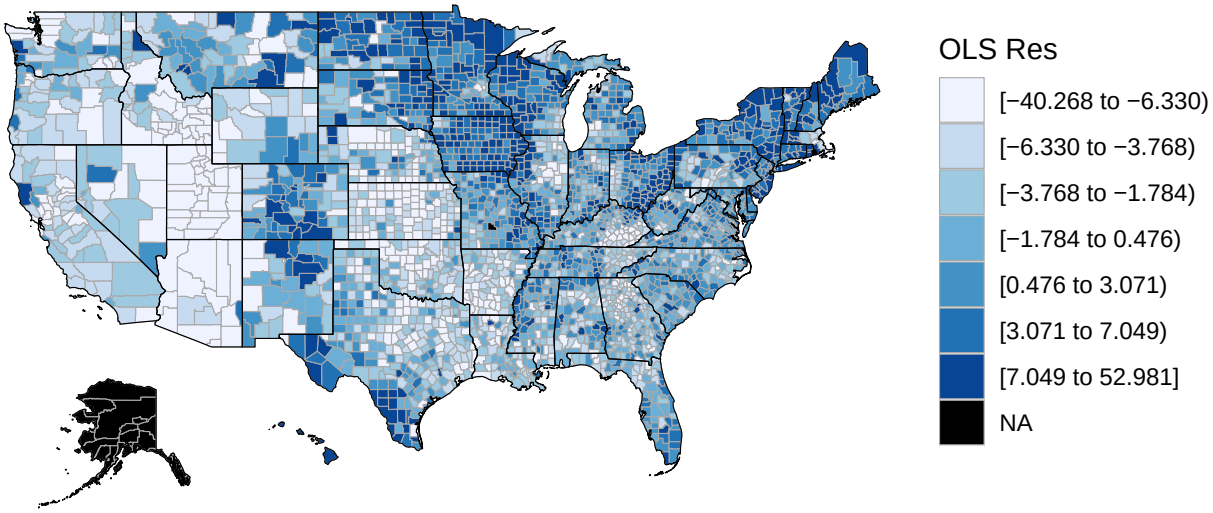
ST 533/433 : Homework 4 Solutions

October, 2020

Plot

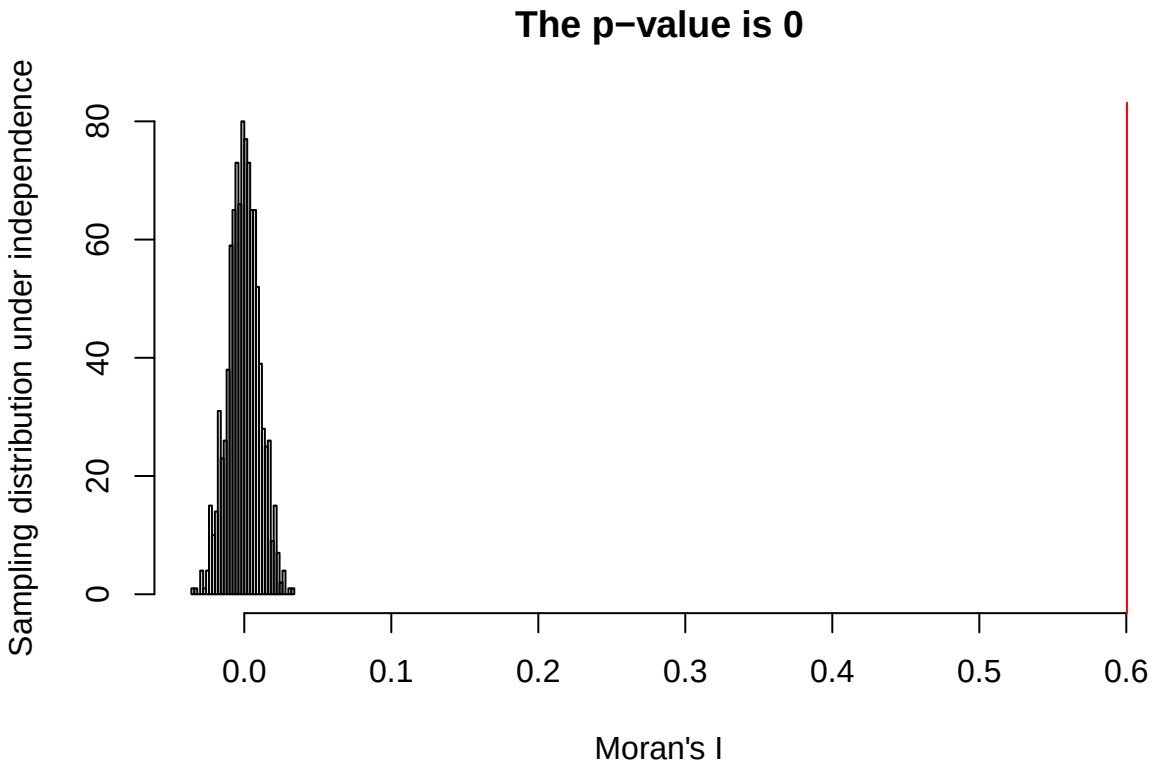


From the map, the residuals look spatially correlated.

Moran's I

Border Adjacency:

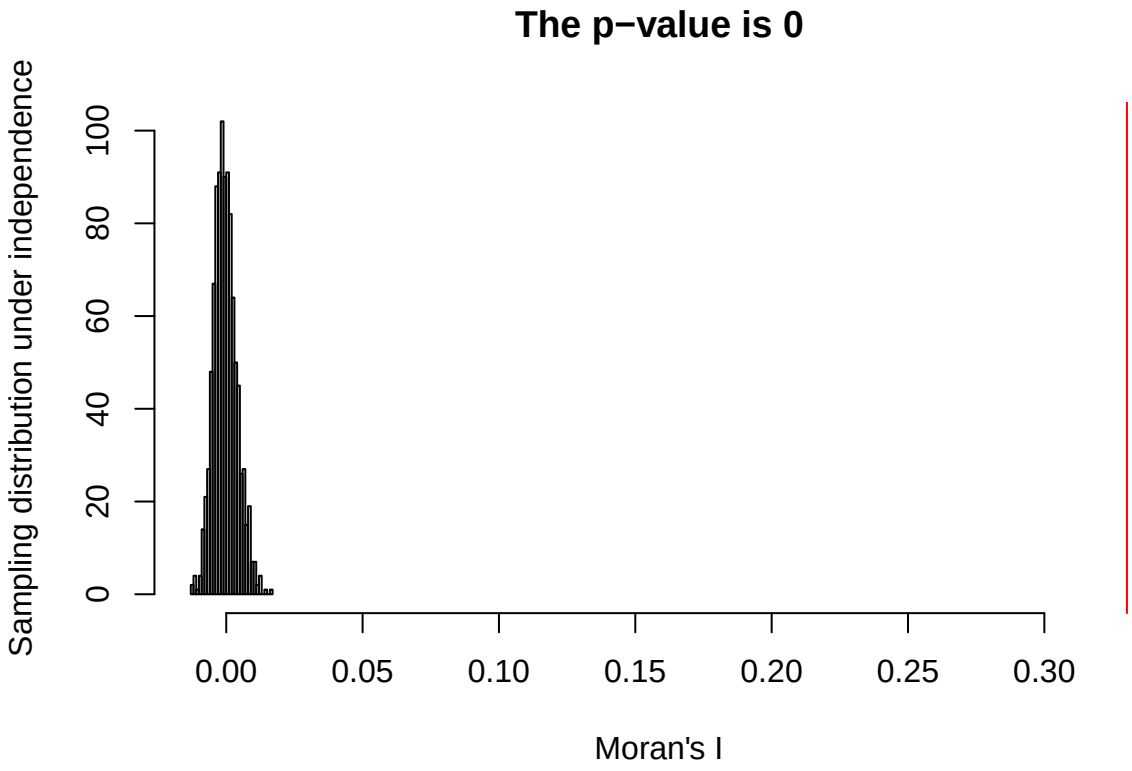
```
## [1] 0.6004
```




```
## [1] 0
```

Distance Adjacency:

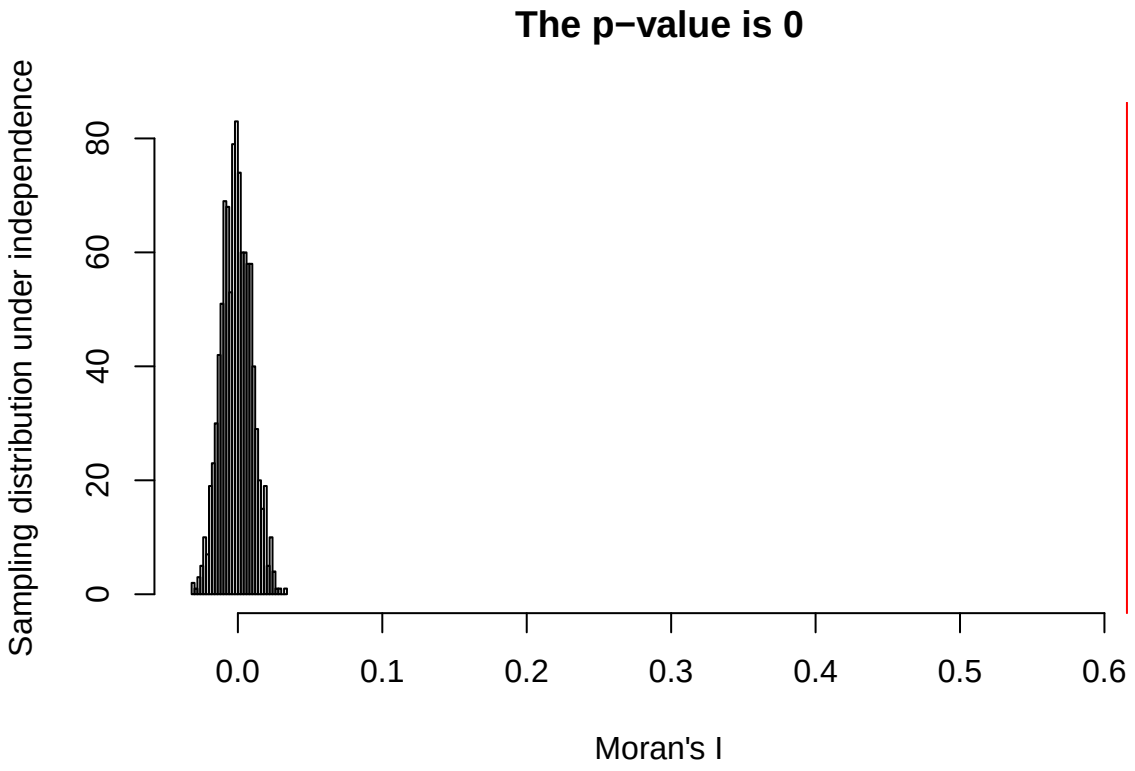
```
## [1] 0.3303
```




```
## [1] 0
```

Nearest Neighbour Adjacency:

```
## [1] 0.6156
```



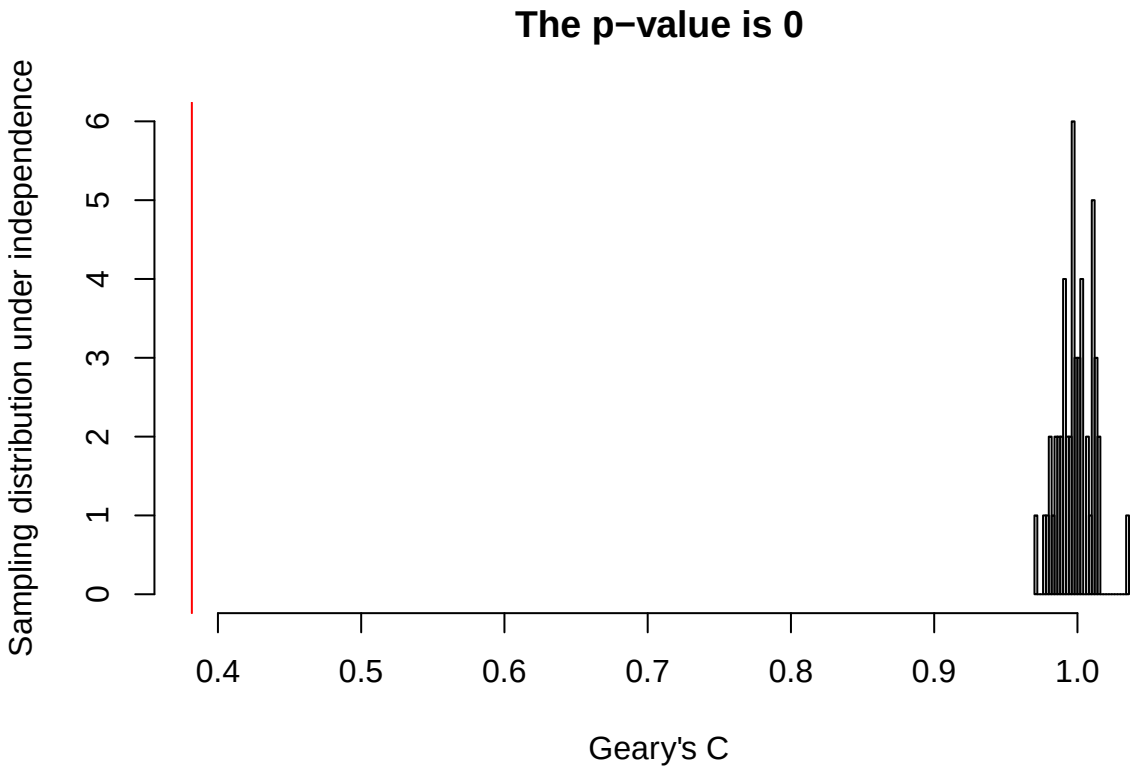

```
## [1] 0
```

In all the cases, from the value of Morans I statistic and the p-values we can say that clearly there is significant spatial autocorrelation.

Geary's C

Border Adjacency:

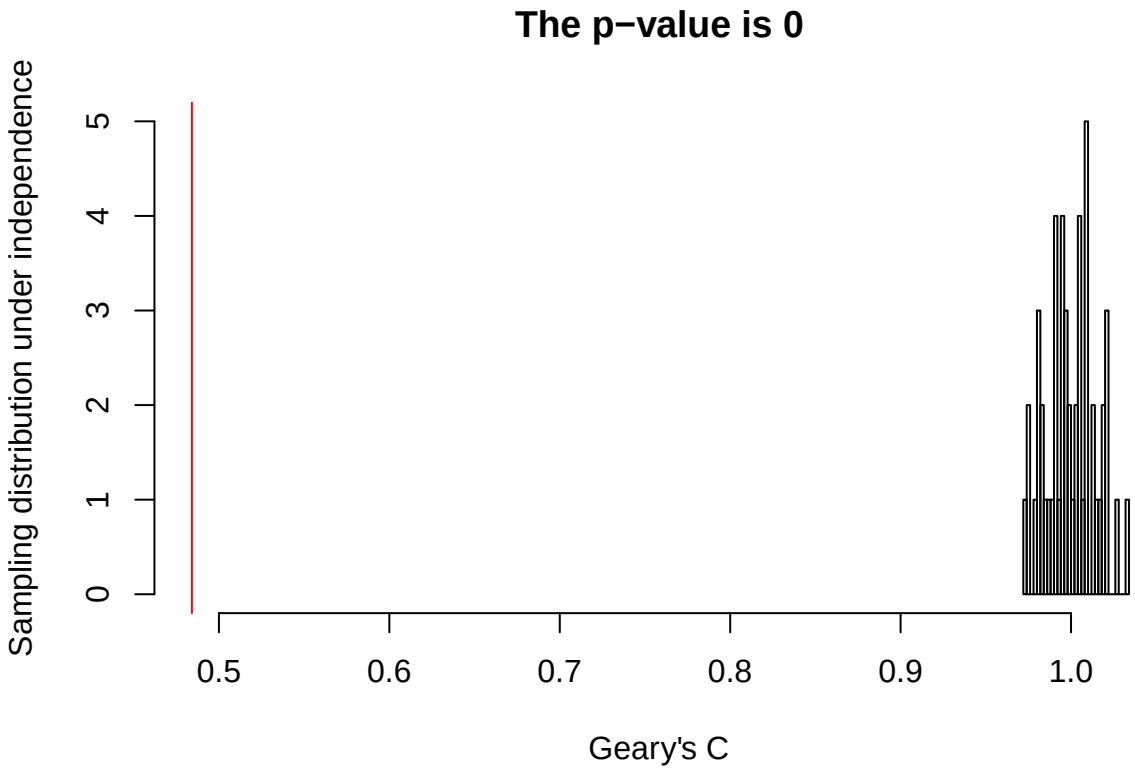
```
## [1] 0.3818
```




```
## [1] 0
```

Distance Adjacency:

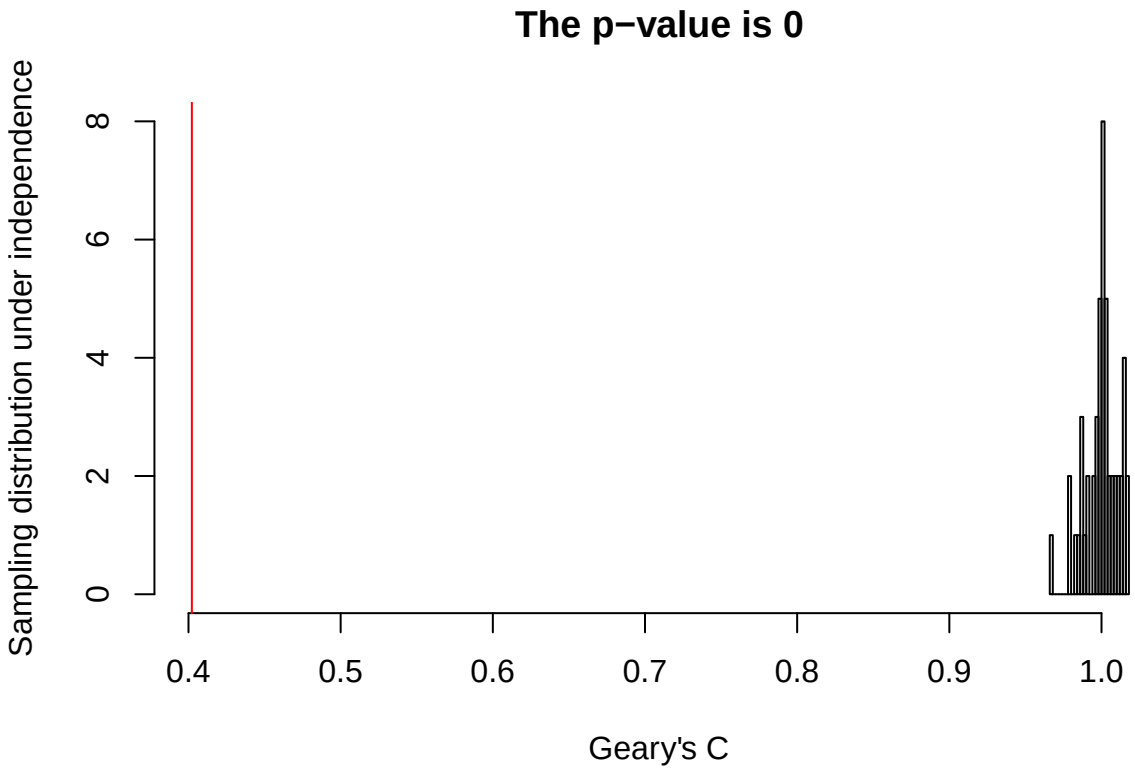
```
## [1] 0.4841
```




```
## [1] 0
```

Nearest Neighbour Adjacency:

```
## [1] 0.4022
```




```
## [1] 0
```

From the value of Geary's C statistic and the p-values we can say that clearly there is significant spatial autocorrelation.

Since, smaller values of Geary C statistic and higher values of Moran's I statistic signify spatial correlation, the results from the analysis are consistent using different weights. Computationally, Geary's C statistic takes a longer time for the p-values.

Codes

```
library(tidyverse)
library(housingData)
library(fields)
library(readr)
library(parallel)

load("C:/Users/Downloads/election_2008_2016.RData")
county_adj2010 <- read_csv("C:/Users/Downloads/county_adjacency2010.csv")

#Removing missing values
miss <- is.na(Y)
Y <- Y[!miss]
X <- X[!miss,]
fips <- fips[!miss]
```

```r
all_dat <- all_dat[!miss,]

#Linear regression
lreg <- lm(Y ~ X)

#Residuals
res <- lreg$residuals

# Define a function to make county maps
county_plot <- function(fips,Y,main="",units=""){
  library(choroplethr)
  temp  <- as.data.frame(list(region=fips,value=Y))
  suppressWarnings(county_choropleth(temp,title=main,legend=units))
}

#Plotting the residuals
county_plot(fips,res,main="Plot of residuals",units="OLS Res")

#Border adjacency

#Subset to only include counties in our dataframe
sub_adj <- as.matrix(county_adj2010[county_adj2010$fipscounty %in% fips &
                                    county_adj2010$fipsneighbor %in% fips,])

#Creating the ADJ matrix
ADJ1 <- matrix(0,nrow(all_dat),nrow(all_dat))
for(i in 1:nrow(sub_adj)){
  a1         <- which(fips==sub_adj[i,2])
  a2         <- which(fips==sub_adj[i,4])
  ADJ1[a1,a2] <- 1
}
diag(ADJ1) <- 0


#Creating coordinates from fips
f <- as.numeric(paste(geoCounty[,1]))
s <- matrix(NA,nrow(all_dat),2)
for(i in 1:nrow(all_dat)){
  these <- which(fips[i]==f)
  if(length(these)==1){
    s[i,1] <- geoCounty[these,4]
    s[i,2] <- geoCounty[these,5]
  }
}

#Distance adjacency

d <- rdist.earth(s) # Distance in miles
d[is.na(d)] <- Inf  # Set Alaska and HI to be infinitely far away
diag(d)     <- Inf  # Make sure counties don't neighbor themselves
nearest     <- apply(d,1,min)
D           <- max(nearest[nearest<Inf])
```

```r
ADJ2 <- ifelse(d <= D,1,0)

#K nearest neighbour adjacency

K    <- 5
ADJ3 <- matrix(0,nrow(all_dat),nrow(all_dat))
for(i in 1:nrow(all_dat)){
  ADJ3[i,]  <- rank(d[i,]) <= K
}
ADJ3 <- ADJ3 + t(ADJ3)          # Make ADJ symmetric
ADJ3 <- ifelse(ADJ3==2,1,ADJ3) # Make ADJ binary


#Moran's I

MoranI <- function(x,W){
  r   <- scale(x)
  rWr <- t(r)%*%W%*%r
  I   <- rWr/sum(W)
  return(as.vector(I))}

MoranI_pval <- function(x,W,nrep=1000,plot=TRUE){
  n    <- length(x)
  Io   <- MoranI(x,W)
  samp <- rep(0,nrep)
  for(i in 1:nrep){
    samp[i] <- MoranI(rnorm(n),W)
  }
  p <- mean(samp>Io)
  if(plot){
    alldat <- c(samp,Io)
    hist(samp,xlim=range(alldat),breaks=25,
         xlab="Moran's I",
         ylab="Sampling distribution under independence",
         main=paste("The p-value is",round(p,3)))
    abline(v=Io,col=2)
  }
  return(p)}

#Moran's I statistic
MoranI(res,ADJ1)
MoranI(res,ADJ2)
MoranI(res,ADJ3)

#Moran's I p value
MoranI_pval(res,ADJ1)
MoranI_pval(res,ADJ2)
MoranI_pval(res,ADJ3)


#Geary's C

GearyC <- function(iter=1,x,W){
```

```
  r   <- scale(x)
  diff.mat <- matrix(0, length(r), length(r))
  for (i in 1:length(r)){
    for (j in 1:length(r)){
      diff.mat[i,j] <- (r[i] - r[j])^2*W[i,j]
    }
  }
  C <- sum(diff.mat)/(2*sum(W))
  return(as.vector(C))}

# GearyC_pval <- function(x,W,nrep=1000,plot=TRUE){
#   n    <- length(x)
#   Co   <- GearyC(x=x,W=W)
#
#   samp <- rep(0,nrep)
#   for(i in 1:nrep){
#     samp[i] <- GearyC(x=rnorm(n),W=W)
#   }
#
#   p <- mean(samp<Co)
#   if(plot){
#     alldat <- c(samp,Co)
#     hist(samp,xlim=range(alldat),breaks=25,
#          xlab="Geary's C",
#          ylab="Sampling distribution under independence",
#          main=paste("The p-value is",round(p,3)))
#     abline(v=Co,col=2)
#   }
#   return(p)}

#We have used 8 cores for parallelising this function.
#Since it was taking time. To use without the parallel
#version, see the GearyC_pval function above.

GearyC_pval <- function(x,W,nrep=1000,c=8,plot=TRUE){
  n    <- length(x)
  Co   <- GearyC(x=x,W=W)
  ncores <- c
  cl <- makeCluster(ncores)
  clusterExport(cl,c())
  clusterEvalQ(cl,{
  })
  clusterExport(cl, list("GearyC","n"))
  samp <- parSapply(cl,1:nrep, function(t) GearyC(iter=t,x=rnorm(n),W=W))
  stopCluster(cl)
  p <- mean(samp<Co)
  if(plot){
    alldat <- c(samp,Co)
    hist(samp,xlim=range(alldat),breaks=25,
         xlab="Geary's C",
         ylab="Sampling distribution under independence",
         main=paste("The p-value is",round(p,3)))
    abline(v=Co,col=2)
```

```
  }
  return(p)}

#Geary's C statistic
GearyC(x=res,W=ADJ1)
GearyC(x=res,W=ADJ2)
GearyC(x=res,W=ADJ3)

#Geary's C p value
GearyC_pval(x=res,W=ADJ1)
GearyC_pval(x=res,W=ADJ2)
GearyC_pval(x=res,W=ADJ3)
```