

Homework 2

Stat 597a: Spatial Models

Claire Kelling

Due September 28, 2017

Problem 1:

First, read over this section. There is one key change we will make compared to Wikle's analysis, which is that I have first transformed the coordinates from longitude and latitude to UTM coordinates. This is because the `geoRglm` package only uses Euclidean distances. Using it with longitude and latitude will not give accurate distances between the points. Another way we could do it, if we were coding this from scratch, would be to keep longitude and latitude, but work with great circle distances when calculating the distance matrix.

Given this new coordinate system, the prior for the range parameter ϕ that we'll use is a discrete uniform distribution from 500 to 300,000, in increments of 500. The reasons for the discrete prior are again due to the constraints of the `geoRglm` package.

With this change, write down the three layers of the hierarchical model as defined on slide 8 of Lecture 8. That is, what distributions make up the data model, process model, and prior model?

From the lecture, we know there are going to be 3 components to our model: the data model, the process model, and the parameter model. That is, $f(\eta, \theta | \mathbf{Y}) \propto f(\mathbf{Y}, \eta, \theta) \times f(\eta | \theta) \times \pi(\theta)$.

According to the textbook, Hierarchical Model with Spatial Data, we will model this data according to the generalized linear spatial modeling framework.

So, the data model is as follows:

$$f(Y(s_i) | \lambda(s_i), \phi) \sim \text{independent } \text{Poisson}(\lambda(s_i)), i = 1, \dots, m$$

where we employ a Gaussian spatial process model to describe the spatial variation in $\lambda(s_i)$ and use the canonical log-link function. So, $\log(\lambda(s_i)) = \beta + \eta(s_i)$ where $\eta(s_i)$ is a Gaussian process with mean 0, variance σ_n^2 and correlation function $r(s_i, s_j; \phi)$. Specifically, $r_\eta(s_i, s_j; \phi) = \exp(-||s_i - s_j||/\phi)$. We were told in the problem statement that ϕ follows a discrete uniform distribution from 500 to 300,000, in increments of 500. Therefore, the process model is as follows:

$$\log(\lambda(s_i)) = \beta + \eta(s_i) \text{ where}$$

$$f(\eta(s_i) | \phi) \sim \text{GP}(\text{mean} = 0, \text{variance} = \sigma_n^2, \text{correlation function} = r_\eta(s_i, s_j; \phi) = \exp(-||s_i - s_j||/\phi)$$

and the parameter model is as follows:

$$\pi(\phi) \sim \text{Uniform}(500, 300,000) \text{ with breaks as described above.}$$

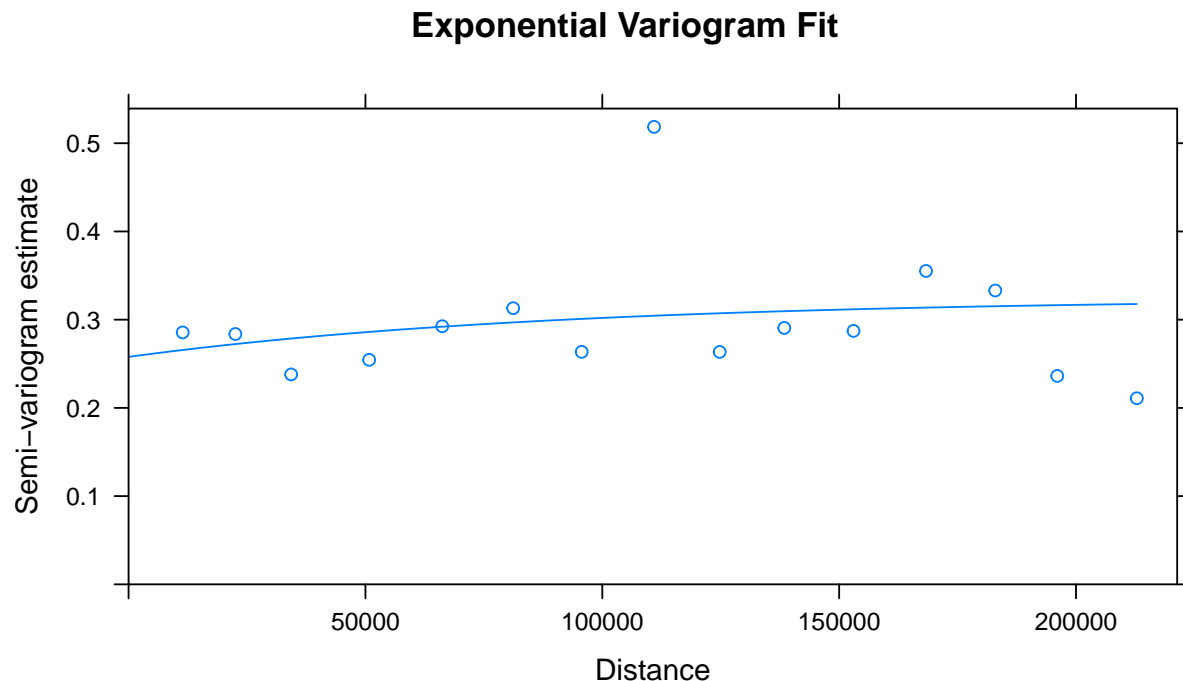
Also, the parameter model for β is flat, so $\pi(\beta) \propto 1$ and σ^2 has a uniform prior, according to the book.

Problem 2

Transform the original observations using $Z_i = \log(Y_i)$ and use classical geostatistical techniques to get preliminary estimates of σ^2 and ϕ by treating the Z_i as normally distributed given η (i.e. fit a model with a nugget, and extract just the estimates for σ^2 and ϕ).

We have our estimates of σ^2 as 0.06637062 and our estimate of ϕ to be 91436.2.

```
plot(vg, fitvg.3, xlab = "Distance", ylab = "Semi-variogram estimate", layout=c(2,1), main = "Exponential Variogram Fit")
```



Above, we have also included the fitted exponential variogram, which seems like a pretty good fit of our data. Therefore, we will proceed into the MCMC.

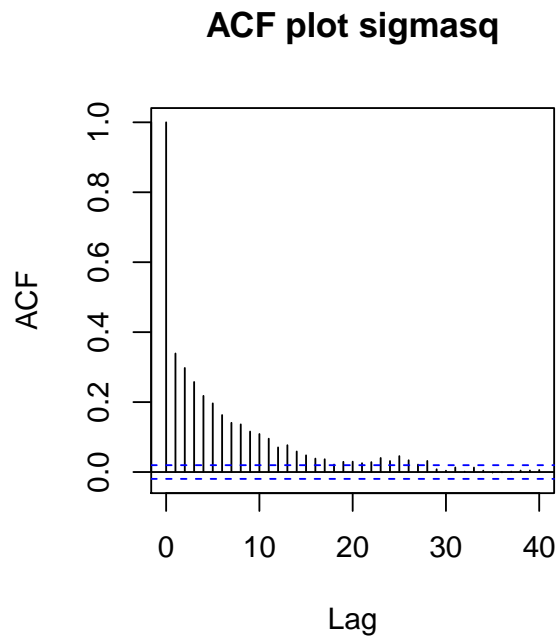
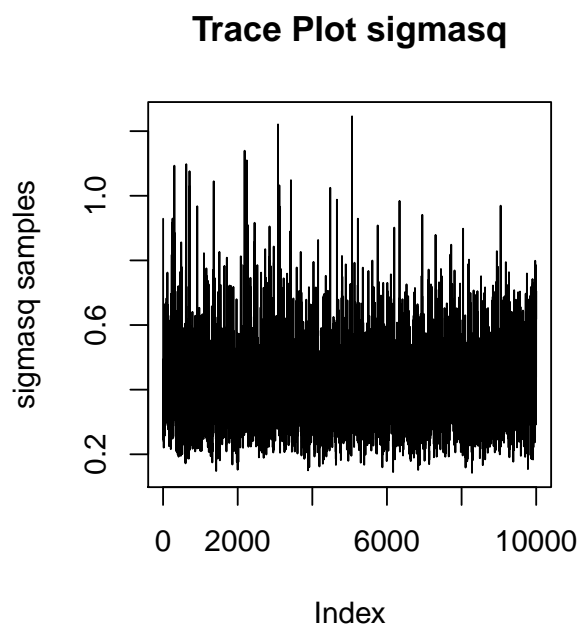
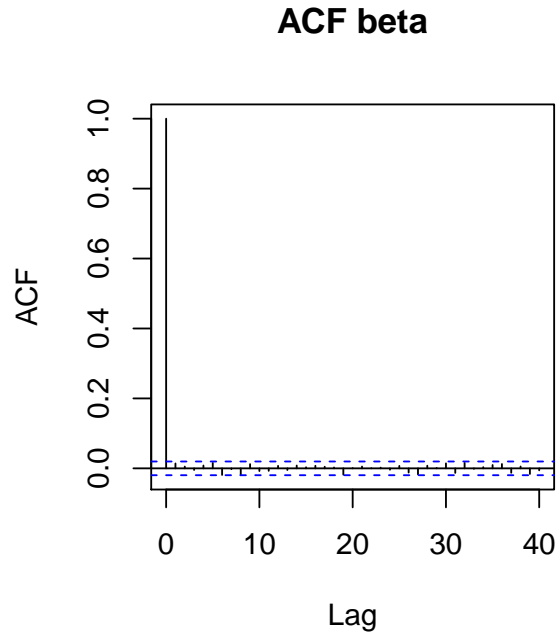
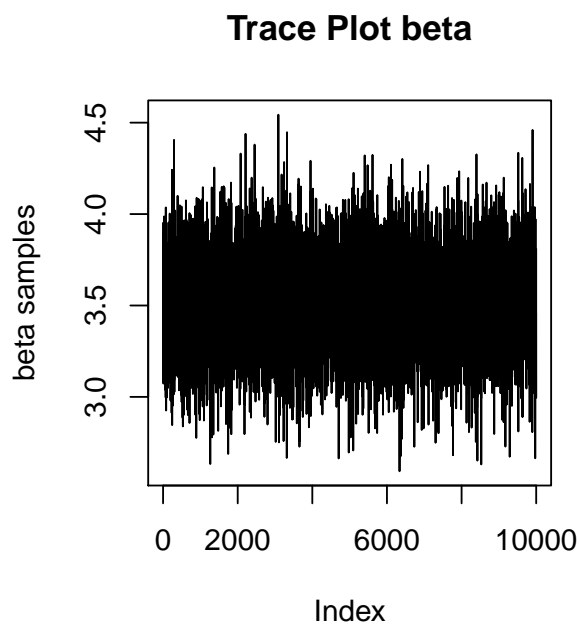
Problem 3

Use `model.glm.control`, `prior.glm.control`, `mcmc.control`, and `pois.krige.bayes` to run an initial MCMC chain, fixing ϕ at your estimate from (1). The goal here is to experiment with changing `S.scale` to achieve an acceptance rate of about 60% for the process samples, which is optimal for the algorithm `pois.krige.bayes` is using to sample the vector of process values.

The choice of `niter = 100000` and `thin = 10` is just to keep things manageable at this stage. We will eventually run a much longer chain. I suggest you always keep `burn.in = 0` and then discard the initial samples yourself after seeing the results.

After I ran the MCMC, I include ACF and trace plots below. It was not too difficult to achieve an acceptance rate of 60% in this case, although this was exceedingly difficult to do later on. I also note that in the ACF and trace plots below, there is not too much autocorrelation. There is a bit more autocorrelation in the trace and ACF plots for σ^2 . For this example, I achieved an acceptance rate that is also included below.

```
## iter.numb  Acc.rate
##    1.0e+05   5.8e-01
```



Problem 4

Now duplicate and modify the code from (3) to include sampling ϕ . Experiment with changing `S.scale` and `phi.scale` to get acceptance rates of about 60% and 25%, respectively. (Note:

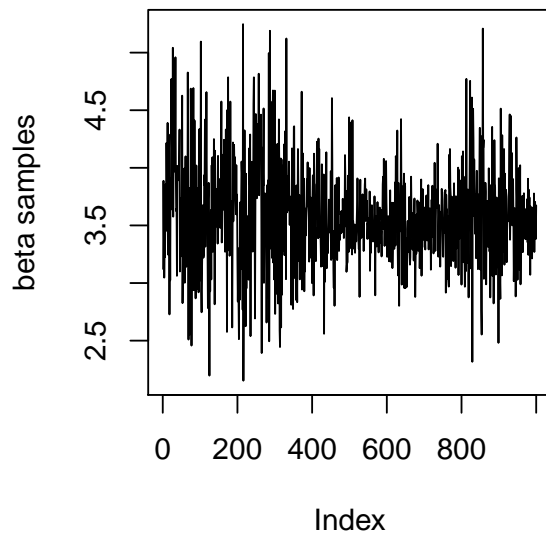
I found the acceptance rates fluctuated over the course of my chain. Just aim to get in the right ballpark.) I suggest you also take `thin = 100` and `n.iter = 100000` here. Make trace plots and ACF plots of the parameters σ^2 , ϕ , and β . You will likely see a LOT of autocorrelation.

As in Problem 3, I have run the MCMC code and I include the ACF and trace plots below. I also include the acceptance rates below. For this iteration, I had a very difficult time trying to find the values of `S.scale` and `phi.scale` that would lead anywhere close to the “acceptable” acceptance rates.

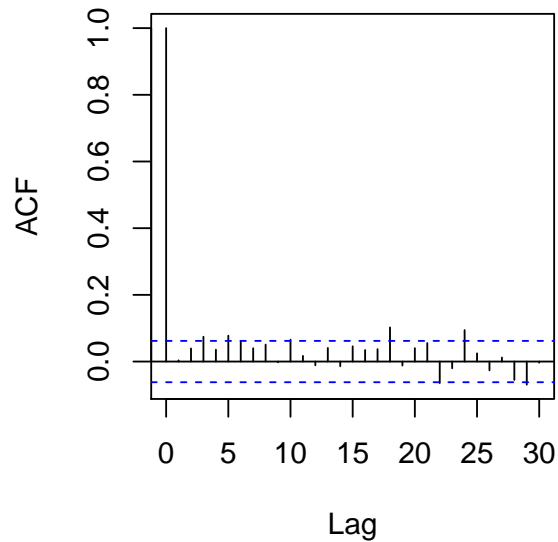
When I look at the acf and trace plots again for this example, I notice that there is quite a bit of autocorrelation, as we expected to see. This is especially relevant for the plots of σ^2 and ϕ , with ϕ having the most autocorrelation.

```
##      iter.numb      Acc.rate Acc.rate.phi
##      1.00e+05      6.56e-01  2.14e-01
```

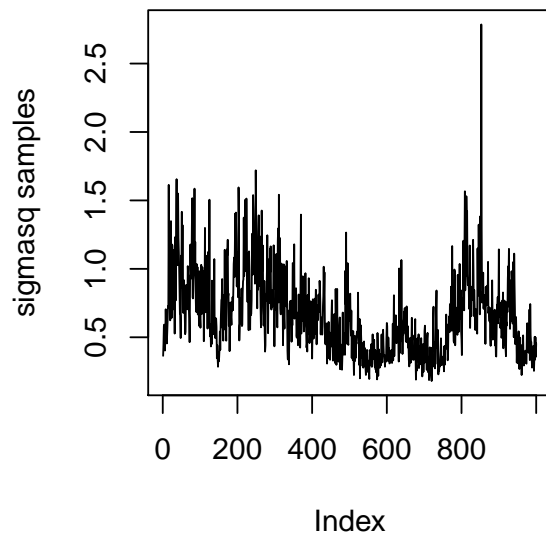
Trace Plot beta



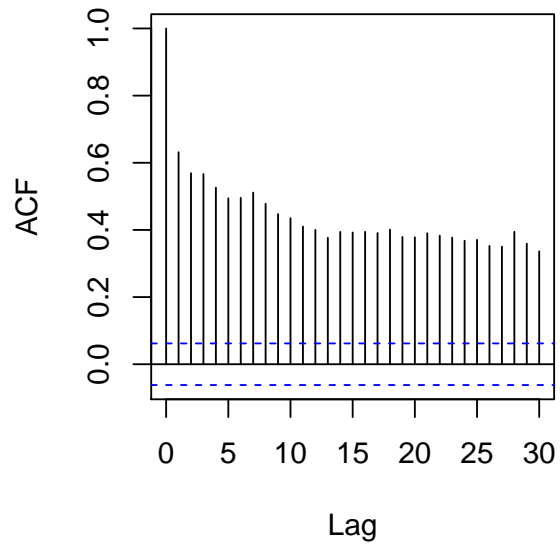
ACF plot beta



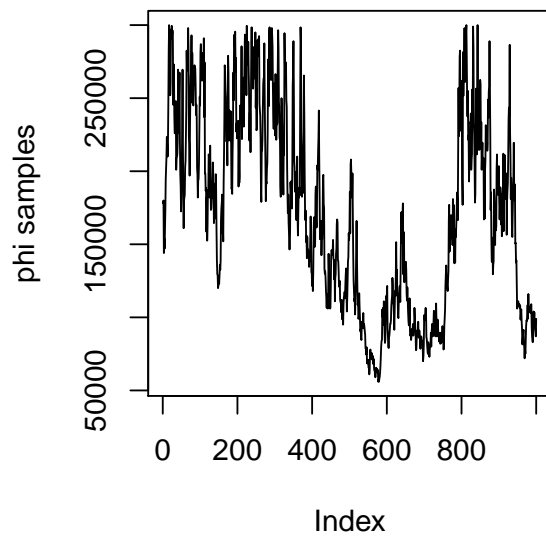
Trace Plot sigmasq



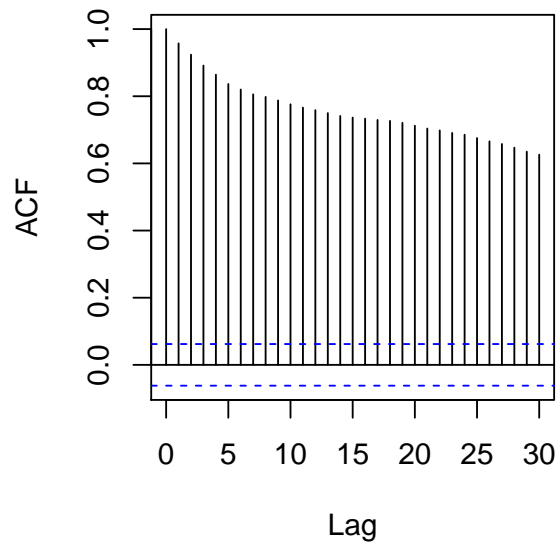
ACF plot sigmasq



Trace Plot phi



ACF plot phi



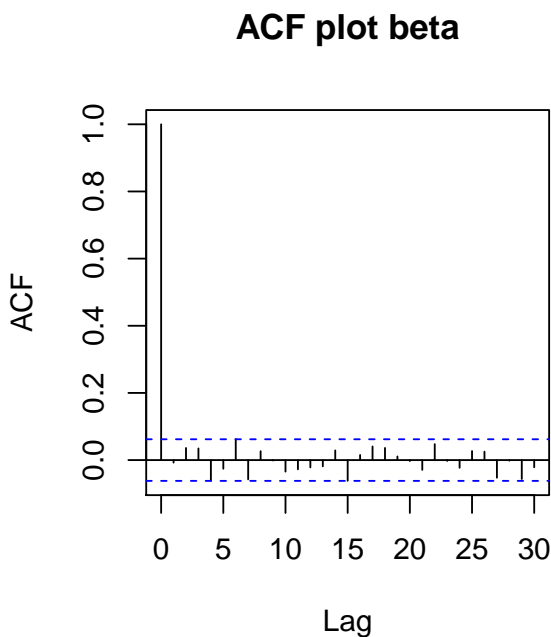
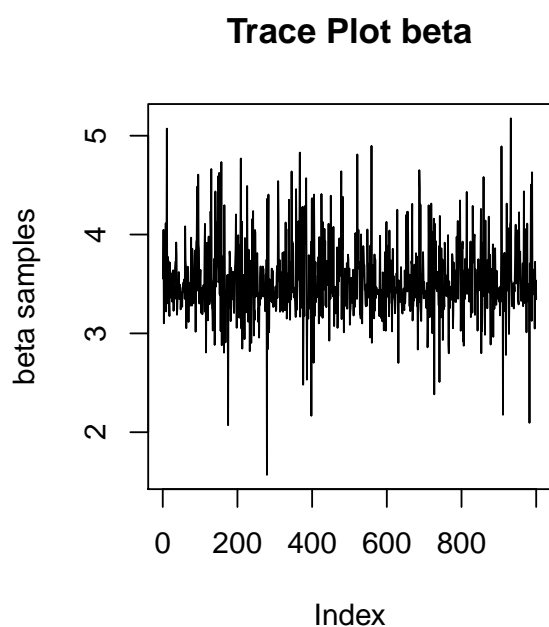
Problem 5

There is not much beyond changing $S.\text{scale}$ and $\phi.\text{scale}$ that we can do to reduce the auto-correlation. So we will run a long chain and subsample it. Run the same code as in part (4), increasing to $\text{thin} = 1e4$ and $\text{n.iter} = 1e7$. Now do the following with your sampled parameters.

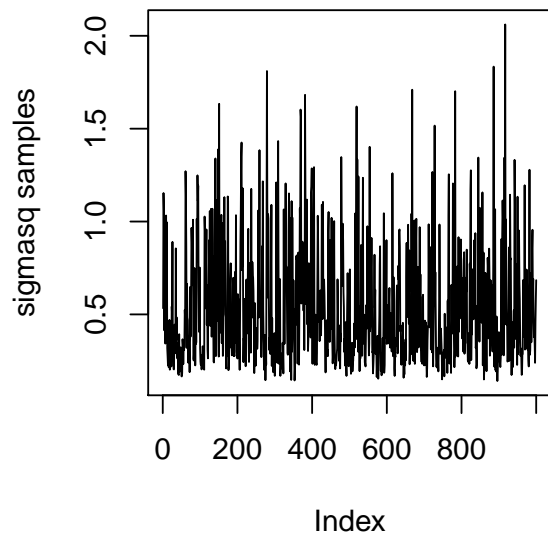
- Make trace plots and ACF plots. Choose a burn-in to discard and make them again.
- Calculate the effective sample size you have for each parameter. If they are not at least 100 for each parameter, go back and run a longer chain.
- Plot the marginal posterior distributions for each parameter using histograms and/or kernel density estimators.

Once again, I include my acceptance rate below as well as my trace and ACF plots. For this example, I chose my burnin value to be 50,000 I notice that the plots vary quite a bit less after I get rid of the burnin.

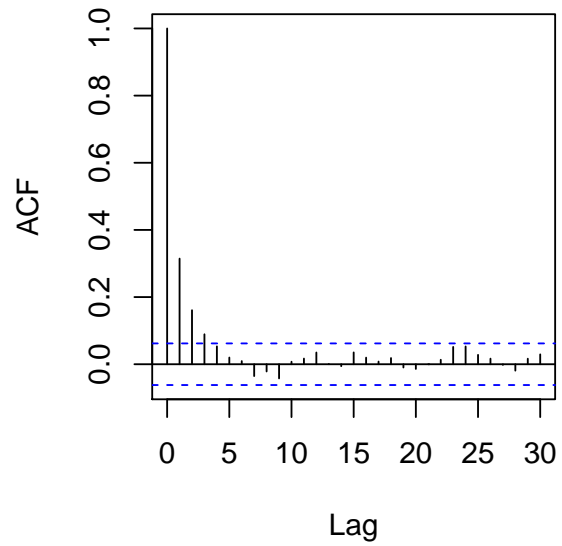
```
##      iter.numb      Acc.rate Acc.rate.phi
##      1.00e+07      2.87e-01  5.83e-01
```



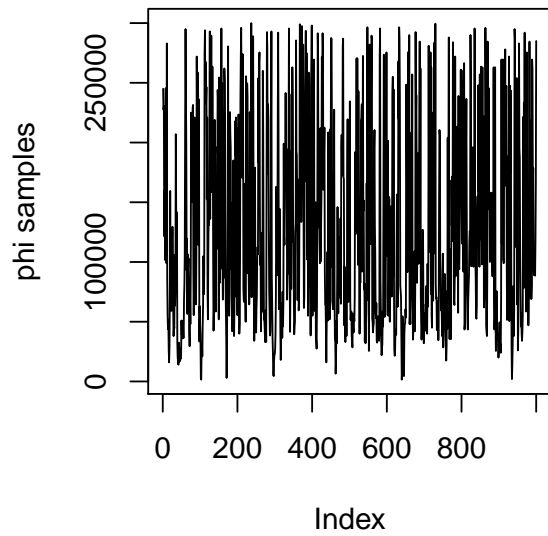
Trace Plot sigma_{sq}



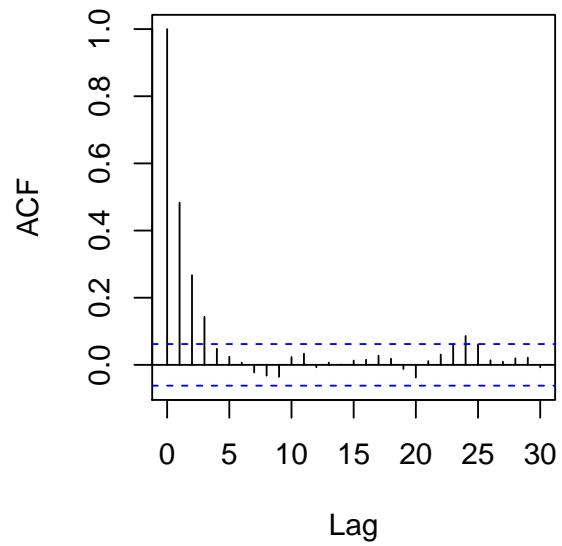
ACF plot sigma_{sq}



Trace Plot phi

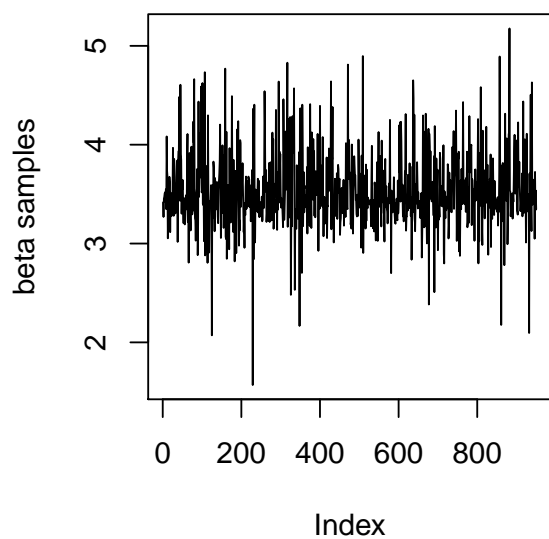


ACF plot phi

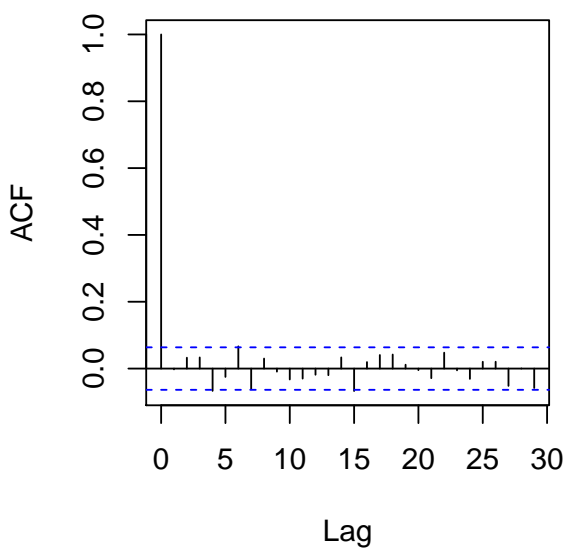


I remove the burnin and re-create the plots.

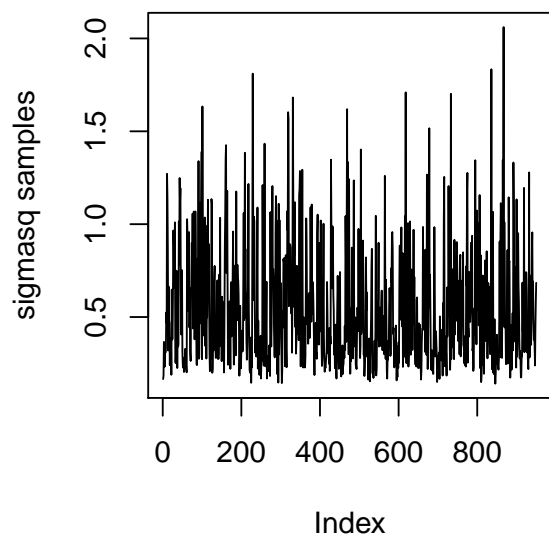
Post Burnin Trace Plot beta



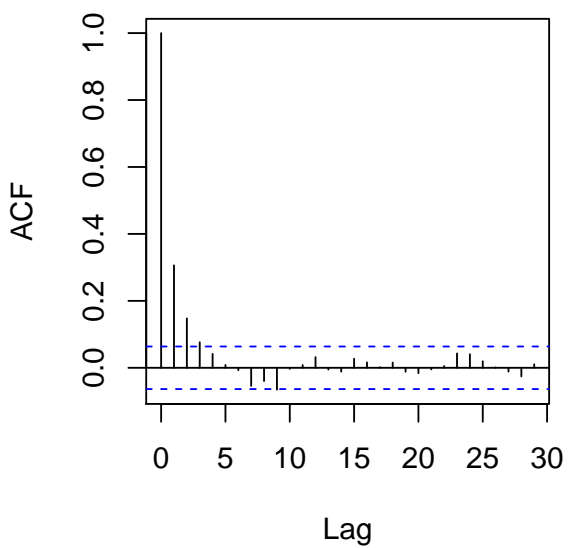
Post Burnin ACF plot beta

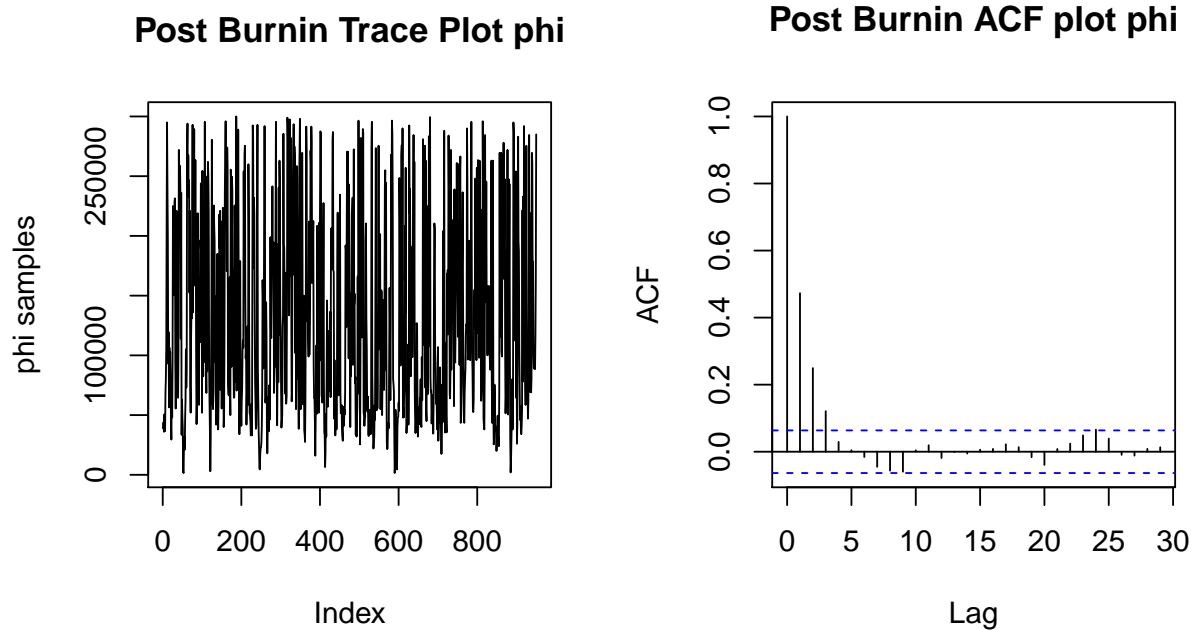


Post Burnin Trace Plot sigmasq



Post Burnin ACF plot sigmasq





My effective sample sizes for each parameter are included below, and are all above 100.

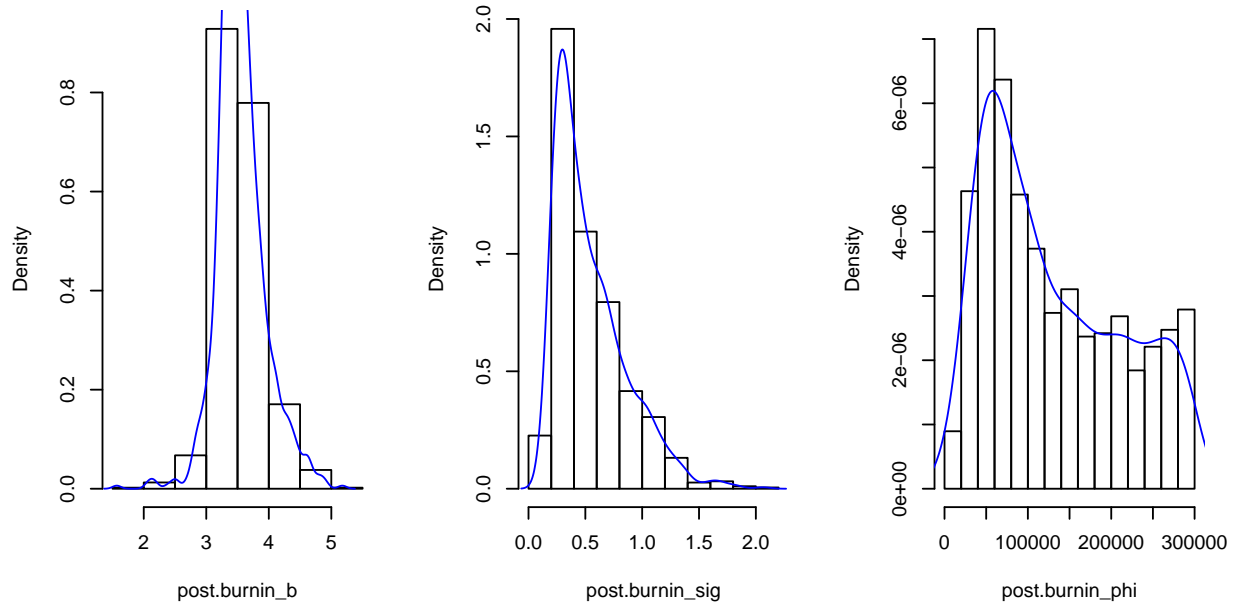
I have also included plots for the marginal posterior distributions for each parameter, which includes a histogram and the kernel density esimators.

```
## [1] "Sigmasq effective sample size is 443.92056108212"
```

```
## [1] "Phi effective sample size is 310.473594820257"
```

```
## [1] "Beta effective sample size is 840.117554999214"
```

ta histogram with kernel density estimation histogram with kernel density estimation histogram with kernel density estimation

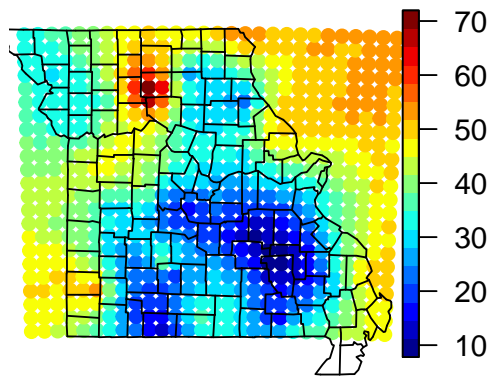


Problem 6

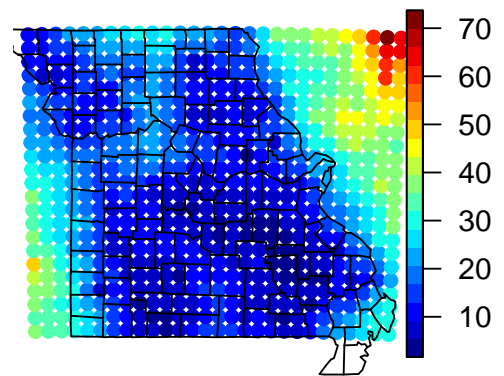
Do one final run of the chain, the same as in (5), but this time specifying the burn-in value you chose in (5) and also modifying the code to include prediction at the locations in `dove.grid`. Create two color or grayscale plots, showing the posterior mean and standard deviation for the underlying mean surface ($\exp \eta$) at each location in `dove.grid`.

Once again, I have made a final run of the chain, as in (5), but I specified the burnin value that I chose in (5) in the function. I also included my two plots with the posterior mean and standard deviation of the underlying mean surface below. I also included a plot of the original observations. I see that the mean surface corresponds pretty well with this graph and that the standard deviation is highest where there aren't many observations. It just so happens that the lowest standard deviations (where there are more observations) occur where there are also low dove counts, in the middle of the state.

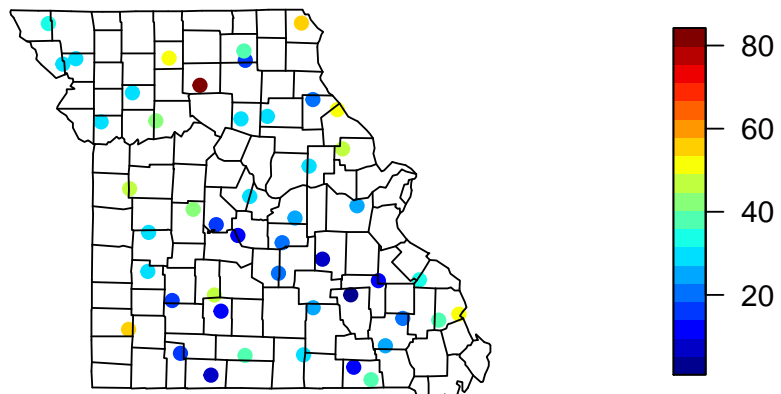
Posterior mean



Standard Deviation



Actual Data Points, log transform ($z_i = \log(\text{count}_i)$)



```
##  
#Problem 2  
##  
  
#load the data  
load("C:/Users/ckell/OneDrive/Penn State/2017-2018/597/spatial_statistics_597/Homework 2/data/dove.RData")
```

```

#transform the original observations
dove$z <- log(dove$counts)

#use classical geostatistical techniques to get preliminary estimates of  $\sigma^2$  and  $\phi$ 

#Fitting the preliminary linear model
linmod <- lm(z~coords.x1+coords.x2, data=dove)
#linmod <- lm(z~1, data=dove)
summary(linmod)
#storing the residuals so I can plot them
dove$resid <- linmod$resid

## Nonparametric estimation of the variogram
vg <- variogram(resid ~ 1, data = dove)#, width=75)
#print(vg)
plot(vg, xlab = "Distance", ylab = "Nonparametric Semi-variogram estimate", width=5, main = "Exponential")

## Fitting the variogram parametrically
# fitvg <- fit.variogram(vg, vgm(1, "Exp", range=47, nugget=1), fit.method = 2)
# print(fitvg)
# fitvg.2 <- fit.variogram(vg, vgm(1, "Exp", 47, 0.05))
# print(fitvg.2)
fitvg.3 <- fit.variogram(vg, vgm("Exp"))
print(fitvg.3)

s2.hat <- fitvg.3$psill[2]
phi.hat <- fitvg.3$range[2]
tau2.hat <- fitvg.3$psill[1]

##
#Problem 3
##

dove_mcmc <- as.geodata(dove)

#run an initial MCMC chain, fixing phi at your estimate from
#achieve an acceptance rate of about 60% for the process samples
sim.model <- model.glm.control(cov.model= "exponential")#, kappa = NA, not required for exponential
#0.017
set.seed(132)
sim.prior <- prior.glm.control(phi.prior = "fixed", phi = phi.hat)
sim.mcmc <- mcmc.control(S.scale = 0.0098, Htrunc = "default", S.start = "random",
                        burn.in = 0, thin = 10, n.iter = 100000)
sim.posterior <- pois.krige.bayes(dove_mcmc, model=sim.model, prior=sim.prior,
                                mcmc.input=sim.mcmc)#, keep.mcmc.sim=TRUE)

nrow(sim.posterior$posterior$acc.rate)
sim.posterior$posterior$acc.rate[100,]

##
#Problem 3 images
##
sim.posterior$posterior$acc.rate[100,]
par(mfrow=c(1,2))

```

```

#ACF and trace plots
plot(sim.posterior$posterior$beta$sample, type = "l", ylab = "beta samples", main = "Trace Plot beta")
acf(sim.posterior$posterior$beta$sample, main = "ACF beta")

plot(sim.posterior$posterior$sigma$sample, type = "l", ylab = "sigma samples", main = "Trace Plot sigma")
acf(sim.posterior$posterior$sigma$sample, main = "ACF plot sigma")

##
#Problem 4
##

#Now duplicate and modify the code from (3) to include sampling $phi$.
sim.model <- model.glm.control(cov.model= "exponential")#, kappa = NA, not required for exponential
sim.prior <- prior.glm.control(phi.prior = "uniform", phi.discrete = seq(500,300000,500))
# Experiment with changing S.scale and phi.scale to get acceptance rates of about 60%
# and 25%, respectively.
set.seed(134)
sim.mcmc <- mcmc.control(S.scale = 0.008, phi.scale=6.5e7, Htrunc = "default",
                        S.start = "random",
                        burn.in = 0, thin = 100, n.iter = 100000)
sim.posterior <- pois.krige.bayes(dove_mcmc, model=sim.model, prior=sim.prior,
                                mcmc.input=sim.mcmc)

mean(sim.posterior$posterior$acc.rate[,2])
mean(sim.posterior$posterior$acc.rate[,3])

nrow(sim.posterior$posterior$acc.rate)
sim.posterior$posterior$acc.rate[100,]

##
#Problem 4 images
##

sim.posterior$posterior$acc.rate[100,]

par(mfrow=c(1,2))
#acf and trace plots for beta
plot(sim.posterior$posterior$beta$sample, ylab = "beta samples", type = "l", main = "Trace Plot beta")
acf(sim.posterior$posterior$beta$sample, main = "ACF plot beta")

#acf and trace plots for sigma
plot(sim.posterior$posterior$sigma$sample, ylab = "sigma samples", type = "l", main = "Trace Plot sigma")
acf(sim.posterior$posterior$sigma$sample, main = "ACF plot sigma")

#acf and trace plots for phi
plot(sim.posterior$posterior$phi$sample, ylab = "phi samples", type = "l", main = "Trace Plot phi")
acf(sim.posterior$posterior$phi$sample, main = "ACF plot phi")

##
#Problem 5
##

#Run the same code as in part (4), increasing to thin = 1e4 and n.iter = 1e7.
sim.model <- model.glm.control(cov.model= "exponential")#, kappa = NA, not required for exponential

```

```

sim.prior <- prior.glm.control(phi.prior = "uniform", phi.discrete = seq(500,300000,500))
# Experiment with changing S.scale and phi.scale to get acceptance rates of about 60%
# and 25%, respectively.
set.seed(134)
sim.mcmc <- mcmc.control(S.scale = 0.008, phi.scale=6.5e7, Htrunc = "default",
                        S.start = "random",
                        burn.in = 0, thin = 1e4, n.iter = 1e7)
sim.posterior <- pois.krige.bayes(dove_mcmc, model=sim.model, prior=sim.prior,
                                mcmc.input=sim.mcmc)

n <- nrow(sim.posterior$posterior$acc.rate)
sim.posterior$posterior$acc.rate[n,]

##
#Problem 5 images
##

sim.posterior$posterior$acc.rate[n,]

par(mfrow=c(1,2))
#acf and trace plots for beta
plot(sim.posterior$posterior$beta$sample, ylab = "beta samples", type = "l", main = "Trace Plot beta")
acf(sim.posterior$posterior$beta$sample, main = "ACF plot beta")

#acf and trace plots for sigmasq
plot(sim.posterior$posterior$sigmasq$sample, ylab = "sigmasq samples", type = "l", main = "Trace Plot sigmasq")
acf(sim.posterior$posterior$sigmasq$sample, main = "ACF plot sigmasq")

#acf and trace plots for phi
plot(sim.posterior$posterior$phi$sample, ylab = "phi samples", type = "l", main = "Trace Plot phi")
acf(sim.posterior$posterior$phi$sample, main = "ACF plot phi")

#choose a burnin to discard and make them again
burnin <- 1:50
#length(sim.posterior$posterior$sigmasq$sample)
post.burnin_sig <- sim.posterior$posterior$sigmasq$sample[-c(burnin)]
post.burnin_b <- sim.posterior$posterior$beta$sample[-c(burnin)]
post.burnin_phi <- sim.posterior$posterior$phi$sample[-c(burnin)]

par(mfrow=c(1,2))
#acf and trace plots for beta
plot(post.burnin_b, ylab = "beta samples", type = "l", main = "Post Burnin Trace Plot beta")
acf(post.burnin_b, main = "Post Burnin ACF plot beta")

#acf and trace plots for sigmasq
plot(post.burnin_sig, ylab = "sigmasq samples", type = "l", main = "Post Burnin Trace Plot sigmasq")
acf(post.burnin_sig, main = "Post Burnin ACF plot sigmasq")

#acf and trace plots for phi
plot(post.burnin_phi, ylab = "phi samples", type = "l", main = "Post Burnin Trace Plot phi")
acf(post.burnin_phi, main = "Post Burnin ACF plot phi")

#calculate the effective sample size for each parameter
##if they are not all at least 100 for each parameter, go back and run a longer chain

```

```

print(paste("Sigmasq effective sample size is ", ess(post.burnin_sig)))
print(paste("Phi effective sample size is ", ess(post.burnin_phi)))
print(paste("Beta effective sample size is ", ess(post.burnin_b)))

#plot the marginal and posterior distributions for each parameter using histograms and/or kernel density
#par(mfrow = c(1,3))
#
par(mfrow=c(1,3))
hist(post.burnin_b,freq=F, main ="beta histogram with kernel density estimator")
points(density(post.burnin_b), type = 'l', col = 'blue')
hist(post.burnin_sig, freq=F, main = "sigmasq histogram with kernel density estimator")
points(density(post.burnin_sig), type = 'l', col = 'blue')
hist(post.burnin_phi,freq=F, main = "phi histogram with kernel density estimator")
points(density(post.burnin_phi), type = 'l', col = 'blue')

##
#Problem 6
##

#Do one final run of the chain, the same as in (5), but this time specifying the burn-in value you chose
sim.model <- model.glm.control(cov.model= "exponential")#, kappa = NA, not required for exponential
sim.prior <- prior.glm.control(phi.prior = "uniform", phi.discrete = seq(500,300000,500))
#S.scale = 0.5, 0.2, 0.7
# Experiment with changing S.scale and phi.scale to get acceptance rates of about 60%
# and 25%, respectively.
set.seed(134)
sim.mcmc <- mcmc.control(S.scale = 0.008, phi.scale=6.5e7, Htrunc = "default",
                        S.start = "random",
                        burn.in = 50e3, thin = 1e4, n.iter = 1e7)

#keep MCMC sample
sim.posterior <- pois.krige.bayes(dove_mcmc, model=sim.model, prior=sim.prior,
                                mcmc.input=sim.mcmc, locations=dove.grid, output =
                                output.glm.control(keep.mcmc.sim=TRUE, sim.predict = TRUE))

pred_sim <- sim.posterior$predictive$simulations
#test2 <- sim.posterior$posterior$simulations

post_med <- sim.posterior$predictive$median
uncertainty <- sim.posterior$predictive$uncertainty

#pred_sim <- exp(pred_sim)

post_mean <- apply(pred_sim, 1, mean)
post_sd <- apply(pred_sim, 1, sd)

##
#Problem 6 images
##
ploteqc <- function(spobj, z, breaks, ...){
  pal <- tim.colors(length(breaks)-1)
  fb <- classIntervals(z, n = length(pal),
                      style = "fixed", fixedBreaks = breaks)
  col <- findColours(fb, pal)

```

```

plot(spobj, col = col, ...)
image.plot(legend.only = TRUE, zlim = range(breaks), col = pal)
}

library(rgdal)
utms <- SpatialPoints(cbind(dove.grid$x, dove.grid$y),
                      proj4string=CRS("+proj=utm +zone=15"))
dove.longlat <- spTransform(utms, CRS("+proj=longlat"))

utms2 <- SpatialPoints(dove, proj4string=CRS("+proj=utm +zone=15"))
?spTransform
dove.longlat2 <- spTransform(utms2, CRS("+proj=longlat"))

par(mfrow=c(1,2))

ploteqc(dove.longlat, post_mean, seq(min(post_mean),max(post_mean),length=20), pch=19)
map("county", region = "missouri", add = TRUE)
title("Posterior mean")

ploteqc(dove.longlat, post_sd, seq(min(post_sd),max(post_sd),length=20), pch=19)
map("county", region = "missouri", add = TRUE)
title("Standard Deviation")
points(dove)

par(mfrow=c(1,1))

ploteqc(dove.longlat2, dove$counts, seq(min(dove$counts),max(dove$counts),length=20), pch=19)
map("county", region = "missouri", add = TRUE)
title("Actual Data Points, log transform (z_i= log(count_i))")

# dove$counts <- exp(dove$counts)
# ploteqc(dove.longlat2, dove$counts, seq(min(dove$counts),max(dove$counts),length=20), pch=19)
# map("county", region = "missouri", add = TRUE)
# title("Actual Data Points, exp(z_i)")

# plot(dove)
#
# range(post_med)
# breaks <- seq(8, 70, by = 0.01)
#
#
# ploteqc(dove.grid, post_med, breaks, pch = 19)
# map("county", region = "missouri", add = TRUE)
# title(main = "Posterior Median")
#
#

```



```
# range(uncertainty)
# breaks <- seq(2, 35, by = 0.01)
# ploteqc(dove.grid, uncertainty, breaks, pch = 19)
# #map("county", region = "missouri", add = TRUE)
# title(main = "Posterior Mean")
# points(dove)
```