Appendix

Code to Generate Indicated Graphs in Lecture Notes

```
#########################################################
## Read in S04 data
#########################################################
so4data = read.table('http://homepage.stat.uiowa.edu/~dzimmer/spatialstats/so4.dt')
names(so4data) = c('lat1', 'lat2', 'lat3', 'long1', 'long2', 'long3', 'y')
attach(so4data)
v=lat1+lat2/60+lat3/3600;
u=-1*(long1+long2/60+long3/3600)
so4x = u
so4y = v


#########################################################
## "SO4 Map", page 41
#########################################################

par(mfrow = c(2,1))
plot(so4x,so4y, xlim = c(-140,-60), ylim = c(25,50),pch = 19,
cex = .5, xlab = "latitude", ylab = "longitude")
library(maps)
map("state")
points(so4x,so4y,pch = 19, cex = .5)



#########################################################
##"NC SIDS neighbor map", page 43
#########################################################

library(spdep)
library(maptools)
data(nc.sids) # read in data from spdep package
# creates a list object of the data
nc.sids <- readShapePoly(system.file("etc/shapes/sids.shp", package="spdep")[1],
  ID="FIPSNO", proj4string=CRS("+proj=longlat +ellps=clrk66"))

rn <- sapply(slot(nc.sids, "polygons"), function(x) slot(x, "ID"))
# create a "neighbor" object of the data
ncCC89nb <- read.gal(system.file("etc/weights/ncCC89.gal", package="spdep")[1],
  region.id=rn)
plot(nc.sids, border = "white",axes = T)
plot(ncCC89nb, coordinates(nc.sids),add = T, col="black")
```

```
#########################################################
## "Marginal Plots", page 47
#########################################################

plot(u, y, pch = 19, cex = .5, xlab = "longitude", ylab = "s04dep",
    main = "sulfate deposition versus longitude")

plot(v, y, pch = 19, cex = .5, xlab = "latitude", ylab = "s04dep",
    main = "sulfate depostition versus latitude")


#########################################################
## "Row/Column Mean/Median versus Index", page 48
#########################################################

library(gstat)
data(coalash)


xmeans = aggregate( coalash ~ x, data = coalash, mean )
par(mfrow = c(2,2))
plot(xmeans,pch = 19, main = "Column mean versus column index", xlab = "Column Index",
    ylab = "Column Mean")
ymeans = aggregate( coalash ~ y, data = coalash, mean )
plot(ymeans,pch = 19,main = "Row mean versus Row index", xlab = "Row Index",
    ylab = "Row Mean")
xmedian = aggregate( coalash ~ x, data = coalash, median)
plot(xmedian,pch = 19, main = "Column median versus column index", xlab = "Column Index",
    ylab = "Column median")
ymedian = aggregate( coalash ~ y, data = coalash, median )
plot(ymedian,pch = 19,main = "Row median versus Row index", xlab = "Row Index",
    ylab = "Row Median")



#########################################################
## "SO4 above or below Median", page 49
#########################################################

mediandata = cbind(u,v,y)
par(mfrow = c(2,1))
plot(mediandata[y > median(y),][,1],mediandata[y > median(y),][,2],xlim = c(-140,-60),
    ylim = c(25,50), cex = .5, xlab = "longitutde", ylab = "latitude")
points(mediandata[y <= median(y),][,1],mediandata[y <= median(y),][,2], pch = 19, cex = .5)
legend(-140,35, c("s04dep_above_median","s04dep_below_median"),pch = c(1,19),
    col=c("black","black"), pt.cex = 1, cex = .6)


#########################################################
##"Contour Plot", page 50
#########################################################

library(akima)
interp.so4 = interp(u,v,y)
dev.new(height=7,width=8)
contour(interp.so4, nlevels = 12,labcex = .8)
points(u,v, pch = 19, cex = .6)
```

```
##################################################################
##"Gray Scale Plot", page 51
##################################################################

library(akima)
interp.so4 = interp(u,v,y)
dev.new(height=7,width=8)
image(interp.so4,xlim = c(-126,-66), ylim = c(23,52), col = gray((11:2)/12),
    xlab = "longitude", ylab = "latitude")


##################################################################
## "Lag data plot", page 53
##################################################################

lag1dt = matrix(c  (4,5,2,1,5,5,4,1,5,3,1,2,4,2,1,1),   nrow = 4, ncol = 4)
plot(lag1dt[,-4], lag1dt[,-1],  pch = 19)
lag2dt = matrix(c  (4,1,5,1,1,5,2,4,5,1,3,2,2,4,1,5),   nrow = 4, ncol = 4)
plot(lag2dt[,-4], lag2dt[,-1],  pch = 19)


##################################################################
## "ACF plot", page 56
##################################################################

set.seed(33)
pretend1 = rnorm(100)
acf(pretend1)
```

Additional Plots from Lecture Notes

```
#########################################################
## Plot for page 2 of notes
## Creates a 3D scatter plot of the SO4 data in the United States
#########################################################

library(scatterplot3d)
scatterplot3d(u,v,y, pch = 19,cex.symbols = .75,angle =120, box = F,
    grid = F,type = 'h', xlab = "longitude", ylab = "latitude", zlab = "s04dep")


#########################################################
## Plot for page 3 of the notes
## Plot the numerical responses of the coalash data on an appropriate grid
#########################################################

library(gstat)
data(coalash)
plot(coalash$x, coalash$y, col = "white")
text(coalash$x, coalash$y, label = round(coalash$coalash,1))


#########################################################
## Plot for page 4 of notes
## Based on the atriplex data(matrix of 1's and 0's),
##code to create a black and white plot
#########################################################

atriplex = matrix(scan("atriplex.dt.txt"),nrow = 16,by =T)
par(pty = "s")
plot(1,1,bty = "n", axes = F, type = "n", xlim = c(0,16), ylim = c(0,16),
xlab = "", ylab = "")
plot(1,1,bty = "n", axes = F, type = "n", xlim = c(0,16), ylim = c(0,16),
xlab = "", ylab = "")
for (i in 1:16){
for (j in 1:16){
if (atriplex[i,j] == 1){
polygon(c(i-1,i-1,i,i),c(j-1,j,j,j-1), border = T, density = -1, col = "black")
}}}
segments(0,0,16,0)
segments(0,0,0,16)
segments(0,16,16,16)
segments(16,0,16,16)


#########################################################
## Plot for page 5 of notes
##NC sids gray scale map with freeman Tukey Square root
##transformation adjusted data
#########################################################

library(maps) # for map
library(spdep) # for nc.sids data
library(maptools) # for readShapePoly
data(nc.sids)
nc.sids <- readShapePoly(system.file("etc/shapes/sids.shp", package="spdep")[1],
```

```
  ID="FIPSNO", proj4string=CRS("+proj=longlat +ellps=clrk66"))
FT = sqrt(1000)*(sqrt(nc.sids$SID74/nc.sids$BIR74) + sqrt((nc.sids$SID74+1)/nc.sids$BIR74))
breaks = c (0,2.0,3.0,4.0,7.0)
sidgrps = cut(FT, breaks)
palette(gray.colors(4))
map("county","north carolina", fill = T, col = sidgrps)

##########################################################
## Variogram cloud and boxplot plots for page 58 of notes
##########################################################

library(gstat)
df = data.frame(u = u, v = v,y = y)
g = gstat(formula = y~1,locations = ~v+u, data = df, maxdist = 25)
dev.new(height=4,width=8)
plot(variogram(g, cloud = T), xlim = c(0,20.5), col = "black")

distVsGamma =data.frame(dist = variogram(g, cloud = T)$dist, gamma = variogram(g, cloud = T)$gamma)
cuts = cut(distVsGamma$dist, breaks = 20, dig.lab = 3)
distVsGamma = data.frame(distVsGamma, cuts)
aggdata = aggregate(distVsGamma$dist, by = list(distVsGamma$cuts),FUN = median)
dev.new(height=4,width=8)
boxplot(gamma ~ cuts, data = distVsGamma, names = round(aggdata$x,0))



##########################################################
## Plots for pages 92-100 of notes
## Covariance models
##########################################################
library(geoR)

##########################################################
##Triangular Model, theta1 = 1, theta2 = .5
##########################################################
vector = seq(from = 0, to = .5, by= .001)
plot(vector, 1-(vector/.5),ylim = c(0,1),xlim = c(0,1), xlab = "Distance",
    ylab = "Covariance", main = "Triangular",cex = .5)
vector = seq(from = .5, to = 1, by = .001)
points(vector, rep(0,length(vector)), cex = .5)

##########################################################
## Spherical model, theta1 = 1, theta2 = .5
##########################################################
vector = seq(0,1,by = .001)
spherical = cov.spatial(vector, "spherical", cov.pars = c(1,.5))
plot(vector, spherical, xlab = "Distance", ylab = "Covariance",
    main = "Spherical", cex = .5, pch = 19)

##########################################################
## Exponential model, theta1 = 1, theta2 = .5
##########################################################
vector = seq(0,1,by = .001)
exponential = cov.spatial(vector, "exponential", cov.pars = c(1,.5))
plot(vector, exponential, xlab = "Distance", ylab = "Covariance",
    main = "Exponential", cex = .5, pch = 19)
```

```
#########################################################
## Gaussian model, theta1 = 1, theta2 = .5
#########################################################
vector = seq(0,1,by = .001)
gaussian = cov.spatial(vector, "gaussian", cov.pars = c(1,.5))
plot(vector, gaussian, xlab = "Distance", ylab = "Covariance",
    main = "Gaussian",cex = .5, pch = 19)

#########################################################
## Rational Quadratic Model, theta1 = 1, theta2 = 1
#########################################################
curve(1 - (x^2)/(1+(x^2)/1), ylim = c(0,1), xlim = c(0,5), lwd = 3,xlab = "Distance",
    ylab = "Covariance",main = "Rational Quadratic")

#########################################################
## Cosine Model, theta1 = 1, theta2 = 4*pi
#########################################################
curve(cos(x/(4*pi)), ylim = c(-1,1), xlim = c(0,100), lwd = 3,xlab = "Distance",
   ylab = "Covariance",main = "Cosine")
abline(0,0, lty = 4)

#########################################################
##Wave  Model, theta1 = 1, theta2 = .5
#########################################################
vector = seq(0,10,by = .001)
wave = cov.spatial(vector, "wave", cov.pars = c(1,.5))
plot(vector, wave, xlab = "Distance",
   ylab = "Covariance",main = "Wave", cex = .5, xlim = c(0,10),ylim = c(-1,1), pch = 19)
abline(0,0)

#########################################################
##Matern Model, theta1 = 1, theta2 = .5, kappa = .5, 1.5, 2.5
#########################################################
vector = seq(0,1,by = .001)
matern1= cov.spatial(vector, "matern", cov.pars = c(1,.5), kappa = .5)
matern2= cov.spatial(vector, "matern", cov.pars = c(1,.5), kappa = 1.5)
matern3= cov.spatial(vector, "matern", cov.pars = c(1,.5), kappa = 2.5)
plot(vector, matern1, xlab = "Distance", ylab = "Covariance",main = "Matern",
    cex = .5, pch = 19)
points(vector, matern2, xlab = "Distance", ylab = "Covariance",main = "Matern",
    cex = .5, pch = 19, col = "red")
points(vector, matern3, xlab = "Distance", ylab = "Covariance",main = "Matern",
    cex = .5, pch = 19, col = "blue")
text (.28,.5, expression(paste(kappa, " = .5")))
text(.40, .75,expression(paste(kappa, " = 1.5")))
text(.51, .9, expression(paste(kappa, " = 2.5")))

#########################################################
##Nugget Model, theta1 = 1, theta2 = .5, kappa = .5, 1.5, 2.5
#########################################################
vector = seq(0,1,by = .001)
nugget = cov.spatial(vector, "pure.nugget", cov.pars = c(1,1))
plot(vector, nugget, xlab = "Distance", ylab = "Covariance",main = "Nugget",
    cex = .5, pch = 19)


#########################################################
```

```
## Plots for pages 113-121 of notes
## Semivariogram Models
#########################################################

library(gstat)

#########################################################
##Triangular model, theta1 = 1, theta2 = .5
#########################################################
vector = seq(from = 0, to = .5, by= .001)
plot(vector, vector/.5,cex = .5,ylim = c(0,1),xlim = c(0,1), xlab = "Distance",
    ylab = "Semivariance",main = "Triangular")
vector = seq(from = .5, to = 1, by = .001)
points(vector, rep(1,length(vector)), cex = .5)

#########################################################
##Spherical model, theta1 = 1, theta2 = .5
#########################################################
spherical  = show.vgms(min = 0.0000001, max = 3, n = 1000, sill = 1, range = 1,
    models = "Sph", nugget = 0,
    plot = FALSE,  as.groups = FALSE)
plot(spherical$distance, spherical$semivariance,pch = 19,cex = .5,  xlab = "Distance",
    ylab = "Semivariance", main = "Spherical")

#########################################################
##Exponential model, theta1 = 1, theta2 = .5
#########################################################
exponential = show.vgms(min = 0.0000001, max = 3, n = 1000, sill = 1, range = 1,
    models = "Exp", nugget = 0,
    plot = FALSE,  as.groups = FALSE)
plot(exponential$distance, exponential$semivariance,pch = 19,cex = .5, xlab = "Distance",
    ylab = "Semivariance", main = "Exponential")

#########################################################
##Gaussian model, theta1 = 1, theta2 = .5
#########################################################
gaussian= show.vgms(min = 0.0000001, max = 3, n = 1000, sill = 1, range = 1,
    models = "Gau", nugget = 0,
    plot = FALSE,  as.groups = FALSE)
plot(gaussian$distance, gaussian$semivariance,pch = 19,cex = .5, xlab = "Distance",
    ylab = "Semivariance", main = "Gaussian")

#########################################################
##Cosine model, theta1 = 1, theta2 = 4*pi
#########################################################
curve(1 - cos(x/(4*pi)), ylim = c(0,3),  xlim = c(0,100),  lwd = 3,xlab = "Distance",
    ylab = "Semivariance",main = "Cosine")

#########################################################
##Linear model, theta1 = 1
#########################################################
linear  = show.vgms(min = 0.0000001, max = 3, n = 1000, sill = 1, range = 1,
    models = "Lin", nugget = 0,
    plot = FALSE,  as.groups = FALSE)
plot(linear$distance, linear$semivariance,pch = 19,cex = .5, xlab = "Distance",
    ylab = "Semivariance", main = "Linear")
```

```
##################################################################
##Power model, theta1 = 1, theta2 = .5, 1, 1.5
##################################################################
power = show.vgms(min = 0.0000001, max = 3, n = 1000, sill = 1, range = c(.5,1,1.5),
    models = "Pow", nugget = 0,
    plot = FALSE,  as.groups = FALSE)
plot(power$distance, power$semivariance,pch = 19,cex = .5, xlab = "Distance",
    ylab = "Semivariance", main = "Power ", col = power$model)
text(2,1.25,  expression(paste(theta[2], " = .5")))
text(2,  1.82,expression(paste(theta[2], " = 1")))
text(1.65,  2.5,  expression(paste(theta[2], " = 1.5")))

##################################################################
##Logarithmic model, theta1 = 1
##################################################################
curve(log(x), xlim = c(1,8),ylim = c(0,3), lwd = 3,xlab = "Distance",
    ylab = "Semivariance",main = "Logarithmic")

##################################################################
##Nugget model, theta1 = 1
##################################################################
nugget = show.vgms(min = .0000001, max = 3, n = 1000, sill = 1, range = 1,
        models = "Nug", nugget = 0,
        plot = FALSE, as.groups = FALSE)
plot(nugget$distance, nugget$semivariance, ylim = c(0,1.5),pch = 19, cex = .5,
    xlab = "Distance", ylab = "Semivariance", main = "Nugget")


##################################################################
## Plots on page 150 of notes
## United States S04 map, gray scale map
## 2nd order and 5th order fitted polynomial surfaces of s04 data
##################################################################

par(mfrow = c(2,2))
library(maps)
map("state")
points(so4x,so4y,pch = 19, cex = .5)

library(akima)
interp.so4 = interp(u,v,y)
image(interp.so4,xlim = c(-126,-66), ylim = c(23,52),
 col = gray((11:2)/12), xlab = "longitude", ylab = "latitude")


library(lattice)
df = data.frame(y = y, u = u, v = v)
 fit  = lm(y ~  poly(u,v,degree = 2) , data = df)
data = data.frame(u = u , v = v, z = fit$fitted)
s <- with(data,interp(u,v,z))
 trellis .par.set("axis.line", list (col=NA,lty=1,lwd=1))
plot(with(s,wireframe(z,row.values=x,col.values=y, xlab = "longitude", ylab = "latitude",
zlab = "z",scales = list(arrows = FALSE), aspect = c(61/87, 0.5))), newpage = F, position = c(0,0,.5,.6))


 fit  = lm(y ~  poly(u,v,degree = 5) , data = df)
data = data.frame(u = u , v = v, z = fit$fitted)
```

```
s <- with(data,interp(u,v,z))
 trellis .par.set("axis.line", list (col=NA,lty=1,lwd=1))
plot(with(s,wireframe(z,row.values=x,col.values=y, xlab = "longitude", ylab = "latitude",
 zlab = "z",scales = list(arrows = FALSE), aspect = c(61/87, 0.5))),, newpage = F, position = c(.5,0,1,.6))

##################################################################
## Plot on page 161 of notes
## Estimating the mean function by LOWESS
## Data are sliced by longitude
##################################################################
df = data.frame(y = y, u = u, v = v)
## first order data by longitude
newdata = df[order(u),]

##Then slice data into six equal  slices
NumPoints = nrow(newdata)/6
span1 = newdata[1:NumPoints,]
span2 = newdata[(NumPoints+1):(2*NumPoints),]
span3 = newdata[(2*(NumPoints)+1):(3*NumPoints),]
span4 = newdata[(3*(NumPoints)+1):(4*NumPoints),]
span5 = newdata[(4*(NumPoints)+1):((5*NumPoints)),]
span6 = newdata[(5*(NumPoints)+1):(6*NumPoints),]

##use these slices to  fit  lowess  at  each  slice
par(mfrow= c(3,2))
loess.out = loess(y~v,span = .4,data = span1, control = loess.control(surface = "direct"))
fitted = predict(loess.out, span1, se = TRUE)
plot(loess.smooth(x = span1$v,y = fitted$fit), type = "l", ylim = c(-.5,3.0),  main = "Longitude (-124,-112)",
xlab = "Latitude", ylab = "S04 concentration")
loess.out = loess(y~v,span = .4,data = span2, control = loess.control(surface = "direct"))
fitted = predict(loess.out, span2, se = TRUE)
plot(loess.smooth(x = span2$v,y = fitted$fit), type = "l", ylim = c(-.5,3.0),  main = "Longitude (-112, -105)",
 xlab = "Latitude", ylab = "S04 concentration")
loess.out = loess(y~v,span = .4,data = span3, control = loess.control(surface = "direct"))
fitted = predict(loess.out, span3, se = TRUE)
plot(loess.smooth(x = span3$v,y = fitted$fit), type = "l", ylim = c(-.5,3.0),  main = "Longitude (-105, -93)",
xlab = "Latitude", ylab = "S04 concentration")
loess.out = loess(y~v,span = .4,data = span4, control = loess.control(surface = "direct"))
fitted = predict(loess.out, span4, se = TRUE)
plot(loess.smooth(x = span4$v,y = fitted$fit), type = "l", ylim = c(-.5,3.0),  main = "Longitude (-93, -87)",
xlab = "Latitude", ylab = "S04 concentration")
loess.out = loess(y~v,span = .4,data = span5, control = loess.control(surface = "direct"))
fitted = predict(loess.out, span5, se = TRUE)
plot(loess.smooth(x = span5$v,y = fitted$fit), type = "l", ylim = c(-.5,3.0),  main = "Longitude (-87, -80)",
xlab = "Latitude", ylab = "S04 concentration")
loess.out = loess(y~v,span = .4,data = span6, control = loess.control(surface = "direct"))
fitted = predict(loess.out, span6, se = TRUE)
plot(loess.smooth(x = span6$v,y = fitted$fit), type = "l", ylim = c(-.5,3.0),  main = "Longitude (-80, -68)",
xlab = "Latitude", ylab = "S04 concentration")

##################################################################
## Simulation envelopes using Ghat, page 367 of notes
##################################################################

par(mfrow = c(2,2))
pines = read.table("pines.dt", col.names = c('lat','long'))
lat  = pines$lat
```

```
long = pines$long
pines.ppp = ppp(lat, long, c(0,1), c(0,1))
plot(pines, pch = 19, main = "Pines")
genv = envelope(pines.ppp, fun = Gest, nsim = 99, nrank =1)
plot(genv, main = expression(paste("Simulation envelopes using ",hat(G))),legend = FALSE)


redwoods = read.table("redwoods.dt", col.names = c('lat','long'))
lat = redwoods$lat
long = redwoods$long
redwoods.ppp = ppp(lat, long, c(0,1), c(0,1))
plot(redwoods, pch = 19, main = "Redwoods")
genv = envelope(redwoods.ppp, fun = Gest, nsim = 99, nrank =1)
plot(genv, main = expression(paste("Simulation envelopes using ",hat(G))), legend = FALSE)

##################################################################
## Simulation envelopes using Fhat, page 370 of notes
##################################################################

fenv = envelope(pines.ppp, fun = Fest, nsim = 99, nrank =1)
plot(fenv, main = expression(paste("Simulation envelopes using ",hat(F))),legend = FALSE,width = 3, height = 3)
mtext("Pines")


fenv = envelope(redwoods.ppp, fun = Fest, nsim = 99, nrank =1)
plot(fenv, main = expression(paste("Simulation envelopes using ",hat(F))),legend = FALSE,width = 3, height = 3)
mtext("Redwoods")


##################################################################
## Simulation envelopes using Khat, page 374 of notes
##################################################################
kenv = envelope(pines.ppp, fun = Kest, nsim = 99, nrank =1)
plot(kenv, main = expression(paste("Simulation envelopes using ",hat(K))),legend = FALSE,width = 3, height = 3)
mtext("Pines")



kenv = envelope(redwoods.ppp, fun = Kest, nsim = 99, nrank =1)
plot(fenv, main = expression(paste("Simulation envelopes using ",hat(K))),legend = FALSE,width = 3, height = 3)
mtext("Redwoods")

##################################################################
## Plot on page 442 of notes
##################################################################
hick <- lansing[lansing\$marks=="hickory",]
hick <- as.points(cbind(hick\$x,hick\$y))
maple <- lansing[lansing\$marks=="maple",]
maple <- as.points(cbind(maple\$x,maple\$y))
r <- 1:70/100
poly <- cbind(c(0,0,1,1),c(0,1,1,0))
myK12 <- k12hat(hick,maple,poly,r)
plot(r,sqrt(myK12/pi)-r,xlim=c(0,0.7),ylim=c(-.1,.1))
my12env <- Kenv.tor(hick,maple,poly,99,r)
lines(r,sqrt(my12env\$upper/pi)-r)
lines(r,sqrt(my12env\$lower/pi)-r)
```

```
##########################################################
## Plot on page 448 of notes
##Splancs code within R for implementing a Monte Carlo test (Diggle and Chetwynd's test)
## for random labelling on the Lansing Woods hickory and maple data:
##########################################################
unitsquare <- spoints(c (0,0,1,0,1,1,0,1))
hick.khat <- khat(hick.spp,unitsquare,seq(0,1,.01))
maple.khat <- khat(maple.spp,unitsquare,seq(0,1,.01))
khat.diff <- hick.khat-maple.khat
plot(seq(0,1,.01),khat.diff,xlab="distance",ylab="K\_11 - K\_22",type="l")
diff.lab <- Kenv.label(hick.spp,maple.spp,unitsquare,nsim=99,seq(0,1,.01))
lines(seq(0,1,.01), diff.lab\$upper,lty=2)
lines(seq(0,1,.01), diff.lab\$lower,lty=2)
```