NTM Workshop

University of Queensland, Australia

Day 2

Schedule:

8:30 – 10:30	Introduction to Weather Source, merging disease/climate data.
10:30 – 11:00	Morning tea– GPH catering
11:00 – 12:30	Workshop: Exploratory data analysis (EDA).
12:30 – 13:30	Lunch – GPH catering
13:30 – 15:30	Workshop: Introduction to spatiotemporal models.
15:30 – 16:00	Afternoon tea – GPH catering
16:00	Workshop: Daily review/catch up on exercises. End of day 2

Introduction to Weather Source, LLC. data

R exercises:

1. Download the exercise files in Box onto your laptop. Unzip the files if necessary (R does not like zipped/compressed files).

1a. Why are there multiple historical meteorology files? Open a few, or get a clue from the file names.

1b. Load the files into R. You will want to start with the historic meteorology files:

Merge the historical meteorology files into one:

- Tell R to list the file names that match the meteorology file naming convention (the ones with "northamerica_" in the file name)

  file_pattern <- paste("northamerica-20", 10:22, sep = "") #give R the pattern

  file_names <- list.files(pattern = file_pattern[ii]) #Tell R to list files that match

- Use **lapply()** to read the .csv files and bind the rows together

  met<- lapply(file_names, read.csv) %>% bind_rows()

The **%>%** operator is called a "pipe" operator, and using it is termed "piping." The code above could be read as "the object **met** is assigned to a list of files that are being read in a loop by **read.csv()** and then piped to a **bind_rows()** function."

Essentially, the code is looping through the list of files in **file_names**, reading the files using **read.csv()** and then binding the files together into one object called **met**.

The other files for different data sets are complete: they have all the data for every year combined already. You can load these individually using **read.csv()**

Example: **air <- read.csv("cams_reanalysis_airquality_monthly.csv")**

Running into trouble? Make sure you have set your R working directory to the same location as where your files are located.

2. Explore the data, using **head()**, **tail()**, **summary()**, **str()**, and, if you want, **view()**

You should have three data sets to explore: historical meteorology, air quality, and land use.

*The rest of the exercise code is written with the following object names assigned to these data sets: **met** for historical meteorology, **air** for air quality, and **land** for land use. You can call them whatever you want for your own code, but will have to also change any code you copy/paste from this file.*

3. Merge the data files into one.

3a. What variables should you merge by?

3b. Remember the function **left_join()**. I recommend using it for this task. First, take a look at the data. Which dataset should be your foundational data set (ie: the one you merge other data to)? If using **left_join()**, the foundational data set should be listed first (to the left).

3c. Did any errors or messages pop up? Is the resulting data frame what you would expect? Take a look at the land use variables.

4. You may notice that some of the data is in daily format, while other data is in monthly or yearly formats.

4a. Consider aggregating daily data to monthly format. For this, you may need to create a month-year variable by which to aggregate:

```
met$year_month <- paste(substr(met$date_valid_std, 6,7), substr(met$date_valid_std, 1, 4),sep = "/")
```

What is the code above doing? Take a look at **?paste** and **?substr** to view the help files for these functions.

4b. You can aggregate multiple columns of data using the **aggregate()** command. Type **?aggregate** into the console and view the help file for the function.

4c. View a summary of the data stored in **met**. How should we aggregate the variables? For some, such as temperature or humidity, we may want to take the mean. For others, such as precipitation, we may want the sum.

4d. Aggregate the data. The example below takes the column indices (or column number) of variables that I will aggregate to monthly averages. I list these variables within the **c()** function, so **aggregate()** knows I want all these variables to be aggregated.

The arguments to aggregate() are

1) the variables to be aggregated,

2) a list with the variables you want the data aggregated *by*, and

3) the function, such as mean or sum, that you want to aggregate with.

met_means <- aggregate(met[,c(3:4,7:51,55:61)], list(met$fips, met$year_month), mean)

The above code instructs R to aggregate columns 3, 4, 7 through 51, and 55 through 61 according to the fips code and year_month variables. It asks R to do this by calculating the mean in each fips code for each year_month.

For variables like precipitation, you can run a similar function, using sum instead of mean. Once you have **met_means** and **met_sums** defined, you can merge them together.

4e. Did you get any errors or messages? Does the result look like it makes sense?

Advanced: This is just one way to aggregate the data. You may choose to instead aggregate each data type separately, before merging the data.

Advanced exercise: Start at exercise 1 and write code that reads, combines, and aggregates historical meteorological data by fips and year/month all at once. You can do this by **defining a function** or running a **for loop**. Whatever works for you!