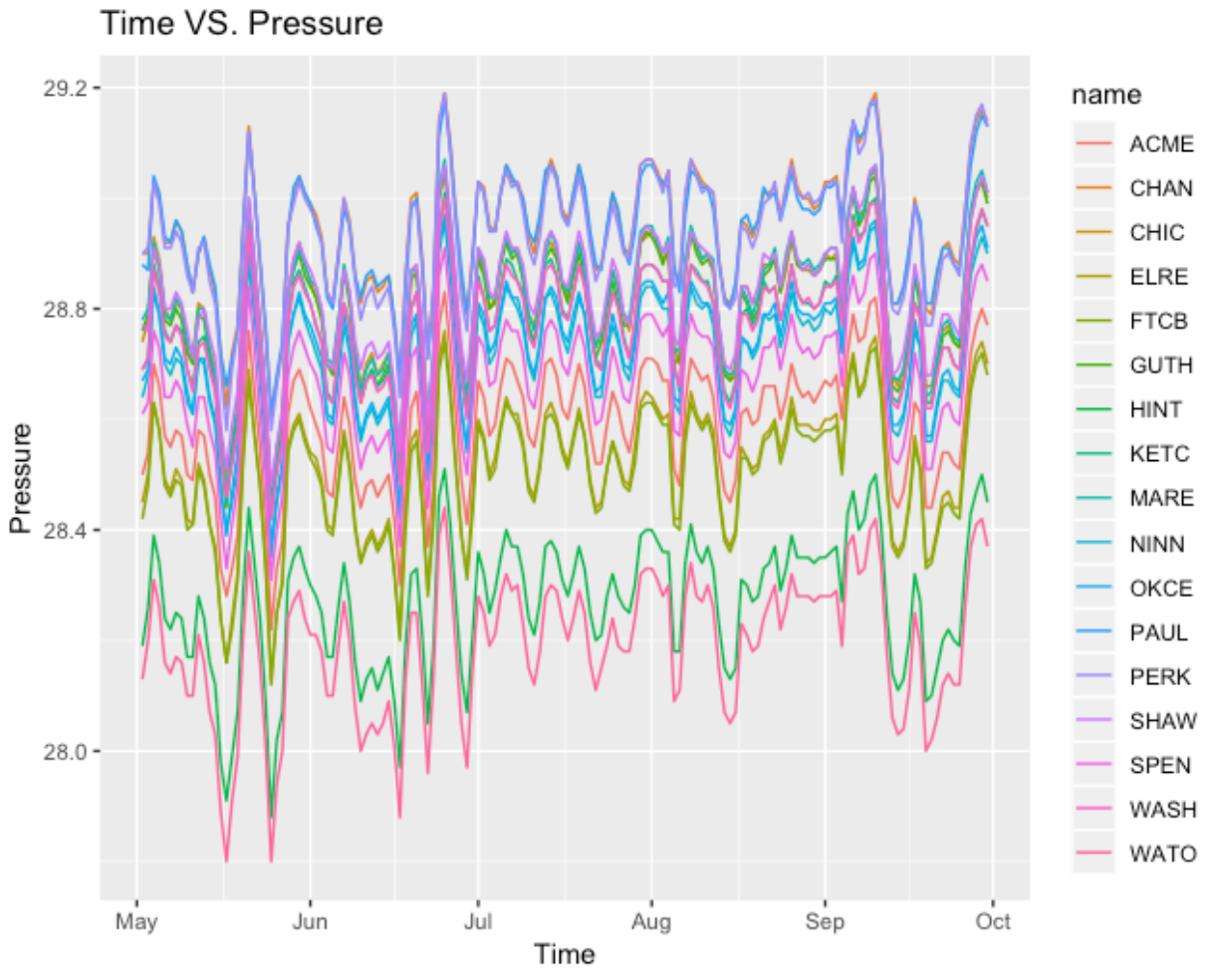
MAST 90083 Assignment

The aim of this report is to learn about models for weather data by predicting weather at some location using measurements from the nearby weather stations. The dataset used in building model in this report are *geoinfo_stations* (include 17 stations info) and *weather data* (152 entries from 05-2017 to 09-2017).

Step 1:
First, we need to deal with two given tables by making them into one table and adding with lag variables. The final table has 2584 rows and 67 columns. The rows contain 152 time recordings for each 17 stations. The variables in columns include time, name, value, city, nlat, Elon, elev, lags and so on. The R code and plots of Time VS. Pressure for this step are shown as below.

```
##Combine two tables
weather = melt(weather_data,id.vars = c('YEAR','MONTH','DAY'),variable.name = "name")
weather$time = as.Date(paste(weather$YEAR,weather$MONTH,weather$DAY,sep = '-'))
geo_weather = inner_join(weather,geoinfo_stations, by = c('name'= 'stid'))
geo_weather = geo_weather[,c(6,4,5,7:ncol(geo_weather))]

##Draw Time VS. Pressure plots
gg = ggplot(geo_weather, aes(time, value)) +
  geom_line(aes(col = name)) +
  labs(title = 'Time VS. Pressure', x = "Date" , y = "Pressure")

##add lag into the table
geo_weather = geo_weather %>% group_by(name) %>%
  mutate(lag1 = lag(value,n = 1),lag2 = lag(value,n = 2),lag3 = lag(value,n = 3),lag4 =
           lag(value,n = 4),lag5 = lag(value,n = 5),lag6 = lag(value,n = 6),lag7 = lag(value,n = 7))
geo_weather_lag = geo_weather[,c('time','name','value','lag1','lag2','lag3')]
```

Code 1: deal with tables and plots of Time VS. Pressure



Second, we fit a linear model and assume NLAT, ELON and ELEV as spatial covariates. The result shows Result 1 as below. The P-value of nlat and elon are greater than 0.05, implying they are not significant in this model. Therefore, we choose elev as our spatial covariate.

```
> summary(m1)

Call:
lm(formula = value ~ nlat + elon + elev + lag1, data = geo_weather)

Residuals:
      Min        1Q    Median        3Q       Max
-0.255034 -0.042000 -0.002603  0.049037  0.275794

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.755e+00  8.769e-01   8.844   <2e-16 ***
nlat        -1.439e-03  4.259e-03  -0.338    0.736
elon         4.006e-03  7.083e-03   0.566    0.572
elev        -7.853e-04  7.104e-05 -11.054   <2e-16 ***
lag1         7.552e-01  1.318e-02  57.314   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.07752 on 2562 degrees of freedom
  (17 observations deleted due to missingness)
Multiple R-squared:  0.8984,    Adjusted R-squared:  0.8983
F-statistic:  5666 on 4 and 2562 DF,  p-value: < 2.2e-16
```

Result 1: summary of linear model

Third, we apply **acf( )** function to check the autocorrelation for original data and residuals of fitted model. The 17 plots are shown below.

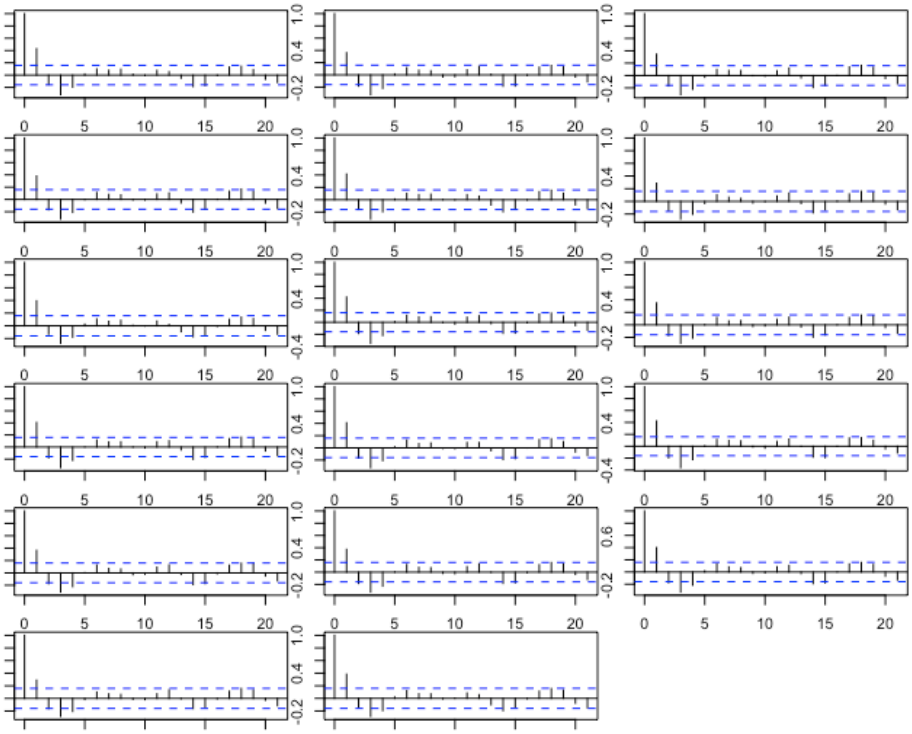


Figure 2: plots of original data

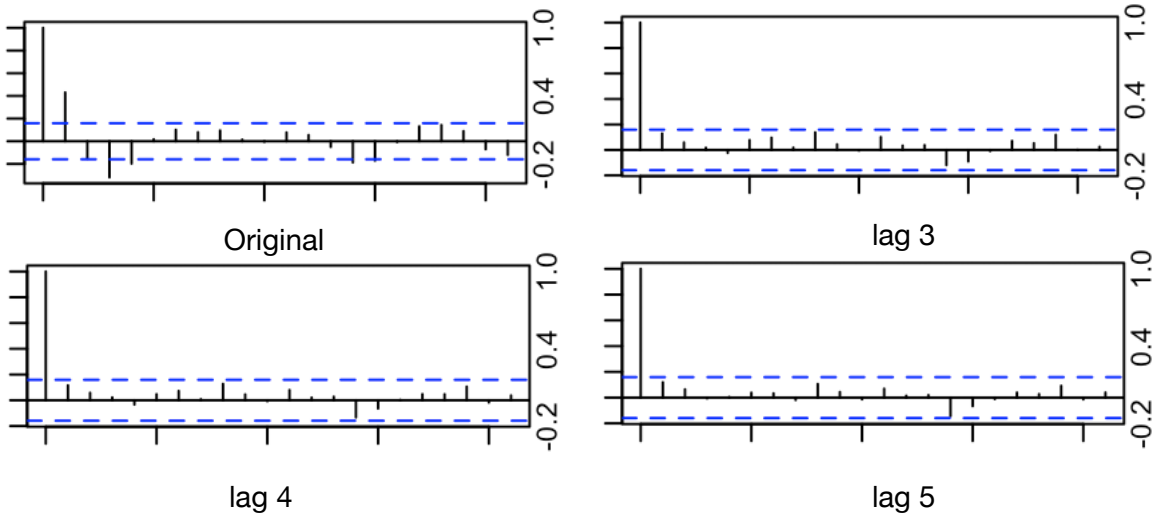
From Figure 2, we can see that some values exceed the dotted blue line which means that there is a strong autocorrelation for original pressure data. Hence we need to reduce the serial dependency by adding more lag variables. We need to select the best lag variables from 7 lag

variables. Hence, we fit the model by adding lag variables one by one and the code are shown in Code 2 below.

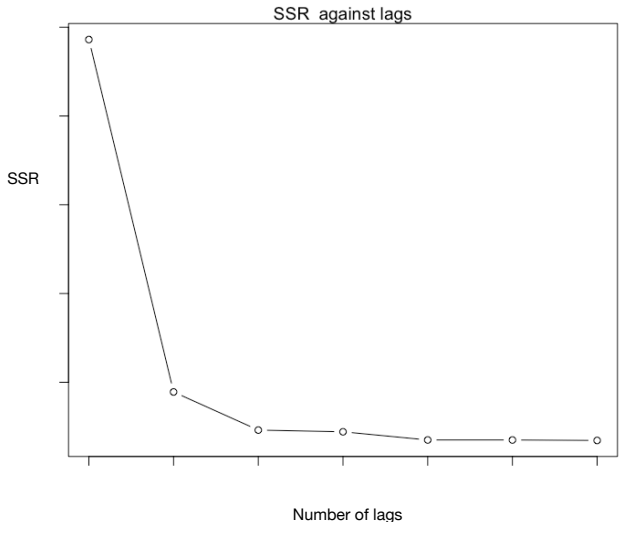
```
### remove the elon and nlat
m1 = lm(value ~ elev +lag1 ,data = geo_weather)
m2 = lm(value ~ elev +lag1+lag2 ,data = geo_weather)
m3 = lm(value ~ elev +lag1+lag2+lag3 ,data = geo_weather)
m4 = lm(value ~ elev +lag1+lag2+lag3+lag4 ,data = geo_weather)
m5 = lm(value ~ elev +lag1+lag2+lag3+lag4+lag5 ,data = geo_weather)
m6 = lm(value ~ elev +lag1+lag2+lag3+lag4+lag5+lag6 ,data = geo_weather)
m7 = lm(value ~ elev +lag1+lag2+lag3+lag4+lag5+lag6+lag7 ,data = geo_weather)
```

Code 2: selecting lag variables

Even though adding more lag variables is benefit for reducing the serial dependency, selecting the best and appropriate of lag variables is necessary. After fitting the models one by one and comparing the results, adding 3 lag variables are the best choice since there is an apparent changes, value drops quickly, from lag 1 model to lag 3 model.



Fourth, we take a look at the residual in lag models and calculate the sum of square of residuals(SSR). The plot of SSR against number of lags are shown below.



Step 2:

First, we standardized the vector of residuals. The code are shown below. For each station, we calculate its mean, residual and standard deviation, then we subtract the mean from the residual and divided by the standard deviation to get the standardized residuals.

```
std_res.list = list()

##Standardize residuals of the fitted model for each station
for(i in 1:7){
  r = matrix(eval(parse(text = paste('m_all_',i,'$residuals',sep = ''))),ncol = 17)
  m = apply(r,2,mean)
  se = apply(r,2,sd)
  std_res.list[length(std_res.list)+1] <- list((r - m)/se)
}
```

Second, histogram and bivariate scatter plots are drawn as Figure 3.

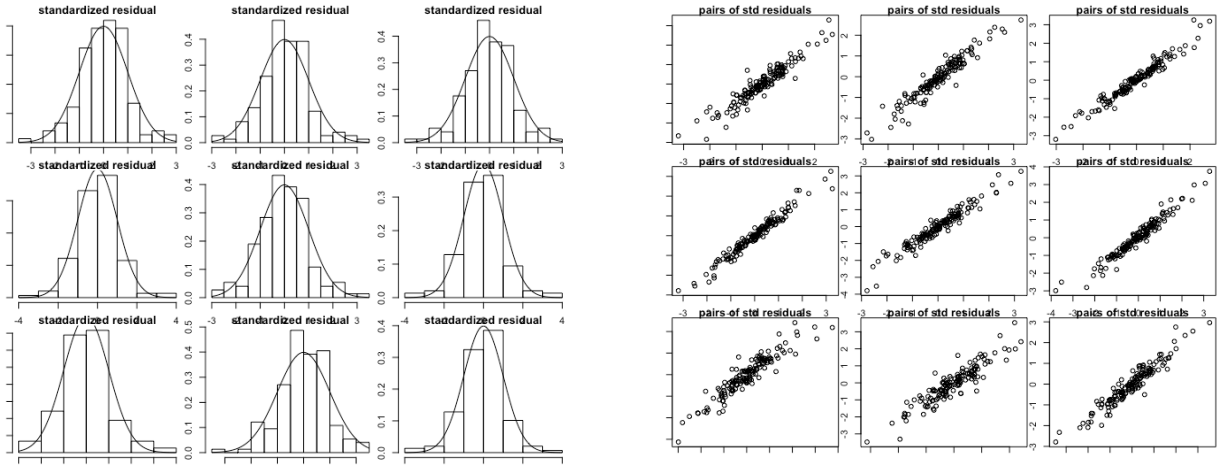


Figure 3

Since the pairs of residuals in the scatter plot show a highly correlation relationship among different stations, therefore, joint normal distribution is suitable for residuals.

Step 3:

We calculate the distance between station and get a matrix with diagonal equals zero. The code and output are shown below.

```
dist = matrix(data = rep(0,17*17), nrow = 17, ncol = 17)     ## generate a 17*17 distance matrix
for(i in 1:17){
  for(j in 1:17){
    dist[i,j] = distVincentyEllipsoid(c(geoinfo_stations$elon[i],geoinfo_stations$nlat[i]),c(geoinfo_stations$elon[j],geoinfo_stations$nlat[j])) / 100000
  }
}    ##geoinfo_stations$elon: longitude; geoinfo_stations$nlat: latitude
> dist
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]      [,9]     [,10]     [,11]     [,12]     [,13]     [,14]     [,15]     [,16]     [,17]
 [1,] 0.0000000 1.4524049 0.26768007 0.8212401 0.5533760 1.2557836 0.8584529 0.3901360 1.5759185 0.18845317 0.8956387 0.7341867 1.5900675 1.1585921 1.0239060 0.4978953 1.2344610
 [2,] 1.4524049 0.0000000 1.22177931 1.1227923 1.6099511 0.6487261 1.5320560 1.5239194 0.5871328 1.29133223 0.6310541 1.1096631 0.4426596 0.3451571 0.5022413 0.9892539 1.5716642
 [3,] 0.2676801 1.2217793 0.00000000 0.5832826 0.5193408 0.9881839 0.7195678 0.5750981 1.3099092 0.07946749 0.6372938 0.7183607 1.3291662 0.9541418 0.7690892 0.3633479 1.0559770
 [4,] 0.8212401 1.1227923 0.58328257 0.0000000 0.5907665 0.6041728 0.4098515 1.1580152 0.9391693 0.64887825 0.5261085 1.1812592 1.0236827 1.0087225 0.6303264 0.7839851 0.5498428
 [5,] 0.5533760 1.6099511 0.51934082 0.5907665 0.0000000 1.1849080 0.3725120 0.9404572 1.5235967 0.51017471 0.9792628 1.2280224 1.5938784 1.4019012 1.1114847 0.8815630 0.7707891
 [6,] 1.2557836 0.6487261 0.98818388 0.6041728 1.1849080 0.0000000 0.9930654 1.4873386 0.3393918 1.06759923 0.4181433 1.2779583 0.4233669 0.7213447 0.3627274 0.9622865 0.9453119
 [7,] 0.8584529 1.5320560 0.71956784 0.4098515 0.3725120 0.9930654 0.0000000 1.2456550 1.3153698 0.74893299 0.9233955 1.4252205 1.4164207 1.3987227 1.0361637 1.0366643 0.3986661
 [8,] 0.3901360 1.5239194 0.57509812 1.1580152 0.9404572 1.4873386 1.2456550 0.0000000 1.7760150 0.51615217 1.0819998 0.5330259 1.7561541 1.1902533 1.1885993 0.5502418 1.6132774
 [9,] 1.5759185 0.5871328 1.30990921 0.9391693 1.5235967 0.3393918 1.3153698 1.7760150 0.0000000 1.38925339 0.6951707 1.4965830 0.1651279 0.8120982 0.5910451 1.2327422 1.2103794
[10,] 0.1884532 1.2913322 0.07946749 0.6488782 0.5101747 1.0675992 0.7489330 0.5161522 1.3892534 0.00000000 0.7145627 0.7178645 1.4073980 1.0151108 0.8454291 0.3938361 1.1011333
[11,] 0.8956387 0.6310541 0.63729378 0.5261085 0.9792628 0.4181433 0.9233955 1.0819998 0.6951707 0.71456271 0.0000000 0.8664575 0.6947096 0.4834598 0.1356018 0.5461178 1.0454477
[12,] 0.7341867 1.1096631 0.71836075 1.1812592 1.2280224 1.2779583 1.4252205 0.5330259 1.4965830 0.71786450 0.8664575 0.0000000 1.4330837 0.7647417 0.9227184 0.3985077 1.7186094
[13,] 1.5900675 0.4426596 1.32916622 1.0236827 1.5938784 0.4233669 1.4164207 1.7561541 0.1651279 1.40739798 0.6947096 1.4330837 0.0000000 0.7089567 0.5717476 1.2065418 1.3451008
[14,] 1.1585921 0.3451571 0.95414183 1.0087225 1.4019012 0.7213447 1.3987227 1.1902533 0.8120982 1.01511077 0.4834598 0.7647417 0.7089567 0.0000000 0.4075242 0.6727414 1.5245634
[15,] 1.0239060 0.5022413 0.76908918 0.6303264 1.1114847 0.3627274 1.0361637 1.1885993 0.5910451 0.84542915 0.1356018 0.9227184 0.5717476 0.4075242 0.0000000 0.6422642 1.1226886
[16,] 0.4978953 0.9892539 0.36334785 0.7839851 0.8815630 0.9622865 1.0366643 0.5502418 1.2327422 0.39383612 0.5461178 0.3985077 1.2065418 0.6727414 0.6422642 0.0000000 1.3201832
[17,] 1.2344610 1.5716642 1.05597704 0.5498428 0.7707891 0.9453119 0.3986661 1.6132774 1.2103794 1.10113329 1.0454477 1.7186094 1.3451008 1.5245634 1.1226886 1.3201832 0.0000000
```

Then, we compute the sigma matrix and log-likelihood function for the multivariate normal density. The code are shown below.

```
library(mvtnorm)
##Calculate sigma1[i,j] = exp(-r*dist[i,j]^ a)
## d: distance; resid: residuals; theta: arbitrary input parameter of r and alpha
## compute the log-likelihood function for the multivariate normal density
f <- function(theta,d,resid){
  sigma1 = exp(-theta[1]* d** theta[2])
  loglik = sum(dmvnorm(resid, mean=rep(0,17), sigma = sigma1, log=TRUE))
  if(is.infinite(loglik)){loglik = sign(loglik)*1e12}
  return (-loglik)
}
## minimize the loglik function with respect to r and α, subject to
## constraints r > 0 and 0 < α ≤ 2 by using method = "L-BFGS-B"
optim(par = c(1, 2), fn = f, method="L-BFGS-B", lower = c(0,0), upper = c(Inf, 2), d = dist, resid = std_res.list[[4]])
```

Finally, we minimize the loglik function. The output shows that the estimated maximum likelihood of r is 0.1635427 and of α is 0.3896342.

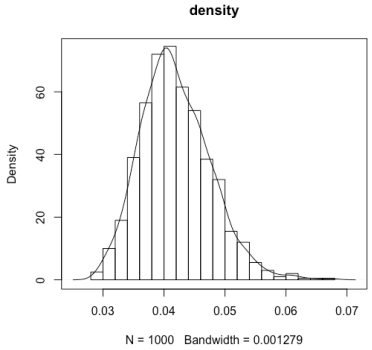
```
> optim(par = c(1, 2), fn = f, method="L-BFGS-B", lower = c(0,0), upper = c(Inf, 2), d = dist, resid = std_res.list[[4]])
$par
[1] 0.1635427 0.3896342

$value
[1] 1319.359

$counts
function gradient
      33       33

$convergence
[1] 0

$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

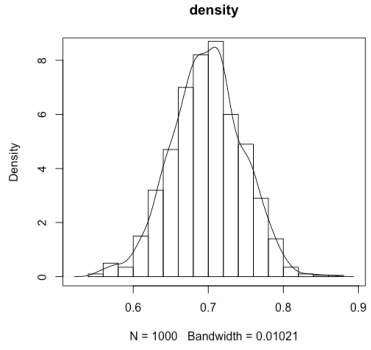
Step 4:

We use the bootstrap of residual method to approximate 95% confidence intervals for r and α in this step. First, we deal with our data by removing null value in this table. Then, we fit the linear model to get the residuals. The result of replicating 1000 times of our bootstrap is shown below.

```
> boot2

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = ori.res, statistic = samplemean, R = 1000)


Bootstrap Statistics :
      original       bias    std. error
t1* 0.03973709  0.002173863 0.005686107
t2* 0.73413435 -0.036189910 0.047118506
```

The plots of densities are shown in the following figures.





The final bootstrap confidence interval are shown below.

```
> boot.ci(boot2,index = c(1,1))
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = boot2, index = c(1, 1))

Intervals :
Level      Normal              Basic             Studentized
95%   ( 0.0264,  0.0487 )   ( 0.0254,  0.0475 )   ( 0.0274,  0.0484 )

Level     Percentile            BCa
95%   ( 0.0319,  0.0541 )   ( 0.0298,  0.0491 )
Calculations and Intervals on Original Scale
Some BCa intervals may be unstable
```

```
> boot.ci(boot2,index = c(2,2))
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = boot2, index = c(2, 2))

Intervals :
Level      Normal              Basic             Studentized
95%   ( 0.6780,  0.8627 )   ( 0.6821,  0.8641 )   ( 0.6839,  0.8774 )

Level     Percentile            BCa
95%   ( 0.6041,  0.7861 )   ( 0.6807,  0.8636 )
Calculations and Intervals on Original Scale
Warning : BCa Intervals used Extreme Quantiles
Some BCa intervals may be unstable
```