Figure 1: Schematic of Honeywell Particulate Matter Sensor. A laser light source illuminates a particle as it is pulled through the detection chamber. As particles pass through the laser beam, the light reflects off the particles and is recorded on the photo or light detector. The light is then analyzed and converted to an electrical signal to calculate particle concentration.

1 Introduction

1.1 How is particulate matter analyzed?

The PM sensor has several components to measure the particles in the air – remember, we don't know what the make up of these particles might be – but we can get an estimate of their size distribution. We have written the code to collect particles smaller than 1μm, 2.5μm and 10μm.

Figure 1 is a schematic of the particle sensor made by Honeywell (HPM Series — Particulate Matter Sensors 32322550HP). The one we are using is made by Plantower. As a Chinese company, most of their literature is in Chinese, so I didn't find something that I could interpret for us.

Our sensors can generate several categories of PM data. Using a laser and light defraction, the number and size of the particles can be estimated (Figure 2).

2 Data Collection

2.1 What data is collected by the PMS5003?

The PMS5003 generates the following data:

Data 1 refers to PM1.0 concentration unit μg/m^{3}CF=1standard particle)[1]

Data 2 refers to PM2.5 concentration unit μg/m^{3}CF=1standard particle

Data 3 refers to PM10 concentration unit μg/m^{3}CF=1standard particle

---

[1]The Asterisk on Data 1 and Data 4 is defined on the same page as "Note: CF=1 should be used in the factory environment"
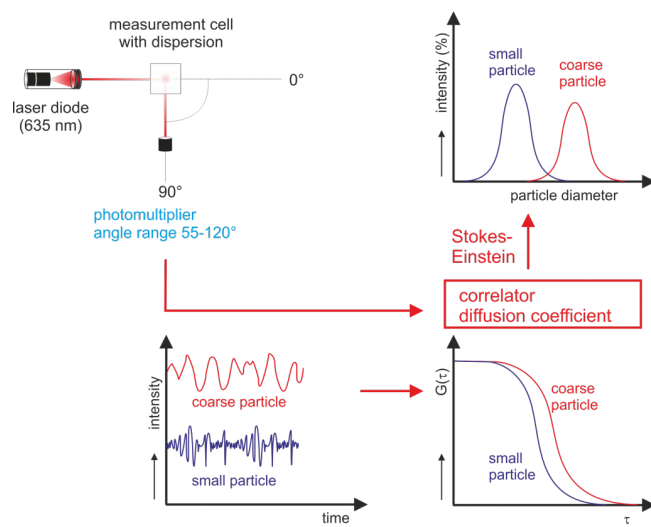
Figure 2: In particle size measurements using light scattering (DLS), a laser beam is scattered on very small, finely dispersed particles in a highly diluted matrix. The scattered light of each particle will then interfere with each other. Since the particles constantly change locations, the position of the scattering centers changes with respect to each other and the interferences lead to small fluctuations in scattering intensity (this explains the name "dynamic" light scattering).

Data 4 refers to PM1.0 concentration unit μg/m^{3}under atmospheric environment

Data 5 refers to PM2.5 concentration unit μg/m^{3}under atmospheric environment

Data 6 refers to concentration unit (under atmospheric environment) μg/m^3

Data 7 indicates the number of particles with diameter beyond 0.3 μm in 0.1 L of air.

Data 8 indicates the number of particles with diameter beyond 0.5 μm in 0.1 L of air.

Data 9 indicates the number of particles with diameter beyond 1.0 μm in 0.1 L of air.

Data 10 indicates the number of particles with diameter beyond 2.5 μm in 0.1 L of air.

Data 11 indicates the number of particles with diameter beyond 5.0 μm in 0.1 L of air.

Data 12 indicates the number of particles with diameter beyond 10 μm in 0.1 L of air

The first set are labeled "standard", while the second set are labeled "atmospheric environment". What do these measure and how do the "standard" ones differ from the "atmospheric environment" ones?

It has to do with the density of the air used for calculations. "Standard" refers to the concentration "corrected" to the "standard atmosphere" which in the US is DEFINED as "having a temperature of 288.15 K at the sea level 0 km geo-potential height and 1013.25 hPa". NOTE: 288.15 K equals 25 C and Pa are "pascals" which is a meausure of pressure.

On the other hand, the "ambient conditions" are just as the air is "now" (whatever temperature and pressure there is). Now what does that mean. . .

Air being a gas, it is compressible which means that it changes its volume when the pressure changes so when you report concentrations as mass per volume of air it is relevant at what pressure that volume is calculated. For example, if you have a bunch of particles rising in the air in a bubble (no loss of particles, no addition, they're just riding a bubble up in the air) then, as they rise, the pressure drops so what was 1cc at the ground it is now 2cc so the concentration of what was in the bubble is now half without anything actually changing other than the ambient pressure. So, it is common to report concentrations (of anything) as "x mg per standard m3" and because we scientist don't like to write much (current example excluded) you'll usually see the "standard" being dropped because it is "implicit".

For gases it is also common to report concentrations as "parts per million" or "ppm" and that metric is independent of the volume of air as it represents

the number of molecules of the gas in a million molecules of air (including the target gas).

In conclusion, I use the "standard" readings for reporting but keep the "ambient conditions" for analysis. I haven't used these things in high altitudes so for all my deployments the standard and ambient are very similar. I have written code to extract the standard values only – but you can certain add the ambient if you want to see if there is a difference – could be important if you are above 1000 ft in elevation.

3   Collecting the data

3.1   Extract Data From Pi

Once the data have been collected, you can extract from the Pi – Unfortunately, there are several ways to do this and one of them are easier using the VNC interface.

3.1.1   Using RealVNC to transfer data files

1. Use RealVNC and VNC into the Raspberry Pi using the Pi's IP adress in the RealVNC search toolbar.

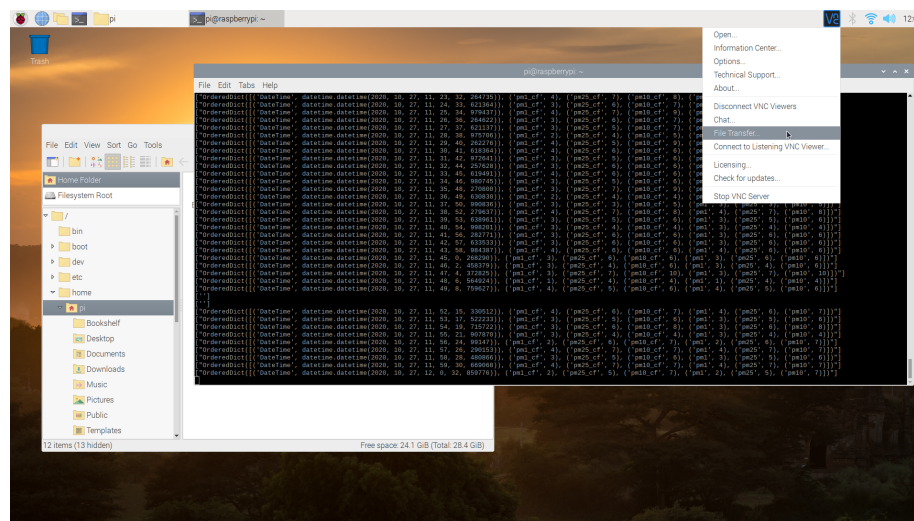2. Controlling the Raspberry Pi, right click on the VNCServer icon in the toolbar in the upper left near the clock.

Figure 3:

3. Select **"File Transfer..."**

4. In the "File Transfer" popup window, select **"Send Files"** in the lower left.

Figure 4:

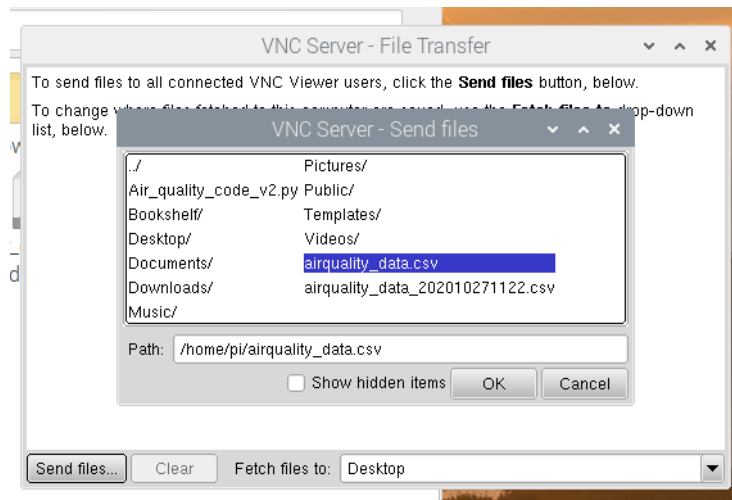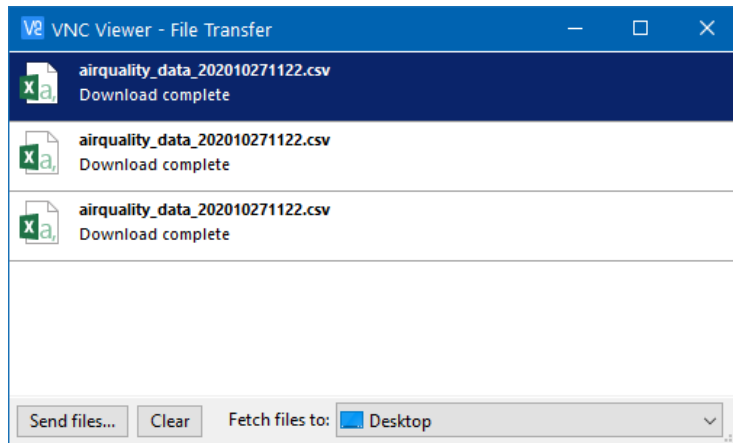5. Then navigate to the file you want to send in the "Send Files" popup and click **"OK"**.

Figure 5:

6. There should be another popup, on your main computer, that will show download manager with your file in it. By default, RealVNC transfers files to your desktop.

3.1.2 Alternative Methods For PCs

Alternatively, there serveral other alternatives. For example, this webpage describes the issues and 5 not great options to get data.

After several attempts to create a cloud folder that you could upload, I found it was far to slow and unreliable. Thus, I recommend using Putty and the following line commands:

```
psftp> open Pi@XXX.XXX.XXX.XXX
```

and you will be prompted to enter the password. Once you are in, you can see your files by entering

```
psftp>ls *.csv
```

where all your csv files should be shown.

Then you'll need to change your local drive (the default one is write protected, creating a signficant obstacle. So, use

```
psftp>lpwd
```

to see your default directly. We are doing to use the download directory and we can change the directory using:

```
psftp>lcd c:/Downloads
```

Then we will get the files...

```
psftp>get airquality_data.csv
```

and it will be transfered to your local computer... ! Yay! It only took me 5 hours to figure out and hopefully for you less than 5 min.

3.1.3   FOR Macs

3.2   Processing the data

We create a script to process the data and allow you to create a reasonably user-friendly dataframe to analyze the data.

However, before running the code below, we need to remove all the double quotes, which get in the way of how "csv" files are read.

I opened the document up in rstdudio and did a search for " and replaced with nothing, i.e. don't enter anythning. Then replace all.

```
filepath.csv = "/home/CAMPUS/mwl04747/github/EJnPi/data/Air_Quality.csv"
filepath.csv = "/home/CAMPUS/mwl04747/github/EJnPi/data/Complete_airquality_data2.csv"
rawdata = read.csv(filepath.csv)

names(rawdata)= c("X1", "X2", "Month", "Day", "Hour",
    "Minute", "Second", "X3", "X4", "pm1_cf", "X5", "pm25_cf", "X6",
    "pm10_cf", "X7", "pm1", "X8", "pm25", "pm10.", "X9")

rawdata$pm1_cf = as.numeric(gsub('[()]', '', rawdata$pm1_cf))
rawdata$pm25_cf = as.numeric(gsub('[()]', '', rawdata$pm25_cf))
rawdata$pm10_cf = as.numeric(gsub('[()]', '', rawdata$pm10_cf))
as.Date(with(rawdata, paste("2020", Month, Day,sep="-")), "%Y-%m-%d")[1]

## [1] "2020-10-21"

library(lubridate)

##
## Attaching package:  'lubridate'
## The following object is masked from 'package:base':
##
##     date

rawdata$DateTime = with(rawdata, ymd_hms(paste("2020", Month,
  Day, Hour, Minute, Second, sep= '-')))
```

As always, I check my data!

```
rawdata[sample(1:nrow(rawdata), 5),] # random 10 rows, all columns

##                               X1                      X2 Month Day Hour Minute
## 912   OrderedDict([('DateTime'  datetime.datetime(2020    10  22   23     37
## 497   OrderedDict([('DateTime'  datetime.datetime(2020    10  21   23     42
## 452   OrderedDict([('DateTime'  datetime.datetime(2020    10  21   22     47
## 1610  OrderedDict([('DateTime'  datetime.datetime(2020    10  25   16      5
## 993   OrderedDict([('DateTime'  datetime.datetime(2020    10  23    6     45
```

```
##       Second        X3        X4 pm1_cf          X5 pm25_cf          X6 pm10_cf
## 912       35 547808)) ('pm1_cf'    11 ('pm25_cf'       21 ('pm10_cf'       26
## 497       20 110389)) ('pm1_cf'    36 ('pm25_cf'       75 ('pm10_cf'       95
## 452        7 318497)) ('pm1_cf'    38 ('pm25_cf'       72 ('pm10_cf'       86
## 1610      19 461461)) ('pm1_cf'     1 ('pm25_cf'        2 ('pm10_cf'        3
## 993       15 728351)) ('pm1_cf'    23 ('pm25_cf'       44 ('pm10_cf'       50
##           X7  pm1       X8 pm25    pm10.    X9              DateTime
## 912  ('pm1'  11) ('pm25'  21) ('pm10'  26)]) 2020-10-22 23:37:35
## 497  ('pm1'  28) ('pm25'  51) ('pm10'  71)]) 2020-10-21 23:42:20
## 452  ('pm1'  29) ('pm25'  50) ('pm10'  66)]) 2020-10-21 22:47:07
## 1610 ('pm1'   1) ('pm25'   2) ('pm10'   3)]) 2020-10-25 16:05:19
## 993  ('pm1'  22) ('pm25'  38) ('pm10'  48)]) 2020-10-23 06:45:15

# Remove Variables
cleandata = subset(rawdata, select=c(DateTime, pm1_cf, pm25_cf, pm10_cf))
```
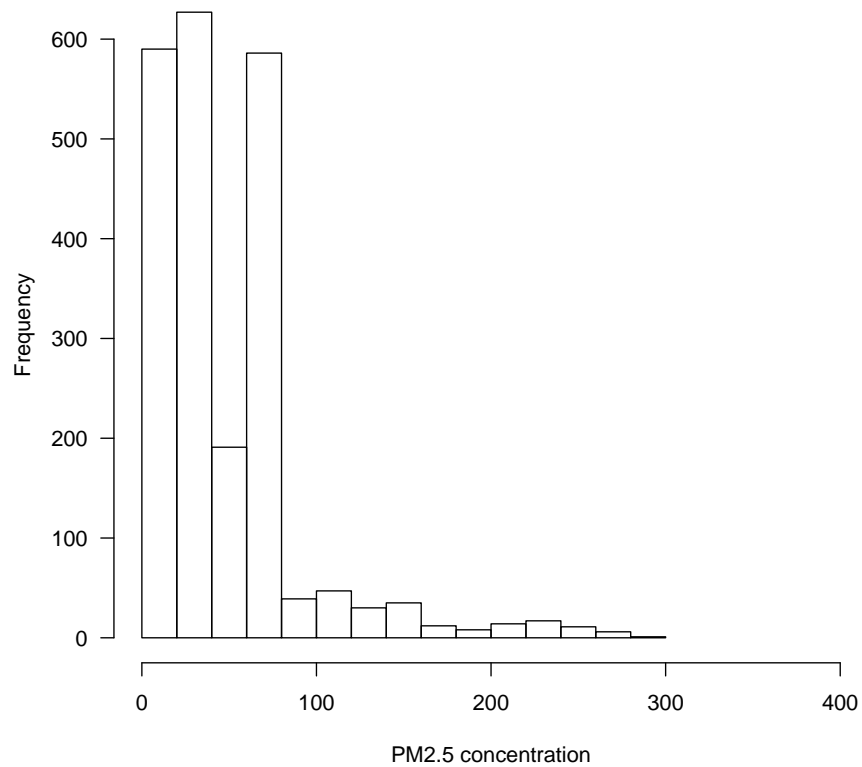
3.3   Plot Data

As usual, it's a good idea to get a sense of the variability of the data.

```
hist(cleandata$pm25_cf, xlab="PM2.5 concentration",
    main="Histogram of PM2.5 in Claremont", las=1, xlim=c(0,400))
```
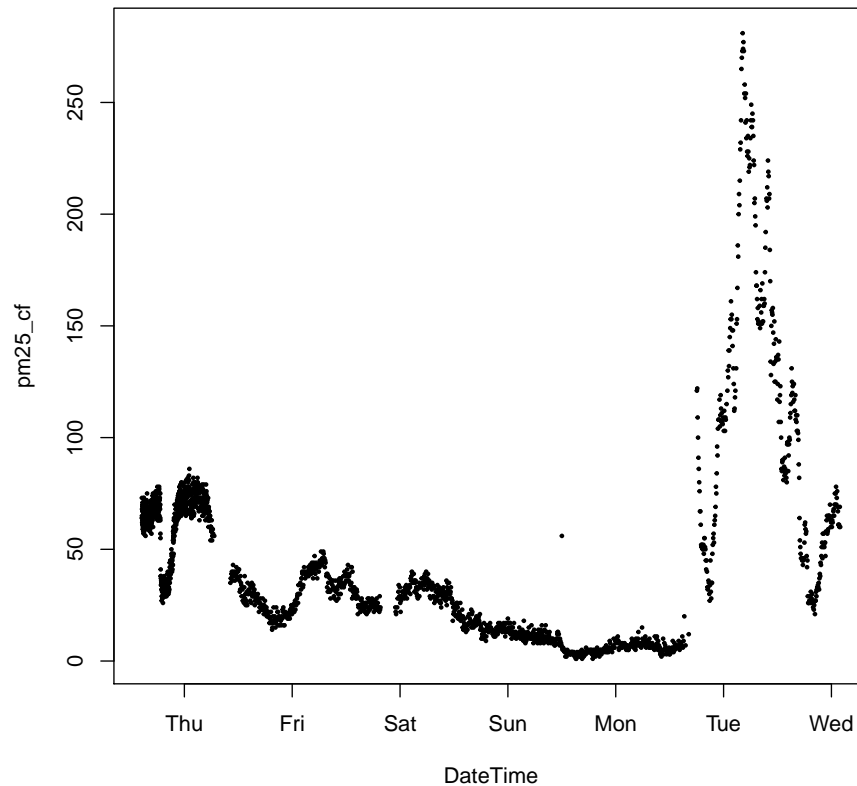
Histogram of PM2.5 in Claremont

In my case, I have a very odd bimodal distribution and an long tail to boot. So, I am going plot on a time series to see if I can figure out what in the world is going on.

```
plot(pm25_cf~DateTime, data=cleandata, pch=20, cex=.5)
```

Except for some breaks in the data, we can certainly see a diurnal pattern, where the concentrations seem highest after midnight.

Let's see if we can add times to the plot!

```
plot(pm25_cf~DateTime, data=cleandata, pch=20,
    cex=.5, las=1, ylab="PM2.5", xlab="Time",
    ylim=c(0,300))#, xaxt='y')
library(Hmisc)

## Loading required package:   lattice
## Loading required package:   survival
## Loading required package:   Formula
## Loading required package:   ggplot2
##
## Attaching package:   'Hmisc'
## The following objects are masked from 'package:base':
##
```

```
##    format.pval, units
abline(h=35, col="red")

rect(as.POSIXct("2020-10-21 7:00:00 UTC"), -20, as.POSIXct("2020-10-21 18:00:00 UTC"), 330,

rect(as.POSIXct("2020-10-22 7:00:00 UTC"), -20, as.POSIXct("2020-10-22 18:00:00 UTC"), 330,

rect(as.POSIXct("2020-10-23 7:00:00 UTC"), -20, as.POSIXct("2020-10-23 18:00:00 UTC"), 330,

rect(as.POSIXct("2020-10-24 7:00:00 UTC"), -20, as.POSIXct("2020-10-24 18:00:00 UTC"), 330,

rect(as.POSIXct("2020-10-25 7:00:00 UTC"), -20, as.POSIXct("2020-10-25 18:00:00 UTC"), 330,

rect(as.POSIXct("2020-10-26 7:00:00 UTC"), -20, as.POSIXct("2020-10-26 18:00:00 UTC"), 330,

rect(as.POSIXct("2020-10-27 7:00:00 UTC"), -20, as.POSIXct("2020-10-27 18:00:00 UTC"), 330,

rect(as.POSIXct("2020-10-28 7:00:00 UTC"), -20, as.POSIXct("2020-10-28 18:00:00 UTC"), 330,
```
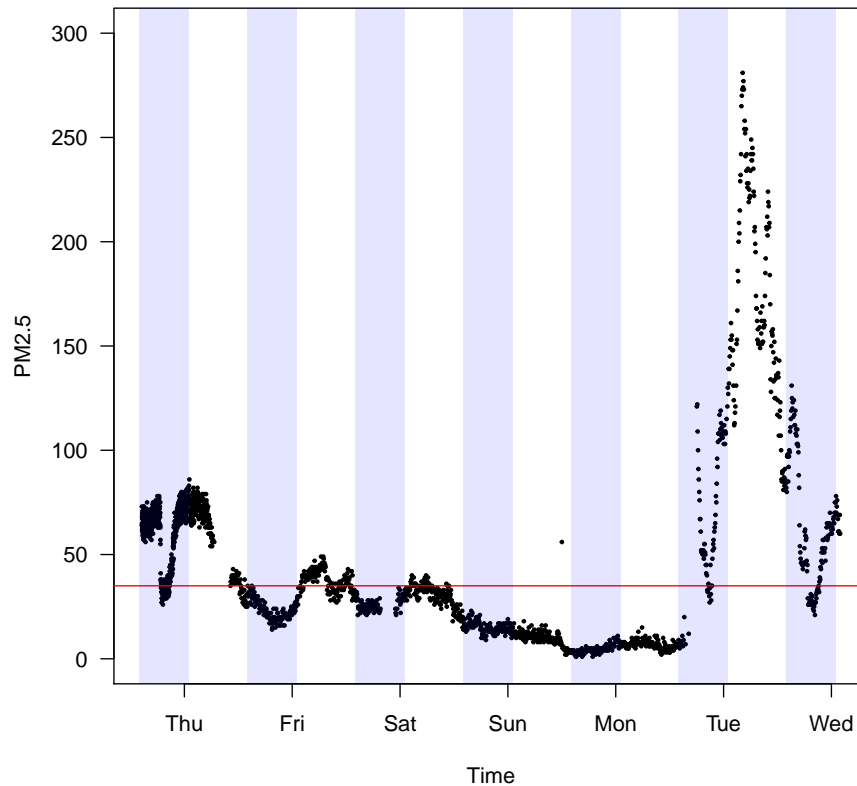
```
#minor.tick(nx=4, tick.ratio=1)
#axis.lab=c(as.Date(c("2020-10-20", "2020-10-21", "2020-10-24")))
#axis(1, cleandata£DateTime, format(cleandata£DateTime, "%H"), cex.axis = .7,  tck=-.05)


#seq(min(cleandata£DateTime), max(cleandata£DateTime), by="days")

#rect(as.POSIXct("2020-10-21 8:00:00 UTC"), 0, as.POSIXct("2020-10-22 18:00:00 UTC", 300)

#axis.Date(3, at=seq(min(cleandata£DateTime), max(cleandata£DateTime), by="days"), format="
#axis(3, cleandata£DateTime, format(cleandata£DateTime, "%a"), cex.axis = .7,  tck=0)#, at=
#axis.Date(1, at = seq(as.Date("2020-10-21 14:32:25 UTC"), as.Date("2020-10-2714:32:25 UTC
#axis.Date(1, at = seq(as.Date("2001/1/1"), max(random.dates)+6, "days"), labels = FALSE, t

#r <- as.POSIXct(round(range(time), "hours"))
#axis.POSIXct(1, at = seq(r[1], r[2], by = "hour"), format = "%H")
```

Not perfect, but good enough – sows a that Claremont's Air Quality was terrible on Wednesday night and Thursday morning.

4 Comparing Data to EPA Stations

4.1 Using previous collected data

```
# TBD
```

4.2 Comparing μ to EPA Data

The population mean (monthly average from the EPA data) is a parameter estimate. We can determine if the values you collected fall within the parameter estimate and it's 95% confidence intervals.

```
mean(cleandata$pm1_cf)
```

```
## [1] 28.02078
```

Okay, now let's talk about signficant figures. The Pi only reports to the nearest whole value. Read about the rules in Wikipedia – https://en.wikipedia.org/wiki/Significant_figures. In general, we should report to the whole number as well. However, some statiticians suggest that the mean and standard deviation can include one extra significant figure, allowing the reader to see the "directionality" of the mean toward the precision. However, for our purposes, I suggest we follow the standard rule and round to the nearest whole number and avoid implying additional precision.

4.3 More Intricate Analysis

You could look to see if the there is a different during the time of the day or day of the week. You could move you sensor around in and outside your house and see if air quality is better / worse or the same....

We can help you with r code if you'd like to follow up with these additional questions.