MLE: Lab 4

Shahryar Minhas

February 5, 2015

Serial Collinearity (most of this is cobbles together from a variety of sites I like)

What is it?

There are times, especially in time-series data, that the CLR assumption of

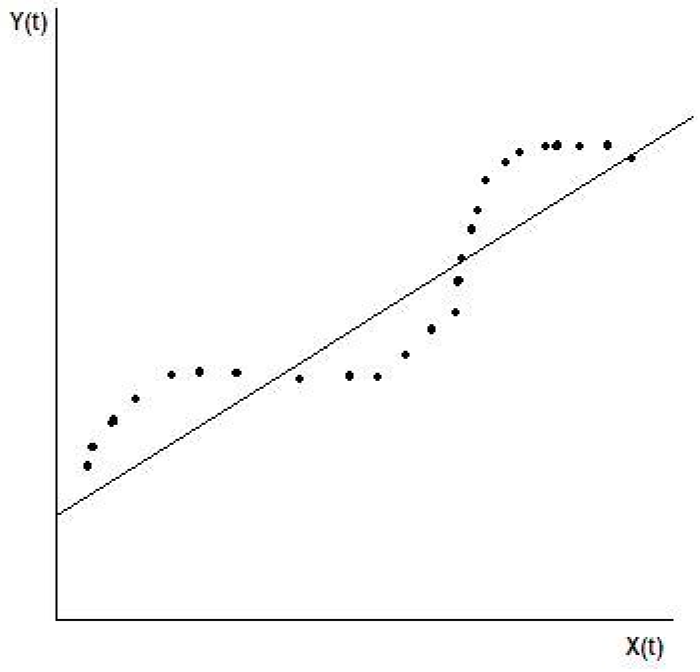
$$corr(\epsilon_t, \epsilon_{t-1}) = 0$$

is broken. This is known in econometrics as Serial Correlation or Autocorrelation. This means that

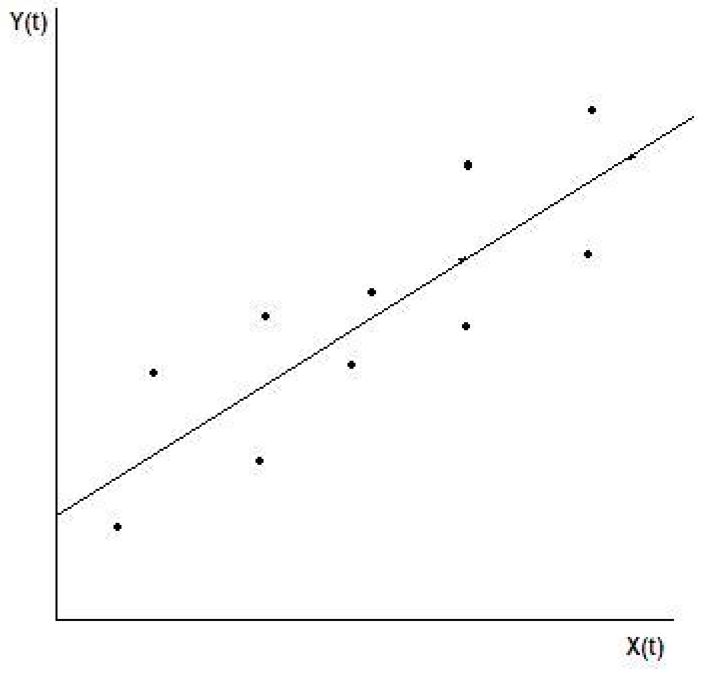
$$corr(\epsilon_t, \epsilon_{t-1}) \neq 0$$

and there is a pattern across the error terms. The error terms are then not independently distributed across the observations and are not strictly random.

Positive autocorrelation example:



Negative autocorrelation example:



When the error term is related to the previous error term, it can be written in an algebraic equation.

$$\epsilon_t = \rho\epsilon_{t-1} + u_t$$

where ρ is the autocorrelation coefficient between the two disturbance terms, and u is the disturbance term for the autocorrelation. This is known as an Autoregressive Process.

$$-1 < \rho = corr(\epsilon_t, \epsilon_{t-1}) < 1$$

The u is needed within the equation because although the error term is less random, it still has a slight random effect.

Dealing with autocorrelation

To deal with autocorrelation we can try a number of things. They are the beyond scope of this class for now, but I recommend that you test for it at least.

Causes

- Besides, three factors can also lead to serially correlated errors. They are: (1) omitted variables, (2) ignoring nonlinearities, and (3) measurement errors. For example, suppose a dependent variable Y_t is related to the independent variables X_{t1} and X_{t2}, but the investigator does not include X_{t2} in the model. The effect of this variable will be captured by the error term ϵ_t . Because many time series exhibit trends over time, X_{t2} is likely to depend on $X_{t-1,2}, X_{t-2,2}, \ldots$ This will translate into apparent correlation between ϵ_t and $\epsilon_{t-1}, \epsilon t - 2, \ldots$, thereby violating the serial independence assumption. Thus, growth in omitted variables could cause autocorrelation in errors.
- Serial correlation can also be caused by misspecification of the functional form. Suppose, for example, the relationship between Y and X is quadratic but we assume a straight line. Then the error term ϵ_t will depend on X^2 . If X has been growing over time, ϵ_t will also exhibit such growth, indicating autocorrelation.

- Example: Consider stock market data. The price of a particular security or a stock market index at the close of successive days or during successive hours is likely to be serially correlated.
- Example: Consider the consumption of electricity during different hours of the day. Because the temperature patterns are similar between successive time periods, we can expect consumption patterns to be correlated between neighboring periods. If the model is not properly specified, this effect may show up as high correlation among errors from nearby periods.
- Systematic errors in measurement can also cause autocorrelation. For example, suppose a firm is updating its inventory in a given period. If there is a systematic error in the way it was measured, cumulative inventory stock will reflect accumulated measurement errors. This will show up as serial correlation.

Consequences

- Coefficients are still unbiased $E(\epsilon_t) = 0$, $cov(X_t, u_t) = 0$
- True variance of $\hat{\beta}$ is increased, by the presence of autocorrelations.
- Estimated variance of $\hat{\beta}$ is smaller due to autocorrelation (biased downward).
- A decrease in se($\hat{\beta}$) and an increase of the t-statistics; this results in the estimator looking more accurate than it actually is.
- R^2 becomes inflated.

Testing for serial autocorrelation

Durbin Watson or Breusch Godfrey tests are commonly used.

```r
# from WDI lets get some data
wbVars=c("BX.KLT.DINV.CD.WD","SP.POP.TOTL", "NY.GDP.DEFL.KD.ZG",
  "NY.GDP.PCAP.KD", "NY.GDP.MKTP.KD.ZG")
wbData=WDI(country='all', indicator=wbVars,
  start=1988, end=2010, extra=T)
names(wbData)[4:8]=c('fdi', 'population', 'inflation', 'gdpCap', 'gdpGr')

# log everything
# wbData$fdi = log(wbData$fdi + abs(min(wbData$fdi, na.rm=T)) + 1)
# wbData$population = log(wbData$population + abs(min(wbData$population, na.rm=T)) + 1)
# wbData$gdpCap = log(wbData$gdpCap + abs(min(wbData$gdpCap, na.rm=T)) + 1)

# take a peak at your data
head(wbData)

# Lets throw out non country units
wbData$cname = countrycode(wbData$iso2c, 'iso2c', 'country.name')
wbData = wbData[!is.na(wbData$cname),]
head(wbData)
unique(wbData$cname)

# Good lets construct some model for FDI
modData=na.omit(wbData[,c('fdi', 'population', 'inflation', 'gdpCap', 'gdpGr','year', 'cname')])
dim(wbData); dim(modData)
```

```r
modData=modData[which(modData$cname=='China'),]
mod1=lm(fdi ~ population + inflation + gdpCap + gdpGr, data=modData)
summary(mod1)

# from lmtest lets examine serial correlation using dwtest and bgtest
dwtest(mod1)
bgtest(mod1)

# Visualize residuals versus fitted values
check=data.frame(pred=mod1$fitted.values, res=mod1$residuals, yr=modData$year)

# Add some color
ggplot(check, aes(x=pred, y=res, color=yr)) + geom_point()
```

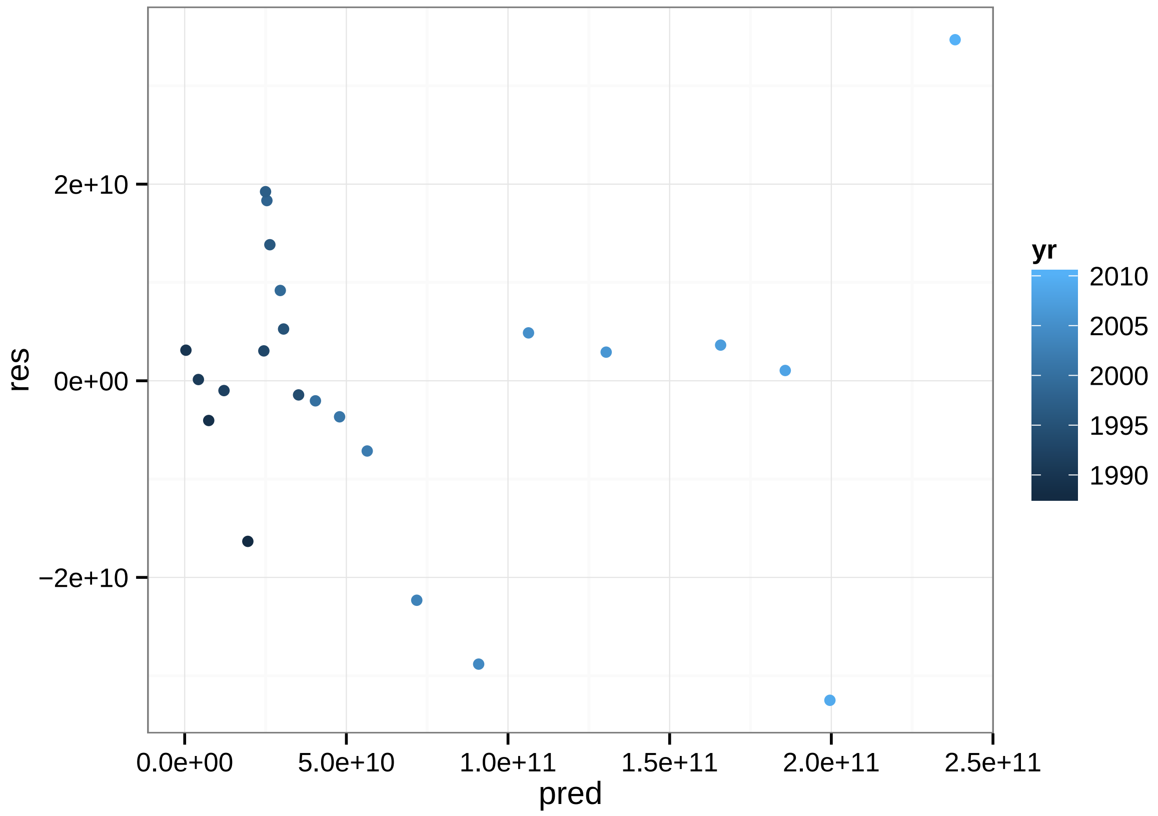


Figure 1:

```r
ggplot(check, aes(x=yr, y=res)) + geom_point()
```

```r
# Lets add a line for the mean
meanres=data.frame(mean=tapply(check$res, check$yr, FUN=mean ), yr=sort(unique(check$yr)) )
ggplot(check, aes(x=yr, y=res)) + geom_point() + geom_line(data=meanres, aes(x=yr,y=mean))
```

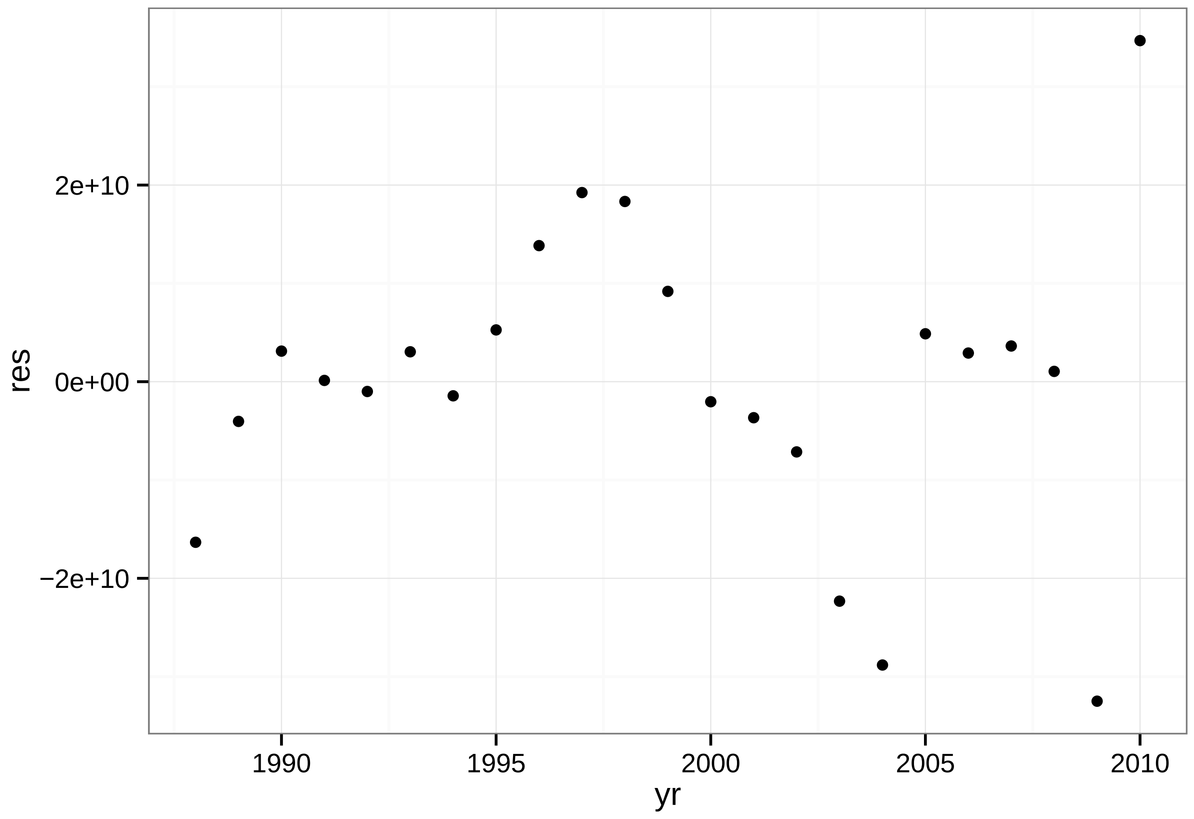


Figure 2:

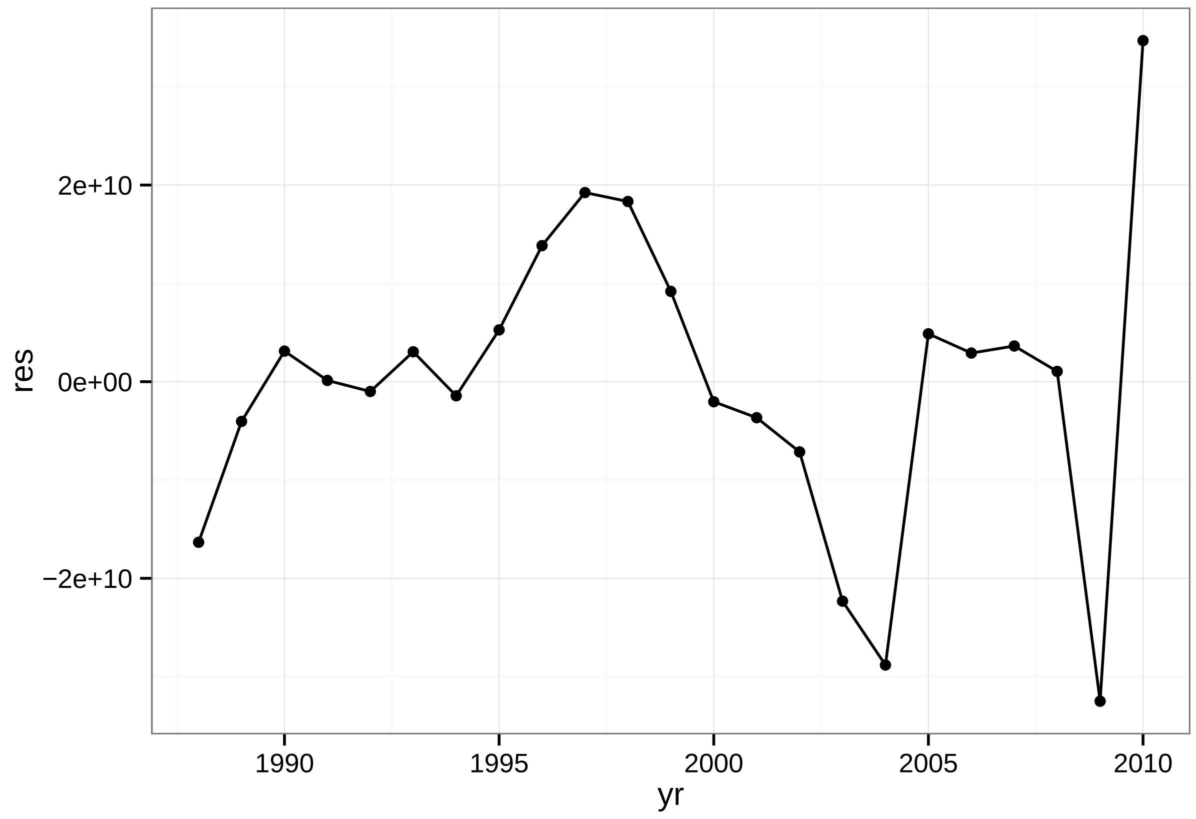


Figure 3:

Spatial Autocorrelation

Same issues as serial correlation.

- Example: The city of St. Paul has a spike of crime and so they hire additional police. The following year, they found that the crime rate decreased significantly. Amazingly, the city of Minneapolis, which had not adjusted its police force, finds that they have a increase in the crime rate over the same period.

Moran's statistic

Moran's I is a measure of spatial autocorrelation–how related the values of a variable are based on the locations where they were measured.

To calculate Moran's I, we will need to generate a matrix of inverse distance weights. In the matrix, entries for pairs of points that are close together are higher than for pairs of points that are far apart.

```r
# Load a helpful dataset
setwd(labPath)
load('panel.rda')

# Finding distances
capdist=distmatrix(as.Date('2000-1-1'), type='capdist',  tolerance=0.5, useGW=FALSE)
capdist[1:5, 1:5]

# lets see if we can calculate Moran's I for FDI
# lets focus on 2000 data
fdi=wbData[which(wbData$year == 2000), c('cname', 'iso3c', 'fdi')]
fdi=na.omit(fdi)

# Lets match countries with values in distance matrix
# First we need to get ccodes for the fdi dataset because that is what is used
# in our distance matrix
fdi$ccode=panel$ccode[match(toupper(fdi$cname), panel$cname)]

# Some countries did not match, lets just drop 'em
sum(is.na(fdi$ccode))
fdi=na.omit(fdi)

# Reorder rows of fdi dataframe
fdi=fdi[order(fdi$ccode),]

# Now lets isolate the fdi and capdist matrix to only
# countries that exist in both objects
matches=intersect(fdi$ccode, rownames(capdist))
fdi=fdi[which(fdi$ccode %in% matches),]
capdist=capdist[matches, matches]

# Like the example above we have to invert the distance matrix
invcapdist = 1/capdist
diag(invcapdist) = 0

# Now lets Moran's test
Moran.I(fdi$fdi, invcapdist)
```

Some fancy stuff. Spatial statistics is a huge field. If you're interested in it follow the work of Roger Bivand.

```
# More fancy stuff
# Could also use spdep to make the Moran's plot
cshp00 = cshp(date=as.Date('2000-1-1'), useGW=TRUE)

# Remove polygons from cshp00
cshp00 = cshp00[which(cshp00$COWCODE %in% matches),]

# Centroid coordinates
coords=coordinates(cshp00)

# Get list of neighbors
cshp00nb=poly2nb(cshp00, queen=TRUE)
cshp00nbBin=nb2listw(cshp00nb, style="W", zero.policy=TRUE)

# Plot centroid connections
plot(cshp00nbBin,coords=coords,pch=19, cex=0.1, col="gray")
```

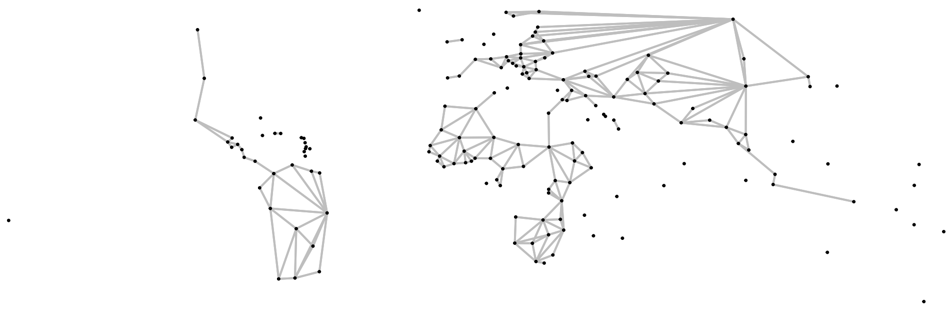


Figure 4:

```
# Moran test again
moran.test(fdi$fdi, listw=cshp00nbBin, zero.policy=T)
geary.test(fdi$fdi, listw=cshp00nbBin, zero.policy=T)

# Moran plot
moran.plot(fdi$fdi, listw=cshp00nbBin, zero.policy=T, pch=16, col="black",cex=.5, quiet=F, labels=as.cha
```

Map Visualizations using cshapes

```
## Warning in gpclibPermit(): support for gpclib will be withdrawn from
## maptools at the next major release
```

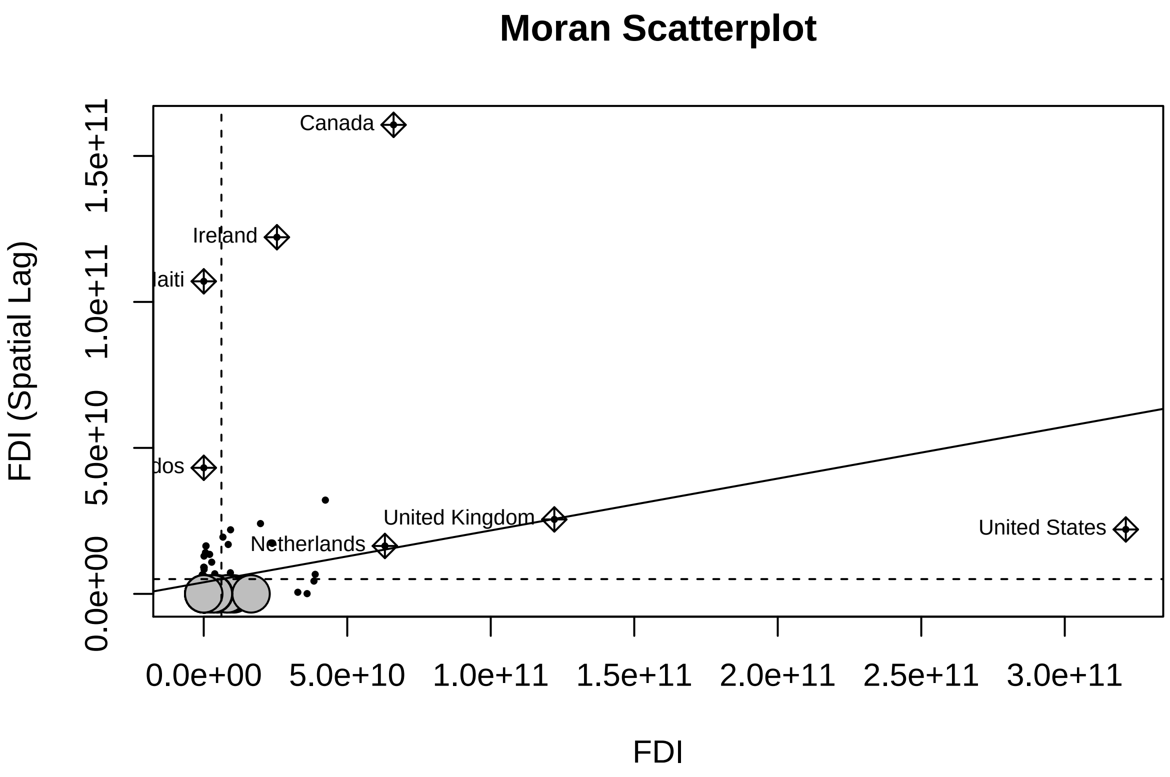
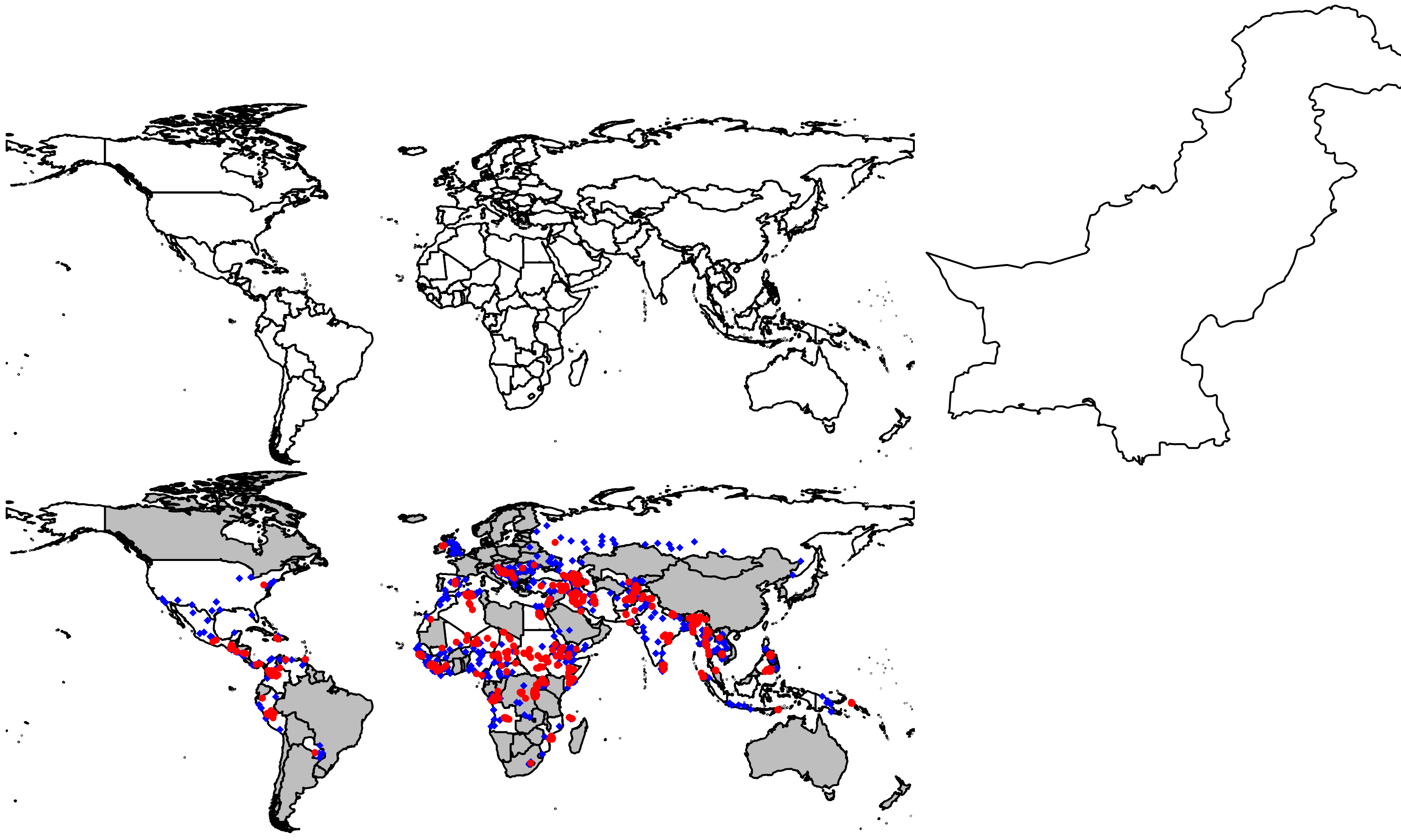

Moran Scatterplot
FDI (Spatial Lag)
FDI
Canada
Ireland
laiti
dos
United Kingdom
Netherlands
United States
0.0e+00
5.0e+10
1.0e+11
1.5e+11
2.0e+11
2.5e+11
3.0e+11
0.0e+00
5.0e+10
1.0e+11
1.5e+11

Figure 5:



Lets focus in on conflict over time in India.

```r
load("cityTotPopLatLongvFinal.rda")
prio=read.csv("ConflictSite 4-2010_v3 Dataset.csv")

fYrCty$cname = toupper( countrycode(
  fYrCty$cname,"country.name","country.name") )
prio$Conflict.territory = toupper( countrycode(
    prio$Conflict.territory,"country.name","country.name") )

cname="INDIA"
worldmap=cshp(date=as.Date("1990-01-01"),useGW=F)
ccode = countrycode(cname,"country.name","cown")
cntryShape = worldmap[worldmap$COWCODE==ccode,]
newprio = na.omit(prio[prio$Conflict.territory==cname,])

gpclibPermit()
```

```
## Warning in gpclibPermit(): support for gpclib will be withdrawn from
## maptools at the next major release

## [1] TRUE
```

```r
ggmap = fortify(cntryShape, region = "COWCODE")
ggmapData = data.frame("id" = unique(ggmap$id))
ggmapData$id = as.numeric(as.character(ggmapData$id))

temp = ggplot(ggmapData, aes(map_id = id))
temp = temp + geom_map(map=ggmap, fill='white',
  linetype=1, colour='black') + expand_limits(x = ggmap$long, y = ggmap$lat)
temp = temp + geom_point(aes(
    x=fYrCty$cleanLong[fYrCty$cname==cname & fYrCty$Capital==0],
    y=fYrCty$cleanLat[fYrCty$cname==cname & fYrCty$Capital==0]),
    pch=17,size=4,col='darkgrey')
temp = temp + geom_point(aes(
    x=fYrCty$cleanLong[fYrCty$cname==cname & fYrCty$Capital!=0],
    y=fYrCty$cleanLat[fYrCty$cname==cname & fYrCty$Capital!=0]),
    pch=18,size=5,col='darkgrey')
temp = temp + geom_point(aes(x=newprio$Longitude, y=newprio$Latitude,
    color=newprio$Year),size=4)
temp = temp + scale_colour_gradient('',
    low=brewer.pal(9,'Blues')[2],high=brewer.pal(9,'Blues')[9],
    breaks=seq(min(newprio$Year),max(newprio$Year),6))
temp = temp + theme(
  line=element_blank(),title=element_blank(),
  axis.text.x=element_blank(),axis.text.y=element_blank(),
  legend.position='top', legend.key.width=unit(4,"line"),
  panel.grid.major=element_blank(),
  panel.grid.minor=element_blank(), panel.border=element_blank())
temp
```

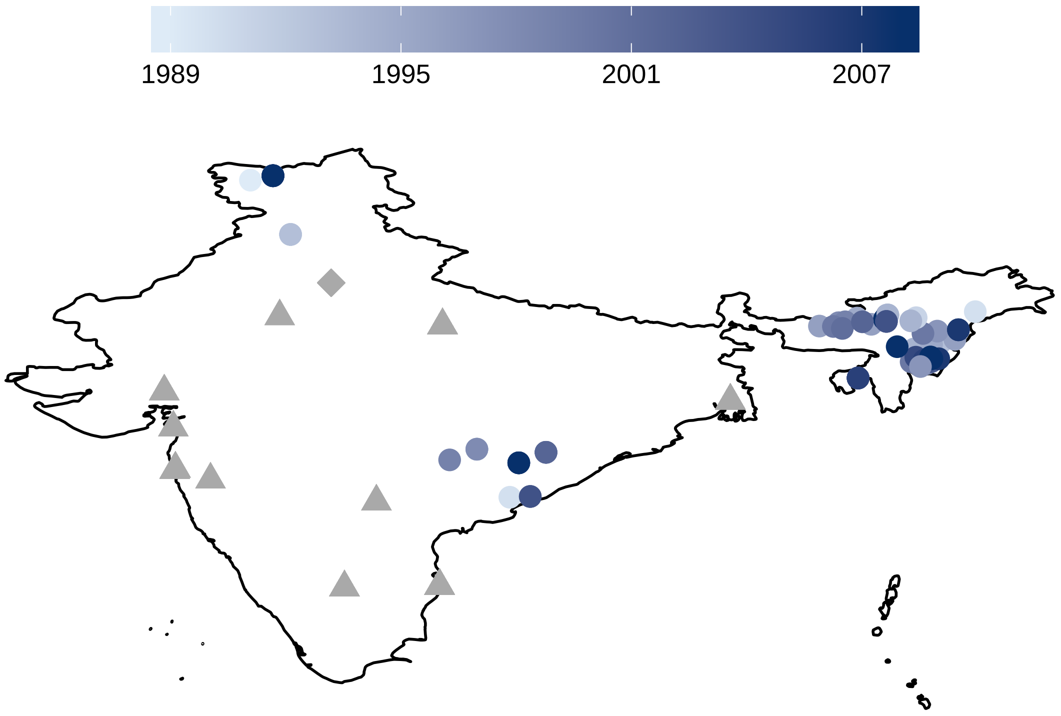


Figure 6:

Constructing spatially weighted regression coefficients

```
spatialBuild <- function(spatList, varData, years, variable, sp_suffix, invert=FALSE){
  varData <- varData
    spatData <- NULL

    for(i in 1:length(years)){
        spatMat <- spatList[[i]]
        # rownames for matrices
        distNames <- as.numeric(rownames(spatMat))
        ndistNames <- panel$ccode[match(distNames, panel$GWCODE)]
        rownames(spatMat) <- ndistNames; colnames(spatMat) <- ndistNames

        # Invert
        if(invert){spatMat <- 1/spatMat; spatMat[spatMat==Inf] <- 0}

        # Applying row standardized weights
        dmatDenom <- apply(spatMat,1,sum)
        dmatDenom[dmatDenom==0] <- 1
        spatMat_rowst <- spatMat/dmatDenom

        # Bringing in fdi dataset
        spat_vars <- c('ccode', variable)
        dataYear <- varData[varData$year==years[i], spat_vars]
        dataYear <- dataYear[which(dataYear$ccode %in% ndistNames),]
        o <- as.character(dataYear$ccode)
        spatMat_rowst <- spatMat_rowst[o,o]
        # data rows with NAs that are in distance matrix
        # this is equivalent to just dropping them from teh
        # spatial variable calculation
        dataYear[is.na(dataYear)] <- 0

        for(j in 1:nrow(spatMat_rowst)){
            row_weights <- NULL
            row_weights <- t(t(dataYear[,c(2:ncol(dataYear))]) %*%  spatMat_rowst[j,])
            row_weights2 <- NULL
            row_weights2 <- cbind(row_weights, years[i], dataYear$ccode[j])
            spatData <- rbind(spatData, row_weights2)
        }
    print(years[i])}
    spatData <- data.frame(spatData, row.names=NULL)

    names(spatData) <- c(
        paste(sp_suffix,names(spatData)[1:(length(spat_vars)-1)],sep=''),
        'year','ccode')
    spatData$cyear <- paste(spatData$ccode, spatData$year, sep='')
    spatData
}
```