

ESP106-Lab4

Vickie Bach

January 30, 2026

Lab 4

In this lab we will look at daily tide 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. These are in the "Data" folder. 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 - Monday Jan 27th

1. Create a data frame containing data on the city name and tide gauge ID given above. Your data frame should have 2 columns, one for the city name and one for the gauge ID. It will have 6 rows, one for each city.

```
tide_gauges <- data.frame(city = c("Boston", "New York", "Baltimore",  
  "Charleston", "Miami", "Corpus Christi"), gauge_id = c(8443970,  
  8518750, 8574680, 8665530, 8723214, 8775296))
```

- 2 Create a data frame that combines the data from the 6 tide gauges into a single data frame.

To do this:

2.1 Create a vector with the file names of the 6 csv files in the "Data" folder. Use the function `list.files()`. HINT: if you set `full.names=TRUE` when you call `list.files` it will give you the full path name for each file

2.2 Read in the first of the tide data files (using `read.csv()`) and the first element of the vector of file names you created in a). Store it as an object in your environment, for instance "tidedata"

2.3 If you look at the new data frame, notice it does not have the city name or the gauge ID. We have to add this. Create 2 new columns in your new data frame with the name and gauge ID of the relevant city. Reference the data frame you created in 1 to do this.

2.4 Now we will add the other 5 cities to our data frame. Write a for loop that loops through the 2nd, 3rd, 4th, 5th and 6th cities. For each city, read in that city's data file (using the vector from 2.1), add the 2 columns for city and gauge ID, then use `rbind()` to attach it to your tidedata data frame. For each run of the for loop, we want our tidedata data frame to "grow" with the data from a new city.

```
# 2.1  
files <- list.files(path = "Data", full.names = TRUE)  
  
# 2.2
```

```

tidedata <- read.csv(files[1])

# 2.3
tidedata$city <- tide_gauges$city[1]
tidedata$gauge_id <- tide_gauges$gauge_id[1]

# 2.4
for (i in 2:length(files)) {
  temp <- read.csv(files[i])
  temp$city <- tide_gauges$city[i]
  temp$gauge_id <- tide_gauges$gauge_id[i]
  tidedata <- rbind(tidedata, temp)
}

```

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

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.

3. 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 the lubridate package to tell R to interpret that column as a date. Make sure the lubridate package is installed (install.packages()) and loaded (library("lubridate"))

3.1 Create a new column named "Date" that has the first day of the month for that row in the format YYYY-MM-01 where YYYY is the data in the Year column and MM is the data in the Month column. Hint: Use paste0() to combine data and characters (i.e. the required separators "-")

3.2 Use the ymd() (i.e. year-month-day) function from the lubridate package to convert your new date column to a date object in R

```
library(lubridate)
```

```

##
## Attaching package: 'lubridate'
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

# 3.1
tidedata$Date <- paste0(tidedata$Year, "-", sprintf("%02d", tidedata$Month),
  "-01")

# 3.2
tidedata$Date <- ymd(tidedata$Date)

```

#Please Note: if you get stuck creating the data frame in questions 1-3, I have also provided the full data frame with the lab (tidedata.csv). You can read that in directly and proceed with the rest of the lab.

Now lets use ggplot to make some cool graphs of this data using ggplot.

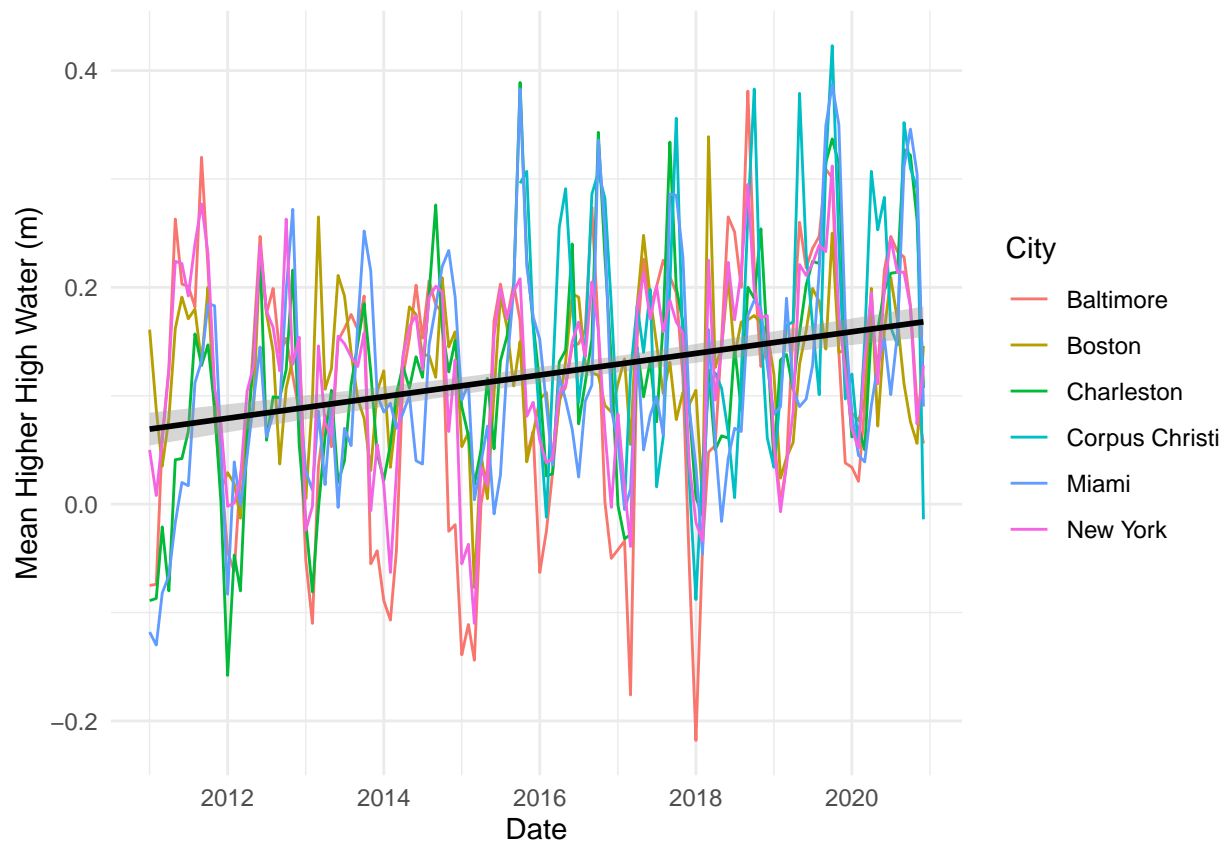
4. Make a plot showing data from all 6 gauges on the same plot. Use different colors to distinguish lines for the different cities. See the example plot uploaded to Canvas (Plot 1)
 - Plot the date on the x axis and MHHW (mean higher high water - i.e. the average daily high water level) on the y axis 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 aesthetic mapping (`aes()`) again within the `geom_smooth` function and add the argument `inherit.aes=FALSE`

```
library(ggplot2) #if you don't already have ggplot2 then install it using install.packages('ggplot2')

ggplot(tidedata, aes(x = Date, y = MHHW, color = city)) + geom_line() +
  geom_smooth(aes(x = Date, y = MHHW), method = "lm", inherit.aes = FALSE,
             color = "black") + labs(x = "Date", y = "Mean Higher High Water (m)",
             color = "City") + theme_minimal()
```

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

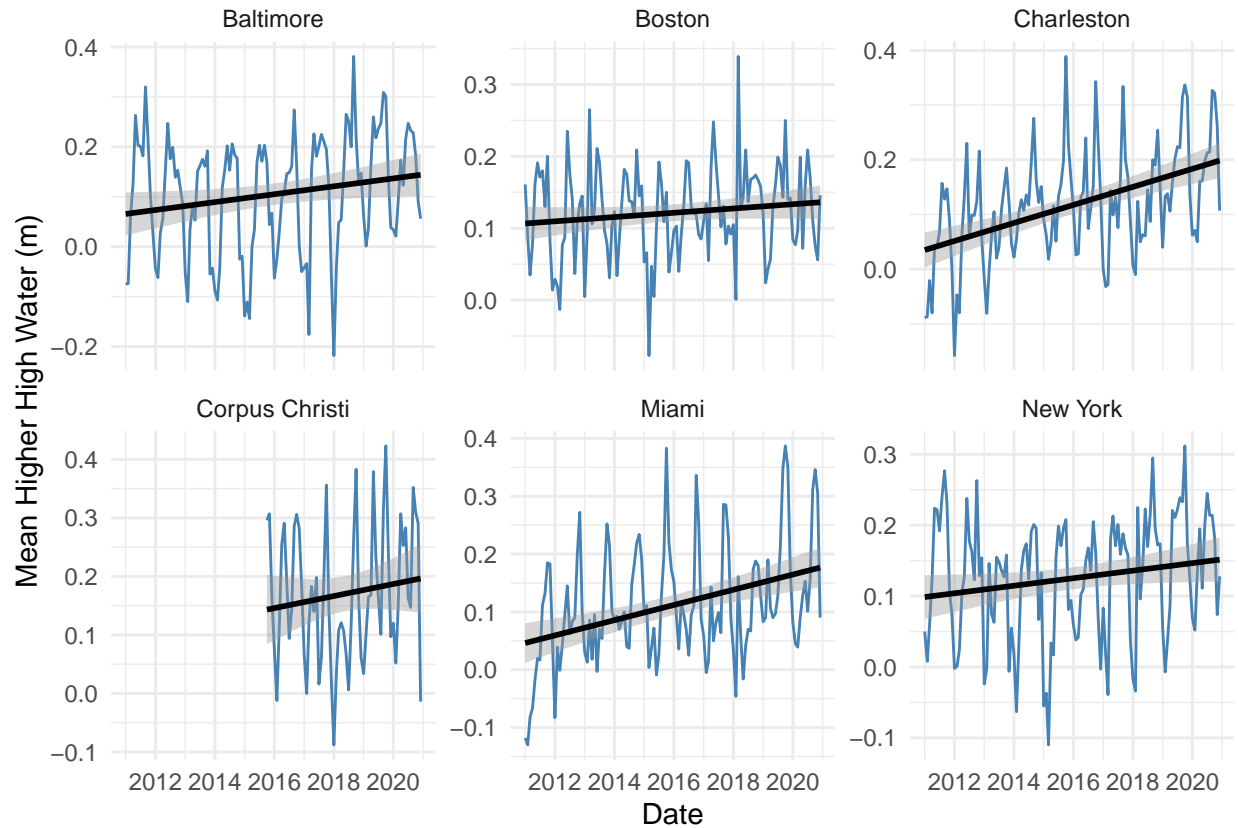


5. Now make a slightly different plot with the same x and y variables, but use `facet_wrap()` to make a subplot separately for each city. Add a best-fit line for each subplot. See the example plot uploaded to Canvas (Plot 2)

*# Hint: you should only need minor modification of the code
from question 4 to make this plot*

```
ggplot(tidedata, aes(x = Date, y = MHHW)) + geom_line(color = "steelblue") +
  geom_smooth(method = "lm", color = "black") + facet_wrap(~city,
             scales = "free_y") + labs(x = "Date", y = "Mean Higher High Water (m)") +
  theme_minimal()
```

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

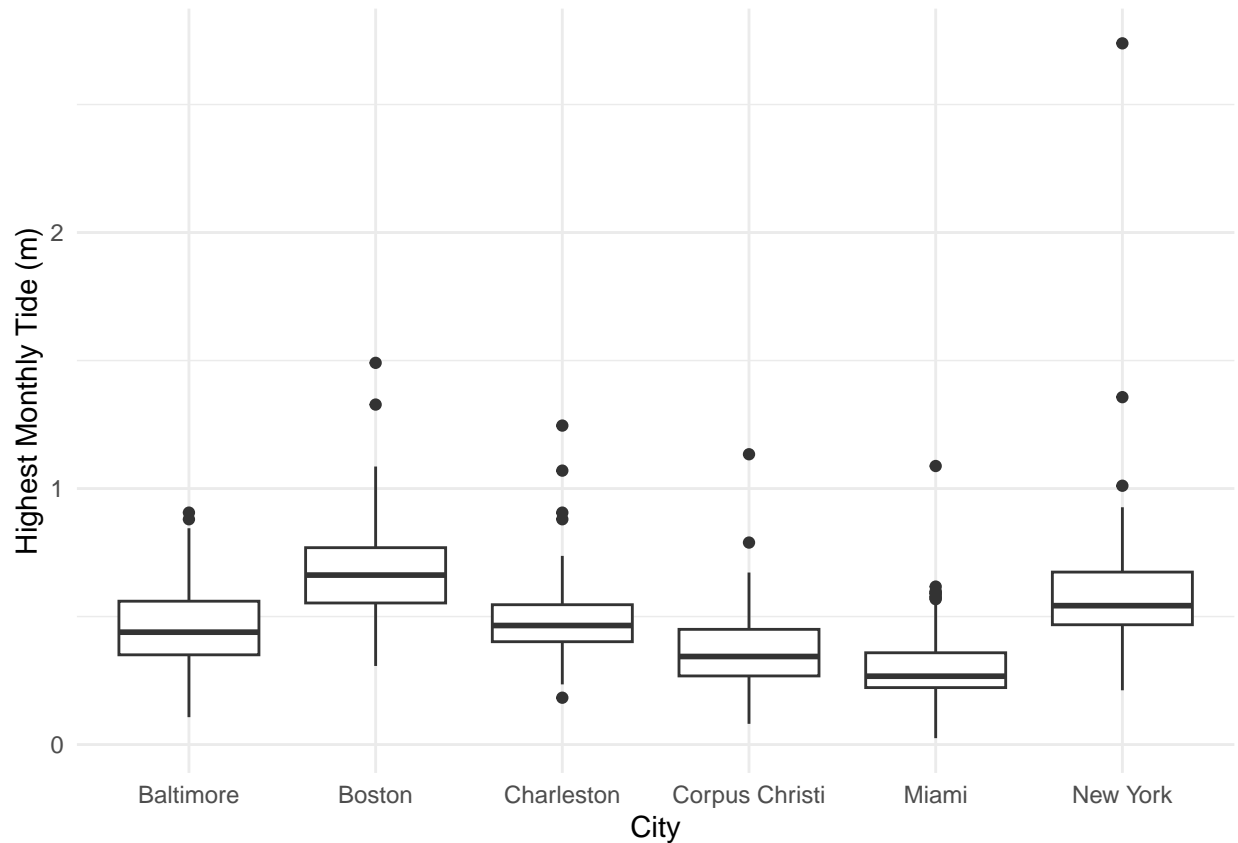


Part 2 - Wednesday Jan 29th

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)

```
ggplot(tidedata, aes(x = city, y = Highest)) + geom_boxplot() +
  labs(x = "City", y = "Highest Monthly Tide (m)") + theme_minimal()
```



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

Hint: The `which.max()` function might be useful here

```
outlier_row <- tidedata[which.max(tidedata$Highest), ]
outlier_row
```

```
##      Year Month Highest  MHHW  MHW  MSL  MTL  MLW  MLLW  DTL  GT
## 141 2012    10   2.739 0.263 0.145 -0.555 -0.587 -1.318 -1.384 -0.561 1.647
##      MN  DHQ  DLQ  HWI  LWI Lowest Inferred    city gauge_id    Date
## 141 1.463 0.118 0.066 0.81 7.23 -1.842      0 New York 8518750 2012-10-01
```

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)

October 2012 (because of 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.

*# Hint: the formula in your `lm()` function is of the form
$y \sim x$ where y here is MHHW and x is your date column*

```
slr_model <- lm(MHHW ~ Date, data = tidedata)
summary(slr_model)

##
## Call:
## lm(formula = MHHW ~ Date, data = tidedata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.35710 -0.06419  0.00207  0.06161  0.27239
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.399e-01  6.002e-02  -5.663 2.23e-08 ***
## Date         2.732e-05  3.552e-06   7.692 5.31e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09572 on 656 degrees of freedom
## Multiple R-squared:  0.08274,    Adjusted R-squared:  0.08134
## F-statistic: 59.17 on 1 and 656 DF,  p-value: 5.314e-14
# The estimated coefficient of the date column is 2.732e^-5
# m/day and it is statistically significant.

# The mean increase in sea-level over the next 10 years
# (2011-2020) is approximately 0.0997 meters across the six
# cities.
```

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

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.

Upload your .Rmd file and you knitted file with the answers and plots to Canvas. Add and commit to your Github repository.

##STRETCH GOAL - Not required for full points

For students 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