ach_estimation

March 12, 2021

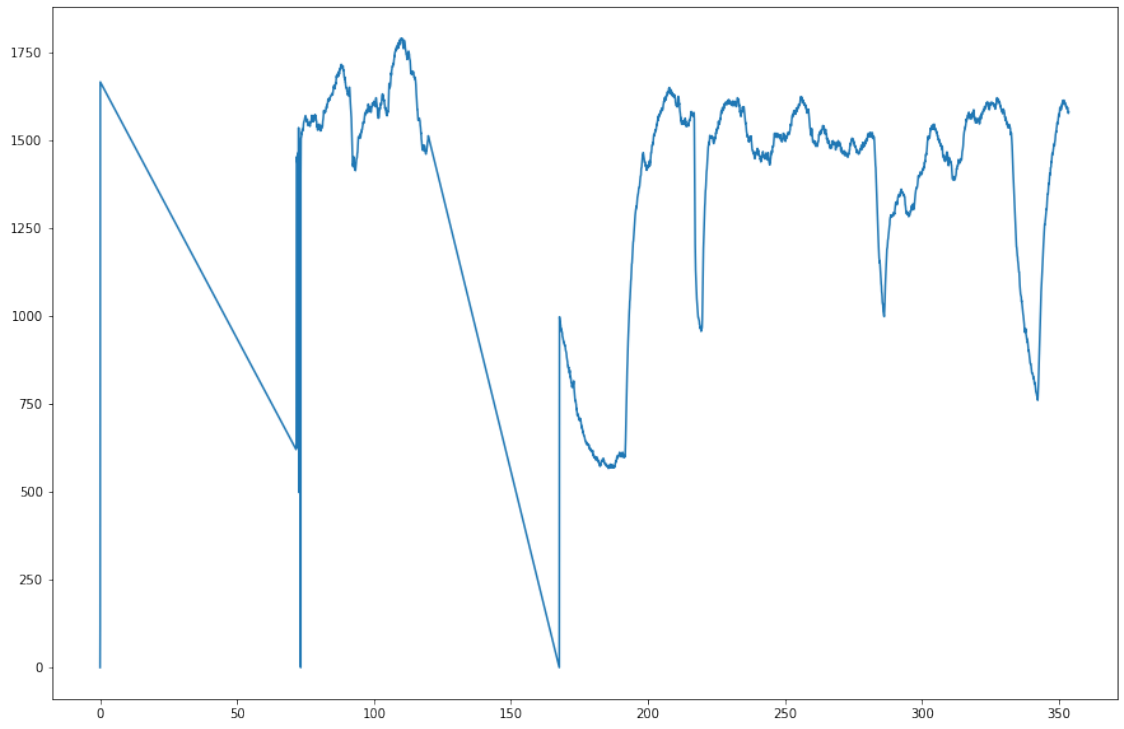
1 Estimating Air Changes Per Hour (ACH)

After a significant amount of CO2 is suddenly injected into a space (e.g. a cannister of CO2 is released for a short time, or a vinegar + baking soda reaction takes place), a CO2 sensor reading in the room will typically rise initially, then exponentially decay at a rate that is related to the effectiveness of ventilation in the room.

Let's begin with our initial dataset, and plot the co2 value vs the time (in hours):

[92]:
```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import math
import numpy as np
from scipy import stats

feed_a_pubkey = "jdwktcu9xt5c"
base_url = "http://co2data.pvos.org/data/"
feed_a_data = pd.read_csv(base_url+feed_a_pubkey+"/csv/")

co2=feed_a_data['co2'].to_numpy()

# get a time array in units of hours
t_hours=(pd.to_datetime(feed_a_data['created']).astype(int)/10**9)/3600
t_hours=(t_hours-t_hours[0]).to_numpy()

plt.figure(figsize=(15, 10))
plt.plot(t_hours,co2)
```

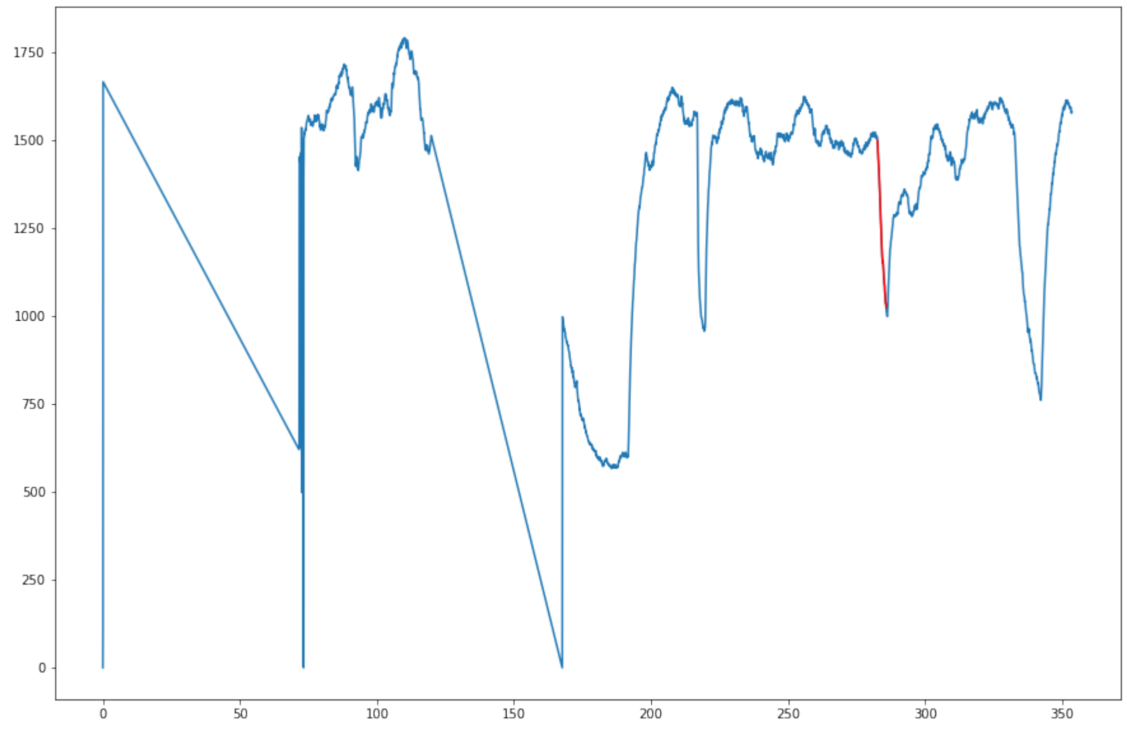
[92]:
```
[<matplotlib.lines.Line2D at 0x7f942d09a1d0>]
```



We'll now create a subset of this data, focusing on a particular exponential decay event. We'll call the array of CO2 values for this subset y, and the array of time values for this subset t, and plot the resultant subset on top of the original dataset.

[109]:
```
index_min = 1961
index_max = 2000

y=co2[index_min:index_max]
t=t_hours[index_min:index_max]

plt.figure(figsize=(15, 10))
plt.plot(t_hours,co2)
plt.plot(t,y,'r')
```

[109]:
```
[<matplotlib.lines.Line2D at 0x7f942cc8f7f0>]
```



The 'time constant' τ (measured in hours) for this exponential decay can -- under certain conditions [REF] -- be considered a rough estimate for the 'Air Changes Per Hour' in the room. That is,

$ACH \approx \tau$

We can estimate τ (and thus the ACH for the room) by: 1. Normalizing y values, so that they range between 0 and 1; 2. Normalizing the t values, so that they start at time = 0; 3. Linearizing the data by taking the natural log 4. Performing linear regression to extract the time constant τ.

These steps are explained and illustrated below, assuming that (as in the above example), you have a subset of your CO2 data in which you assume an exponential decay has occured, where: - t = the time values of this subset of the data, in hours - y = the co2 values of this subset of the data, in PPM

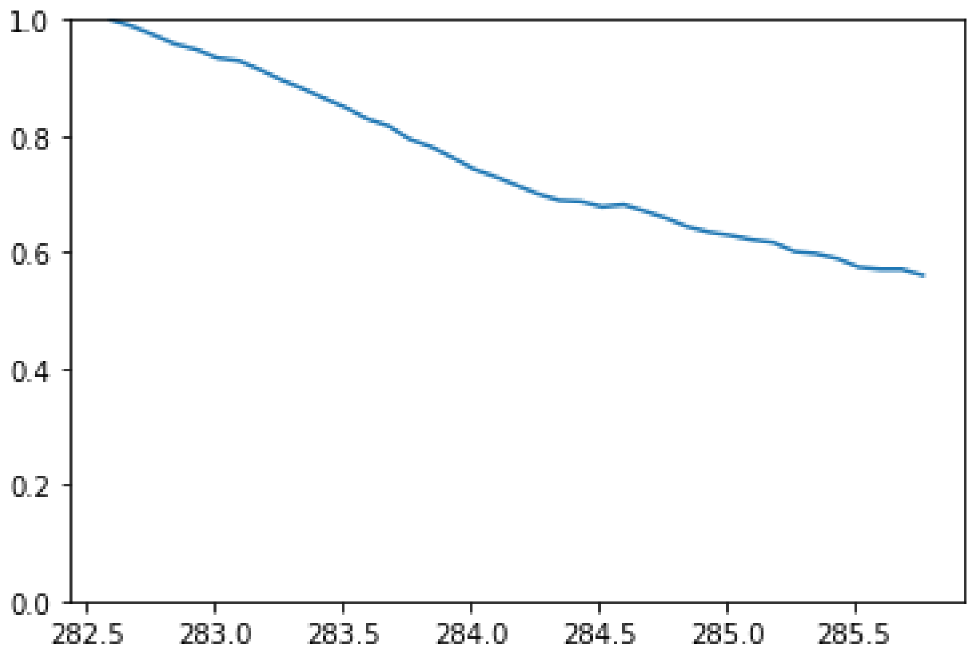
1.1 1. Normalizing the CO2 values (y)

We can normalize the CO2 values (y) by first subtracting off the value to which the CO2 is expected to decay -- typically, the ambient CO2 (approx 420 ppm), $y_{ambient}$ -- and dividing by the maximum y value (i.e., the first y value in the decay, y_0:

$y_{norm} = \frac{y - y_{ambient}}{y_0 - y_{amplitude}}$

[99]:
```
y_ambient=420
y_norm = (y-co2_outside)/(y[0]-y_ambient)
```

```
axes = plt.gca()
axes.set_ylim([0,1])
plt.plot(t,y_norm)
plt.show()
```

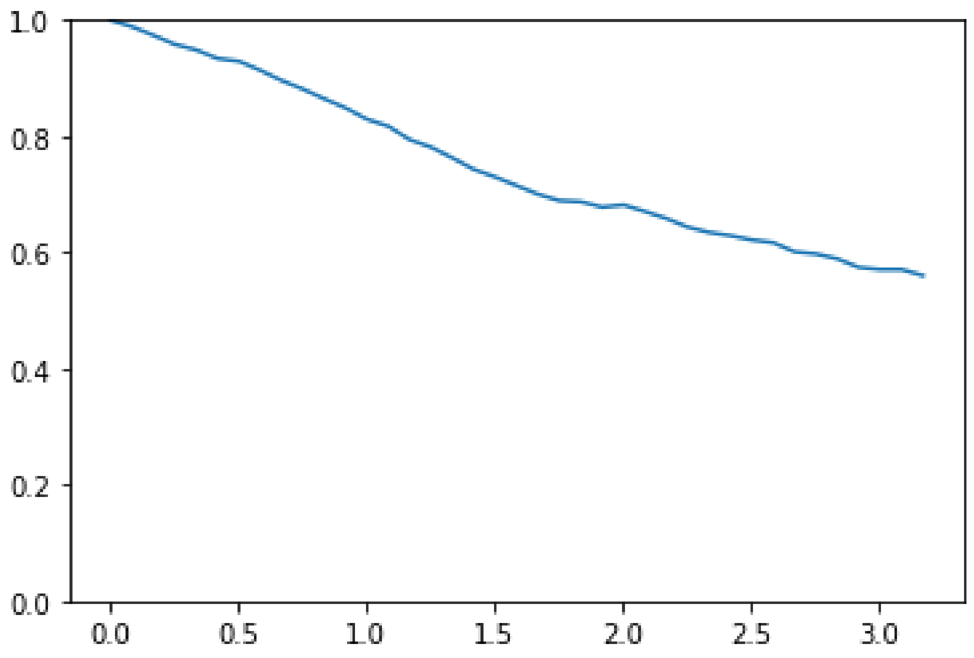


1.2 Normalizing the time values, t

Assuming the time values are in hours, we consider the first data point of our subset to be $t = 0$; we therefore need to subtract off our time at $t = 0$ from our t values:

$t_{norm} = t - t_0$

[101]:
```
t_norm=t-t[0]

axes = plt.gca()
axes.set_ylim([0,1])
plt.plot(t_norm,y_norm)
plt.show()
```



1.3 Finding the time constant, τ

With the data normalized in this way, the equation for the decay becomes:

$y_{norm} = e^{-t_{norm}/\tau}$

Where τ is the time constant of the decay (the value we seek).

Taking the natural log of both sides, we get:

$\ln(y_{norm}) = \frac{-t_{norm}}{\tau}$

Note that this equation then has the form of a straight line,

$z = m * x$

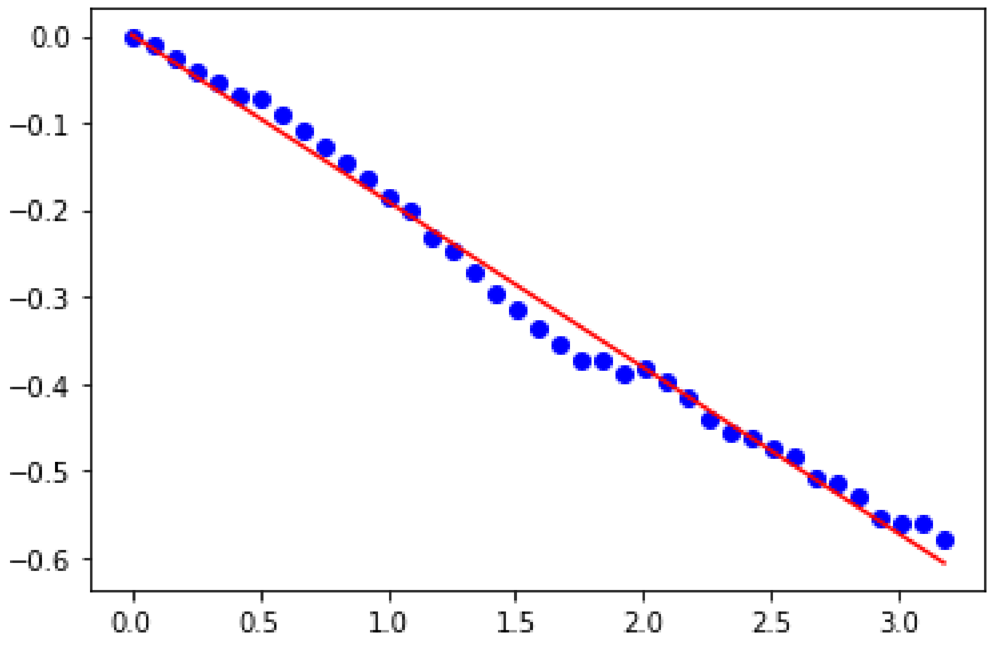
where the slope m is the negative inverse of τ, i.e.:

$\tau = -m^{-1}$

We can then perform a linear fit of $\ln(y_{norm})$ vs. $\frac{-t_{norm}}{\tau}$ to find τ.

[108]:
```
# some plotting housekeeping
fig=plt.figure()
ax=fig.add_subplot(111)

# take the natural log of the normalize y data
log_y_norm = np.log(y_norm)
```

```
# perform a linear regress on the dataset: log_y_norm vs t_norm
slope, intercept, r_value, p_value, std_err = stats.
 ↪linregress(t_norm,log_y_norm)

# plot log_y_norm vs t_norm, along with our linear fit
plt.plot(t_norm,log_y_norm,'bo',label="Data")
plt.plot(t_norm,slope*t_norm+intercept, 'r-',label="Polyfit")

plt.show()

tau = round(-1/slope,2)

print("tau (ACH) =",tau)
```



```
tau (ACH) = 5.22
```