# CaseStudy

...

|= | 1%

| In this lesson we'll apply some of the techniques we learned in this course to study air pollution

| data, specifically particulate matter (we'll call it pm25 sometimes), collected by the U.S.

| Environmental Protection Agency. This website

| https://www.health.ny.gov/environmental/indoors/air/pmq\_a.htm from New York State offers some basic

| information on this topic if you're interested.

...

|== | 2%

| Particulate matter (less than 2.5 microns in diameter) is a fancy name for dust, and breathing in

| dust might pose health hazards to the population. We'll study data from two years, 1999 (when

| monitoring of particulate matter started) and 2012. Our goal is to see if there's been a noticeable

| decline in this type of air pollution between these two years.

...

|=== | 3%

| We've read in 2 large zipped files for you using the R command read.table (which is smart enough to

| unzip the files). We stored the 1999 data in the array pm0 for you. Run the R command dim now to

| see its dimensions.

> dim(pm0)

[1] 117421 5

| Keep up the great work!

|==== | 4%

| We see that pm0 has over 117000 lines, each containing 5 columns. In the original file, at the EPA

| website, each row had 28 columns, but since we'll be using only a few of these, we've created and

| read in a somewhat smaller file. Run head on pm0 now to see what the first few lines look like.

> head(pm0)

V1 V2 V3 V4 V5

1 1 27 1 19990103 NA

2 1 27 1 19990106 NA

3 1 27 1 19990109 NA

4 1 27 1 19990112 8.841

5 1 27 1 19990115 14.920

6 1 27 1 19990118 3.878

| Excellent job!

|===== | 5%

| We see there's some missing data, but we won't worry about that now. We also see that the column

| names, V1, V2, etc., are not informative. However, we know that the first line of the original file

| (a comment) explained what information the columns contained.

...

|====== | 6%

| We created the variable cnames containing the 28 column names of the original file. Take a look at

| the column names now.

> cnames

[1] "# RD|Action Code|State Code|County Code|Site ID|Parameter|POC|Sample Duration|Unit|Method|Date|Start Time|Sample Value|Null Data Code|Sampling Frequency|Monitor Protocol (MP) ID|Qualifier - 1|Qualifier - 2|Qualifier - 3|Qualifier - 4|Qualifier - 5|Qualifier - 6|Qualifier - 7|Qualifier - 8|Qualifier - 9|Qualifier - 10|Alternate Method Detectable Limit|Uncertainty"

| That's the answer I was looking for.

|======= | 7%

| We see that the 28 column names look all jumbled together even though they're separated by "|"

| characters, so let's fix this. Reassign to cnames the output of a call to strsplit (string split)

| with 3 arguments. The first is cnames, the pipe symbol '|' is the second (use the quotation marks),

| and the third is the argument fixed set to TRUE. Try this now.

> cnames <- strsplit(cnames, "|", fixed = TRUE)

| That's a job well done!

|======== | 8%

| The variable cnames now holds a list of the column headings. Take another look at the column names.

> cnames

[[1]]

[1] "# RD" "Action Code"

[3] "State Code" "County Code"

[5] "Site ID" "Parameter"

[7] "POC" "Sample Duration"

[9] "Unit" "Method"

[11] "Date" "Start Time"

[13] "Sample Value" "Null Data Code"

[15] "Sampling Frequency" "Monitor Protocol (MP) ID"

[17] "Qualifier - 1" "Qualifier - 2"

[19] "Qualifier - 3" "Qualifier - 4"

[21] "Qualifier - 5" "Qualifier - 6"

[23] "Qualifier - 7" "Qualifier - 8"

[25] "Qualifier - 9" "Qualifier - 10"

[27] "Alternate Method Detectable Limit" "Uncertainty"

| Excellent work!

|========= | 9%

| Nice, but we don't need all these. Assign to names(pm0) the output of a call to the function

| make.names with cnames[[1]][wcol] as the argument. The variable wcol holds the indices of the 5

| columns we selected (from the 28) to use in this lesson, so those are the column names we'll need.

| As the name suggests, the function "makes syntactically valid names".

> names(pm0) <- make.names(cnames[[1]][wcol])

| Excellent job!

|========= | 10%

| Now re-run head on pm0 now to see if the column names have been put in place.

> head(pm0)

State.Code County.Code Site.ID Date Sample.Value

1 1 27 1 19990103 NA

2 1 27 1 19990106 NA

3 1 27 1 19990109 NA

4 1 27 1 19990112 8.841

5 1 27 1 19990115 14.920

6 1 27 1 19990118 3.878

| Perseverance, that's the answer.

|========== | 11%

| Now it's clearer what information each column of pm0 holds. The measurements of particulate matter

| (pm25) are in the column named Sample.Value. Assign this component of pm0 to the variable x0. Use

| the m$n notation.

> x0 <- pm0$Sample.Value

| You got it!

|=========== | 12%

| Call the R command str with x0 as its argument to see x0's structure.

> str(x0)

num [1:117421] NA NA NA 8.84 14.92 ...

| That's correct!

|============ | 13%

| We see that x0 is a numeric vector (of length 117000+) with at least the first 3 values missing.

| Exactly what percentage of values are missing in this vector? Use the R function mean with

| is.na(x0) as an argument to see what percentage of values are missing (NA) in x0.

> mean(is.na(x0))

[1] 0.1125608

| You are quite good my friend!

|============= | 14%

| So a little over 11% of the 117000+ are missing. We'll keep that in mind. Now let's start

| processing the 2012 data which we stored for you in the array pm1.

...

|============== | 15%

| We'll repeat what we did for pm0, except a little more efficiently. First assign the output of

| make.names(cnames[[1]][wcol]) to names(pm1).

> names(pm1) <- make.names(cnames[[1]][wcol])

| All that practice is paying off!

|=============== | 16%

| Find the dimensions of pm1 with the command dim.

> dim(pm1)

[1] 1304287 5

| You nailed it! Good job!

|================ | 18%

| Wow! Over 1.3 million entries. Particulate matter was first collected in 1999 so perhaps there

| weren't as many sensors collecting data then as in 2012 when the program was more mature. If you

| ran head on pm1 you'd see that it looks just like pm0. We'll move on though.

...

|================= | 19%

| Create the variable x1 by assigning to it the Sample.Value component of pm1.

> x1 <- pm1$Sample.Value

| Perseverance, that's the answer.

|================== | 20%

| Now let's see what percentage of values are missing in x1. As before, use the R function mean with

| is.na(x1) as an argument to find out.

> mean(is.na(x1))

[1] 0.05607125

| You are amazing!

|=================== | 21%

| So only 5.6% of the particulate matter measurements are missing. That's about half the percentage

| as in 1999.

...

|==================== | 22%

| Now let's look at summaries (using the summary command) for both datasets. First, x0.

> summary(x0)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

0.00 7.20 11.50 13.74 17.90 157.10 13217

| Your dedication is inspiring!

|===================== | 23%

| The numbers in the vectors x0 and x1 represent measurements taken in micrograms per cubic meter.

| Now look at the summary of x1.

> summary(x1)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

-10.00 4.00 7.63 9.14 12.00 908.97 73133

| All that hard work is paying off!

|====================== | 24%

| We see that both the median and the mean of measured particulate matter have declined from 1999 to

| 2012. In fact, all of the measurements, except for the maximum and missing values (Max and NA's),

| have decreased. Even the Min has gone down from 0 to -10.00! We'll address what a negative

| measurment might mean a little later. Note that the Max has increased from 157 in 1999 to 909 in

| 2012. This is quite high and might reflect an error in the table or malfunctions in some monitors.

...

|======================= | 25%

| Call the boxplot function with 2 arguments, x0 and x1.

> boxplot(x0,x1)

| Great job!

|======================== | 26%

| Huh? Did somebody step on the boxes? It's hard to see what's going on here. There are so many

| values outside the boxes and the range of x1 is so big that the boxes are flattened. It might be

| more informative to call boxplot on the logs (base 10) of x0 and x1. Do this now using log10(x0)

| and log10(x1) as the 2 arguments.

> boxplot(log10(x0),log10(x1))

Warning messages:

1: In boxplot.default(log10(x0), log10(x1)) : NaNs produced

2: In bplt(at[i], wid = width[i], stats = z$stats[, i], out = z$out[z$group == :

Outlier (-Inf) in boxplot 1 is not drawn

3: In bplt(at[i], wid = width[i], stats = z$stats[, i], out = z$out[z$group == :

Outlier (-Inf) in boxplot 2 is not drawn

| That's the answer I was looking for.

|========================= | 27%

| A bonus! Not only do we get a better looking boxplot we also get some warnings from R in Red. These

| let us know that some values in x0 and x1 were "unloggable", no doubt the 0 (Min) we saw in the

| summary of x0 and the negative values we saw in the Min of the summary of x1.

...

|========================== | 28%

| From the boxplot (x0 on the left and x1 on the right), what can you say about the data?

1: The median of x1 is less than the median of x0

2: The range of x0 is greater than the range of x1

3: The boxes are too small to interpret

4: The mean of x1 is less than the mean of x0

Selection: 1

| Nice work!

|=========================== | 29%

| Let's return to the question of the negative values in x1. Let's count how many negative values

| there are. We'll do this in a few steps.

...

|============================ | 30%

| First, form the vector negative by assigning to it the boolean x1<0.

> negative <- x1<0

| Your dedication is inspiring!

|============================ | 31%

| Now run the R command sum with 2 arguments. The first is negative, and the second is na.rm set

| equal to TRUE. This tells sum to ignore the missing values in negative.

> sum(negative, na.rm = T)

[1] 26474

| Not exactly. Give it another go. Or, type info() for more options.

| Type sum(negative, na.rm = TRUE) at the command prompt.

> sum(negative, na.rm = TRUE)

[1] 26474

| You are really on a roll!

|============================= | 32%

| So there are over 26000 negative values. Sounds like a lot. Is it? Run the R command mean with same

| 2 arguments you just used with the call to sum. This will tell us a percentage.

> mean(negative, na.rm = TRUE)

[1] 0.0215034

| Excellent work!

|============================== | 33%

| We see that just 2% of the x1 values are negative. Perhaps that's a small enough percentage that we

| can ignore them. Before we ignore them, though, let's see if they occur during certain times of the

| year.

...

|=============================== | 34%

| First create the array dates by assigning to it the Date component of pm1. Remember to use the x$y

| notation.

> dates <- pm1$Date

| Keep up the great work!

|================================ | 35%

| To see what dates looks like run the R command str on it.

> str(dates)

int [1:1304287] 20120101 20120104 20120107 20120110 20120113 20120116 20120119 20120122 20120125 20120128 ...

| Perseverance, that's the answer.

|================================= | 36%

| We see dates is a very long vector of integers. However, the format of the entries is hard to read.

| There's no separation between the year, month, and day. Reassign to dates the output of a call to

| as.Date with the 2 arguments as.character(dates) as the first argument and the string "%Y%m%d" as

| the second.

> dates <- as.Date(as.character(dates), "%Y%m%d")

| Nice work!

|================================== | 37%

| Now when you run head on dates you'll see the dates in a nicer format. Try this now.

> head(dates)

[1] "2012-01-01" "2012-01-04" "2012-01-07" "2012-01-10" "2012-01-13" "2012-01-16"

| Your dedication is inspiring!

|=================================== | 38%

| Let's plot a histogram of the months when the particulate matter measurements are negative. Run

| hist with 2 arguments. The first is dates[negative] and the second is the string "month".

> hist(dates[negative], "month")

| Excellent job!

|==================================== | 39%

| We see the bulk of the negative measurements were taken in the winter months, with a spike in May.

| Not many of these negative measurements occurred in summer months. We can take a guess that because

| particulate measures tend to be low in winter and high in summer, coupled with the fact that higher

| densities are easier to measure, that measurement errors occurred when the values were low. For now

| we'll attribute these negative measurements to errors. Also, since they account for only 2% of the

| 2012 data, we'll ignore them.

...

|===================================== | 40%

| Now we'll change focus a bit and instead of looking at all the monitors throughout the country and

| the data they recorded, we'll try to find one monitor that was taking measurements in both 1999 and

| 2012. This will allow us to control for different geographical and environmental variables that

| might have affected air quality in different areas. We'll narrow our search and look just at

| monitors in New York State.

...

|====================================== | 41%

| We subsetted off the New York State monitor identification data for 1999 and 2012 into 2 vectors,

| site0 and site1. Look at the structure of site0 now with the R command str.

> str(site0)

chr [1:33] "1.5" "1.12" "5.73" "5.80" "5.83" "5.110" "13.11" "27.1004" "29.2" "29.5" "29.1007" ...

| Excellent job!

|======================================= | 42%

| We see that site0 (the IDs of monitors in New York State in 1999) is a vector of 33 strings, each

| of which has the form "x.y". We've created these from the county codes (the x portion of the

| string) and the monitor IDs (the y portion). If you ran str on site1 you'd see 18 similar values.

...

|======================================== | 43%

| Use the intersect command with site0 and site1 as arguments and put the result in the variable

| both.

> both <- intersect(site0, site1)

| Great job!

|========================================= | 44%

| Take a look at both now.

> both

[1] "1.5" "1.12" "5.80" "13.11" "29.5" "31.3" "63.2008" "67.1015" "85.55"

[10] "101.3"

| You are amazing!

|========================================== | 45%

| We see that 10 monitors in New York State were active in both 1999 and 2012.

...

|=========================================== | 46%

| To save you some time and typing, we modified the data frames pm0 and pm1 slightly by adding to

| each of them a new component, county.site. This is just a concatenation of two original components

| County.Code and Site.ID. We did this to facilitate the next step which is to find out how many

| measurements were taken by the 10 New York monitors working in both of the years of interest. Run

| head on pm0 to see the first few entries now.

> head(pm0)

State.Code County.Code Site.ID Date Sample.Value county.site

1 1 27 1 19990103 NA 27.1

2 1 27 1 19990106 NA 27.1

3 1 27 1 19990109 NA 27.1

4 1 27 1 19990112 8.841 27.1

5 1 27 1 19990115 14.920 27.1

6 1 27 1 19990118 3.878 27.1

| All that practice is paying off!

|============================================ | 47%

| Now pm0 and pm1 have 6 columns instead of 5, and the last column is a concatenation of two other

| columns, County and Site.

...

|============================================= | 48%

| Now let's see how many measurements each of the 10 New York monitors that were active in both 1999

| and 2012 took in those years. We'll create 2 subsets (one for each year), one of pm0 and the other

| of pm1.

...

|============================================== | 49%

| The subsets will filter for 2 characteristics. The first is State.Code equal to 36 (the code for

| New York), and the second is that the county.site (the component we added) is in the vector both.

...

|============================================== | 51%

| First create the variable cnt0 by assigning to it the output of the R command subset, called with 2

| arguments. The first is pm0, and the second is a boolean with the 2 conditions we just mentioned.

| Recall that the testing for equality in a boolean requires ==, intersection of 2 boolean conditions

| is denoted by & and membership by %in%.

> cnt0 <- subset(pm0, State.Code == 36 & county.site %in% both)

| Keep up the great work!

|=============================================== | 52%

| Recall the last command with the up arrow, and create cnt1 (instead of cnt0). Remember to change

| pm0 to pm1. Everything else can stay the same.

> cnt1 <- subset(pm1, State.Code == 36 & county.site %in% both)

| Keep working like that and you'll get there!

|================================================ | 53%

| Now run the command sapply(split(cnt0, cnt0$county.site), nrow). This will split cnt0 into several

| data frames according to county.site (that is, monitor IDs) and tell us how many measurements each

| monitor recorded.

> sapply(split(cnt0, cnt0$county.site), nrow)

1.12 1.5 101.3 13.11 29.5 31.3 5.80 63.2008 67.1015 85.55

61 122 152 61 61 183 61 122 122 7

| You got it right!

|================================================= | 54%

| Do the same for cnt1. (Recall your last command and change 2 occurrences of cnt0 to cnt1.)

> sapply(split(cnt1, cnt1$county.site), nrow)

1.12 1.5 101.3 13.11 29.5 31.3 5.80 63.2008 67.1015 85.55

31 64 31 31 33 15 31 30 31 31

| Perseverance, that's the answer.

|================================================== | 55%

| From the output of the 2 calls to sapply, which monitor is the only one whose number of

| measurements increased from 1999 to 2012?

1: 29.5

2: 63.2008

3: 85.55

4: 101.3

Selection: 3

| That's a job well done!

|=================================================== | 56%

| We want to examine a monitor with a reasonable number of measurements so let's look at the monitor

| with ID 63.2008. Create a variable pm0sub which is the subset of cnt0 (this contains just New York

| data) which has County.Code equal to 63 and Site.ID 2008.

> pm0sub <- subset(cnt0, County.Code==63 & Site.ID==2008)

| All that practice is paying off!

|==================================================== | 57%

| Now do the same for cnt1. Name this new variable pm1sub.

> pm1sub <- subset(cnt1, County.Code==63 & Site.ID==2008)

| Keep working like that and you'll get there!

|===================================================== | 58%

| From the output of the 2 calls to sapply, how many rows will pm0sub have?

1: 29

2: 122

3: 30

4: 183

Selection: 2

| Keep working like that and you'll get there!

|====================================================== | 59%

| Now we'd like to compare the pm25 measurements of this particular monitor (63.2008) for the 2

| years. First, create the vector x0sub by assigning to it the Sample.Value component of pm0sub.

> x0sub <- pm0sub$Sample.Value

| You got it right!

|======================================================= | 60%

| Similarly, create x1sub from pm1sub.

> x1sub <- pm1sub$Sample.Value

| Excellent work!

|======================================================== | 61%

| We'd like to make our comparison visually so we'll have to create a time series of these pm25

| measurements. First, create a dates0 variable by assigning to it the output of a call to as.Date.

| This will take 2 arguments. The first is a call to as.character with pm0sub$Date as the argument.

| The second is the format string "%Y%m%d".

> dates0 <- as.Date(as.character(pm0sub$Date), "%Y%m%d")

| Excellent job!

|========================================================= | 62%

| Do the same for the 2012 data. Specifically, create dates1 using pm1sub$Date as your input.

> dates1 <- as.Date(as.character(pm1sub$Date), "%Y%m%d")

| Excellent work!

|========================================================== | 63%

| Now we'll plot these 2 time series in the same panel using the base plotting system. Call par with

| 2 arguments. The first is mfrow set equal to c(1,2). This will tell the system we're plotting 2

| graphs in 1 row and 2 columns. The second argument will adjust the panel's margins. It is mar set

| to c(4,4,2,1).

> par(mfrow=c(1,2), mar=c(4,4,2,1) )

| You are quite good my friend!

|=========================================================== | 64%

| Call plot with the 3 arguments dates0, x0sub, and pch set to 20. The first two arguments are the x

| and y coordinates. This will show the pm25 values as functions of time.

> plot(dates0, x0sub, pch = 20)

| That's a job well done!

|============================================================ | 65%

| Now we'll mark the median.

...

|============================================================= | 66%

| Use abline to add a horizontal line at the median of the pm25 values. Make the line width 2 (lwd is

| the argument), and when you call median with x0sub, specify the argument na.rm to be TRUE.

> abline(h = median(x0sub, na.rm = TRUE),lwd=2)

| Excellent job!

|============================================================== | 67%

| Now we'll do the same for the 2012 data. Call plot with the 3 arguments dates1, x1sub, and pch set

| to 20.

> plot(dates1, x1sub, pch = 20)

| All that practice is paying off!

|=============================================================== | 68%

| As before, we'll mark the median of this 2012 data.

...

|================================================================ | 69%

| Use abline to add a horizontal line at the median of the pm25 values. Make the line width 2 (lwd is

| the argument). Remember to specify the argument na.rm to be TRUE when you call median on x1sub.

> abline(h = median(x1sub, na.rm = TRUE),lwd=2)

| You got it right!

|================================================================ | 70%

| Which median is larger - the one for 1999 or the one for 2012?

1: 2012

2: 1999

Selection: 2

| Nice work!

|================================================================= | 71%

| The picture makes it look like the median is higher for 2012 than 1999. Closer inspection shows

| that this isn't true. The median for 1999 is a little over 10 micrograms per cubic meter and for

| 2012 its a little over 8. The plots appear this way because the 1999 plot ....

1: shows different months than those in the 2012 plot

2: displays more points than the 2012 plot

3: shows a bigger range of y values than the 2012 plot

Selection: 3

| Keep working like that and you'll get there!

|================================================================== | 72%

| The 1999 plot shows a much bigger range of pm25 values on the y axis, from below 10 to 40, while

| the 2012 pm25 values are much more restricted, from around 1 to 14. We should really plot the

| points of both datasets on the same range of values on the y axis. Create the variable rng by

| assigning to it the output of a call to the R command range with 3 arguments, x0sub, x1sub, and the

| boolean na.rm set to TRUE.

> rng <- range(x0sub, x1sub,na.rm = TRUE)

| You are really on a roll!

|=================================================================== | 73%

| Look at rng to see the values it spans.

> rng

[1] 3.0 40.1

| Your dedication is inspiring!

|==================================================================== | 74%

| Here a new figure we've created showing the two plots side by side with the same range of values on

| the y axis. We used the argument ylim set equal to rng in our 2 calls to plot. The improvement in

| the medians between 1999 and 2012 is now clear. Also notice that in 2012 there are no big values

| (above 15). This shows that not only is there a chronic improvement in air quality, but also there

| are fewer days with severe pollution.

...

|===================================================================== | 75%

| The last avenue of this data we'll explore (and we'll do it quickly) concerns a comparison of all

| the states' mean pollution levels. This is important because the states are responsible for

| implementing the regulations set at the federal level by the EPA.

...

|====================================================================== | 76%

| Let's first gather the mean (average measurement) for each state in 1999. Recall that the original

| data for this year was stored in pm0.

...

|======================================================================= | 77%

| Create the vector mn0 with a call to the R command with using 2 arguments. The first is pm0. This

| is the data in which the second argument, an expression, will be evaluated. The second argument is

| a call to the function tapply. This call requires 4 arguments. Sample.Value and State.Code are the

| first two. We want to apply the function mean to Sample.Value, so mean is the third argument. The

| fourth is simply the boolean na.rm set to TRUE.

> mn0 <- with(pm0, tapply(Sample.Value, State.Code, mean, na.rm=TRUE))

| You got it!

|======================================================================== | 78%

| Call the function str with mn0 as its argument to see what it looks like.

> str(mn0)

num [1:53(1d)] 19.96 6.67 10.8 15.68 17.66 ...

- attr(\*, "dimnames")=List of 1

..$ : chr [1:53] "1" "2" "4" "5" ...

| Great job!

|========================================================================= | 79%

| We see mn0 is a 53 long numerical vector. Why 53 if there are only 50 states? As it happens, pm25

| measurements for the District of Columbia (Washington D.C), the Virgin Islands, and Puerto Rico are

| included in this data. They are coded as 11, 72, and 78 respectively.

...

|========================================================================== | 80%

| Recall your command creating mn0 and change it to create mn1 using pm1 as the first input to the

| call to with.

> mn1 <- with(pm1, tapply(Sample.Value, State.Code, mean, na.rm=TRUE))

| Excellent job!

|=========================================================================== | 81%

| For fun, call the function str with mn1 as its argument.

> str(mn1)

num [1:52(1d)] 10.13 4.75 8.61 10.56 9.28 ...

- attr(\*, "dimnames")=List of 1

..$ : chr [1:52] "1" "2" "4" "5" ...

| Keep up the great work!

|============================================================================ | 82%

| So mn1 has only 52 entries, rather than 53. We checked. There are no entries for the Virgin Islands

| in 2012. Call summary now with mn0 as its input.

> summary(mn0)

Min. 1st Qu. Median Mean 3rd Qu. Max.

4.862 9.519 12.315 12.406 15.640 19.956

| All that hard work is paying off!

|============================================================================= | 84%

| Now call summary with mn1 as its input so we can compare the two years.

> summary(mn1)

Min. 1st Qu. Median Mean 3rd Qu. Max.

4.006 7.355 8.729 8.759 10.613 11.992

| That's the answer I was looking for.

|============================================================================== | 85%

| We see that in all 6 entries, the 2012 numbers are less than those in 1999. Now we'll create 2 new

| dataframes containing just the state names and their mean measurements for each year. First, we'll

| do this for 1999. Create the data frame d0 by calling the function data.frame with 2 arguments. The

| first is state set equal to names(mn0), and the second is mean set equal to mn0.

> d0 <- data.frame(state=names(mn0), mean=mn0)

| All that practice is paying off!

|=============================================================================== | 86%

| Recall the last command and create d1 instead of d0 using the 2012 data. (There'll be 3 changes of

| 0 to 1.)

> d1 <- data.frame(state=names(mn1), mean=mn1)

| You nailed it! Good job!

|================================================================================ | 87%

| Create the array mrg by calling the R command merge with 3 arguments, d0, d1, and the argument by

| set equal to the string "state".

> mrg <- merge(d0,d1,by = "state")

| Perseverance, that's the answer.

|================================================================================= | 88%

| Run dim with mrg as its argument to see how big it is.

> dim(mrg)

[1] 52 3

| You're the best!

|================================================================================== | 89%

| We see merge has 52 rows and 3 columns. Since the Virgin Island data was missing from d1, it is

| excluded from mrg. Look at the first few entries of mrg using the head command.

> head(mrg)

state mean.x mean.y

1 1 19.956391 10.126190

2 10 14.492895 11.236059

3 11 15.786507 11.991697

4 12 11.137139 8.239690

5 13 19.943240 11.321364

6 15 4.861821 8.749336

| That's correct!

|=================================================================================== | 90%

| Each row of mrg has 3 entries - a state identified by number, a state mean for 1999 (mean.x), and a

| state mean for 2012 (mean.y).

...

|=================================================================================== | 91%

| Now we'll plot the data to see how the state means changed between the 2 years. First we'll plot

| the 1999 data in a single column at x=1. The y values for the points will be the state means.

| Again, we'll use the R command with so we don't have to keep typing mrg as the data environment in

| which to evaluate the second argument, the call to plot. We've already reset the graphical

| parameters for you.

...

|==================================================================================== | 92%

| For the first column of points, call with with 2 arguments. The first is mrg, and the second is the

| call to plot with 3 arguments. The first of these is rep(1,52). This tells the plot routine that

| the x coordinates for all 52 points are 1. The second argument is the second column of mrg or

| mrg[,2] which holds the 1999 data. The third argument is the range of x values we want, namely xlim

| set to c(.5,2.5). This works since we'll be plotting 2 columns of points, one at x=1 and the other

| at x=2.

> with(mrg, plot(rep(1, 52), mrg[, 2], xlim = c(.5, 2.5)))

| That's the answer I was looking for.

|===================================================================================== | 93%

| We see a column of points at x=1 which represent the 1999 state means. For the second column of

| points, again call with with 2 arguments. As before, the first is mrg. The second, however, is a

| call to the function points with 2 arguments. We need to do this since we're adding points to an

| already existing plot. The first argument to points is the set of x values, rep(2,52). The second

| argument is the set of y values, mrg[,3]. Of course, this is the third column of mrg. (We don't

| need to specify the range of x values again.)

> with(mrg, points(rep(2, 52), mrg[, 3]))

| Your dedication is inspiring!

|====================================================================================== | 94%

| We see a shorter column of points at x=2. Now let's connect the dots. Use the R function segments

| with 4 arguments. The first 2 are the x and y coordinates of the 1999 points and the last 2 are the

| x and y coordinates of the 2012 points. As in the previous calls specify the x coordinates with

| calls to rep and the y coordinates with references to the appropriate columns of mrg.

> segments(rep(1, 52), mrg[, 2], rep(2, 52), mrg[, 3])

| Excellent job!

|======================================================================================= | 95%

| We see from the plot that the vast majority of states have indeed improved their particulate matter

| counts so the general trend is downward. There are a few exceptions. (The topmost point in the 1999

| column is actually two points that had very close measurements.)

...

|======================================================================================== | 96%

| For fun, let's see which states had higher means in 2012 than in 1999. Just use the mrg[mrg$mean.x

| < mrg$mean.y, ] notation to find the rows of mrg with this particulate property.

> mrg[mrg$mean.x < mrg$mean.y, ]

state mean.x mean.y

6 15 4.861821 8.749336

23 31 9.167770 9.207489

27 35 6.511285 8.089755

33 40 10.657617 10.849870

| Great job!

|========================================================================================= | 97%

| Only 4 states had worse pollution averages, and 2 of these had means that were very close. If you

| want to see which states (15, 31, 35, and 40) these are, you can check out this website

| https://www.epa.gov/enviro/state-fips-code-listing to decode the state codes.

...

|========================================================================================== | 98%

| This concludes the lesson, comparing air pollution data from two years in different ways. First, we

| looked at measures of the entire set of monitors, then we compared the two measures from a

| particular monitor, and finally, we looked at the mean measures of the individual states.

...

|=========================================================================================== | 99%

| Congratulations! We hope you enjoyed this particulate lesson.

...

|============================================================================================| 100%

| Would you like to receive credit for completing this course on Coursera.org?

1: Yes

2: No

## Plots















