ES 207 Spring 2016

Homework 7 Time & Space:

Updated March, 2016

Table of Contents

Description

Time and space remain at the crux of environmental data analysis. In his 1989 address to the *Ecological Society of America* and upon receiving the society's highest award, Si Levin (1992) stated:

> To address such phenomena, we must find ways to quantify patterns of variability in space and time, to understand how patterns change with scale (e.g., Steele 1978, 1989, Dagan 1986), and to understand the causes and consequences of pattern (Levin 1989, Wiens 1989). This is a daunting task that must involve remote sensing, spatial statistics, and other methods to quantify pattern at broad scales; theoretical work to suggest mechanisms and explore relationships; and experimental work, carried out both at fine scales and through whole-system manipulations, to test hypotheses. Together, these can provide insights as to how information is transferred across scales, and hence how to simplify and aggregate models.

This homework assignment is intended to get you comfortable working with both temporal and spatial dimensions. While this effort will only be scratching the surface, think about aggregation along the two primary axes of the environment: time and space.

Homework Exercise (4 Points total)

These exercises rely on lab data found in the files section of CATCOURSES.

Your document should have the following sections, and provide written explanations formatted in RMarkdown that explains your code, output and graphics in the following format:

NAME
CLASS
DATE

Homework Assignment 7

Objective Statement: [What are you trying to accomplish?]

Methods: [In general terms, what analyses are you doing?]
Data: [What are the data and where did they come from?]
Code: [In specific terms, what is the code that was used to conduct the analysis?]

Results: [What do the results show? Numerical evidence and graphic evidence are required.]

Discussion: [What do the results mean?]

Limitations: [What are the limitations, caveats, and assumptions of the analysis?]

OBJECTIVE

In November 2014, voters were asked to pass a $7Bn bond to better secure California's water future such that times of scarcity (e.g., drought) could be better balanced by times of abundance (e.g., floods). As a new water resources manager for the California Department of Water Resources, you have been asked to explore the temporal and spatial dynamics of discharge in the Cosumnes River, the last major undammed river of the Sierra Nevada. One idea that has been floated by your boss is to examine the possibility of localized recharge of the groundwater aquifer by purposeful flooding of the floodplain. This approach could have the added benefit of promoting fish growth (as suggested by Jeffres et al. 2008) via primary productivity dynamics (Ahearn et al. 2006). One concern is that with the recent drought, it might not flood often enough to be worth the expense of constructing the set-back levees. Thus the first step in your analysis is to provide exploratory data analysis of the time series data of discharge at the USGS gaging station located at Michigan Bar (MHB), with a final analysis of determining what percentage of years it floods more than 100 days (a flood is defined as discharges >= 800 cubic feet per second). If at least ⅓ of all years flood for this period of time, localized aquifers will benefit. Your initial report may help DWR determine in setback levees are a good use of Prop 1. funds.

Step 1 - Data Retrieval and troubleshooting

The California Data Exchange Center (CDEC) hosts several parameters related to real-time river conditions across California. For example, a list of river stage monitors hosted on CDEC can be found below at the following link:

http://cdec.water.ca.gov/misc/stages.html

Historical data from individual stations can be aggregated into a `.csv` file and downloaded through the following query tool, provided that the user knows the Station ID and the Sensor Number.

http://cdec.water.ca.gov/cgi-progs/queryCSV

The sensor number can be obtained from station metadata. The following link lists station metadata for the USGS-operated gage on the Cosumnes River (at Michigan Bar):

http://cdec.water.ca.gov/cgi-progs/selectQuery?station_id=MHB&sensor_num=&dur_code=D&start_date=02%2F29%2F2016+00%3A00&end_date=now

A successful query will yield a URL similar to the following:

http://cdec.water.ca.gov/cgi-progs/queryCSV?station_id=MHB&dur_code=D&sensor_num=41&start_date=2010/10/01&end_date=2015/09/30

Notice the structure of the URL. Can you identify the Station ID, Sensor Number, starting and ending dates? For this system, the query metadata is incorporated into the request URL. We can change these values manually to construct arbitrary requests without using the web-form interface.

Since the `read.table` function in R can accept a URL as an input, we create a function that programmatically creates valid “query URLs” according to our station, sensor, and date ranges of interest.

A function `getCDEC()` has been provided that accomplishes the above objective. You can run it by sourcing the `getCDEC.R` file in RStudio and calling the function as follows:

```
get.CDEC("MHB","D","41","2011/10/01","2012/09/30")
```

However, there are a couple of things wrong with the function. First, according to the instructions on `Line 10` of the function, the user should format the start/end dates as `YYYY-MM-DD`. However, as you may have noticed, we have to use the `YYYY/MM/DD` format, since that matches the date format used in the query URLs.

Regardless, the function seems to work properly until you attempt to instruct the function to save a `.csv`. Later in the function (`Line 57`), the date strings are used to create the name of the `.csv` file saved to disk. Since, filenames cannot contain slash characters, the `write.csv` function fails. There are many ways to fix this—one method uses a combination of `strptime` and `as.Character` to re-format the date strings into a form that is appropriate for filenames. The fixed function is available in `getCDEC_fixed.R`. Please note the changes on Lines 52-54 and become familiar with the `strptime` function.

The gage at Michigan Bar is operated by USGS. While CDEC records go back to 1993, USGS hosts records back to 1907 at the National Water Information System. You can find daily discharge records from the 1908-2015 water years at the following link:

http://waterdata.usgs.gov/nwis/dv?cb_00060=on&format=rdb&site_no=11335000&referred_module=sw&period=&begin_date=1907-10-01&end_date=2015-09-30

Do you notice similar metadata in the USGS URL?

Make your own function similar to `get.CDEC()` that obtains water data from USGS using `read.table`, and assigns the data it to your local workspace. The function should be named `get.dailyUSGS()` and should take 4 input arguments:

1. The site number (11335000 for Michigan Bar)
2. The sensor parameter (00060 for discharge)
3. The start date
4. The end date

The function should be very short, and should only require two commands.

Now, we will need to change the formatting of your data table.

Step 2 - Formatting and EDA

Notice the header for the data uses 2 lines. The `read.table` function does not compensate for this, but instead reads in the second header line as if it were a data row (and thus the first row in our dataframe). Let us remove this using the following (where `waterdat` is the data we loaded earlier):

```
waterdat = droplevels(waterdat[-1,]) #remove first row
```

(Think about why `droplevels` is used here - even though we removed the first row, it still had an influence on the summary of the data)

View the structure of the data frame and notice the column names (relate these to what you read in the metadata/comment section of the dataset). Rename the last 2 columns to something more sensible:

```
colnames(waterdat)[4:5] = c("discharge_cfs","dat_qvalue")
```

All data were read in as “Factors”. Convert the “datetime” column to `POSIXct` (A date/time class) and the now “discharge_cfs” column to numeric values:

```
waterdat$datetime = as.POSIXct(waterdat$datetime, tz = "PST")
#you must convert a factor to a character, then to double...
waterdat$discharge_cfs = as.double(as.character(waterdat$discharge_cfs))
```

Note: the metadata mentioned the location of this dataset, which time zone should you use? Timezone is set by the `tz` parameter. You might not even need to specify the `tz` parameter.

Create a new column called “discharge_afd” that represents “(Acre feet) per day” calculated from the “discharge_cfs” column (research the conversion). Run an EDA to make sure value ranges make sense.

(Advanced users can incorporate these formatting steps into their `get.dailyUSGS()` function, but beware of special cases. Different USGS gage parameters have different column names and types. If your modified function is only intended for use on certain gage parameters, be sure to stipulate that in the function comments.)

Step 3 - Time Series using Cosumnes River discharge

Create year, month and day columns and populate them appropriately using the datetime column. “Water Year” is the nomenclature that hydrologists use to designate the period of October 1 – September 30 of a given year. So the 2016 Water Year started October 1, 2015, for example. This is helpful when winter runoff is the predominant time of year for flow events.

```
waterdat$year = as.numeric(substr(waterdat$datetime,1,4))
waterdat$month = as.numeric(substr(waterdat$datetime,6,7))
waterdat$day = as.numeric(substr(waterdat$datetime,10,11))
```

```
#water year
waterdat$wy = ifelse(waterdat$month > 9, waterdat$year + 1,
waterdat$year)
waterdat$wym = ifelse(waterdat$month > 9, waterdat$month - 9,
waterdat$month + 3)
```

Remove leap days (that is, those years that are considered “leap years”, remove that extra day) by subsetting using the following condition:

```
(month != 2) && (day != 29)
```

Create a time series for the total, mean and maximum discharge by month using the `ts` function (`?ts()`).

```
#Total discharge by year (using sum)
waterdat.mo.afs.sum = ts(as.vector(tapply(waterdat$discharge_afd,
list(waterdat$wym, waterdat$wy), sum)), frequency=12)
```

Do the same for monthly average of daily discharge (cfs) and monthly maximum of daily discharge (cfs). Plot the monthly hydrograph results for the following years:

1) the driest year on record
2) the wettest year on record
3) the year with the highest daily average discharge

Examine the temporal autocorrelation in the monthly discharge using the following `ts()` functions for both raw (untransformed) data and `log10()` transformed data:
`acf(ts)`, `plot(stl(ts))`
What is the seasonal autocorrelation structure?

Step 4 - Calculate Floods

Calculate the frequency of days that exceed a daily average discharge of 800 cfs (cubic feet per second) for each year and calculate the percentage of the historical record that is greater than or equal to 100 days of flooding.

For the initial EDA, plot the entire time series and add a horizontal line at 800 cfs flooding threshold (e.g., `abline(thresh, col = "red")`)

Summarize the total number of flood events per year, here are some potential code snippets:

```
# note, this code is incomplete
mhb$flood = mhb$cfs >= 800
mhb.yr.flood.sum$fldcnt = ts(as.vector(tapply(mhb$flood,
```

```
list(mhb$wy), sum)), start=1907)

sum(mhb.yr.flood.sum$fldcnt >= 100)
```

Resources

Data

- http://waterdata.usgs.gov/nwis/dv?cb_00060=on&format=rdb&site_no=11335000&referred_module=sw&period=&begin_date=1907-10-01&end_date=2015-09-30
- `getCDEC.R`
- `getCDEC_fixed.R`

Keywords

`read.table, droplevels, as.numeric, ifelse, acf, ts`