TESTING FOR SPATIAL AUTOCORRELATION IN MODEL FITTING

Mike Antolin

November 5, 2007

Getting Started

The R package is supported online by a series of servers that contain the main GUI (program), and a series of additional packages useful for specialized analyses. We will first open the R program then load packages onto each of your computers the necessary.

The most generally useful package is in the basic R, the MASS library.

```
# Load main statistical library
library(MASS)
```

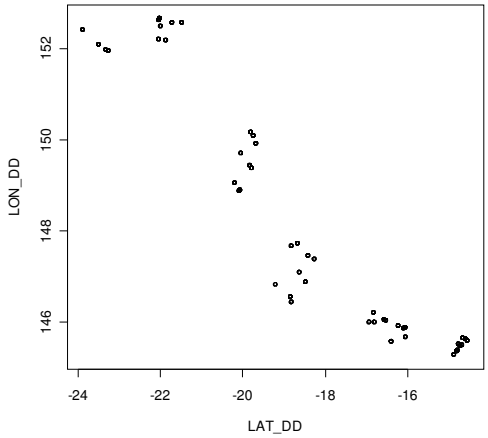
Let's load the coral data (Bruno et al. 2007[1]) and make some nice plots to see what they look like.

First, change the default directory to the place where you have the data file, by going to the "File" pulldown and then the "Change dir..." button.

```
### Import data

coral<-read.table("GBR_WS_data.csv",sep=",",header=T)
attach(coral)
names(coral)
X11()               # create a graphics window for each plot
plot(LAT_DD,LON_DD)
```

[1]Bruno, J.F., E.R. Selig, K.S. Casey, C.A. Page, B. L. Willis, C.D. Harvell, H. Sweatman A. M. Melendy. 2007. Thermal Stress and Coral Cover as Drivers of Coral Disease Outbreaks. PLoS Biology.



Now let's make a couple of other neat looking plots of the same data, just for fun. First, use the "Packages" pulldown to get the `scatterplot3d` and `geoR` packages.

```
# Make some fancy plots for exploratory data analysis
library(geoR)
library(scatterplot3d)

# make data columns used by points.geodata
coral$coords = cbind(LAT_DD,LON_DD)
coral$data = CASES

X11()
plot(LAT_DD, LON_DD, type= "n",main="Number of Cases",
ylab="Latitude",xlab="Longitude")
text(jitter(LAT_DD), jitter(LON_DD),CASES,cex=0.75)
X11()
points.geodata(coral,cex.var=CORAL_COVER,cex.min=.1,cex.max=3,
main="Coral Cover",ylab="Latitude",xlab="Longitude")
X11()
points.geodata(coral,cex.var=WSSTA,cex.min=.1,cex.max=3, main="Ocean Temperature",
ylab="Latitude",xlab="Longitude")
X11()
scatterplot3d(LAT_DD, y=LON_DD, CASES, type="h", main="Number of Cases",
ylab="Latitude",xlab="Longitude")
```

Testing the data for fit to the Poisson and negative binomial distributions:

Parasite data are quite often NOT normally distributed, so data should be plotted and tested for fits to either Poisson (random) or the negative binomial distribution (aggregated, or clumped).

First fit to a Poisson distribution:

```
# Make some definitions and descriptions of the data
avecases<-mean(CASES)                    # average cases defined
varcases<-var(CASES)                     # variance of cases
sd(CASES)
summary(CASES)
sample_size=length(CASES)
```

```
sample_size


# Create data tables for analysis
frequencies<-table(CASES)
frequencies

ucases<-as.vector(names(frequencies)) # get levels from the table into a vector
ycases<-as.numeric(ucases)            # change them from a vector to a number
maxlev<-nlevels(as.factor(ycases))    # number of levels in frequency table
maxlev                                # this will be used for plotting and binning

#  plot the fit to Poisson distribution
X11()
par(mfrow=c(1,2))
barplot(frequencies,ylab="Frequency",xlab="Obs. Cases")
barplot(dpois(0:(maxlev-1),avecases)*sample_size,names=as.character(0:(maxlev-1)),
ylab="Frequency",xlab="Exp. Cases")
varmean=varcases/avecases
varmean
par(new=T)
par(mfrow=c(1,1))
if (varmean>1) text(12,25,"Looks like this is NOT Poisson distributed!",
cex=1,col="RED",font=4)
```

Now check the negative binomial distribution

```
#  plot the fit to to a negative binomial
X11()
par(mfrow=c(1,2))
barplot(frequencies,ylab="Frequency",xlab="Obs. Cases")
barplot(dnbinom(0:(maxlev-1),1,mu=avecases)*sample_size,names=as.character(0:
(maxlev-1)),ylab="Frequency",xlab="Exp. Cases")
k<-avecases^2/(varcases-avecases)
par(new=T)
par(mfrow=c(1,1))
if (k<1) text(12,20,"Looks like this could be negative binomial!",
cex=1,col="RED",font=4)
```

Make a sweet plot!

```
# Plot double a historgram with both observed and expected on it
X11()
exp<-dnbinom(0:(maxlev-1),1,mu=avecases)*sample_size
both<-numeric(2*maxlev)
both[1:(2*maxlev) %% 2!=0]<-frequencies
both[1:(2*maxlev) %% 2==0]<-exp
lab<-character(2*maxlev)
lab[1:(2*maxlev) %% 2==0]<-as.character(0:(maxlev-1))
par(mfrow=c(1,1))
barplot(both,col=rep(c("white","grey"),maxlev),names=lab,ylab="Frequency",xlab="Cases"
)
legend(90,70,c("obs","exp"),fill=c("white","grey"))
par(new=T)
# make fancy X-axis
axis(1,at=c(1:(2*maxlev)),
     label=labels[1:(2*maxlev) %% 2==0]<-as.character(0:((2*maxlev)-1)),
     lwd=2,pos=-1.5,cex.axis=0.5)
```

How about a statistical test for goodness of fit?

```
##  Do an approximate test of goodness of fit to the negative binomial distribution

# make categories
cs<-factor(0:(maxlev-1))

#levels(cs)   # this will check how many "bins" you've made

#  make your expected and observed categories, and calculate chi-squared
ef<-as.vector(tapply(exp,cs,sum))
of<-as.vector(tapply(frequencies,cs,sum))
chisq<-sum((of-ef)^2/ef)
ef
of

Prob<-1-pchisq(chisq,(nlevels(cs)-3))
Prob

# Write the result of your test on your plot
par(new=T)
par(mfrow=c(1,1))
mtext(paste("Chi-squared  = ",chisq," with probability P =",Prob),
side=3,cex=1,col="RED",font=4)
```

Fitting a GLM:

```
#  Use GLM to test effects of temperature and coral cover  on whiting, between regions
#  original model
library(MASS)

#  Fit a GLM using quasi-Poisson errors
fit1 = glm(CASES ~ WSSTA + CORAL_COVER + WSSTA*CORAL_COVER + factor(SECTOR),
family=quasipoisson)
coef(fit1)
summary(fit1)

# Estimate the “k” parameter for the negative binomial distribution
ybar=predict(fit1,type="response")
theta.ml(CASES, ybar)

theta1=0.3052719
#  Fit a GLM with negative binomial errors
fit2=glm(CASES ~ WSSTA + CORAL_COVER + WSSTA*CORAL_COVER +factor(SECTOR),
family=negative.binomial(theta=theta1))

coef(fit2)
summary(fit2)
plot(WSSTA,fit2$resid,xlab="WSSTA")
```

Testing residuals of the model for additional spatial autocorrelation

Moran’s *I*, a measure of spatial autocorrelation to identify departures from spatial randomness over all samples in the system:

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

y_i and y_j are values of your variable at any points i and j
n is the number of sampling points,
w_{ij} is the spatial proximity between points i and j,
W is the sum of all n^2 values of w_{ij}, the spatial weights matrix.

A positive I would indicate that additional local processes not accounted for in the model could be acting.

First load the spatial library:

```
##### Install Spatial Stats library
S_HOME="RSpatial/"
source(paste(S_HOME,"Spatial_start.R",sep=""))
Spatial_start()
.First<-function(){Spatial_start()
```

Create the spatial weights matrix, in this case a rescale inverse of distance between points.

```
spt.wt<-spwtdist(LAT_DD,LON_DD,rescale=T)
```

Run the spatial analysis and make the diagnostic plots:

```
morani(fit2$resid,spt.wt)

#Plots to test for spatial autocorrelation
par(mfrow=c(2,2))
plot(WSSTA,fit2$resid,xlab="WSSTA")
plot(fit2$resid, CASES -fit2$resid,xlab="Residual",ylab="Predicted value")
title("Predicted values vs residuals",cex=.7)
plot(fit2$resid,spt.wt %*% fit2$resid, xlab="Residual",ylab="Weighted
residuals")
title("W * RESIDUALS VS. RESIDUALS",cex=.7)
cor(fit2$resid,spt.wt %*% fit2$resid)
```