**OPENAIR PRIMER**

TEMPLATE: Please copy content/questions to your own document.

Go to File → Make a Copy → Save to your own Drive.

Your name(s):

**INSTALLATION**

1. (Slide 3 - *Install Openair*)  
   Install *openair* – “an R package developed for the purpose of analyzing air quality data or more generally atmospheric composition data.”  
     
   Open RStudio and type:  
     
   install.packages("openair")  
     
   Then press enter to execute – *openair* along with other necessary packages will begin downloading immediately.
2. References:  
   User friendly documentation: <https://bookdown.org/david_carslaw/openair/>  
   Github repo: <https://github.com/davidcarslaw/openair>

**EXPLORE OPENAIR**

1. (Slide 4 - *Call Libraries*)  
   Open a new script file and name it openair\_primer\_firstname\_lastname.R to keep track of these function calls for your reference.  
     
   Import the necessary libraries:  
   **library**(openair) # vis pkg  
   **library**(tidyverse) # helps vis pkg
2. (Slide 5 - *Get Data*)  
   We’re going to download our data from *openair* via an API, which is a repository of data that can be accessed directly from code commands. Specifically, the data that we will be using is from the UK's [Automatic Urban and Rural Network (AURN)](https://uk-air.defra.gov.uk/networks/network-info?view=aurn).  
     
   Run the following to import your data:  
   data <- importAURN(site = "my1", year = 2000:2005)  
     
   If this doesn’t work, you can download *openair\_primer\_dataset.csv* from Canvas and use readcsv() function.  
   data <- read.csv(file.choose())
   1. For the project, you will have some pre-cleaned data, but it is important to know about openair’s capabilities (and the capabilities of APIs) for future use!
3. (Slide 6 - *View Data*)  
   Once the data is imported, you can click on the object in your workspace to view it, or run the command head(data) to see some of the data.
   1. What do you notice? (Any interesting formatting of entries? Any columns you can recognize before going further?)
   2. (Slide 7 - *Columns*)  
      There are 13 variables in the dataset. Complete the following table with variable name to match the given meaning, format, and/or unit of measurement (done as a class)

| **Variable Name** | **Meaning** |
| --- | --- |
|  | The name of the collection location |
|  | The code for importing this site through the API |
|  | Date and time listed as yyyy-mm-dd hh:mm:ss |
|  | The level of carbon monoxide in μg/m^3 |
|  | The level of other nitrous oxides (not no2 or no) in μg/m^3 |
|  | The level of nitrogen dioxide in μg/m^3 |
|  | The level of nitrogen monoxide in μg/m^3 |
|  | The level of ozone in μg/m^3 |
|  | The level of sulfur dioxide in μg/m^3 |
|  | The level of particulate matter of 10 micron size in μg/m^3 |
|  | The level of particulate matter of 2.5 micron size in μg/m^3 |
|  | The speed of the wind in m/s |
|  | The direction of the wind in compass degrees (0º north, clockwise around) |

1. (Slide 9 - *Plot 1: timePlot*)  
   Now let’s first check our data by creating a simple plot: the value of our variables over time. The best way to do this clearly and concisely is to use the openair function timePlot() (plots the pollutant nox by default)  
     
   timePlot(data)  
     
   To specify one or more variables to plot, as well as the option to average over time:  
     
   timePlot(data,   
    pollutant = c("no", "no2", "nox", "o3"),   
    avg.time = "month")

Once plotted, you may need to adjust your plots window to fully see the data. You can do this by clicking and dragging the bar dividing the files window and plots window.

* 1. Do you see any gaps in this data? If so, why do you think they are there? If not, what would you do if there were gaps?
  2. Do you see any pattern of peaks or valleys in this data? Why do you think they are there?
  3. You might notice that the time plots don’t have titles. What argument can you use from the R intro assignment to create a plot title? Try this out with an appropriate title of your choice, and submit your code

1. (Slide 10 - *Plot 2: windRose*)  
   Next, we’ll create one of the most important plots: a windrose. Run the command windRose().  
     
   windRose(data)
   1. (Slide 11 - *windRose Questions*)  
      This plot produces 12 bars jutting out on a polar plot. What do the directions of these bars correspond to?
   2. The legend below the plot lists a range of values for each color with units of (m s-1), what do the values correspond to (think about what variable has those units)?
   3. (Slide 12 - *windRose Questions*)  
      The plot is titled “Frequency of counts by wind direction (%).” If this is the case, what do the lengths of the bars represent (look at the units on the concentric rings)?
   4. In the lower right corner of the plot are values for the “mean” and “calm”, what do these mean (hint: mean is in m s-1 and you can check if the wind speed ever hits zero by making a plot of wind speed)?
   5. (Slide 13 - *windRose Questions*)  
      Finally, using the above information as necessary, explain what a windRose shows (1-2 sentences).
   6. (Slide 14 - *windRose Arguments*)  
      As before, we can use internal arguments to specify further.  
        
      windRose(data, type = "year", layout = c(3, 2))  
        
      How does type = “year” separate the data?
   7. How about type = “pm10”?  
        
      windRose(data, type = "pm10", layout = c(3, 2))  
        
      What is the data binned by now?
   8. Explore the different variations of windroses given type and variable. Export one that you produced of your choice, something that piques your interest. Explain what your windRose shows. Are there any conclusions you can draw?
2. (Slide 15 - *pollutionRose*)  
   We can also make a variation of this called a pollution rose (nox by default).  
     
   pollutionRose(data)  
     
   To focus on a specific pollutant, use the pollutant argument:  
     
   pollutionRose(data, pollutant = "no")  
     
   To link with another pollutant, use the type argument:  
     
   pollutionRose(data, pollutant = "no", type = "no2", layout = c(4, 1))  
     
   To segment (remove the spaces between the bars) and normalize:  
     
   pollutionRose(data, pollutant = "nox", seg = 1, normalise = TRUE)
   1. The 12 bars correspond to the same data aspect as in the windRose, but now what does the shading correspond to?
   2. What does setting the segment to 1 and turning on normalization help you to see? (Hint: are the sizes of the colored regions relative to something? Also, think about how a pollution rose was employed by the article you read in assignment 2.)
   3. This is the second time you’ve seen the layout argument, what does the argument accept and what does this do?
3. (Slide 16 - *pollutionRose challenge*)  
   Using the command pollutionRose() and knowledge of R and *openair* syntax, make a pollution rose for only the year 2002 that is tracking the level of pm10.  
     
   Submit your code you used to create this plot, including the commands to set up the file and import the data.
4. (Slide 17 - *Plot 4: calendarPlot*)  
   Next we will create some other visualizations of pollution over time. Begin with the calendar plot using the command:  
     
   calendarPlot(data)

If you wish, you can also add the wind as a vector to each day on the calendar:

calendarPlot(data, annotate = "ws")

1. Now, using your knowledge of R and openair, make a calendar plot for just the year 2003 tracking nitrous oxide only. Paste your code (script/data setup not necessary) and your plot below.
2. Imagine you made a timePlot with a monthly average of co for the whole time period (you can even do this now if you wish). Why might the calendarPlot be a useful next step?
3. (Slide 18 - *Plot 5: polarAnnulus*)  
   As a broader view of pollutants over time, we can create a polar annulus to view pollution on various time scales

Begin with these 3 kinds:

polarAnnulus(data, poll = "nox", period = "season", main = "Season")

polarAnnulus(data, poll = "nox", period = "weekday", main = "Weekday")

polarAnnulus(data, poll = "nox", period = "hour", main = "Hour")

1. First, how are these three most obviously different (think about what different arguments they have)?
2. What do the colors correspond to? What does the location of that color correspond to?
3. Why do we need to make a ring (an annulus) rather than just a full circle (what information might be lost as we approach the center of the circle and the circumference decreases)?
4. (Slide 20 - *Plot 6: scatterPlot*)  
   Finally, we’ll create a common plot for any science, the scatter plot. This command exists in base R, but openair has embellished it for our use.

Use the command scatterPlot to compare “no” on the x axis and “nox” on the y axis

scatterPlot(data, x = "...", y = "...")

It may be valuable to subset your data from a specific year first. You can do this by selecting the rows manually as you learned in the introduction to R, or using openair’s handy “selectByDate” function. Because of the full date datatype, it is more difficult to get the right year using logical selection as we did previously

data2000 <- selectByDate(data, year = 2000)

(Slide 21 - *scatterPlot (cont)*)  
Even with this reduction, our scatterPlot may be messy because of overlapping points. We can make our scatterplot a hexbin type to log these overlapping points differently.

Add to your scatterPlot command the arguments method="hexbin" and col="jet" to do this.

1. How did “hexbin” change the scatter plot? How is this helpful?
2. Experiment with other color scales (try help(openColours) and see the available “schemes”). Which scale is your favorite for a heatmap?
3. Do you see any trend from the scatter plot? If so, what? You might consider if the variables have a linear relationship, if that relationship is positive or negative, etc.
4. (Slide 23 - *Closing Exercise: Your Choice!*)  
   As a closing exercise, go to the openair manual and find one plot that we **have not** seen or used today. Run the command to produce that plot for your data, paste it, and describe what the plot shows. The link to the manual is [here](https://bookdown.org/david_carslaw/openair/), and remember that all openair functions use similar internal arguments.