Workshop 03

September 24, 2024

We started to talk about subsetting data.frames last week, however we were running out of time!

Subsetting Data Frames

First, it is **absolutely critical** that you place the `.csv` file in the same directory as this `.qmd` file if you want this to work.

```r
ap <- read.csv(file = "5240-workshop03-data.csv", header = TRUE)
str(ap)
```

```
'data.frame':	1094 obs. of  2 variables:
 $ o3 : num  5.43 9.1 3.95 7.4 5.78 ...
 $ no2: num  18.4 23.3 18.1 27.6 28.1 ...
```

- This data.frame consists of 2 variables: ozone and nitrogen dioxide measurements.
- The units on these variables is parts per million (ppm).
- The data consists of daily measurements from 1981 to 1984.
- NO_2 is actually lagged by 1-day; i.e., each row of O_3, NO_2 measurements is ozone at time t and NO_2 at time $t-1$.
- This is real data from the National Air Pollution Surveilance (NAPS) network run in Canada by Environment and Climate Change Canada (ECCC).
- The measurements are of ground-level pollutants.

```r
plot(x = 1:1094, y = ap$no2,
     type = 'l', xlab = "Days Since Collection Start",
     ylab = "NO2 (ppb)")
     abline(h = 60 , col = "red",lty=2)
```

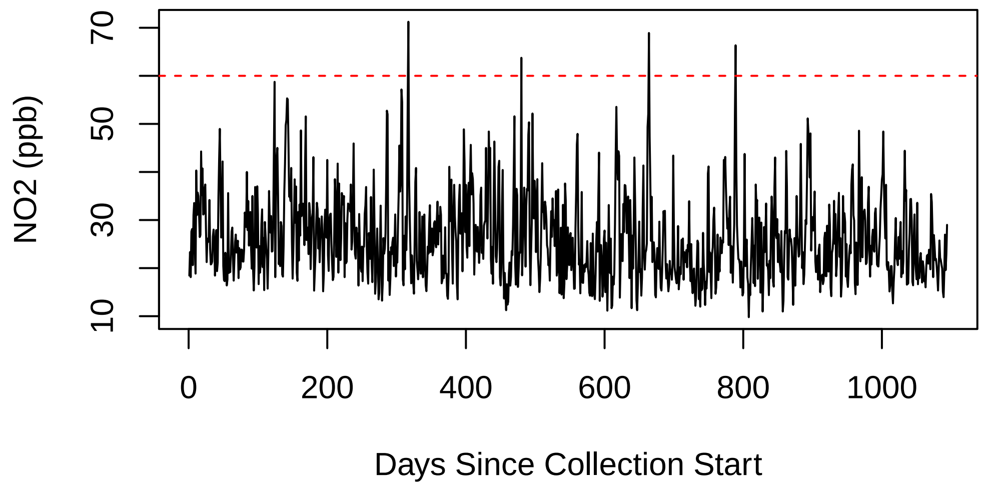

Figure 1: Measurements of NO2 per day since start of collection.

Back to Subsetting

What if we know that any value of NO2 being greater than 60 is an error in the measurement or recording device.
(We will just pretend that this is true for these examples.)

Logical Comparisons

When we do comparison of things, this results in what is known as a Boolean variable (in R, these take on the values `TRUE` and `FALSE` and are called `logical` variable types).

We use logic! If we compare two things in `R`, it returns logical answers: `TRUE` means **YES**, and `FALSE` means **NO**.
The expressions that return `TRUE` or `FALSE` are called *logical expressions*.

Let's *pretend* that any value of NO2 over 60 would be considered an error from the instrument. How can we find *where* that value is?

```r
answers <- ap$no2 > 60
str(answers)
```

```
 logi [1:1094] FALSE FALSE FALSE FALSE FALSE FALSE ...
```

```
answers[310:330] # here we are selecting the 310 through 330's element of the vector
```

```
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

If we removed the [310:330] thing above, it would print 1095 answers. Way too many to look at on our own! But when we print just a few, we see a TRUE. Can you see it? There's more than one in the data set. How do we know that?

```
sum(answers)
```

```
[1] 4
```

When we add variables, R will do its best to convert those variable types into numeric variables. In most programming languages, TRUE is 1, while FALSE is 0. So, if we sum a logical vector, we will end up counting the number of TRUE values!

In our case, this is the number of observations that are larger than 60.

Now, here's the really interesting thing. We're going to break this off because it's so important.

REALLY IMPORTANT FACT

If you use a logical vector as a selection operation (i.e., to subset), TRUE values get selected, and FALSE values get ignored.

Example

Consider the numbers 1 through 20. We would like to identify the ones that are greater than 11. Obviously you can just do this, but how would you tell R to do it?

```
our_numbers <- 1:20
our_logic <- ( our_numbers > 11 )
our_numbers
```

```
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```r
our_logic
```

```
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
[13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```r
our_numbers[ our_logic ]
```

```
[1] 12 13 14 15 16 17 18 19 20
```

So by creating a logical comparison (`our_logic`), we are able to subset the vector `our_numbers` and only keep the ones that are greater than 11.

Back to the NO2 data

We want to keep **only** the elements of NO2 that are less than 60. How do we do this?

Implement this here:

```r
lt_60 <- ap$no2 < 60
ap_lt60 <- ap$no2[lt_60]
length(ap_lt60)
```

```
[1] 1090
```

```r
length(ap$no2)
```

```
[1] 1094
```

```r
# Proportion
sum(lt_60) / length(ap$no2)
```

```
[1] 0.9963437
```

Notice how this vector is only 1090 elements long now? The original data set was 1094! We have deleted 4 observations - fake outliers!

Final Note: Accessing Data Frames, and Subsetting by Observation

What we just did was actually subset a **vector**, since every column is a vector and we used the `$` operator to return a single column from the data frame.
However, `data.frame` objects aren't vectors:
they are 2D objects, like spreadsheets. When subsetting a data frame we have to specify the **rows** and the **columns**!

```
no_outliers_dat <- ap[ our_logic, ]
str(no_outliers_dat)
```

```
'data.frame':   489 obs. of  2 variables:
 $ o3 : num  2.33 5.33 2.44 4.7 7 ...
 $ no2: num  31 35.7 35 31.5 26.5 ...
```

What did we do here? We took `our_dat` (the spreadsheet: five variables, 1095 observations) and told R that we wanted to keep only the rows where the NO2 was <= 60, and (because nothing was specified after the comma) to keep *all* of the columns. Without the comma, it would break, so be careful!

Linear Models

Scatter Plots

- Create a scatterplot of O_3 on the y-axis vs. NO_2 on the x-axis (use the `plot` function);
- Add appropriate axis labels (`xlab` and `ylab`);

```
plot(ap$no2,ap$o3,type = "p",
     xlab = "No2/ppm",
     ylab = "O3/ppm",
     col= "blue",
     main = "Ozone Vs No2")
```

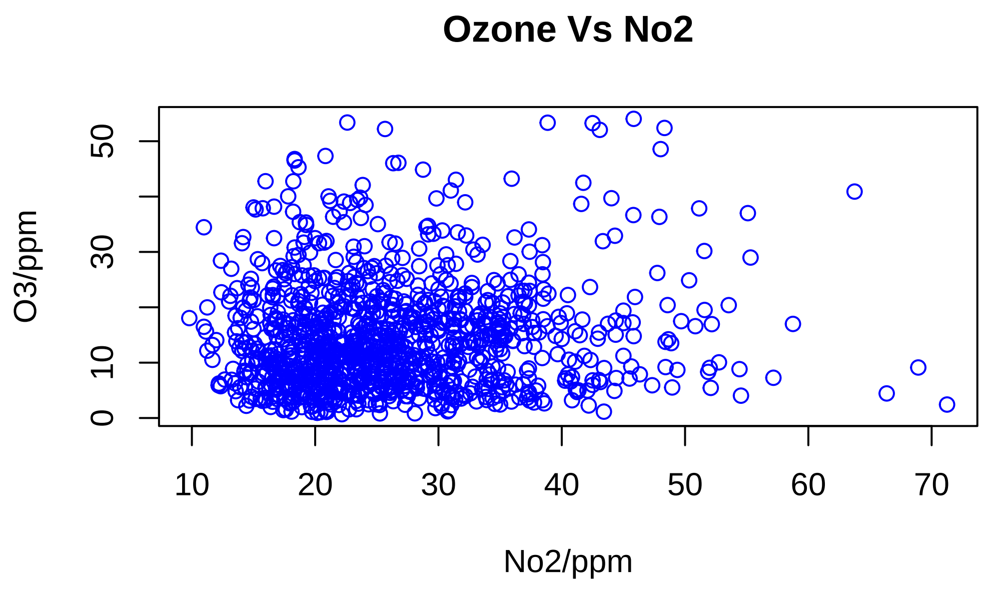

Figure 2

How would you describe the relationship between O_3 and NO_2?

Correlation

Calculate the correlation between O_3 and NO_2:

```
corr <- cor(ap$o3,ap$no2,method="pearson")
# dependent 1st and independent 2nd
print(corr)
```

```
[1] 0.08724986
```

Does this value match your intuition from the plot?
Is this a strong linear relationship?
In what direction?

Fit a Linear Model

Us the `lm` function to fit a linear model using `o3` as the response variable.
Use the `summary` function to print out the model information.

```r
linear_model <- lm(o3 ~ no2 , data = ap)
summary(linear_model)
```

```

Call:
lm(formula = o3 ~ no2, data = ap)

Residuals:
    Min      1Q  Median      3Q     Max 
-16.327  -7.100  -1.684   4.712  39.192 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 12.05590    0.90137  13.375  < 2e-16 ***
no2          0.09411    0.03252   2.894  0.00388 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.629 on 1092 degrees of freedom
Multiple R-squared:  0.007613,  Adjusted R-squared:  0.006704 
F-statistic: 8.377 on 1 and 1092 DF,  p-value: 0.003876
```

Write the Model Equation

To create math equations, we use dollar signs.

For example, we could write: we estimated the β coefficients in the simple linear regression model between O_3 and NO_2.

We can also use "display style" equations:

$$\hat{y} = 12.056 + 0.094.x$$

Now, write the model equation for the regression model that you just fit by pulling the coefficients out of the `summary()` above, or by using another method. Round the coefficients to something reasonable!

Model equation here. Delete me first.

Plot the Data and the Linear Regression Line

Next, use the `plot` function again to plot the data and add the line of best fit (the model regression line that we just fit) to the plot.

You can do this by using the `abline` function. The only argument *needed* is the model object that you fit!

You might also want to change the colour of the line; this can be done by adding the `col = "blue"` (or whatever colour!) to the function.

Predictions

Lastly, let's predict the O_3 concentration given the following NO_2 measurements: 10, 20, 25, 30, 50, 70.

Create a data frame with a column named `no2` that contains those values.

Next, use the `predict` function in order to obtain predicted values of O_3.

Let's assume that O_3 levels above 35 ppm would be considered a risk to health.
Would any of our predictions be considered a risk to health? Use a logical expression to determine that!
(Yes, I know you could just look at the predictions :)).