampled: 99 of 1000, 9.90%
## Sampled: 199 of 1000, 19.90%
## Sampled: 299 of 1000, 29.90%
## Sampled: 399 of 1000, 39.90%
## Sampled: 499 of 1000, 49.90%
## Sampled: 599 of 1000, 59.90%
## Sampled: 699 of 1000, 69.90%
## Sampled: 799 of 1000, 79.90%
## Sampled: 899 of 1000, 89.90%
## Sampled: 999 of 1000, 99.90%
```

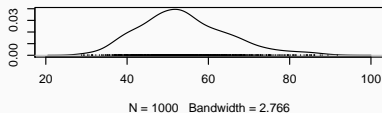
Parameter values

```
m$p.theta.recover.samples %>% mcmc() %>% plot()
```

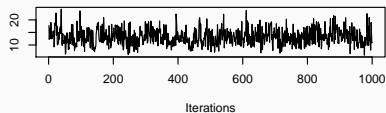
Trace of sigma.sq



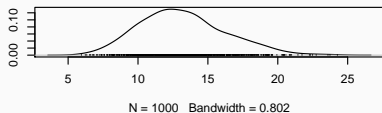
Density of sigma.sq



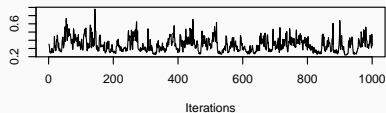
Trace of tau.sq



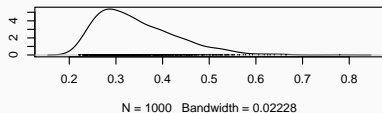
Density of tau.sq



Trace of phi

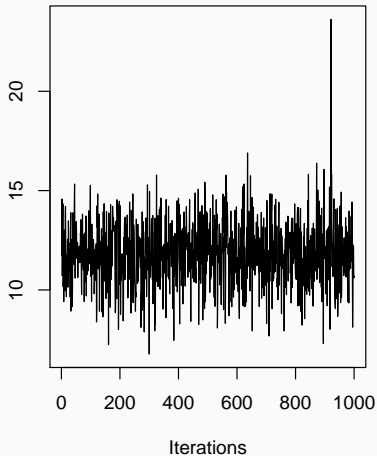


Density of phi

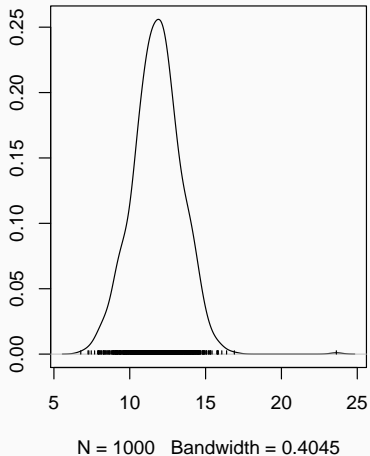


```
m$p.beta.recover.samples %>% mcmc() %>% plot()
```

Trace of (Intercept)



Density of (Intercept)



Predictions

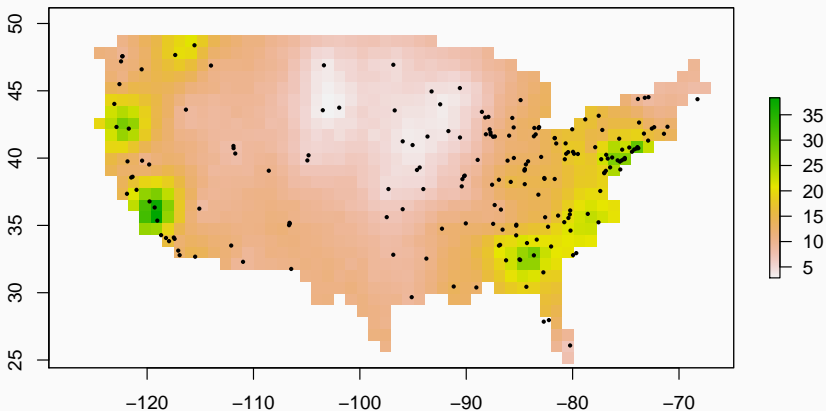
```
m_pred = spPredict(m, pred_coords, pred.covars = matrix(1, nrow=nrow(pred_co
                start=n_samp/2+1, thin=(n_samp/2)/1000)

## -----
## General model description
## -----
## Model fit with 191 observations.
##
## Prediction at 900 locations.
##
## Number of covariates 1 (including intercept if specified).
##
## Using the exponential spatial correlation model.
##
## -----
##          Sampling
## -----
## Sampled: 100 of 1000, 9.90%
## Sampled: 200 of 1000, 19.90%
## Sampled: 300 of 1000, 29.90%
## Sampled: 400 of 1000, 39.90%
## Sampled: 500 of 1000, 49.90%
## Sampled: 600 of 1000, 59.90%
## Sampled: 700 of 1000, 69.90%
## Sampled: 800 of 1000, 79.90%
```



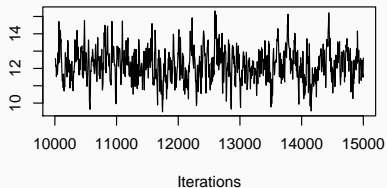
```
splm_pm25_pred = r  
splm_pm25_pred[cells] = m_pred_summary$post_mean
```

```
plot(splm_pm25_pred)  
points(coords, pch=16, cex=0.5)
```

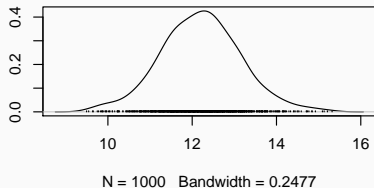


```
## model{
##   for(i in 1:length(y)){
##     y[i] ~ dnorm(beta + w[i], tau)
##     mu_w[i] <- 0
##   }
##
##   for(i in 1:length(y)){
##     for(j in 1:length(y)){
##       Sigma_w[i,j] <- sigma2_w * exp(-phi * d[i,j])
##     }
##   }
##   Sigma_w_inv <- inverse(Sigma_w)
##   w ~ dmnorm(mu_w, Sigma_w_inv)
##
##   beta ~ dnorm(0, 1/1000)
##   sigma2_w ~ dgamma(2, 2)
##   sigma2 ~ dgamma(2, 2)
##   tau <- 1/sigma2
##   phi ~ dunif(3/14, 6)
## }
```

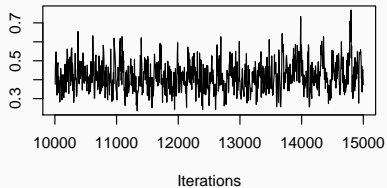
Trace of beta



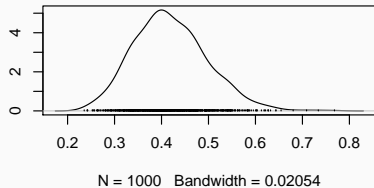
Density of beta



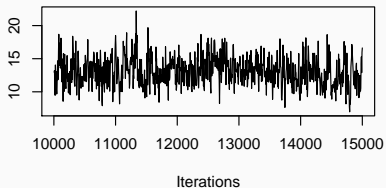
Trace of phi



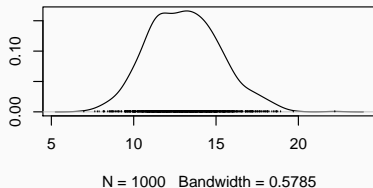
Density of phi



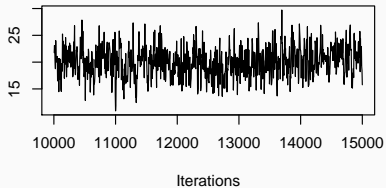
Trace of sigma2



Density of sigma2



Trace of sigma2_w



Density of sigma2_w

