Chapter 2

Climate Data Interpolation

Contents

The purpose of this chapter is to bring in climate data from weather stations or aquatic monitoring stations that are often in a text files or from spreadsheet software. These files often may be numerous separate files that you only want to summarize by station, annually, monthly, or some other time period. We are going to show you how to bring these files into R, clean up data, and create your own GIS layer for use in modeling efforts. The example used is from a project that attempted to predict snowshoe hare (*Lepus americanus*) presence or absence in Pennsylvania and determination of how habitats occupied may change due to changes in climate and temperature.

We obtained shapefiles from a variety of sources (see Section 2.1 for details) and weather station data from the National Oceanic and Atmospheric Administration's National Climatic Data Center and cleaned up data outside of R (see Section 2.2 for details). Data were collected from 102 weather stations in and around Pennsylvania in order to determine mean snow depths (SNWD), mean maximum temperatures (TMAX) and mean minimum temperatures (TMIN) for the month of January from 1995 to 2005. Data were used only from stations with records for at least 10 of the 11 Januaries covered by the time range. For each weather station, records for each of the three climate variables were included only if the data covered at least 95 percent of total January days. These criteria resulted in data from 66 stations for snow depth, 69 stations for maximum temperature, and 68 stations for minimum temperature. This data was edited outside of R and the resulting text files were then combined in R (Section 2.3). Alternatively, we can obtain data directly from NOAA (Section 2.4) and clean it up in R prior to moving forward thus eliminating the need for Section 2.3.

We entered spatial coordinates for the weather stations along with climate data that met the selection criteria into program R a geographic information system (Section 2.5; ArcMap 9.3). Data from these stations were then used to create maps that showed the range of mean snow depth, mean maximum temperature, and mean minimum temperature for the month of January across the state. This was accomplished with interpolation using the *kriging* method in ArcMap. The interpolations included data from a total of 46 weather stations outside Pennsylvania in order to avoid errors associated with boundary issues. The interpolated maps were then used to assign the appropriate climate data for each location sampled for presence of snowshoe hare in 2004.

These data were then used to select a model for predicting occupancy probability of

snowshoe hare. Tested models examined occupancy as a function of habitat type and one of four additional correlated variables (elevation, minimum Jan temp, max Jan temp, and Jan snow depth). The model including mean minimum temp (TMIN) was found to be the best one, and the co-efficients for that model, which varied by habitat type, were included as a shapefile (county param) to calculate occupancy probability based on habitat type and TMIN across the range of snowshoe hare counties (Figure 4).

2.1 Incorporating background spatial layers

There is no formal exercise here so no R script. This is just some background information to assist with the remaining exercises in this section

1. PA_Counties and Snowshoe_Counties: Basic PA shapefile with county boundaries and a clip of only the counties with hare harvest reported.
2. Data_TMIN, Data_TMAX, and Data_SNWD: These are the files that result from assigning interpolation values to points. Attribute table for each will have TMIN, TMAX or SNWD values, which are then added to the study's database.
3. Weather_Stations_All: This shapefile is created from the data produced in R, namely the mean January values for TMIN, TMAX, and SNWD for each station from Jan. 1995 to Jan. 2005.
4. Weather_Stations_TMIN, Weather_Stations_TMAX, Weather_Stations_SNWD: These are the files that are used for interpolating weather variables across the state. The Weather_Stations_All shapefile is split into three shapefiles, one each for TMIN, TMAX and SNWD. Not every weather station has data for each of the three weather variables, and 0 values must be taken out in order for the interpolation's to be correct.
5. 2004_Study_Locs: Shapefile contains point data for all locations in 2004 snowshoe hare study. These points are used to assign interpolation values for new shapefiles (Data_TMIN, Data_TMAX, Data_SNWD).
6. pa_krig_tmin, pa_krig_tmax, pa_krig_snwd: These are the interpolation shapefiles, based on Weather_Stations_TMIN, Weather_Stations_TMAX, Weather_Stations_SNWD.
7. counties_hab: Contains the four forest types for the snowshoe hare counties: Coniferous, Mixed, Deciduous, Transitional.
8. county_param: Assigns values to each forest type based on Duane's model. These values will be used to determine occupancy probability when the model is plugged into map algebra.

2.2 Accessing climate data

There is no formal exercise here so no R script. All files downloaded in this section are available for Exercise 2.3 so only accessing climate data for practice or to download data specific to your study area and objectives

1. Let's find some weather data and explore the format available by copying the NOAA data link into a web browser `http://www.ncdc.noaa.gov/cdo-web/` or just select the link here: NOAA Data

2. Select the Mapping Tool Application box then All Maps tab, then choose "GHCN-Daily" all in a separate window.

3. Use polygon to select a defined area, check "GHCND" under "GHCN Daily", and scroll/zoom with the Polygon tool that allows for selecting areas outside PA boundaries for inclusion. This is done in order to create a better interpolation for SnowDepth, TMIN TMAX

4. Selection will bring up all weather stations in area. Look under "Period of Record to determine whether it has data for Jan. 1 1995 Jan. 31 2005 If 1 year out of the 11 is missing, save it. Greater than 1, skip it. Select all that match dates and click "Get Selected Data"

5. On next page, make "Output date range" as 01 Jan 1995 to 01 Jan 2005 and select the "Custom GHCN-Daily CSV" box and select continue.
   Then select:
   SnowDepth (under Precipitation)
   TMIN (under Air Temperature)
   TMAX (under Air Temperature)
   Be sure to include Include Station Name and Geographic Location (Lat Long Elev)

6. Wait for email with link then click and Save file as .csv

   Below are specific instructions when retrieving data for a specific station with known GHCND code:

   1. Go to NOAA Climate Data Online NOAA Data
   2. Select data search Enter GHCND code (but don't include "GHCND:" portion of code in search term) Select "Daily GHCND" as data set
   3. If dates match desired range (see 4), select station and hit continue If not, remain on page and enter new GHCND code in search box 4. Start date for this project: Jan. 1, 1995 End date: Jan. 31, 2005 5. Select "Custom GHCN Daily CSV" format option, hit continue
   6. Under Precipitation, select PRCP, SNOW, SNWD
   7. If have Air Temperature, select TMAX, TMIN
   8. Select Station Name in "Additional Output Options" and hit continue
   9. You will receive two emails - one confirming order and one with link to data
   10. Click link, save data as .csv file with Station Name as file name
   11. Click on Data Search, repeat process until have data for all weather stations

2.3 Cleaning raw climate data

1. Exercise 2.3 - Download and extract zip folder into your preferred location

2. Set working directory to the extracted folder in R under File - Change dir...

3. No packages are needed for this exercise, these are base R functions

4. Now open the script "Read_FilesScript.R" and run code directly from the script

5. For each csv file, save as Excel Worsheet 1997-2003 if importing to NCSS or keep in csv or txt if using R

6. Take out all non-Jan months for every year

7. Files will need to meets the following criteria but will be addressed in Exercise 2.4:

```
Each  weather station must have records for at least 10 of the 11 Januaries
Each weather station must have at least 95% of daily records for those Januaries
This means at least 325 days for 11 seasons and 295 days for 10 seasons

Snow Depth (SNWD) 66 stations
Maximum temp (TMAX) 69 stations
Minimum temp (TMIN) 68 stations
```

8. The code that follows should have all files in the same folder but not the R script or any R files or code will not run. The code below brings in each text file and summarizes the data for each weather station as instructed in the code.

```
# Vector of files names in working directory
files <- list.files(pattern = ".txt")

# Total number of files in working directory (for loop below)
n.files <- length(files)

# Container to hold text files
files.list <- list()

# (populate the container files.list with weather data sets
files.list <- lapply(files, read.table, header =T, sep="\t")

# Set up matrix for weather station summary  data
m1 <- matrix(NA,ncol=8,nrow=n.files)

# Loop for running through all weather station files
for(i in 1:n.files){

    #Assign elevation
        m1[i,1] <- files.list[[i]][1,10]

    #Assign Lat
        m1[i,2] <- files.list[[i]][1,11]

    #Assign Long
        m1[i,3] <- files.list[[i]][1,12]

    #Calculate mean snow depth
        SNWD_mm <- mean(files.list[[i]][,7],na.rm=T)

    #Convert snow depth mean to inches
        SNWD_in <- SNWD_mm/25.4

    #Assign snow depth
        m1[i,4] <- SNWD_in

    #Calculate mean maximum temp
        TMAX_C <- mean(files.list[[i]][,8],na.rm=T)

    #Convert max temp to F
```

```
        TMAX_F <- TMAX_C*0.18 + 32

    #Assign max temp
        m1[i,5] <- TMAX_F

    #Calculate mean minimum temp
        TMIN_C <- mean(files.list[[i]][,9],na.rm=T)

    #Convert min temp to F
        TMIN_F <- TMIN_C*0.18 + 32

    #Assign min temp
        m1[i,6] <- TMIN_F

    #Reassign GHCN number
        GHCN <- toString(files.list[[i]][1,1])

    #Assign Station Name
         m1[i,7] <- GHCN

    #Reassign Station Name
         SN <- toString(files.list[[i]][1,2])

   #Assign Station Name
         m1[i,8] <- SN
}

colnames(m1) <- c("Elevation","Lat","Long","SNWD","TMAX","TMIN","Station")
write.csv(m1,paste(".","\\output.csv",sep=""))

#Removes quotes from output table below
m1 <-noquote(m1)

#Results of code that summarizes the 6 weather stations
m1
    Elevation  Lat       Long    SNWD    TMAX   TMIN   GHCN       Station
[1,] 520       42.249  -77.758  15.558  31.969 13.299 USC00300085 ALFRED
[2,] 457.2     42.1    -78.749   7.698  30.175 14.466 USC00300093 ALLEGANYSP
[3,] 452       42.303  -78.018   5.293  30.872 11.687 USC00300183 ANGELICA
[4,] 341.4     42.348  -77.347   4.122  32.247 13.549 USC00300448 BATH
[5,] 80.2      40.833  -75.083   NaN    36.457 19.621 USC00280734 BELVIDERE
```

2.4 Using data in R

1. Exercise 2.4 - Download and extract zip folder into your preferred location
2. Set working directory to the extracted folder in R under File - Change dir...
3. First we need to load the packages needed for the exercise

```
library(adehabitatHR)
```

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

4. Now open the script "Weather-Station-Code_Snowfall.R" and run code directly from the script

5. This code is designed to process data downloaded from NOAA Data This version looks at weather stations that provide snowfall data in and around Pennyslvania. 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.

```
### remove other objects from the working session ###
rm(list=ls())
```

6. 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"))

head(WS)
```

7. 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), ]

#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)

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

#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]),]
head(PulledCoords)
dim(PulledCoords)
```

```
CoordChart <- ddply(PulledCoords, c('STATION'), function(x) c(Lat=x$LATITUDE,
     Long=x$LONGITUDE))
head(CoordChart)

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

#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))
head(YearlySnow)
tail(YearlySnow)
dim(YearlySnow)

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

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

#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)))
head(WinterYears)
tail(WinterYears)
dim(WinterYears)

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

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

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

#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)
head(Complete.Records)

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

#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)
head(CoordChart)
dim(CoordChart)
dim(Complete.Records)

#Combine Complete.Records and CoordChart. Make sure each STATION matches in row
#Delete any rows that don't match. Shouldn't be any. If # of rows in Final.Values
#is less than number of rows in CoordChart, there is a problem (but note that
#number 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)
dim(CoordChart)

head(Final.Values)

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

8. 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)
```

9. 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")
proj4string(counties)
plot(counties)

#Project Weather Stations to match Counties
Albers.crs <-CRS("+proj=aea +lat_1=29.3 +lat_2=45.3 +lat_0=23 +lon_0=-96
     +x_0=0 +y_0=0 +datum=NAD83 +units=m +no_defs +ellps=GRS80 +towgs84=0,0,0")
stations <- spTransform(merge, CRS=Albers.crs)

#Plot Weather Stations over counties
points(stations)

stations@data$MAS <- stations@data$MeanAnnualSnowfall
str(stations)

str(counties)

## create a grid onto which we will interpolate:
xx = spsample(counties, type="regular", cellsize=10000)
class(xx)

points(xx, bg="red", cex=.5,col="red")

#Convert to a SpatialPixels class
gridded(xx) <- TRUE
class(xx)

plot(xx)
```

```
points(stations, bg="red", cex=.5,col="red")

#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:
## first get the range in data
 x.range <- as.integer(range(stations@coords[,1]))
 y.range <- as.integer(range(stations@coords[,2]))

## now expand to a grid with 500 meter spacing:
grd <- expand.grid(x=seq(from=x.range[1], to=x.range[2], by=5000),
     y=seq(from=y.range[1], to=y.range[2], by=5000) )

## convert to SpatialPixel class
 coordinates(grd) <- ~ x+y
 gridded(grd) <- TRUE

## test it out:
plot(grd, cex=0.5)
points(stations, pch=1, col='red', cex=0.7)
title("Interpolation Grid and Sample Points")

x <- krige(stations@data$MAS~1, stations, xx)
class(x)
image(x)
points(stations, pch=1, col='blue', cex=0.7)
```

2.5 Importing dynamically downscaled global climate data

This exercise will provide some code for manipulating climate change data from the Regional Climate Downscaling by copy the link into your browser: `http://regclim.coas.oregonstate.edu/data-access/index.html` or just select the link here: Regional Climate Downscaling. IMPORTANT: For each climate projection, must change name in first command and file name in last command.

1. Exercise 2.5 - Download and extract zip folder into your preferred location
2. Set working directory to the extracted folder in R under File - Change dir...
3. First we need to load the packages needed for the exercise

   ```
   library(ncdf4)
   ```

4. Now open the script "NetCDF_Script.R" and run code directly from the script
5. Open netCDF and setting verbose=true provides details about the data in the netcdf file including the varid. You need to know the varid to select the variable you want to extract/summarize. Note: the dimensions x, y, time also get a varid so you will need to subtract 3 from the varid of interest to get the correct one.

   ```
   dat <- nc_openf("Monthly_AvgMinTemp_1995-99_MPI.nc", write=TRUE,
   ```

```
     readunlim=TRUE, verbose=TRUE)

# Read data this loads all the data from the downloaded variable into the
# tmin object
tmin <- dat$var[[1]]
tmin

#####################################
# The following illustrates how to read the data
#####################################
print(paste(tmin$name))  #in this case the 'field name' is TAMIN

# Grab data for TAMIN variable and place in object df1
df1 <- ncvar_get(dat, tmin)

head(df1, n = 10L) # head(x, n = 6L, ...); head returns the first data  entries,
#x is the object, n sets the
# number of entries displayed. tail returns  the last of the data entries

# Dimensions of df1 (x, y, time)
dim(df1)

# Dimensions can also be examined one at a time
dim(df1)[1] # number of x grids (36)
dim(df1)[2] # number of y grids (21)
dim(df1)[3] # number of months in file (49)
# NOTE: FILE INCLUDES MONTHS OTHER THAN JANUARY (Jans are 1,13,25,37,49)

# Check first element
df1[1,1,1]

# Check first January for all x,y
df1[,,1]

# Create a new matrix which is monthly averages for each grid cell. Make the new
#matix the same size (i.e. same number of rows and columns as there are in the
#dataframe df1
sum1 <- array(data=NA, c(dim(df1)[1],dim(df1)[2] ))
dim(sum1)

# Create January mean TAMIN for each x-y coordinate
for(i in 1:dim(df1)[1]){ # loop over x-coords
for(j in 1:dim(df1)[2]){ # loop over y-coords
sum1[i, j] <- (df1[i,j,1]+df1[i,j,13]+df1[i,j,25]+df1[i,j,37]+df1[i,j,49])/5
}
}

# head(sum1)  ## useful for large files
sum1

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

```
############################################################
# Create netcdf file from sum1 (contains matrix of new data)
############################################################
# Get x and y coordinates from original "dat" ncdf file
x  = ncvar_get(nc=dat,varid="x")
y  = ncvar_get(nc=dat,varid="y")

# Check dimensions
length(x)
length(y)
dim(sum1)

## define the netcdf coordinate variables - note that these are coming
#from the dat file with actual values
dim1 = ncdim_def( "X","meters", as.double(x))
dim2 = ncdim_def( "Y","meters", as.double(y))

## define the EMPTY (climate) netcdf variable and define names that will
#be used in the var.def.ncdf function
# Define climate variable names
new.name <- 'mintemp'
# Define units of measurement for variable
units <- 'degreesC'
# Define long name for variable
long.name <- 'Jan average min temperature'

varz = ncvar_def(new.name,units, list(dim1,dim2), -1,
          longname=long.name)

#associate the netcdf variable with a netcdf file
#put the variable into the file, and close

nc.ex = nc_create("MPI1999-95.nc", varz )
ncvar_put(nc.ex, varz, sum1)
nc_close(nc.ex)
```