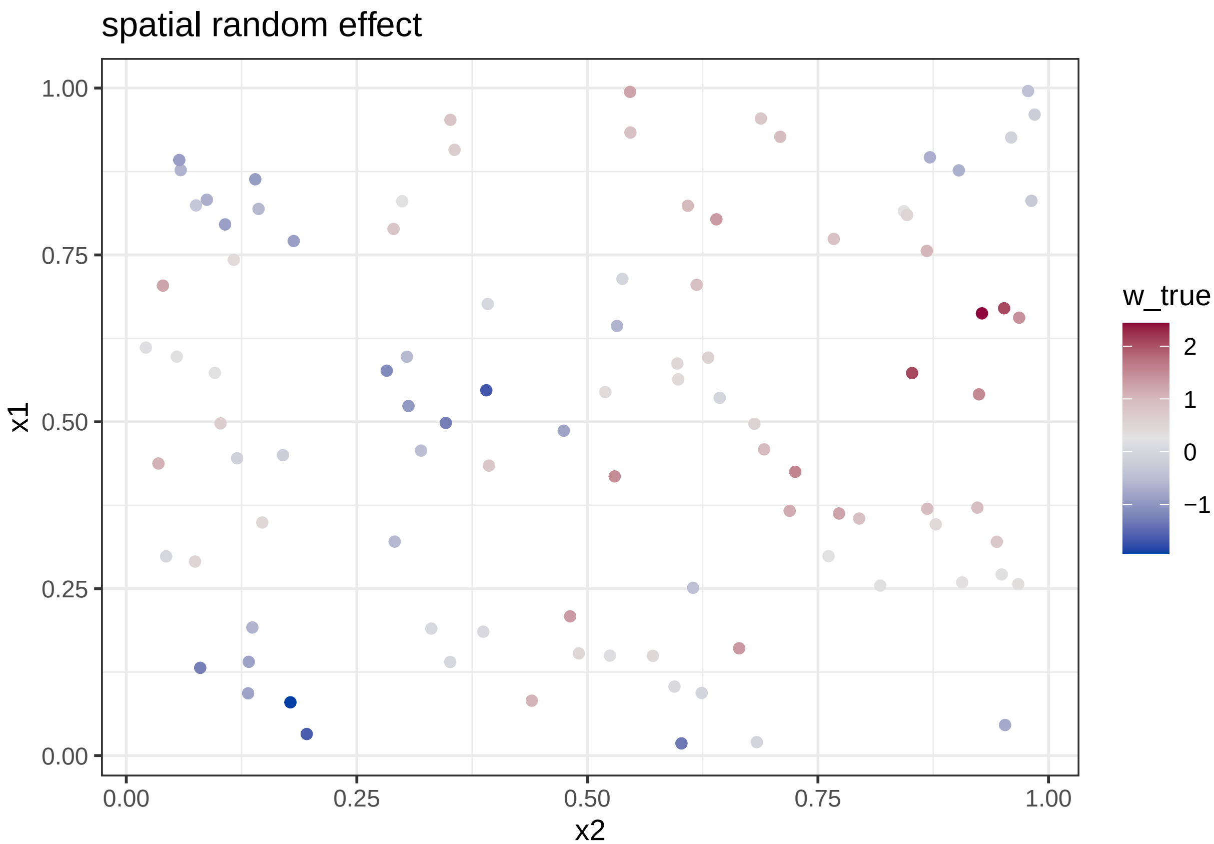
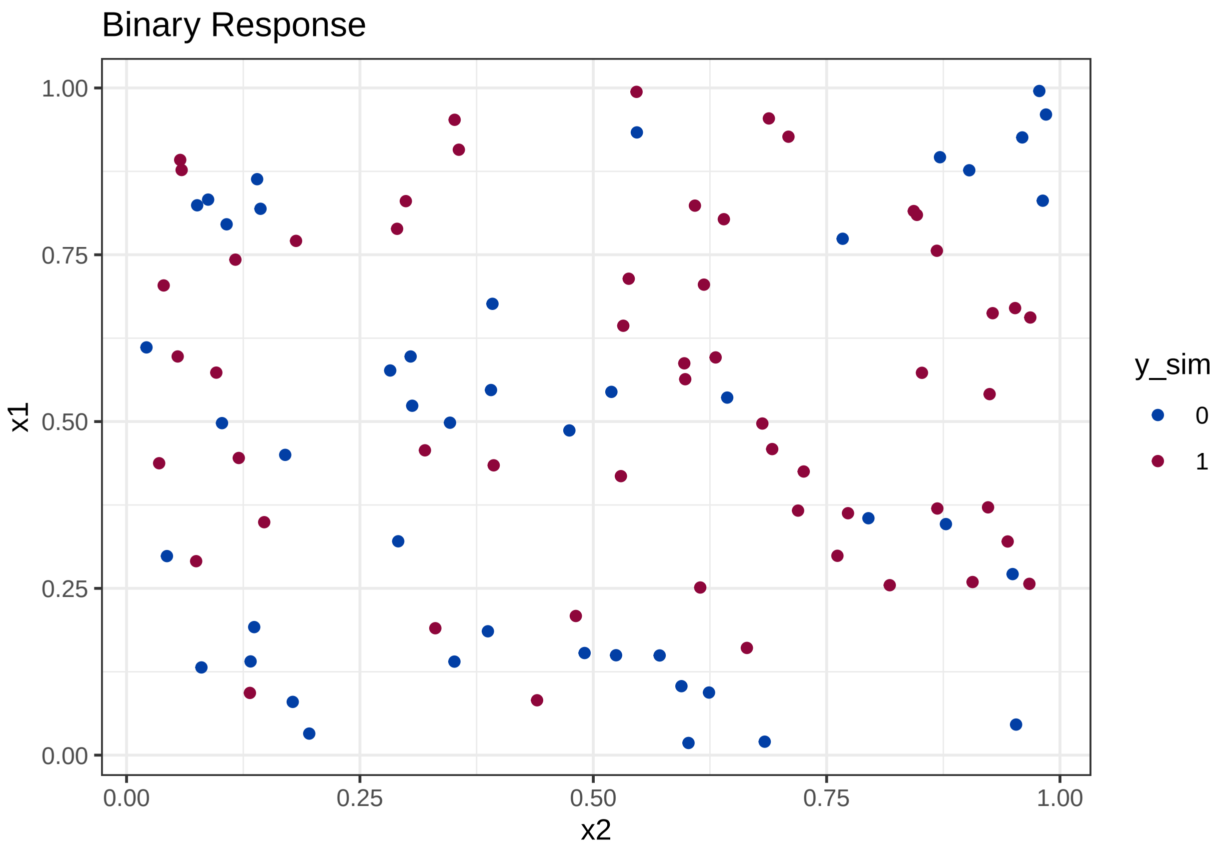
Spatial GLMs Demo

1. Simulate and visualize spatial random effects for binary data: No Covariates

```r
N <- 100
x1 <- runif(N)
x2 <- runif(N)
phi_true <- .2
sigmasq_true <- 1

d <- dist(cbind(x1,x2), upper = T, diag = T) %>% as.matrix
H <- sigmasq_true * exp(- d / phi_true)
w_true <- rmnorm(1,0,H)
p_true <- pnorm(w_true)
y_sim <- rbinom(N,1,p_true)

sim1_dat <- tibble(x1 = x1, x2 = x2, w_true = w_true, p_true = p_true, y_sim = as.factor(y_sim))

sim1_dat %>% ggplot(aes(y = x1, x = x2, color = w_true)) +
  geom_point() + theme_bw() +
  scale_color_gradientn(colours = colorspace::diverge_hcl(7)) +
  ggtitle('spatial random effect')
```




```
sim1_dat %>% ggplot(aes(y = x1, x = x2, color = y_sim)) +
  geom_point() + theme_bw() +
  ggtitle('Binary Response') + scale_color_manual(values=c("#023FA5", "#8E063B"))
```



2. Fit a model for this setting

```
##
## // The input data is a vector 'y' of length 'N'.
## data {
##   int<lower=0> N;
##   int<lower=0,upper=1> y[N];
##   matrix[N,N] d;
## }
##
## // The parameters accepted by the model. Our model
## // accepts two parameters 'mu' and 'sigma'.
## parameters {
##   vector[N] w;
##   real <lower = 0.1, upper = .8> phi;
##   real<lower = 0> sigmasq;
## }
##
## transformed parameters {
##   real<lower = 0, upper = 1> p[N];
##   vector[N] mu;
##   corr_matrix[N] Sigma;
##
##   for (i in 1:N) {
##     p[i] = Phi(w[i]);
##     mu[i] = 0;
```

```
##   }
##   for(i in 1:(N-1)){
##    for(j in (i+1):N){
##      Sigma[i,j] = exp((-1)*d[i,j]/ phi);
##      Sigma[j,i] = Sigma[i,j];
##    }
##  }
##  for(i in 1:N) Sigma[i,i] = 1;
## }
##
## // The model to be estimated.
## model {
##   w ~ multi_normal(mu, sigmasq * Sigma);
##   for (i in 1:N){
##     y[i] ~ bernoulli(p[i]);
##   }
##   sigmasq ~ inv_gamma(5,5);
## }
```

```
## Inference for Stan model: spatial_probit_demo.
## 2 chains, each with iter=5000; warmup=2500; thin=1;
## post-warmup draws per chain=2500, total post-warmup draws=5000.
##
##         mean se_mean   sd 2.5%  25%  50%  75% 97.5% n_eff Rhat
## phi     0.33    0.01 0.19 0.11 0.17 0.27 0.46  0.76   235 1.01
## sigmasq 1.19    0.03 0.64 0.50 0.80 1.05 1.40  2.66   339 1.01
##
## Samples were drawn using NUTS(diag_e) at Wed Mar 17 11:55:05 2021.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

3. Alternatively, spGLM can be used

Code below is extracted from help file. Note that phi is the inverse of how we have talked about it.

```
################################
##Spatial binomial
################################

##Generate binary data
coords <- as.matrix(expand.grid(seq(0,100,length.out=8), seq(0,100,length.out=8)))
n <- nrow(coords)

phi <- 3/50
sigma.sq <- 2

R <- sigma.sq*exp(-phi*as.matrix(dist(coords)))
w <- rmvn(1, rep(0,n), R)

x <- as.matrix(rep(1,n))
beta <- 0.1
p <- 1/(1+exp(-(x%*%beta+w)))
```

```r
weights <- rep(1, n)
weights[coords[,1]>mean(coords[,1])] <- 10

y <- rbinom(n, size=weights, prob=p)

##Collect samples
fit <- glm((y/weights)~x-1, weights=weights, family="binomial")
beta.starting <- coefficients(fit)
beta.tuning <- t(chol(vcov(fit)))

n.batch <- 200
batch.length <- 50
n.samples <- n.batch*batch.length

m.1 <- spGLM(y~1, family="binomial", coords=coords, weights=weights,
             starting=list("beta"=beta.starting, "phi"=0.06,"sigma.sq"=1, "w"=0),
             tuning=list("beta"=beta.tuning, "phi"=0.5, "sigma.sq"=0.5, "w"=0.5),
             priors=list("beta.Normal"=list(0,10), "phi.Unif"=c(0.03, 0.3), "sigma.sq.IG"=c(2, 1)),
             amcmc=list("n.batch"=n.batch, "batch.length"=batch.length, "accept.rate"=0.43),
             cov.model="exponential", verbose=TRUE, n.report=10)
```

```
## ----------------------------------------
##  General model description
## ----------------------------------------
## Model fit with 64 observations.
##
## Number of covariates 1 (including intercept if specified).
##
## Using the exponential spatial correlation model.
##
## Number of MCMC samples 10000.
##
## Priors and hyperpriors:
##  beta normal:
##      mu: 0.000
##      sd: 10.000
##
##
##  sigma.sq IG hyperpriors shape=2.00000 and scale=1.00000
##
##  phi Unif hyperpriors a=0.03000 and b=0.30000
##
## Adaptive Metropolis with target acceptance rate: 43.0
## -------------------------------------------------
##      Sampling
## -------------------------------------------------
## Batch: 10 of 200, 5.00%
##  parameter   acceptance  tuning
##  beta[0]     62.0        0.12076
##  sigma.sq    42.0        0.47561
##  phi     74.0        0.55814
## -------------------------------------------------
## Batch: 20 of 200, 10.00%
##  parameter   acceptance  tuning
```

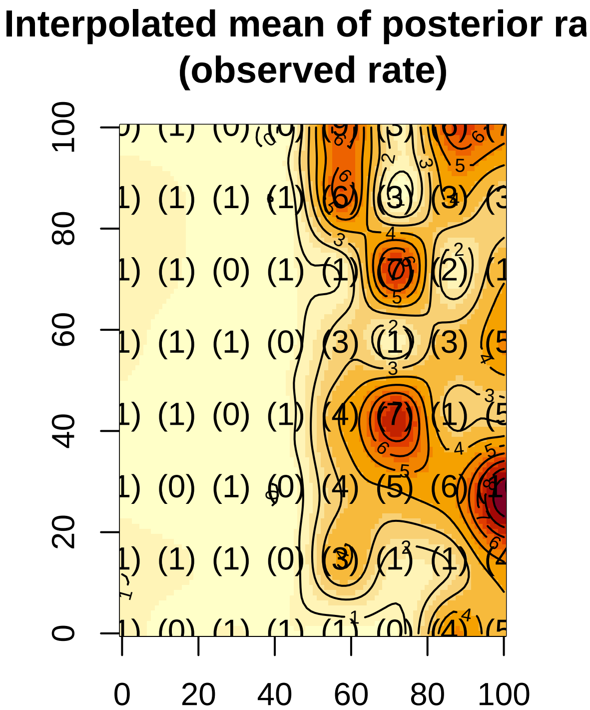
```
##  beta[0]     72.0        0.13346
##  sigma.sq    44.0        0.47561
##  phi     66.0        0.61684
## -------------------------------------------------
## Batch: 30 of 200, 15.00%
##  parameter   acceptance  tuning
##  beta[0]     70.0        0.14750
##  sigma.sq    44.0        0.46620
##  phi     64.0        0.68171
## -------------------------------------------------
## Batch: 40 of 200, 20.00%
##  parameter   acceptance  tuning
##  beta[0]     54.0        0.16301
##  sigma.sq    44.0        0.45697
##  phi     54.0        0.75341
## -------------------------------------------------
## Batch: 50 of 200, 25.00%
##  parameter   acceptance  tuning
##  beta[0]     50.0        0.18016
##  sigma.sq    38.0        0.43905
##  phi     58.0        0.83265
## -------------------------------------------------
## Batch: 60 of 200, 30.00%
##  parameter   acceptance  tuning
##  beta[0]     64.0        0.19910
##  sigma.sq    50.0        0.44792
##  phi     68.0        0.92022
## -------------------------------------------------
## Batch: 70 of 200, 35.00%
##  parameter   acceptance  tuning
##  beta[0]     54.0        0.21569
##  sigma.sq    54.0        0.46620
##  phi     58.0        1.01700
## -------------------------------------------------
## Batch: 80 of 200, 40.00%
##  parameter   acceptance  tuning
##  beta[0]     48.0        0.23837
##  sigma.sq    32.0        0.43905
##  phi     42.0        1.10170
## -------------------------------------------------
## Batch: 90 of 200, 45.00%
##  parameter   acceptance  tuning
##  beta[0]     54.0        0.25822
##  sigma.sq    46.0        0.43035
##  phi     70.0        1.21756
## -------------------------------------------------
## Batch: 100 of 200, 50.00%
##  parameter   acceptance  tuning
##  beta[0]     42.0        0.25311
##  sigma.sq    42.0        0.41348
##  phi     64.0        1.29285
## -------------------------------------------------
## Batch: 110 of 200, 55.00%
##  parameter   acceptance  tuning
```

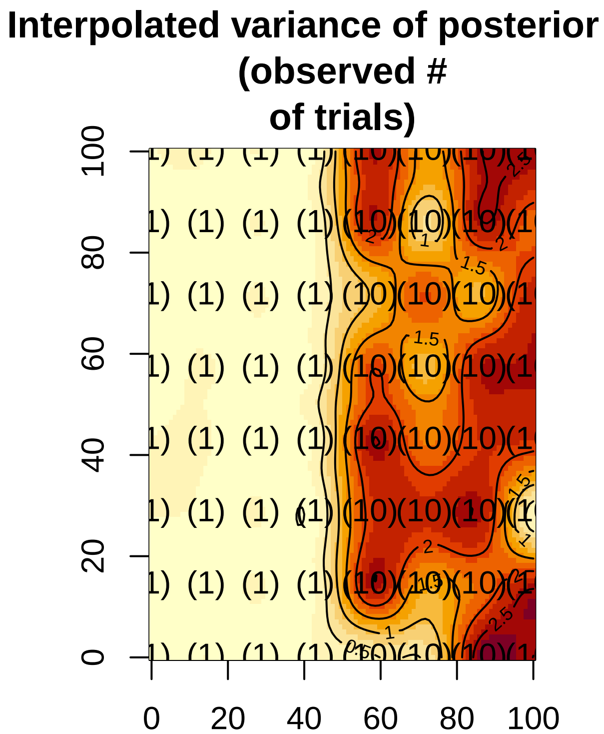
```
##  beta[0]     50.0        0.26876
##  sigma.sq    46.0        0.39727
##  phi     58.0        1.40053
## -------------------------------------------------
## Batch: 120 of 200, 60.00%
##  parameter   acceptance  tuning
##  beta[0]     42.0        0.26344
##  sigma.sq    50.0        0.42183
##  phi     40.0        1.48714
## -------------------------------------------------
## Batch: 130 of 200, 65.00%
##  parameter   acceptance  tuning
##  beta[0]     52.0        0.26876
##  sigma.sq    50.0        0.42183
##  phi     36.0        1.45769
## -------------------------------------------------
## Batch: 140 of 200, 70.00%
##  parameter   acceptance  tuning
##  beta[0]     44.0        0.26876
##  sigma.sq    50.0        0.43905
##  phi     46.0        1.51718
## -------------------------------------------------
## Batch: 150 of 200, 75.00%
##  parameter   acceptance  tuning
##  beta[0]     54.0        0.27973
##  sigma.sq    38.0        0.42183
##  phi     52.0        1.64354
## -------------------------------------------------
## Batch: 160 of 200, 80.00%
##  parameter   acceptance  tuning
##  beta[0]     48.0        0.29114
##  sigma.sq    44.0        0.46620
##  phi     40.0        1.64354
## -------------------------------------------------
## Batch: 170 of 200, 85.00%
##  parameter   acceptance  tuning
##  beta[0]     40.0        0.29114
##  sigma.sq    50.0        0.46620
##  phi     44.0        1.64354
## -------------------------------------------------
## Batch: 180 of 200, 90.00%
##  parameter   acceptance  tuning
##  beta[0]     34.0        0.26876
##  sigma.sq    44.0        0.46620
##  phi     48.0        1.67674
## -------------------------------------------------
## Batch: 190 of 200, 95.00%
##  parameter   acceptance  tuning
##  beta[0]     46.0        0.27973
##  sigma.sq    28.0        0.44792
##  phi     32.0        1.67674
## -------------------------------------------------
## Sampled: 10000 of 10000, 100.00%
## -------------------------------------------------
```

```r
burn.in <- 0.9*n.samples
sub.samps <- burn.in:n.samples

print(summary(window(m.1$p.beta.theta.samples, start=burn.in)))
```

```
##
## Iterations = 9000:10000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 1001
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                  Mean      SD Naive SE Time-series SE
## (Intercept) -0.005949 0.34510 0.010908        0.10437
## sigma.sq     1.074182 0.44161 0.013958        0.07946
## phi          0.114462 0.07004 0.002214        0.01132
##
## 2. Quantiles for each variable:
##
##                 2.5%      25%     50%   75%  97.5%
## (Intercept) -0.86217 -0.18387 0.01522 0.197 0.6769
## sigma.sq     0.36496  0.78100 0.95909 1.320 2.1388
## phi          0.03073  0.06125 0.08929 0.150 0.2799
```

```r
beta.hat <- m.1$p.beta.theta.samples[sub.samps,"(Intercept)"]
w.hat <- m.1$p.w.samples[,sub.samps]

p.hat <- 1/(1+exp(-(x%*%beta.hat+w.hat)))

y.hat <- apply(p.hat, 2, function(x){rbinom(n, size=weights, prob=p.hat)})

y.hat.mu <- apply(y.hat, 1, mean)
y.hat.var <- apply(y.hat, 1, var)

##Take a look
par(mfrow=c(1,2))
surf <- mba.surf(cbind(coords,y.hat.mu),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(surf, main="Interpolated mean of posterior rate\n(observed rate)")
contour(surf, add=TRUE)
text(coords, label=paste("(",y,")",sep=""))

surf <- mba.surf(cbind(coords,y.hat.var),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(surf, main="Interpolated variance of posterior rate\n(observed #
of trials)")
contour(surf, add=TRUE)
text(coords, label=paste("(",weights,")",sep=""))
```


Interpolated mean of posterior ra
(observed rate)


Interpolated variance of posterior r
(observed #
of trials)