2.4 Using Data in R

Manual of Applied Spatial Ecology

3/11/2022

This code is designed to process data downloaded from Climate Date Online. http://www.ncdc.noaa.gov/cdo-web/ This version looks at weather stations that provide snowfall data in and around Pennsylvania. The code pulls out the desired data from the downloaded aggregate weather-station file and calculates the mean annual snowfall per weather station for 12/1/94 - 3/31/05. The data is then exported to a text file for interpolation in ArcGIS. - Bill Kanapaux, PA Cooperative Fish & Wildlife Research Unit

Last modified: Jan. 13, 2014

1. Exercise 2.4 - Download and extract zip folder into your preferred location
2. Set working directory to the extracted folder in R under Session - Set Working Directory…
3. Now open the script "Weather-Station-Code_Snowfall.Rmd" and run code directly from the script
4. First we need to load the packages needed for the exercise

```
library(adehabitatHR)
library(rgdal)
library(gstat)
library(plyr)
```

5. This code is designed to process data downloaded from NOAA Date. This version looks at weather stations that provide snowfall data in and around Pennsylvania. The code pulls out the desired data from the downloaded aggregate weather-station file and calculates the mean annual snowfall per weather station for 12/1/94 - 3/31/05. The data is then exported to a text file for interpolation in R or ArcGIS. First, we read weather station data - note the use of stringsAsFactors=FALSE and na.strings='-9999'

```
WS <-read.table('Weather_Station_Data-Sep_01Dec1994-31March2005.txt',
  stringsAsFactors=FALSE, na.strings='-9999',header=T)
```

```
#Check data
dim(WS)
head(WS)
summary (WS)
```

```
#Reformat DATE and create Year Month Day columns from NewDate column ###
WS$NewDate <- as.Date(as.character(WS$DATE), format("%Y%m%d"))
WS$Year = as.numeric(format(WS$NewDate, format = "%Y"))
WS$Month = as.numeric(format(WS$NewDate, format = "%m"))
WS$Day = as.numeric(format(WS$NewDate, format = "%d"))
```

6. Make a subset of WS that includes only the months of Dec-March with further manipulation of the data for desired output of project objectives.

```
Winter <- WS[WS$Month %in% c(1,2,3,12), ]
```

```r
#For December, add 1 to Year so that Year matches Jan-March in that season ###
Winter <- within(Winter, Year[Month==12] <- Year[Month==12] +1)
```

```r
#Check subset, including random row to make sure only selected months included ###
dim(Winter)
head(Winter)
Winter[699,]
```

```r
#Create a matrix of unique STATION values (GHCND ) with Lat/Long values for later reference.
### Data contains some multiple versions of individual GHCND coordinates. Only want 1 set per.
PulledCoords <- Winter[!duplicated(Winter[,1]),]
CoordChart <- ddply(PulledCoords, c('STATION'), function(x) c(Lat=x$LATITUDE, Long=x$LONGITUDE))

#Get the number of snowfall records for each STATION for each year and name it RecordTotal.
#Note that NA is omitted from the length count
WinterRecords <- ddply(Winter, .(STATION,Year), summarize, RecordTotal = length(na.omit(SNOW)))

#Get the total amount of snowfall per STATION per year and name it YearlySnow
YearlySnow <- ddply(Winter, .(STATION,Year), summarize, Snow = sum(SNOW, na.rm=TRUE))

#Combine WinterRecords and YearlySnow into one matrix
AllWinters <- cbind(WinterRecords,YearlySnow)
AllWinters <- AllWinters[,-4:-5]

#Only include years that have more than 75% of days recorded
WinterDays <- 121
FullWinters <- AllWinters[AllWinters$RecordTotal/WinterDays > 0.75, ]

#Get the number of years with more than 75% of days recorded for each STATION
WinterYears <- ddply(FullWinters, c('STATION'), function(x) c(TotalYears=length(x$Year)))

#Get the total amount of snow for each station for all years
TotalWinterSnow <- ddply(FullWinters, c('STATION'), function(x) c(TotalWinterSnow=sum(x$Snow)))

#Combine WinterYears and TotalWinterSnow into one matrix
SnowCalc <- cbind(WinterYears,TotalWinterSnow)
SnowCalc <- SnowCalc[,-3]

#Get rid of the stations that don't have at least 10 years recorded at >75% of days ###
Complete.Records <- SnowCalc[SnowCalc$TotalYears > 9, ]

#Calculate average annual snowfall and round to nearest mm
Complete.Records$MeanAnnualSnowfall <- Complete.Records$TotalWinterSnow/Complete.Records$TotalYears
Complete.Records$MeanAnnualSnowfall <- round (Complete.Records$MeanAnnualSnowfall, digits = 0)

#Convert SnowDepth from mm to cm
Complete.Records$MeanAnnualSnowfall <- Complete.Records$MeanAnnualSnowfall/10
head(Complete.Records)
```

```
##             STATION TotalYears TotalWinterSnow MeanAnnualSnowfall
## 1 GHCND:USC00072730         11            3493               31.8
## 3 GHCND:USC00079605         10            4967               49.7
## 4 GHCND:USC00181530         11            7232               65.7
## 5 GHCND:USC00181750         11            4828               43.9
## 7 GHCND:USC00182282         11             7496                 68.1
## 8 GHCND:USC00182336         11             8420                 76.5
```

```
#Add a column to CoordChart showing whether each row matches  a STATION in Complete.Records
#Use "NA" for value if no match, then delete rows with "NA" value.
#Number of rows in CoordChart should now equal number of rows in Complete.Records
CoordChart$match <- match(CoordChart$STATION, Complete.Records$STATION, nomatch=NA)
CoordChart <- na.omit(CoordChart)

#Combine Complete.Records and CoordChart. Make sure each STATION matches in row
#Delete any rows that don't match. Shouldn't be any. If number of rows in Final.Values
#is less than number of rows in CoordChart, there is a problem (but note that # of cols does change).
Final.Values <- cbind(Complete.Records,CoordChart)
Final.Values$match2 <- match(Final.Values[  ,1], Final.Values[ ,5], nomatch=NA)
Final.Values <- na.omit(Final.Values)
dim(Final.Values)
```

```
## [1] 144   9
```

```
dim(CoordChart)
```

```
## [1] 144   4
```

```
#Take out unnecessary rows (2nd STATION, match, and match2) and round MeanSnow to 2 decimal places
Final.Values[,5] <- Final.Values[,8] <- Final.Values[,9] <- NULL
```

7. Make data frame to get rid of lists (in R) so can export to text file to use to load weather station points into ArcGIS and skip to Section 2.5.

```
Final.Values <- as.data.frame(lapply(Final.Values,unlist))
```

```
write.table(Final.Values, "MeanSnowData_95-05.txt", sep="\t", row.names=F)
```

8. Alternatively we can conduct interpolation directly in R using the steps below

```
#Need to convert factors to numeric
Final.Values$Longitude <- as.numeric(as.character(Final.Values$Long))
Final.Values$Latitude <- as.numeric(as.character(Final.Values$Lat))

#Here we need to create Spatial points, attach ID and Date Time sorted
data.xy = Final.Values[c("Longitude","Latitude")]
#Creates class Spatial Points for all locations
xysp <- SpatialPoints(data.xy)
proj4string(xysp) <- CRS("+proj=longlat +ellps=WGS84")

#Creates a Spatial Data Frame from
sppt<-data.frame(xysp)

#Creates a spatial data frame of STATION
ID<-data.frame(Final.Values[1])
#Creates a spatial data frame of Mean Annual Snow Fall
MAS<-data.frame(Final.Values[4])
#Merges ID and Date into the same spatial data frame
merge<-data.frame(ID,MAS)
#Adds ID and Date data frame with locations data frame
coordinates(merge)<-sppt
proj4string(merge) <- CRS("+proj=longlat +ellps=WGS84")
```

```
#Import a county layer for study site and check projections
counties<-readOGR(dsn=".",layer="PACountiesAlbers",verbose = FALSE)
proj4string(counties)
```

```
## [1] "+proj=aea +lat_0=23 +lon_0=-96 +lat_1=29.3 +lat_2=45.3 +x_0=0 +y_0=0 +datum=NAD83 +units=m +no_
```

```
EPSG <- make_EPSG()
PAsp <- subset(EPSG, EPSG$code == "6564")
PAsp
```

```
##      code                             note
## 5266 6564 NAD83(2011) / Pennsylvania South
##
## 5266 +proj=lcc +lat_0=39.3333333333333 +lon_0=-77.75 +lat_1=40.9666666666667 +lat_2=39.9333333333333
##                           prj_method
## 5266 Lambert Conic Conformal (2SP)
```

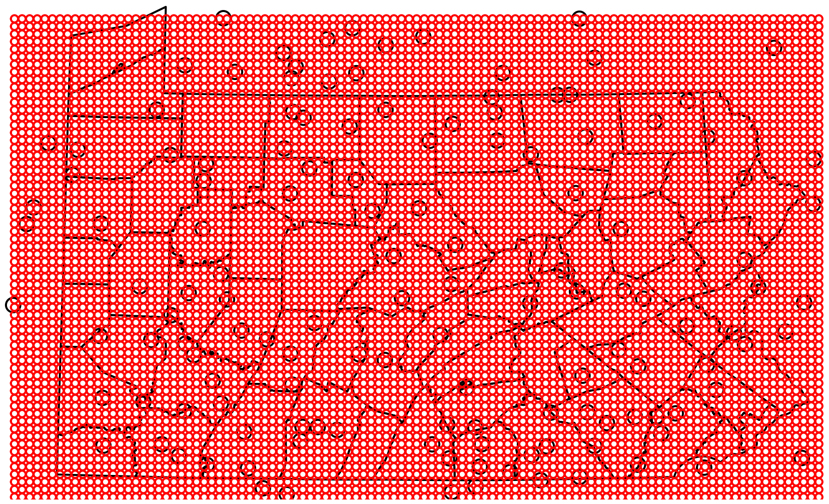
```
#Copy and Paste State Plane projection into code for StatePlane below
#Project Weather Stations and Counties to State Plane
StatePlane <- CRS("+proj=lcc +lat_0=39.3333333333333 +lon_0=-77.75 +lat_1=40.9666666666667 +lat_2=39.93

stations <- spTransform(merge, CRS=StatePlane)
counties <- spTransform(counties, CRS=StatePlane)

stations@data$MAS <- stations@data$MeanAnnualSnowfall
```

```
#Plot out the MAS across the study region
bubble(stations, zcol='MAS', fill=FALSE, do.sqrt=FALSE, maxsize=2, add=T)
```

```
#Create a grid onto which we will interpolate:
xx = spsample(stations, type="regular", cellsize=5000)
class(xx)
```

```
## [1] "SpatialPoints"
## attr(,"package")
## [1] "sp"
```

```
#Plot Weather Stations over counties
plot(counties)
points(stations)
points(xx, bg="red", cex=.5,col="red")
```



```r
#Now expand to a grid with 5000 meter spacing
x.range <- as.integer(range(xx@coords[,1]))
y.range <- as.integer(range(xx@coords[,2]))

grd <- expand.grid(x=seq(from=x.range[1]-50000, to=x.range[2]+50000, by=5000),
                   y=seq(from=y.range[1]-50000, to=y.range[2]+50000, by=5000))

#Convert to SpatialPixel class
coordinates(grd) <- ~ x+y
gridded(grd) <- TRUE
proj4string(grd) <- StatePlane
```

Interpolate grid over sample points directly in R with gstat package

```r
x <- krige(stations@data$MAS~1, stations, grd)
```

```
## [inverse distance weighted interpolation]
```

```r
class(x)
```

```
## [1] "SpatialPixelsDataFrame"
## attr(,"package")
## [1] "sp"
```

```r
image(x)
points(stations, pch=1, col='blue', cex=0.7)
```