ESP106-Lab4

Kyra Liu

Lab 4

In this lab we will look at daily tidy data downloaded from NOAA's Tides and Currents API (Application Programming Interface) for six cities around the US. I used the API to obtain six csv files containing data for tide gauges in each city. The tide gauges have numerical codes that correspond to the city as follows:

1. Boston: 8443970
2. New York: 8518750
3. Baltimore: 8574680
4. Charleston: 8665530
5. Miami: 8723214
6. Corpus Christi: 8775296

Part 1

1. Creating a data frame containing data on the city name and tide gauge ID

```
city.td = data.frame(c("Boston","New York","Baltimore","Charleston","Miami","Corpus Christi"),c(8443970
colnames(city.td) = c("city","tide.id") #renaming columns
```

2a. Using a for-loop to read in the csv files and bind them together into a single data frame. Add a column to the data frame giving the name of the city the data is from.

```
datafile = unzip("ESP106_week4_data.zip") #unzipping file


cities.f = list() #create empty list
c = 1 #initialize counter as 1
for (file in datafile){

#adding city name and reading in file as cc
cc = cbind(read.csv(datafile[c]),city.td$city[c])

  cities.f[[file]] = cc
  #file element of list is equal to cc
  c = c+1 #counter for indexing, increase by 1 with each iteration
}

c.final = do.call(rbind,cities.f) #combining the elements of the list into df
names(c.final)[ncol(c.final)] <- "city" #renaming final column as city
```

modified 2a for stretch challenge

```
#note that eval is false because data will pull from
#noaa source rather than downloaded files
dir.create("tidedatafiles") #creates directory where files will go when downloaded
# for-loop for pulling data from noaa
for (i in 1:nrow(city.td)){ #iterate for each city
  gaugeid = city.td$tide.id[i] #creating gauge id that will go into url with
                               #each iteration
  #creating url using gauge id variable and noaa url parameters
  #note it requests to pull all data between 01/01/2011 and end 01/01/2023

   id_url = paste0("https://api.tidesandcurrents.noaa.gov/api/prod/datagetter?begin_date=20110101&end_da

#downloading csv file to created directory, named with gauge id
download.file(id_url,destfile = paste0("tidedatafiles/", gaugeid,".csv"))
}

fi = list.files("tidedatafiles") #names of files in created directory
cities.f = list() #setting empty list
c = 1 #counter for indexing
for (file in fi){

#adding city name and reading in file as cc
cc = cbind(read.csv(paste0("tidedatafiles/",fi[c])),city.td$city[c])

  cities.f[[file]] = cc
  #file element of list is equal to cc
  c = c+1 #counter for indexing, increase by 1 with each iteration
}

c.final = do.call(rbind,cities.f) #combining the elements of the list into df
names(c.final)[ncol(c.final)] <- "city" #renaming final column as city
```

2b. Take a look at your data frame - is this in a tidy format?

```
head(c.final)
```

```
##                  Year Month Highest  MHHW    MHW    MSL    MTL    MLW   MLLW
## ./8443970.csv.1 2011     1   0.844 0.161 -0.003 -1.450 -1.485 -2.967 -3.070
## ./8443970.csv.2 2011     2   0.686 0.087 -0.054 -1.483 -1.517 -2.981 -3.049
## ./8443970.csv.3 2011     3   0.695 0.035 -0.083 -1.501 -1.539 -2.996 -3.062
## ./8443970.csv.4 2011     4   0.658 0.079 -0.034 -1.456 -1.496 -2.958 -3.042
## ./8443970.csv.5 2011     5   0.719 0.162  0.030 -1.372 -1.409 -2.849 -2.962
## ./8443970.csv.6 2011     6   0.659 0.191  0.045 -1.357 -1.395 -2.836 -2.936
##                    DTL    GT    MN   DHQ   DLQ  HWI  LWI Lowest Inferred   city
## ./8443970.csv.1 -1.454 3.231 2.964 0.164 0.103 3.63 9.85 -3.671        0 Boston
## ./8443970.csv.2 -1.481 3.136 2.927 0.141 0.068 3.59 9.78 -3.703        0 Boston
## ./8443970.csv.3 -1.513 3.096 2.913 0.117 0.066 3.63 9.85 -3.749        0 Boston
## ./8443970.csv.4 -1.481 3.121 2.924 0.113 0.084 3.66 9.85 -3.761        0 Boston
## ./8443970.csv.5 -1.400 3.124 2.879 0.132 0.113 3.68 9.89 -3.549        0 Boston
## ./8443970.csv.6 -1.372 3.127 2.881 0.146 0.100 3.72 9.90 -3.322        0 Boston
```

yes, data is tidy

We are going to examine the question of whether these gauges show evidence of rising sea levels. One of the first things we have to deal with is the issue of dates.

Your data frame right now has one column with a year and one with the month. We are going to combine these into a single column, and use as.Date to formally use Date objects

3a. Changing date column to date object

```
# pasting year and month with 01
cit.dt = paste0(c.final$Year,"-",c.final$Month,"-01")
#combining data frame with new column
c.final = cbind(cit.dt, c.final)
names(c.final)[1] = "Date" #renaming column as date
```

3b. Use as.Date to convert your new date column to a date object in R

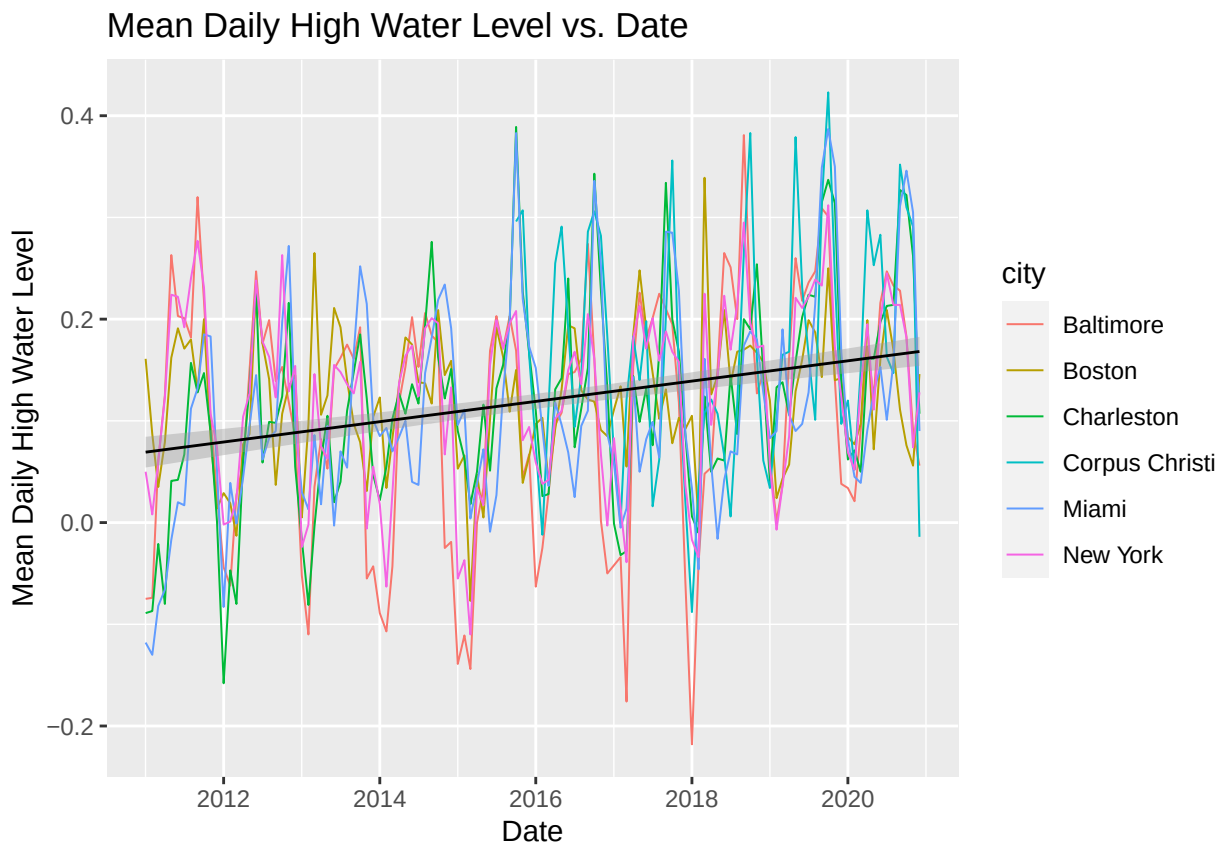
```
c.final$Date = as.Date(c.final$Date) # converting column to date object
```

4. plot showing data from all 6 gauges on the same plot.

- Plot the date on the x axis and MHHW (mean higher high water - i.e. the average daily high water level) on the y axis Make sure to add proper axis labels and units (using +labs(x=““,y=””))
- Add a single best-fit line through the full data set using geom_smooth(method=“lm”) - note that by default ggplot will fit one best fit line for each city. To override this specify the aestetic mapping (aes()) again within the geom_smooth function and add the argument inherit.aes=FALSE

```
library(ggplot2) #loading ggplot
#figure with data from created frame
#x is date, y is mean higher high water, by city
figure = ggplot(data = c.final,aes(x = Date, y = MHHW ,group = city,col = city))
figure +geom_line(linewidth=0.35) +
  labs(y ="Mean Daily High Water Level",x = "Date",
                           title = "Mean Daily High Water Level vs. Date")+
  geom_smooth(method = "lm",aes(x = Date, y = MHHW,group =1),
              col = "black",linewidth = 0.5)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

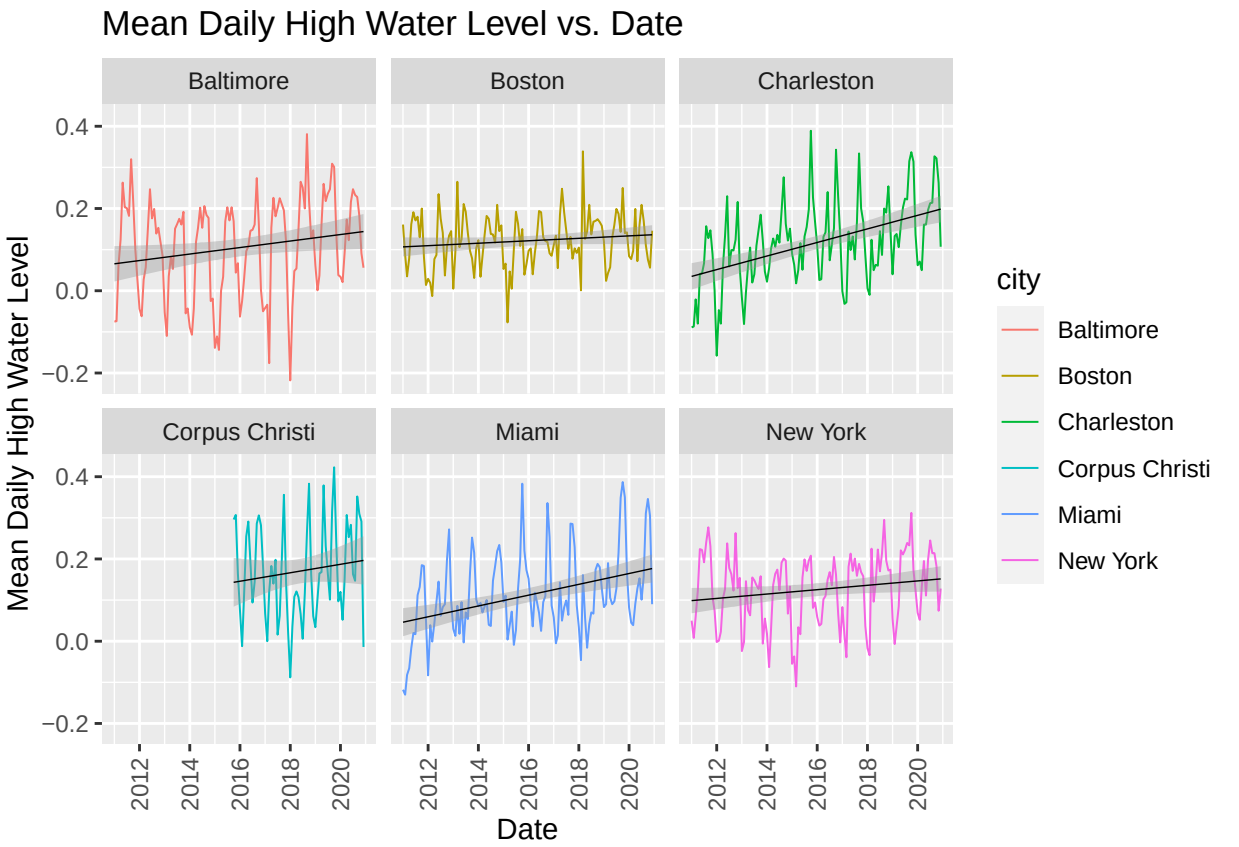


5. Now make a slightly different plot with the same x and y variables, but use facet_wrap() to make a subplot separately for each city. Add a best-fit line for each subplot. See the example plot uploaded to Canvas (Plot 2)

```
figure = ggplot(data = c.final, aes(x = Date, y = MHHW ,group = city,col = city))
figure +geom_line(size=0.35) +labs(y ="Mean Daily High Water Level",x = "Date",
                           title = "Mean Daily High Water Level vs. Date")+ facet_wrap(vars(city))+ #ma
  geom_smooth(method = "lm",aes(x = Date, y = MHHW),col = "black",linewidth = 0.25) +theme(axis.text.x =
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
```

```
## ‘geom_smooth()‘ using formula = ’y ~ x’
```

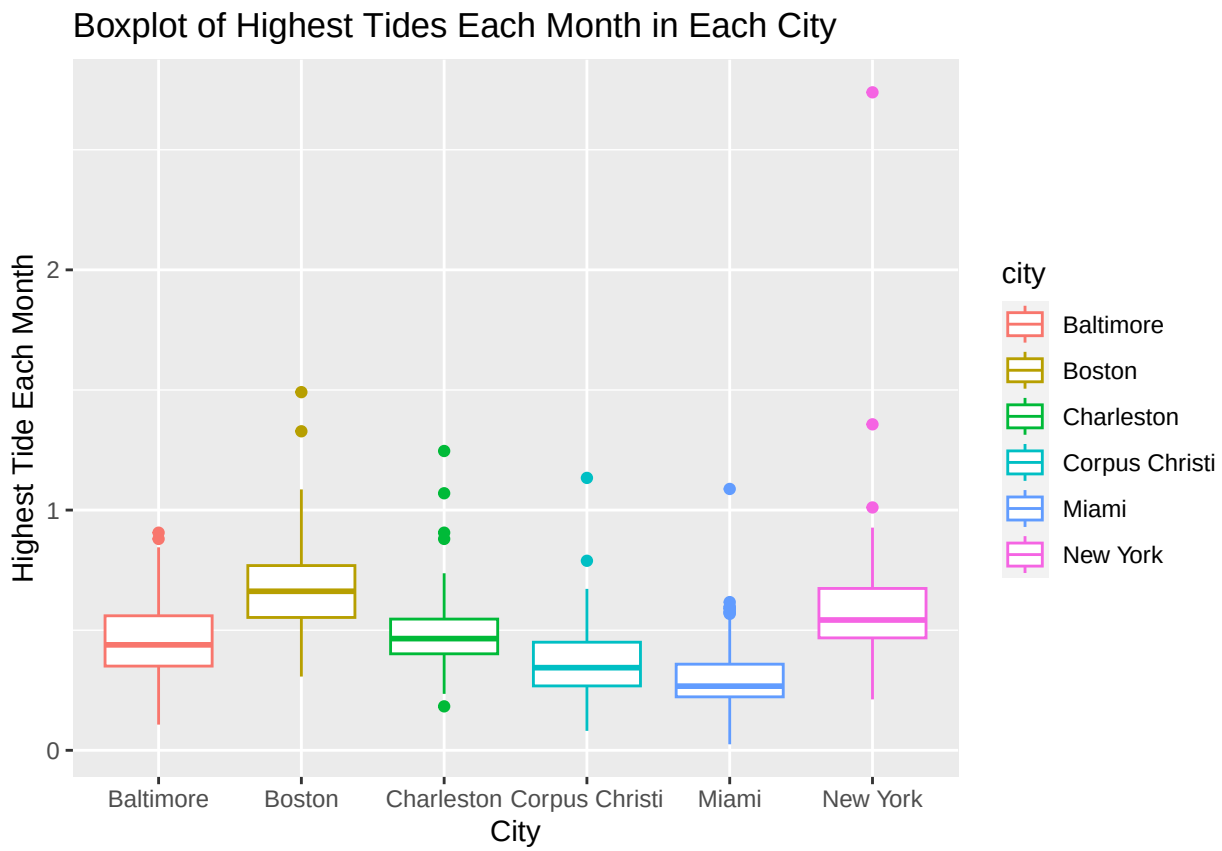


Part 2 - Wednesday

In this part of the lab we will identify some outliers, and practice running regressions

6. Make a box plot showing the distribution of the highest tides each month ("Highest" column in the NOAA data) . (Ideally practice using ggplot by using geom_boxplot() - put the city on the x axis and Highest on the y. But this can also be done in base R). See the example plot on Canvas (Plot 3)

```
figure1 = ggplot(data = c.final, aes(x = city, y = Highest ,group = city,col = city))+geom_boxplot() #m
figure1 +labs(y ="Highest Tide Each Month",x = "City",
                          title = "Boxplot of Highest Tides Each Month in Each City ")
```



Notice the very extreme value in New York City - a major outlier both within New York and compared to all the other cities

7a. Find the row in the data corresponding to this outlier observation

```
max_loc = which.max(c.final$Highest) #finding location of max
```

7b. What month and year did this outlier event occur in? What meteorological event happened in New York in that month that probably caused this outlier event? (Feel free to use Google - I don't expect you to know this off hand)

```
c.final$Date[max_loc] #using location of max to find date
```

```
## [1] "2012-10-01"
```

October of 2012: Hurricane Sandy

Finally, we will fit a linear model to estimate the rate of sea-level rise across these 6 cities.

8a. Fit a linear regression with the mean higher high water (MHHW) as the dependent variable and date (i.e. time) as the independent variable.

```
m.mhhw = lm(MHHW~Date, data = c.final) #fitting linear regression model
```

8b. Give the estimated coefficient of the date column. Is it statistically significant (i.e. has a p-value less than 0.05)?

```
co.dt = m.mhhw$coefficients['Date']
co.dt
```

```
##         Date
## 2.732001e-05
```

```
a = summary(m.mhhw)
```

This coefficient gives us the average increase in high tide levels each day, across all six cities, for this ten year time frame (i.e. the units of the coefficient are in m per day).

8c. Using your estimated coefficient, estimate the mean increase in sea-level over the 10 year time frame from 2011-2020.

```
# yhat = predict.lm(m.mhhw,newdata = )
# mean(yhat)

yhat = m.mhhw$coefficients[['Date']]*2011:2020
mean(yhat)
```

```
## [1] 0.05506347
```

Upload your .Rmd file and you knitted file with the answers and plots to Canvas

STRETCH GOAL

If you are looking for a challenge, have a go downloading the original csv files directly from the NOAA API. Details on the API are here: https://api.tidesandcurrents.noaa.gov/api/prod/

You will want to paste together a URL describing the data you want from the API, then use download.file() to download the data from that URL into a directory on your computer.

The URL you want will have the following form, except you will loop through to replace *GAUGEID* with each of the six tide gauge ID numbers:

paste0("https://api.tidesandcurrents.noaa.gov/api/prod/datagetter?begin_date=20110101&end_date=20201231&station=",*GAUGEID*,"&product=monthly_mean&datum=MHHW&units=metric&time_zone=lst&format=csv"

See if you can make sense of this URL given the options listed at the website describing access to the API

```r
#note eval is false because code is also at beginning of file

dir.create("tidedatafiles")#creates directory where files will go when downloaded
# for-loop for pulling data from noaa
for (i in 1:nrow(city.td)){#iterate for each city
  gaugeid = city.td$tide.id[i]
   id_url = paste0("https://api.tidesandcurrents.noaa.gov/api/prod/datagetter?begin_date=20110101&end_da
   #creating gauge id that will go into url with
                              #each iteration
  #creating url using gauge id variable and noaa url parameters
  #note it requests to pull all data between 01/01/2011 and end 01/01/2023
download.file(id_url,destfile = paste0("tidedatafiles/", gaugeid,".csv"))
}

fi = list.files("tidedatafiles")
```