MAST90083: Computational Statistics and Data Mining

ASSIGNMENT 1

ERANGA CHAMAL GOMES

777833

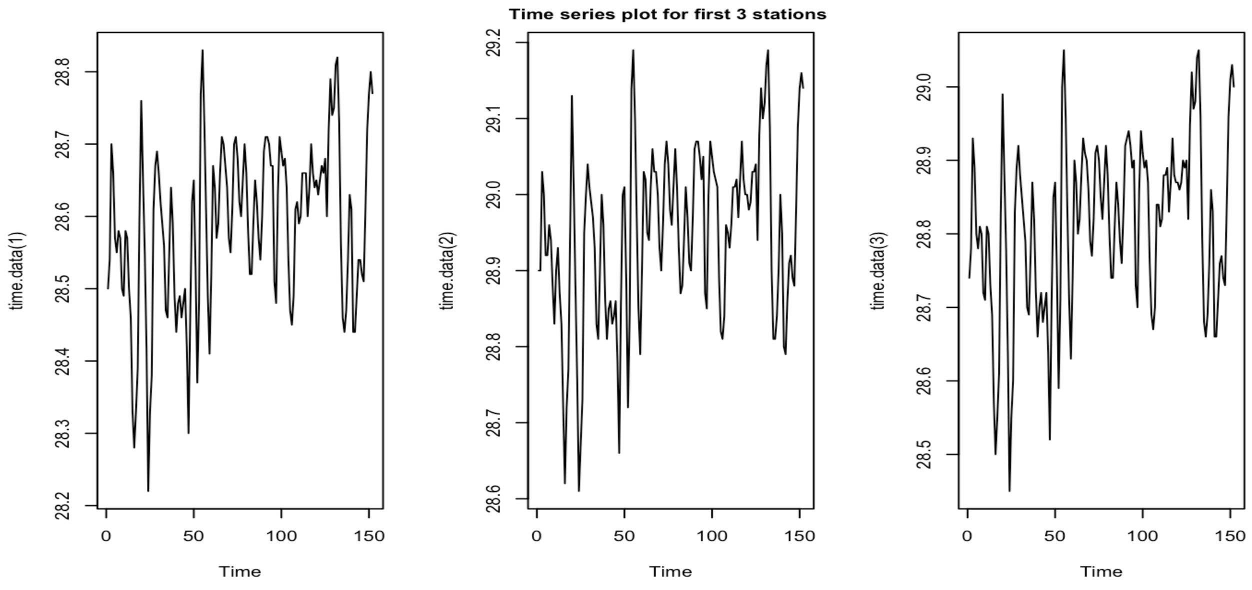
Definitions

In the following report we have only included few sections of graphs when it came to plot as plotting 17 stations took too much space. Therefore, I have made plots for the "first three stations" at most cases which denote station ACME, CHAN and CHIC.

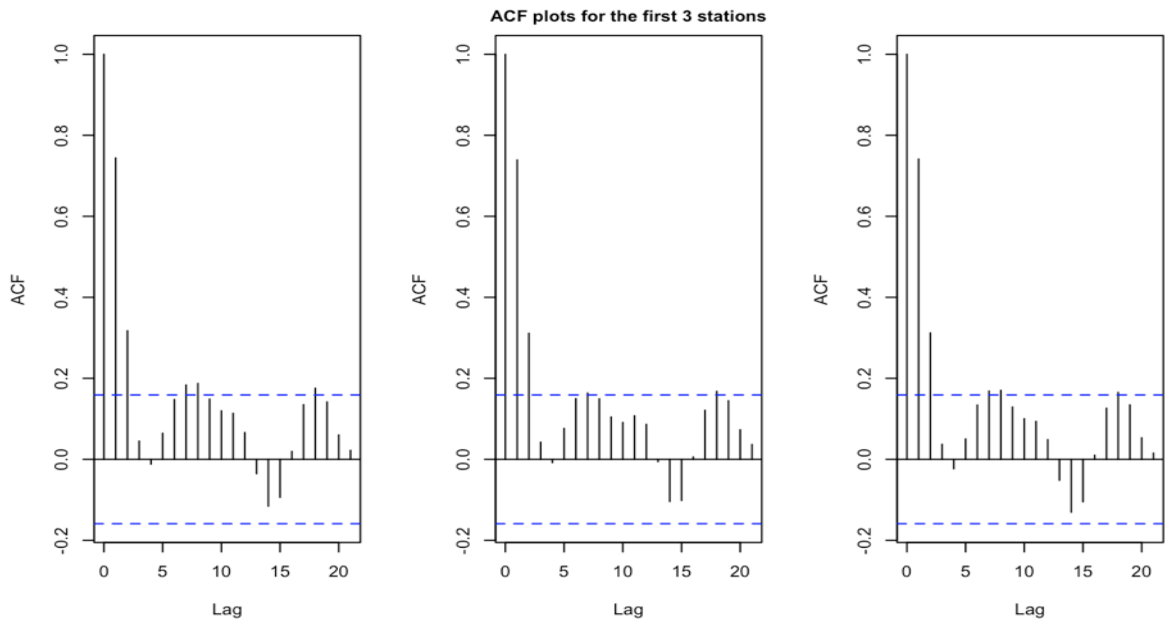
Also, I have only included abstract of my code which deemed to be of high importance in particular sections of the report and if you require any clarification we are able to provide the whole R markdown file for you.

1.Temporal Dependence

We first begin by analysing the time series data for temporal dependence. As the initial step we first observe the time series data for any trend or seasonal effects. From observing the time series plots shown below it can be seen that only random variation was in place. This implied that we did not have to make any differencing the data for seasonal or trend effect.



Next, we proceeded to analyse the Autocorrelation plots (ACF) to seek any serial dependence among reading days. As shown below it showed the first and the second lag to have the most impact. In other words, the previous day's and the day before pressure readings showed significant dependence for today's reading.

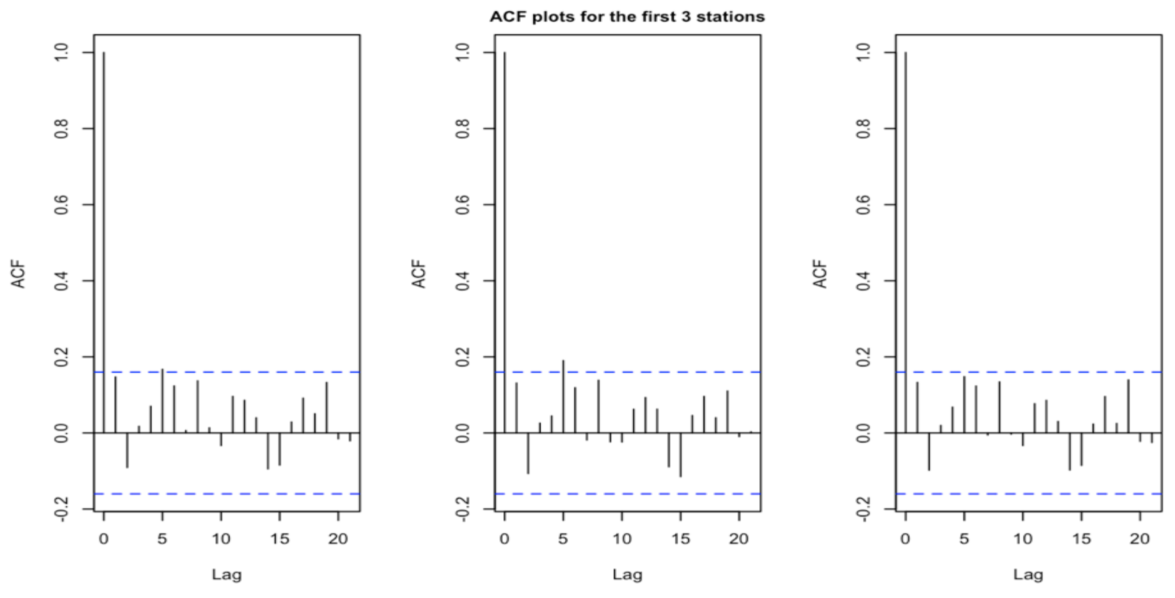


Then we proceeded to create a linear model with two lag variables as shown below. Where Y_j, X_j, X_{j-1} are 150x17 column vectors, where j denotes the jth station.

$$Y_j = \gamma_0 + \beta_0 X_j + \beta_1 X_{j-1}$$

Then we obtained the resulting residuals from the above model and plotted the ACF graph for them. Below you could see the respective results for station 1 to 3. We see the clear distinction that the residuals are now independent, despite having a few lags exceed the 95% confidence interval all else seems to lie below the significance level.

Despite the damping effect of the overall lag behaviour we chose only the first 2 lag variables to be of significance for our purpose. We came to this conclusion after realising that adding further lags to our model did not increase residual independence any further and so in consistent with parsimony 2 lag variables stood to be most ideal.



2.1 Spatial Dependence

Our next objective was to seek and include spatial covariates in our model. From variables which were available for us, we proceeded with station specific spatial variables Elevation, Longitude and Latitude. We first used all three of the variables in our model to see the significance level of each variable, then proceeded to consider alteration of the model.

Having decided on the number of lag variables as 2 from previous section, we fit the following linear models to the whole data set, where spatial covariates Latitude, Longitude and Elevation for station j are denoted by $ELON_{j,}$ $NLAT_j$ and $ELEV_j$ respectively.

$$Y_j = \gamma_0 + \beta_0 X_j + \beta_1 X_{j-1} + \gamma_1 NLAT_j + \gamma_2 ELON_j + \gamma_3 ELEV_j.$$

```
lm.fit = lm(Y~lag1+lag2+Elev+Nlat+Elon) #2 lags with elev, nlat, and elon
summary(lm.fit)
## Call:
## lm(formula = Y ~ lag1 + lag2 + Elev + Nlat + Elon)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.202275 -0.038045 -0.001281  0.043111  0.233579
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.219e+01  7.659e-01  15.920   <2e-16 ***
## lag1         1.144e+00  1.691e-02  67.673   <2e-16 ***
## lag2        -5.279e-01  1.710e-02 -30.867   <2e-16 ***
## Elev        -1.230e-03  6.263e-05 -19.640   <2e-16 ***
## Nlat        -2.337e-03  3.657e-03  -0.639    0.523
## Elon         6.701e-03  6.081e-03   1.102    0.271
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06633 on 2544 degrees of freedom
## Multiple R-squared:  0.9258, Adjusted R-squared:  0.9257
## F-statistic:  6350 on 5 and 2544 DF, p-value: < 2.2e-16
```

However, from analysis of the summary above we found the that the Latitude and Longitude Variables were of least importance as it had a large P value. Therefore, we proceeded with the following linear model in consistent with parsimony including Elevation as the only spatial covariate.

$$Y_j = \gamma_0 + \beta_0 X_j + \beta_1 X_{j-1} + \gamma_1 ELEV_j$$

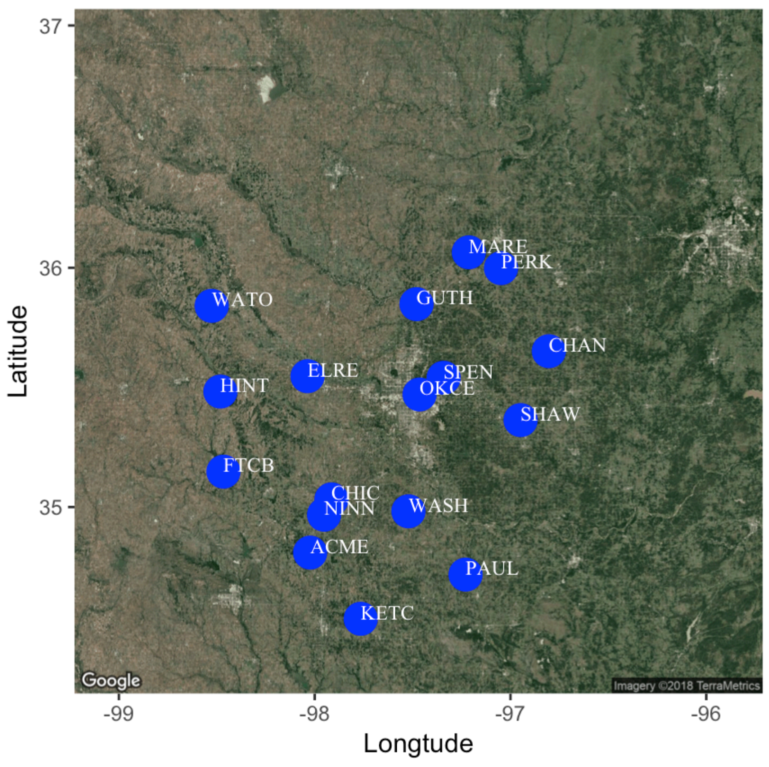
The respective summary for this linear model was as follows

```
Call:
## lm(formula = Y ~ lag1 + lag2 + Elev)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.202569 -0.038137 -0.001095  0.043156  0.234051
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.146e+01  3.633e-01   31.55   <2e-16 ***
## lag1         1.144e+00  1.690e-02   67.70   <2e-16 ***
## lag2        -5.276e-01  1.710e-02  -30.86   <2e-16 ***
## Elev        -1.278e-03  4.534e-05  -28.18   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06632 on 2546 degrees of freedom
## Multiple R-squared:  0.9258, Adjusted R-squared:  0.9257
## F-statistic: 1.059e+04 on 3 and 2546 DF, p-value: < 2.2e-16
```

This shows that with two les predictors were able to achieve the same level of R squared. Therefore, based on R squares specifically we choose the $Y_j = \gamma_0 + \beta_0 X_j + \beta_1 X_{j-1} + \gamma_1 ELEV_j$ to be the most suitable model for our purpose. Keep in mind as reduce the number of predictors the variability of our predicted values decreases as well, providing us with a much narrower confidence interval for our predictions.

2.2 Additional Notes on Spatial Analysis

Objective of this section was to present the possibility of having spatial dependence from the neighbouring station pressure reading to the station being considered. Inclusion of such “spatial lags” in our linear model would make it overly complicated with our objective of fitting a general linear model for all the stations. Thus, this research was conducted purely for additional knowledge and to show the possibility of further improving our model predictability.



To observe the impact from the nearest neighbour we calculate the Spatial auto correlation statistic "Moran's I" and perform a simple hypothesis test. The test statistic is as follows.

$$I = \frac{n}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\,(y_i - \bar{y})(y_j - \bar{y})}{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}}$$

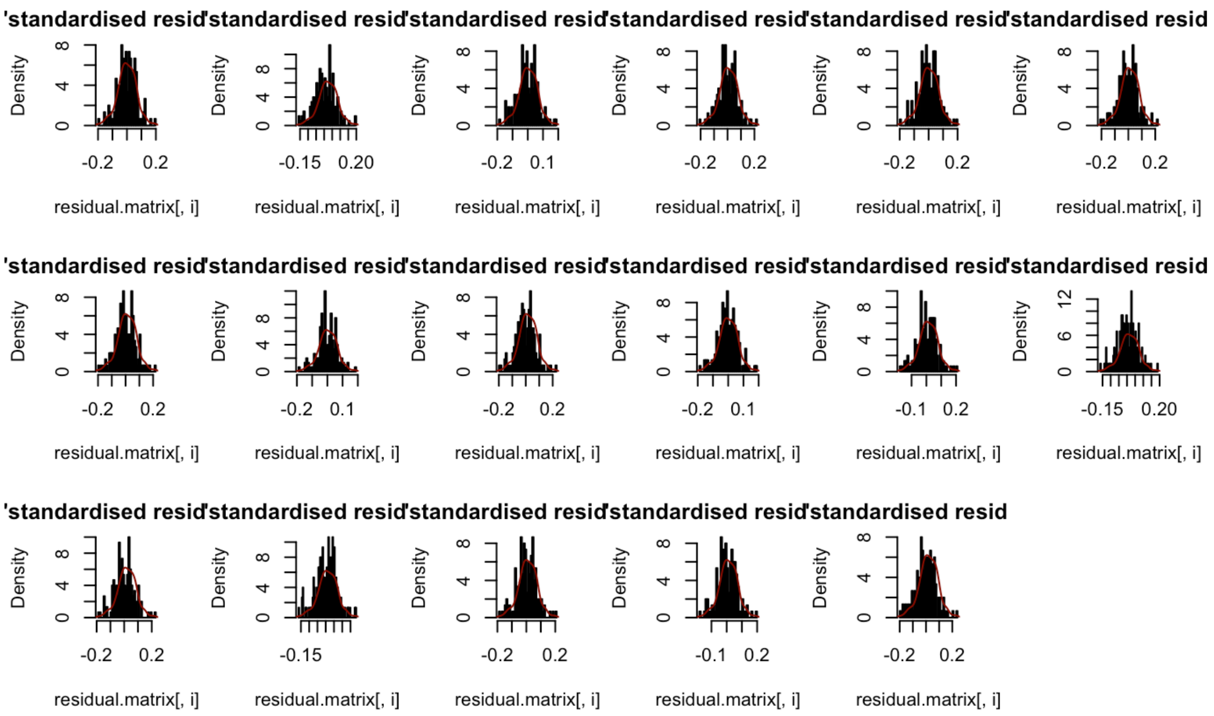
Intuitively the underlying concept behind this statistic can be understood very easily just by observing the formula. We compare for each station mean reading and compare against the neighboring stations mean readings to analyze the covariance. w_{ij} represent the inverse Wight matrix with the diagonal equal to zero, so we give more weight towards the nearest neighbour than itself. Here n represents 17 which is the number of stations.

The null hypothesis of no phylogenetic correlation is tested assuming normality of I under this null hypothesis. If the observed value of I is significantly greater than the expected value, then the values of x are positively autocorrelated. We obtain our observed value to be **0.1381664** and expected value to be -0.0625. which indicates that there is positive correlation. Moreover, p value of **0.0007511611** indicates strong evidence against the Null hypothesis.

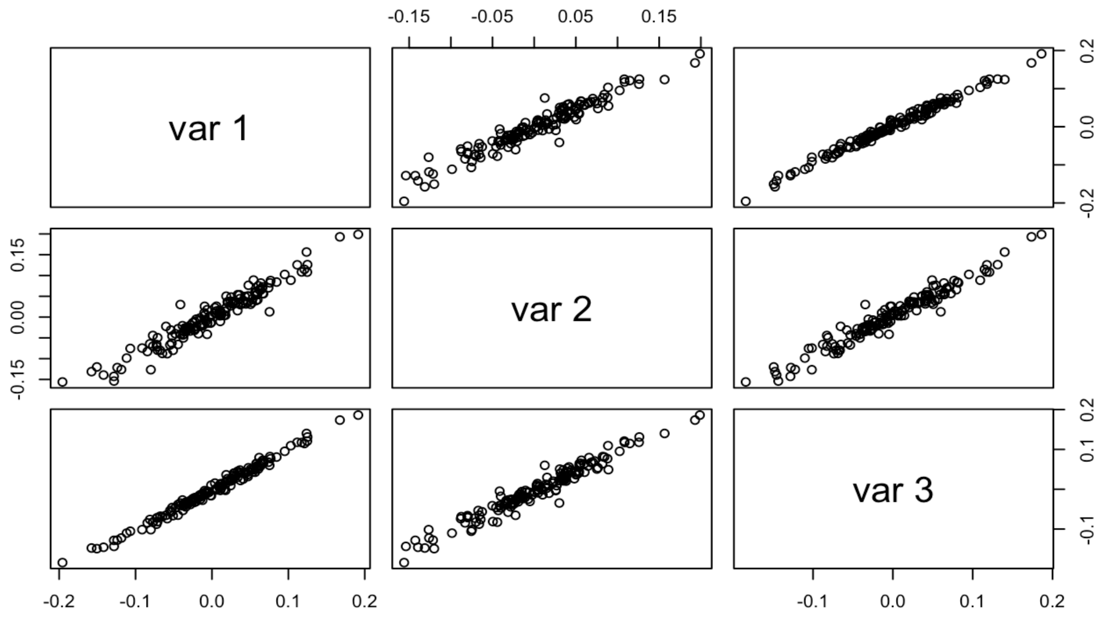
This indicates there exist a further possibility of adding spatial lags for our model to further increase our model accuracy.

3. Analysis of Multivariate Normality of the residuals

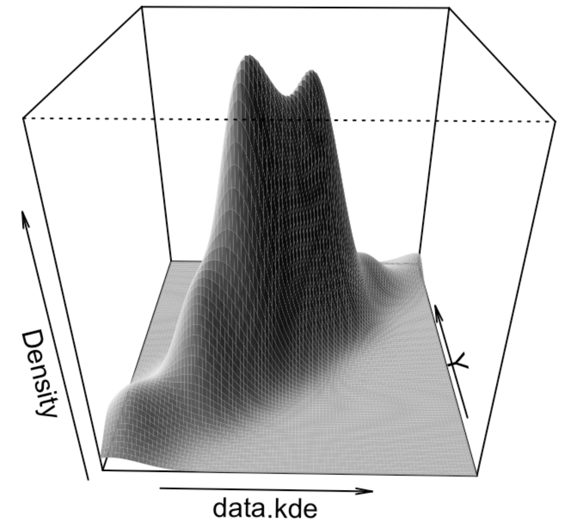
In advance to testing multivariate normality we first standardize the residuals for each station. We do this in order have unit variance for the residuals. Once we obtained the standardized residuals we plot residuals against the normal density for each station as follows.



From above you could see residuals of each station seem to hold the normality assumption. Now we proceed to check if Multivariate normality holds, to do so we begin by plotting some scatter plots for pairwise normality. Shown below are the scatter plots for the first 3 stations.



To make even clear we even managed to do a 3D plotting of the bivariate normal density. Shown below is the bivariate normal density for the residuals of station and station 2 respectively. The R code consists of the required code to plot bivariate normal density between any stations.



Although above plot doesn't show an exact bivariate normality we were able to conduct a few tests to check Multivariate Normality as graphical representation is not possible. We conduct a several multi variate normality tests **(MVT)**. The MVT's we conducted were the Mardia's test, Henze-Zirkler's test, Royston's test and Doornik-Hansen's test.

Results for Mardia's MVT test

```
##              Test         Statistic           p value Result
## 1 Mardia Skewness  702.908265189146 0.999999999989012    YES
## 2 Mardia Kurtosis 0.874382231921406 0.381910130335311    YES
## 3             MVN              <NA>              <NA>    YES
```

This function performs multivariate skewness and kurtosis tests at the same time and combines test results for multivariate normality. If both tests indicate multivariate normality, then data follows a multivariate normality distribution at the 0.05 significance level. Note that this test also ensures univariate Normality for each station as well.

Results for Henze-Zirkler's MVT test

```
##            Test        HZ   p value MVN
## 1 Henze-Zirkler 0.9973432 0.6083299 YES
```

The last column indicates whether dataset follows a multivariate normality or not (i.e, YES or NO) at significance level 0.05.

Results for Royston's MVT test

```
##      Test         H   p value MVN
## 1 Royston 1.107985 0.3775065 YES
```

The last column indicates whether dataset follows a multivariate normality or not (i.e, YES or NO) at significance level 0.05.

Results for Doornik-Hansen's MVT test

```
##             Test        E df   p value MVN
## 1 Doornik-Hansen 24.83652 34 0.8744209 YES
```

The last column indicates whether dataset follows a multivariate normality or not (i.e, YES or NO) at significance level 0.05.

From all of tests we conducted above each test strongly suggests Multi variate Normality holds for the residuals at 95% confidence. Thus, we conclude with strong evidence that Multi Variate normality holds for the residuals of the stations.

4.Parameter Estimation for Covariance Matrix

Now that we ensured Multi variate Normality we proceed on estimating the Alpha and r parameters under the proposed model for the Covariance matrix of the residuals. As you could see from the code below once we derived the log likelihood function We used the Simulated-annealing method for optimizing, which belongs to the class of stochastic global optimization methods. This managed to optimize the parameters with in the given bounds for us. In implementation we decided the initial estimates which would provide us with the minimum log likelihood, as our objective is to find the minimum of the negative log likelihood.

```
#calculating the normalised distance matrix

lonlat_centers=as.matrix(cbind(geo_info$East_Longitude,geo_info$North_Latit
ude))

label=as.array(geo_info$Station_Identifier)

distance.matrix=distm(cbind(lonlat_centers[,1],lonlat_centers[,2]),fun = di
stVincentyEllipsoid)/100000


#creating the log likelihood function and covariance matrix

wrap1=function(v,x){

  a=v[1]
  r=v[2]
  sigma1=exp(-r*distance.matrix^a)
  loglik =abs(sum(dmvnorm(x=x, mean=rep(0,17), sigma =sigma1, log=TRUE)))
#  sigma1=apply(distance.matrix,c(1,2), function(x) exp(-r*(x^a)))
}
#optimisation
ML1=optim(c(0.05,0.5),fn=wrap1,x=residual.matrix)
```

Afterwards we obtained the following values for r > 0 and 0 < α ≤ 2 as follows

```
## [1] "0.371216466013169 Alpha"
## [1] "0.157451187996592 r Value"
```

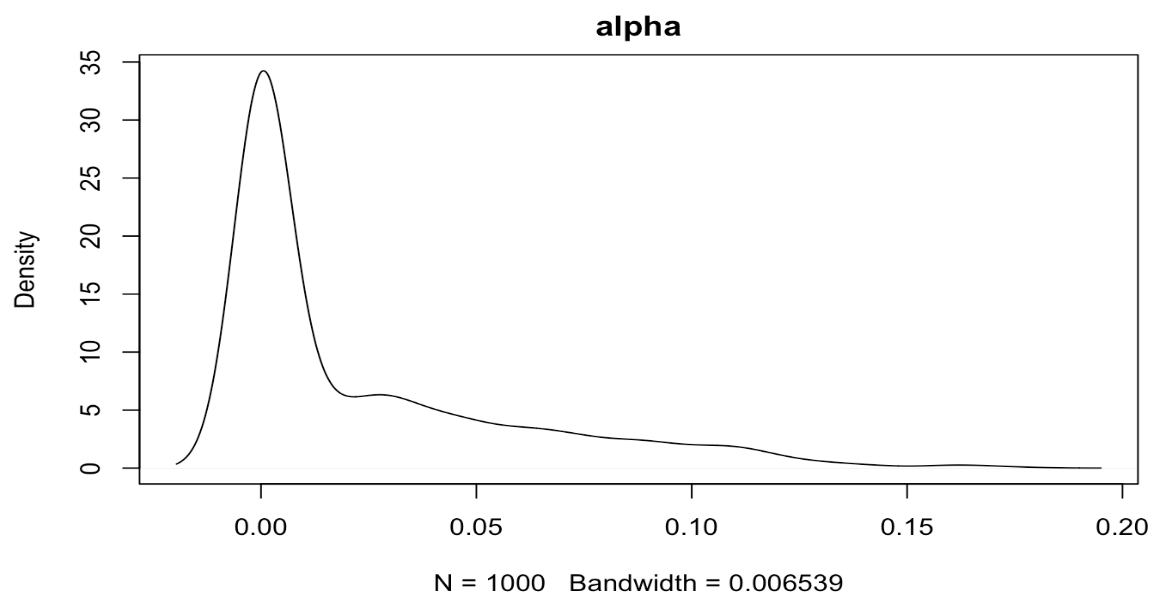
4.1 Constructing Bootstrap Confidence Intervals

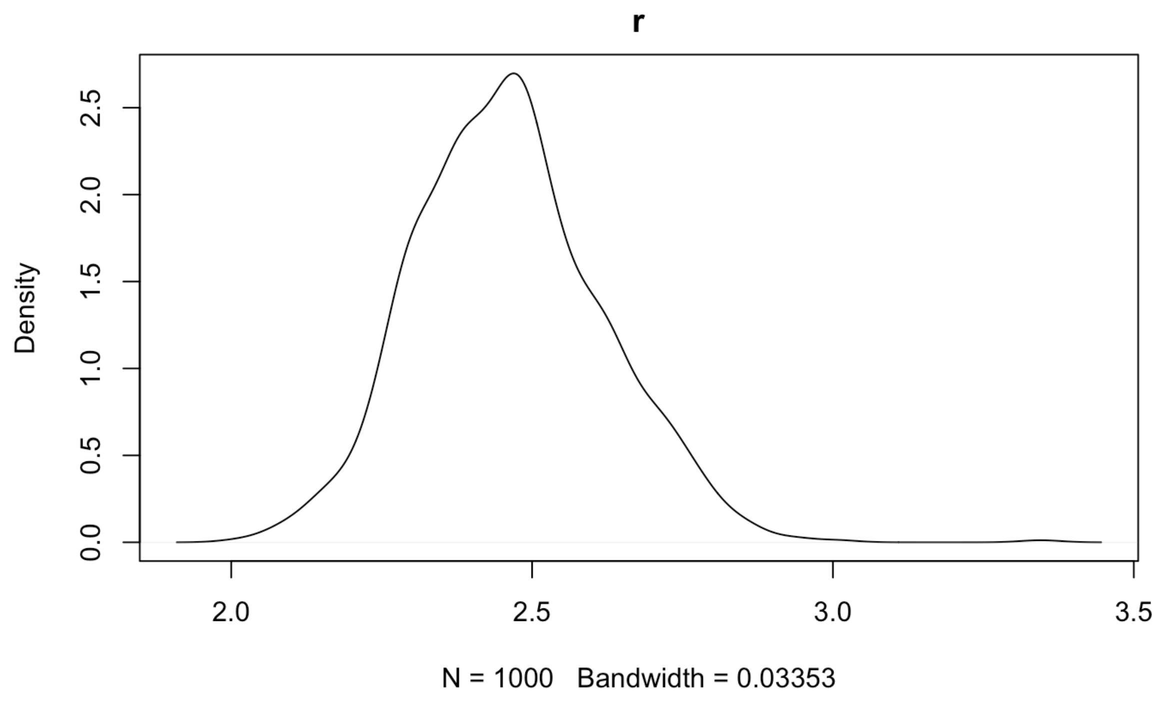
In this particular objective we come across a special issue, that is we have to preserve randomness throughout our bootstrap process. Keeping in mind that were dealing with time series data for 17 different station.

Reminder that in earlier section we proved that the residuals from our model proved to be independent at 95% Confidence. So, we are able to bootstrap the residuals while blocking them into Stations as well as blocking residuals of desired numbered days with in each station residuals to preserve randomness property. This may avoid any volatilities that may arise from stations or certain time periods. The code below shows how we managed to block bootstrap desired number of residuals from each station. From this method we observed as we reduce the size of the block the standard error of the bootstrap estimates as well as bias seems to decrease.

```
#BLOCK BOOTSTRAP RESIDUALS
z=as.data.frame(cbind(response.matrix,predictor.matrix[,-1]))
usePackage("mvtnorm")
usePackage("boot")

#b is the size of the block which has to be a number divisible of 150
boot1=function(x,b){
     temp=lm(x$y~x$lag1+x$lag2+x$gamma1)
    #block sampling
    l=2550/b
    test1=c()
    for(j in 1:l){
      lol1=1+((j-1)*b)
      lol2=b*j
      test1=c(test1,temp$residuals[sample((lol1:lol2),b,replace = TRUE)])
    }
     residuals=test1


     y.star=temp$fitted.values+residuals
     temp1=lm(y.star~x[,2]+x[,3]+x[,4])
    #creating the residual matrix
     residual.matrix=matrix(temp1$residuals,nrow=150,ncol=17)
    #optimising
    wrap1=function(v){
      a=v[1]
      r=v[2]
      loglik =abs(sum(dmvnorm(x=residual.matrix, mean=rep(0,17), sigma =exp
(-r*distance.matrix^a), log=TRUE)))
    }
    ML1=optim(c(0.05,0.5),fn=wrap1)
    return(ML1$par)
  }
#creating a matrix and storing all the estimates we obtained for the 2 para
meters
test1=matrix(NA,100,2)
for(i in 1:100){
  test1[i,]=boot1(z,25)
}
```

However, when we constructed the densities for the bootstrap estimates there was still high volatility among the bootstrap estimates. It maybe, that though bock bootstrap managed to reduce the volatility it still did not managed to capture the whole effect due to the nature of air pressure readings. Shown below are the densities and you could clearly see that this method is failing.



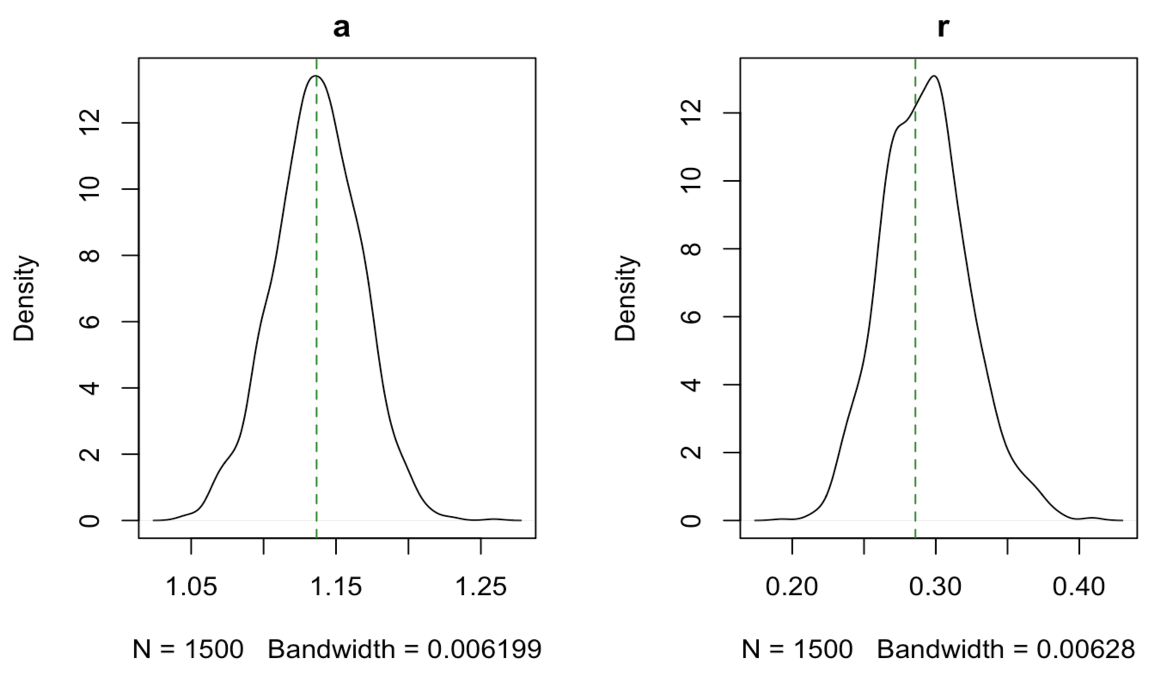


Therefore, we followed the parametric bootstrap residuals approach while still preserving the station specific residuals randomness.

```
test1=matrix(NA,100,2)
for(i in 1:100){
  test1[i,]=boot1(z,25)
}
```

```
#PARAMETRIC BOOTSTRAP

z=as.data.frame(cbind(response.matrix,predictor.matrix[,-1]))

start=c(0.5,1)

lm1.bt = function(x){

  temp = lm(x[,1]~x[,2]+x[,3]+x[,4])

  temp1=temp$residuals

  temp2=temp1/sd(temp1)

  loglik = function(param){

  a = param[1] #a is between 0 and 2

  r = param[2] #r is positive

  sigma1 = exp(-r*distance.matrix^a)

  loglik = abs(sum(dmvnorm(matrix(temp2,ncol = 17), mean = rep(0,17), sigma
 = sigma1, log = TRUE)))

  }

  NM=optim(start,fn = loglik, method = "Nelder-Mead")

  return(NM$par)

}

lm1.gen = function(x,mle){

  a = mle[1]

  r = mle[2]

  sigma1 = exp(-r*distance.matrix^a)

  n = 150

  err = rmvnorm(n, mean = rep(0,17), sigma = sigma1)

  err = as.vector(err)

  y.star = mle[3]+mle[4]*x[,2]+mle[5]*x[,3]+mle[6]*x[,4]+err*sd(lm.fit1$res
iduals)


  return(cbind(y.star,x[,2],x[,3],x[,4])) #create new data matrix

}


mle.list = c(ML1$par,coef(lm.fit1))

set.seed(1234)

boot1 = boot(data=z,statistic=lm1.bt,R=200,sim="parametric",ran.gen=lm1.gen
,mle=mle.list)

#onebt = boot1$statistic(boot1$ran.gen(z,mle.list))

boot1

boot.ci(boot1,type = c("norm","basic",'perc'),index = 1)
boot.ci(boot1,type = c("norm","basic",'perc'),index = 2)
par(mfrow=c(1,2))
plot(density(boot1$t[,1]), main = "density of a")
abline(v=mle.list[1],col="darkred",lty=2)
plot(density(boot1$t[,2]), main = "density of r")
abline(v=mle.list[2],col="darkred",lty=2)
```

Under this method we managed to obtain much more stable and a much clear representative bootstrap estimates than under block bootstrap. As shown by the following densities you could see that. Moreover, the bootstrap CI were not able to capture the true coverage interval as well.



```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS for Alpha
## Based on 1500 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot1, type = c("norm", "basic", "perc"),
##     index = 1)
##
## Intervals :
## Level      Normal              Basic              Percentile
## 95%   ( 1.059,  1.176 )   ( 1.058,  1.179 )   ( 1.075,  1.196 )
## Calculations and Intervals on Original Scale
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS for r
## Based on 1500 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot1, type = c("norm", "basic", "perc"),
##     index = 1)
##
## Intervals :
## Level      Normal              Basic              Percentile
## 95%   ( 1.059,  1.176 )   ( 1.058,  1.179 )   ( 1.075,  1.196 )
## Calculations and Intervals on Original Scale
```

END OF THE REPORT