Yonsei University, Department of Applied Statistics

Spatio Temporal Data Analysis HW1

응용통계학과 신보현 2019321817

April 10, 2020

1.

(a) Using the property of multivariate normal random variable that its linear function also follows multivariate normal, random vector Y follows multivariate normal and it suffices to show its mean vector and covariance matrix.

$$E[Y] = E[\mu + LZ] = \mu,\ Cov(Y) = Cov(\mu + LZ) = LCov(Z)L^{'} = LL^{'} = \Sigma$$

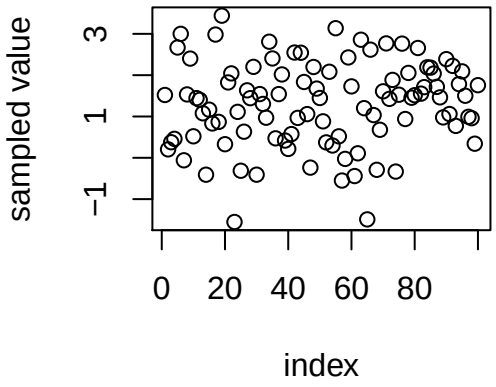
(b)

```
generate_MVN = function(mu, sigma, n){
L = chol(sigma)
n = dim(sigma)[1]
Z = rmvnorm(1, mean=rep(0,n), sigma= diag(n))
Y = mu + L %*% t(Z)
return(Y)
}
```

(c)

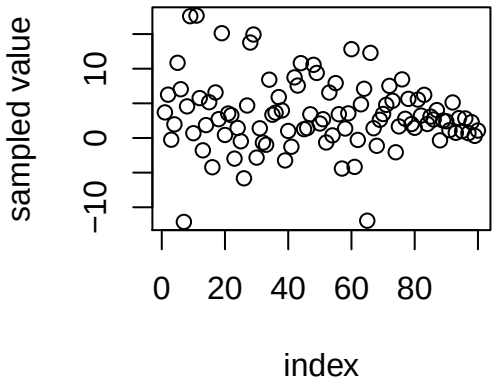
First, I tried independence case in which I chose identity matrix as covariance matrix and second, I generated covarance matrix using runif function. In Both of cases, I fixed Z vector and μ. Below is the R code and output.

```
library(mvtnorm)
set.seed(2019321817)
n = 100
Z = rmvnorm(1, mean=rep(0,n), sigma = diag(n))
generate_MVN_fixed_Z = function(mu, sigma, Z){
L = chol(sigma)
n = dim(sigma)[1]
Y = mu + L %*% t(Z)
return(Y)
}
sigma = diag(n)
plot(1:100, generate_MVN_fixed_Z(rep(1,100), sigma = sigma, Z), xlab='index', ylab='sampled value')
```



As I chose identity matrix, the Gaussian Process shown in the plot has no pattern. It is randomly scattered. Next is the case using generated covariance matrix

```
set.seed(2019321817)
A = matrix(runif(n^2)*2-1, ncol=n)
sigma = t(A) %*% A
plot(1:100, generate_MVN_fixed_Z(rep(1,100), sigma = sigma, Z), xlab='index', ylab='sampled value')
```



The difference is clearly noticed that the generated GP is somewhat correlated. There is pattern narrowing down.

2-(a)

First, to use longitude and lantitude as variables to our model, we do the following coding

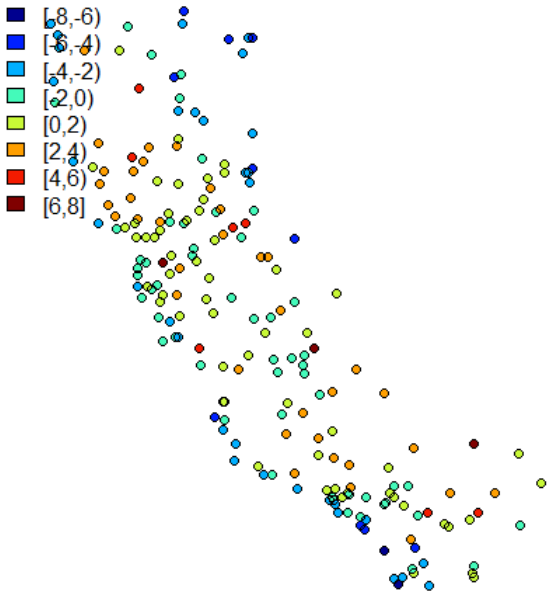
```
library(sp)
library(gstat)
library(fields)
library(classInt)
library(maps)
load("C:/Users/sbh0613/Desktop/20년 1학기/시공간/HW1/CAtemps.RData")
df = cbind(CAtemp, coordinates(CAtemp)) # to use lon, lat variable
```

Now, fit the ordinary least squares, ignoring spatial dependence using lm function in R and calclate the residuals.

```
ols_result = lm(avgtemp ~ lon + lat + elevation, data = df)
fitted = predict(ols_result, newdata = df, na.action = na.pass) # fitted values
ehat = df$avgtemp - fitted # residuals
```

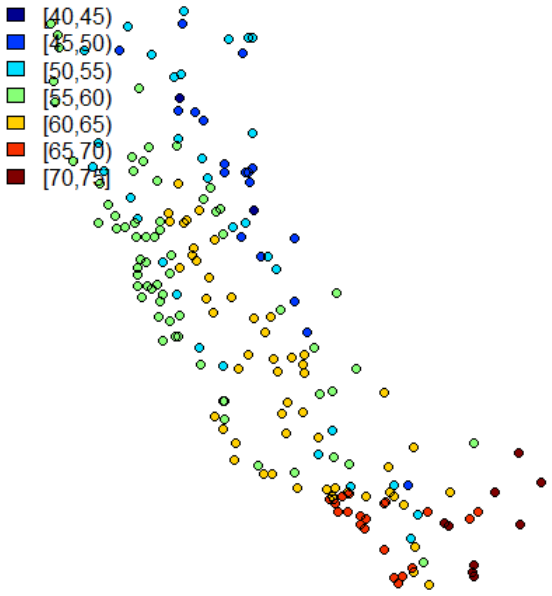
Using the professors' code, plot the residuals by color

```
# function from the STDA2 script
plot.point.ref <- function(spatialdata, vals) {
pal <- tim.colors(10)
ints <- classIntervals(vals, n = 8, style = "pretty") # Determine breakpoints
# also see style options "quantile" and "fisher"
intcols <- findColours(ints, pal) # vector of colors
# if pal doesn't have the same length as # classes, findColours will interpolate
par(mar = rep(0, 4))
plot(spatialdata, col = intcols, pch = 19)
points(spatialdata, pch = 1)
legend("topleft", fill = attr(intcols, "palette"),
legend = names(attr(intcols, "table")), bty = "n")
}
plot.point.ref(df, ehat)
```



We can notice some slight pattern in residual plot in that blue points at the top and left, orange points at the middle, red points at the bottom. Note that similar pattern is noticed in the estimates plot.

```
plot.point.ref(df, fitted)
```

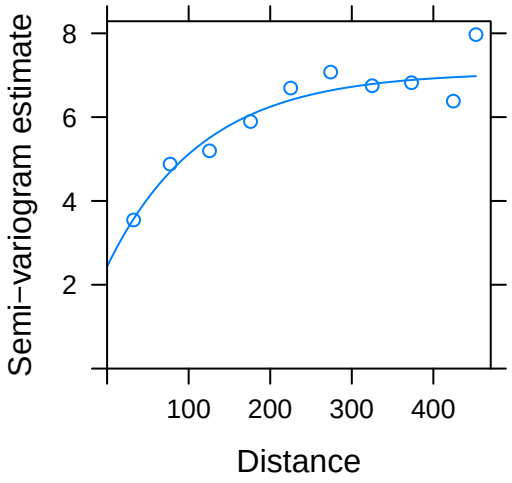


Residuals mean the rest part after fitting assumed model. If residuals show some patterns, then it means that we should include other variables or do another modeling that accounts for the patterns of residuals. In this example, residuals show similar pattern with the fitted value, which denotes spatial dependence.

(b)

We estimate variogram nonparametrically and then fit the exponential variogram to it using weighted least squares, by using following R code.

```
vg = variogram(ehat ~ 1, data = df, width=50)
# fitting exponential variogram using nonparametric estimates
fitvg = fit.variogram(vg, vgm(1, "Exp", 500, 0.05))
plot(vg, fitvg, xlab = "Distance", ylab = "Semi-variogram estimate")
```



The nonparametrically estmated variograms are plotted in points and parametric exponential variograms are drawn as a curve.

We report the parameter estimates for exponential variogram models.

```
# report parameter estimates
s2.hat = fitvg$psill[2]
rho.hat = fitvg$range[2]
tau2.hat = fitvg$psill[1]
print(fitvg)


##   model    psill   range
## 1   Nug 2.434884   0.000
## 2   Exp 4.635659 115.766
```

That is, $\hat{\tau}^2 = 2.434,\ \hat{\sigma}^2 = 4.635,\ \hat{\rho} = 115.76$

(c)

Let s_i be the point in spatial process. Note that in exponential covariance function

$$C(s_i, s_j) = \begin{cases} \sigma^2 exp\{-||s_i - s_j||/\rho\} & if\ s_i \neq s_j \\ \sigma^2 + \tau^2 & if\ otherwise \end{cases}$$

We get the estimates of σ^2, ρ, τ^2 in the proble 2-(b), and $||s_i - s_j||$ is calculated through rdist function. Note that when $s_i \neq s_j$, it menas the off diagonal value of covariance function. Using these clues we can construct covariance function as below.

```
dist_matrix = rdist(coordinates(df))
exp_cov = function(dist_matrix, tau2, s2, rho){
n = dim(dist_matrix)[1]
mat_no_nugget = matrix(rep(0,n*n), ncol=n)
for (i in 1:n){
        for (j in 1:n){
                h = dist_matrix[i,j]
                mat_no_nugget[i,j] = s2 * exp(-h/rho)
        }
}
mat_with_nugget = (s2 + tau2) * diag(n)
return(mat_no_nugget + mat_with_nugget)
}
estimated_cov_mat = exp_cov(dist_matrix, tau2.hat, s2.hat, rho.hat)
```

Also, we create X matrix and put all the pieces together to form $\widehat{\beta}_{GLS}$ using the following form.

$$\widehat{\beta}_{GLS} = \left(\mathbf{X}'\mathbf{\Sigma}(\hat{\theta})^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{\Sigma}(\hat{\theta})^{-1}\mathbf{Y}$$

Note that $\theta = (\rho, \tau^2)$ is parameters associated with spatial dependence. In real data analaysis, we do not know these parameters, so we estimate these parameters through parametric variograms models in 2-(b) and derive the estimated covariance matrix in the above code. To get $\widehat{\beta}_{GLS}$,

```
# gls estimate of beta
X = cbind(CAtemp$elevation, coordinates(CAtemp))
colnames(X) = c('elevation','lon','lat')
Sigma = estimated_cov_mat
Y = CAtemp$avgtemp
beta_hat = solve( t(X) %*% solve(Sigma) %*% X ) %*% t(X) %*% solve(Sigma) %*% Y
```

```
beta_hat
```

```
##                   [,1]
## elevation -0.006523966
## lon       -0.839901954
## lat       -1.134947221
```

(d)

In CAgrid, there are 500 locations and its elevation, so we use the observed $\mathbf{Y}$ to form prediction of $\mathbf{Y}(s_0)$. Note that the joint distribtion of $\mathbf{Y}$ and $\mathbf{Y}(s_0)$ is

$$\begin{pmatrix} \mathbf{Y} \\ \mathbf{Y}(s_0) \end{pmatrix} \sim D\left(\begin{pmatrix} \mathbf{m} \\ \mathbf{m_0} \end{pmatrix}, \begin{pmatrix} \mathbf{\Sigma} & \gamma \\ \gamma' & \Sigma_{new} \end{pmatrix}\right)$$

We estimate the $Cov(\mathbf{Y},\mathbf{Y}(s_0))$ through the exponential covariance function.

$$\widehat{\mathbf{Y}}(s_0) = \mathbf{X}(s_0)\widehat{\beta}_{GLS} + \gamma' \mathbf{\Sigma}^{-1}(\mathbf{Y} - \mathbf{X}\widehat{\beta}_{GLS}) \tag{1}$$

Note that we can estimate γ=$Cov(\mathbf{Y}, \mathbf{Y}(s_0))$ through exponential covariance function, calculating distance between original points and new points and using the estimates, $\hat{\rho}, \hat{\sigma}^2$. Note that nugget $\hat{\tau}^2$ is not needed to calculate covariance.

```
dist_matrix = rdist(coordinates(CAtemp), coordinates(CAgrid))
gamma = exp(- dist_matrix / rho.hat) * s2.hat
```

Now that se derive estimate of γ, we can calculate EBLU of $\mathbf{Y}(s_0)$, which is $\widehat{\mathbf{Y}}(s_0)$ using formula (1).

```
Xpred = cbind(coordinates(CAgrid), CAgrid$elevation)
colnames(Xpred) = c('lon','lat', 'elevation')
ypred = Xpred %*% beta_hat + t(gamma) %*% solve(Sigma) %*% (Y - X %*% beta_hat)
head(ypred)
```

```
##             [,1]
## [1,]  -164.95164
## [2,]  -754.21667
## [3,] -1160.11229
## [4,] -1298.75179
## [5,]  -159.61951
## [6,]   -10.57724
```

Let's derive the estimate of standard error of Z, which is $\widehat{\mathbf{Y}}(s_0) - \mathbf{X}(s_0)'\widehat{\beta}_{GLS}$. Let $\mathbf{\Sigma} = \mathbf{\Sigma}(\hat{\theta})$ for simple notation.

$$\begin{aligned} Var(\widehat{\mathbf{Y}}(s_0) - \mathbf{X}(s_0)\widehat{\beta}_{GLS}) &= Var(\gamma'\mathbf{\Sigma}^{-1}(\mathbf{Y} - \mathbf{X}\widehat{\beta}_{GLS})) \\ &= Var(\gamma'\mathbf{\Sigma}^{-1}\mathbf{Y} - \gamma'\mathbf{\Sigma}^{-1}\mathbf{X}\widehat{\beta}_{GLS}) \\ &= \gamma'\mathbf{\Sigma}^{-1}Var(\mathbf{Y})\mathbf{\Sigma}^{-1}\gamma + \gamma'\mathbf{\Sigma}^{-1}\mathbf{X}Var(\widehat{\beta}_{GLS})\mathbf{X}'\mathbf{\Sigma}^{-1}\gamma - 2Cov(\gamma'\mathbf{\Sigma}^{-1}\mathbf{Y}, \gamma'\mathbf{\Sigma}^{-1}\mathbf{X}\widehat{\beta}_{GLS}) \\ &= first\ term + secod\ term + third\ term \end{aligned}$$

- first term: Note that if we estimate $Var(\mathbf{Y})$ as $\mathbf{\Sigma}(\hat{\theta})^{-1}$

$$\gamma'\mathbf{\Sigma}(\hat{\theta})^{-1}Var(\mathbf{Y})\mathbf{\Sigma}(\hat{\theta})^{-1}\gamma = \gamma'\mathbf{\Sigma}(\hat{\theta})^{-1}\gamma$$

  We already derive $\mathbf{\Sigma}(\hat{\theta})$ in problem 2-(c).

- second term

$$\begin{aligned} \gamma'\mathbf{\Sigma}^{-1}\mathbf{X}Var(\widehat{\beta}_{GLS})\mathbf{X}'\mathbf{\Sigma}^{-1}\gamma &= \gamma'\mathbf{\Sigma}^{-1}\mathbf{X}Var(\left(\mathbf{X}'\mathbf{\Sigma}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{\Sigma}^{-1}Y)\mathbf{X}'\mathbf{\Sigma}^{-1}\gamma \\ &= \gamma'\mathbf{\Sigma}^{-1}\mathbf{X}\left(\left(\mathbf{X}'\mathbf{\Sigma}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{\Sigma}^{-1}\right)Var(\mathbf{Y})\left(\left(\mathbf{X}'\mathbf{\Sigma}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{\Sigma}^{-1}\right)'\mathbf{X}'\mathbf{\Sigma}^{-1}\gamma \end{aligned}$$

- third term

$$Cov(\gamma'\mathbf{\Sigma}^{-1}\mathbf{Y}, \gamma'\mathbf{\Sigma}^{-1}\mathbf{X}\widehat{\beta}_{GLS}) = \gamma'\mathbf{\Sigma}^{-1}Cov(\mathbf{Y}, \widehat{\beta}_{GLS})\left(\gamma'\mathbf{\Sigma}^{-1}\mathbf{X}\right)'$$

  Note that

$$\begin{aligned} Cov(\mathbf{Y}, \widehat{\beta}_{GLS}) &= Cov(\mathbf{Y}, \left(\mathbf{X}'\mathbf{\Sigma}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{\Sigma}^{-1}\mathbf{Y}) \\ &= Var(\mathbf{Y})\left(\left(\mathbf{X}'\mathbf{\Sigma}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{\Sigma}^{-1}\right)' \end{aligned}$$

The calculated standard errors are described below

```
# calculate standard error of Z
first_term = t(gamma) %*% solve(Sigma) %*% gamma
second_term = t(gamma) %*% solve(Sigma) %*% X %*%
                                (solve(t(X) %*% solve(Sigma) %*% X) %*% t(X) %*% solve(Sigma)) %*%
                                Sigma %*%
                                t((solve(t(X) %*% solve(Sigma) %*% X)
                                %*% t(X) %*% solve(Sigma))) %*%
                                t(X) %*% solve(Sigma) %*% gamma
```

```
cov_y_beta =  Sigma %*%
                                t( solve(t(X) %*% solve(Sigma) %*% X) %*% t(X) %*% solve(Sigma) )
third_term = t(gamma) %*% solve(Sigma) %*% cov_y_beta %*%
                          t( t(gamma) %*% solve(Sigma) %*% X)
z_cov = first_term + second_term + third_term
z_std = diag(z_cov)
plot.point.ref(CAgrid, z_std)
```