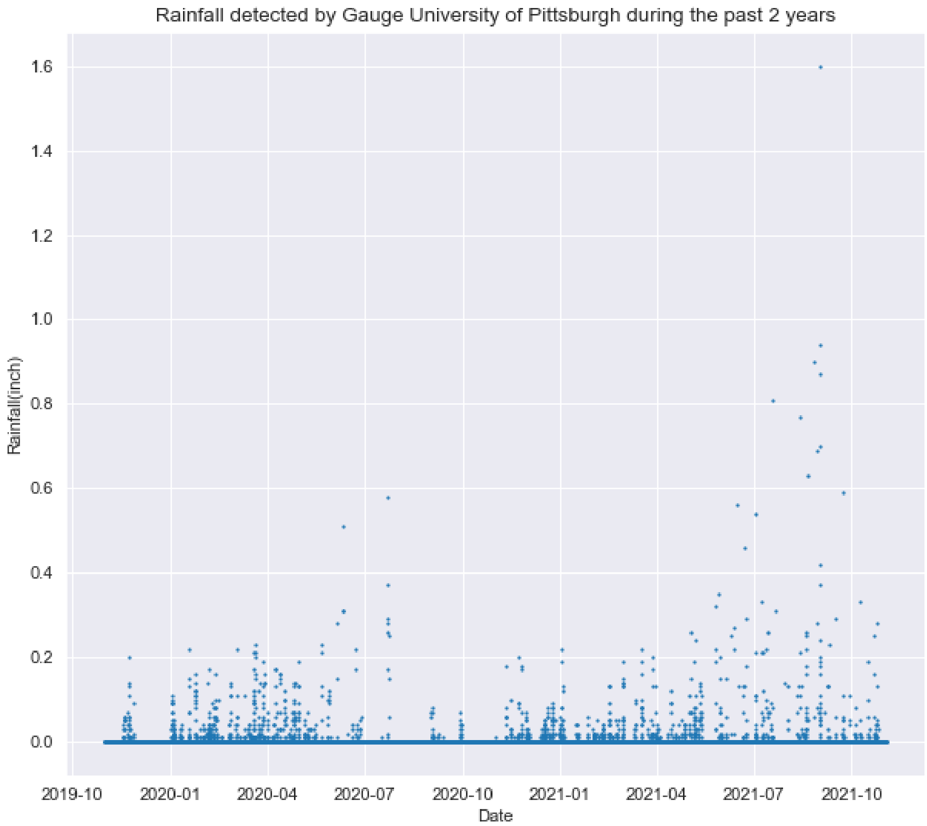
1. Warming up to Time-Series Data Again (15%)

a.

```
In [154]: result['Date'] = result['Date'].astype('datetime64')
          result = result.sort_values(by="Date")
          result = result.set_index("Date")
          result_hourly = result.resample("1H").sum()
```

```
In [155]: import matplotlib.pyplot as plt
          from matplotlib.pyplot import figure
          %matplotlib inline

          fig = figure(figsize = (9,8),dpi = 80)
          x = np.array(result_hourly.index)
          y = result_hourly['University of Pittsburgh']
          y[y<0]=0
          y = np.array(y)
          plt.scatter(x,y,s=1)
          plt.xlabel("Date")
          plt.ylabel("Rainfall(inch)")
          plt.title("Rainfall detected by Gauge UPitts during the past 2 years")
          plt.show()
          fig.savefig("rainfall.png")
```




b.

In the rainfall data points, we can see that there seems to be a daily variation in the amount of rainfall. Therefore, to design an equation to model the linear regression, I would propose $y_i = \alpha * t_i + n_i$ where $y_i$ is the recorded rainfall, $\alpha$ is the parameter, $t_i$ is the time instance with a time sampling interval of 1 day, and $n_i$ is the noise recorded at the time instance.

After setting up the equation, we can now construct the matrix of parameters using

$\alpha$ and y vector with $y_i$. Obtaining the estimated linear regression model through the $\alpha$ matrix and y vector, we can than calculate the residuals, and the mean and standard derivation of the residuals. To remove the outliers, we may set a range using the mean and standard derivation of residuals. Using the Chauvenet's criterion, the range is set to be plus and minus three times the standard deviation from the mean, any data point outside this range is considered an outlier. We will repeat the process until all the outliers are identified and removed.

2. Water in the foundation: yikes! (60%)

(a)

```
In [575]: data_matrix = np.array(data[['Water Level(m)','Temperature(C)']].values,'float')
          beta_vector = np.ones(3651)
          alpha_vector = np.arange(0,3651,1)*144
          A_matrix = np.column_stack((alpha_vector,beta_vector,data_matrix))
          A_matrix.reshape(3651,4)
          pseudo_inverse_A_Matrix = np.linalg.pinv(A_matrix)

In [576]: y_vector = np.array(data['Strain(micro-strain)'].values,'float')

In [577]: W_Matrix = np.matmul(pseudo_inverse_A_Matrix,y_vector)

In [578]: print('Alpha=',W_Matrix[0],' micro-strain/min')
          print('Beta=',W_Matrix[1],'micro-strain')
          print('Gamma=',W_Matrix[3],'micro-strains/meter')
          print('Delta=',W_Matrix[2],'micro-strains/Celsius')
```

$\hat{\alpha}$= -0.00042984263809532115 micro-strain/min

$\hat{\beta}$= -439.50238120742614 micro-strain

$\hat{\gamma}$= 1.4432743355313284 micro-strains/Celsius

$\hat{\delta}$= -53.15723364050697 micro-strains/meter

(b)

```
In [579]: q_3000 = W_Matrix[0]*2999*144+W_Matrix[1]
          print('q_3000 =',q_3000, 'micro-strain')
```

q_3000 = -625.1325035247191 micro-strain

(c)

```
In [580]: q_5y = W_Matrix[0]*5*365*24*60+W_Matrix[1]
          print('q_5y=',q_5y, 'micro-strain')
```

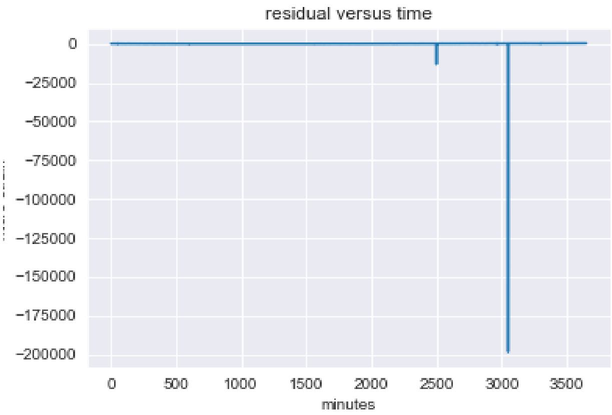
q_5y= -1569.1288341219304 micro-strain

(d)

```
In [680]: # y is the predcition afer linear regression
          y = np.matmul(A_matrix,W_Matrix)
          residual = data['Strain(micro-strain)']-y

In [682]: residual_plot = sns.lineplot(data = residual)
          sns.set_style("darkgrid")
          residual_plot.set_xlabel("minutes")
          residual_plot.set_ylabel("micro-strain")
          residual_plot.set_title("residual versus time")
          residual_plot.figure.savefig("residual versus time")
```

Lih-Shyan (Chris) Fang
lihshyaf




```
In [483]: print('The mean of the residual =',residual.mean())
          residual_mean = residual.mean()
          print('The std of the residual =',residual.std())
          residual_std = residual.std()
```

The mean of the residual = 1.6778227974623325e-12

The std of the residual = 3293.952960065999

(e)

```
In [669]: # iterate the process
outlier_exist = True
drop_index_total=[]
while outlier_exist:
    number_of_rows = len(data)
    # build A_matrix for linear regression
    data_matrix = np.array(data[['Water Level(m)','Temperature(C)']].values,'float')
    beta_vector = np.ones(number_of_rows)
    alpha_vector = np.arange(0,number_of_rows,1)*144
    A_matrix = np.column_stack((alpha_vector,beta_vector,data_matrix))
    A_matrix.reshape(number_of_rows,4)
    # find the psuedo inverse of A_matrix
    pseudo_inverse_A_Matrix = np.linalg.pinv(A_matrix)
    # find the y vector
    y_vector = np.array(data['Strain(micro-strain)'].values,'float')
    # calculate the parameters for linear regression
    W_Matrix = np.matmul(pseudo_inverse_A_Matrix,y_vector)
    # predict y
    y = np.matmul(A_matrix,W_Matrix)
    # calculate residuals
    residual = y-data['Strain(micro-strain)']
    # mean and std of residuals
    print('The mean of the residual = ',residual.mean())
    residual_mean = residual.mean()
    print('The std of the residual = ',residual.std())
    residual_std = residual.std()
    # find the index with outlier datapoints and store it in drop_index[]
    drop_index = []
    threshold_high = residual_mean+3*residual_std
    threshold_low = residual_mean-3*residual_std
    for i in range(0,number_of_rows):
        if i in drop_index_total:
            continue
        if residual[i] > thres_hold_high or residual[i] < threshold_low:
            drop_index.append(i)
    #keep track of all the index dropped throught the iteration
    for i in range(0,len(drop_index)):
        drop_index_total.append(drop_index[i])
        print('The water level:',data['Water Level(m)'][drop_index[i]],
              ',the Temperature:',data['Temperature(C)'][drop_index[i]],',the row is:',drop_index[i])
    #data set drop index
    data.drop(drop_index,inplace=True)
    print('---------------------------------------------------')
    #if no more outliers than exit the iteration
    if not drop_index:
        print('no more outliers')
        outlier_exist = False
```

The mean of the residual = 1.6778227974623325e-12

The std of the residual = 3293.952960065999

The water level: -9.54688299960992, the Temperature: 4.17978837213146, the row is: 2499

The water level: -8.37139121446426, the Temperature: 2.10399519785036, the row is: 3049

-----------------------------------------------------

The mean of the residual = -5.787398817541758e-14

The std of the residual = 14.009994054339375

The water level: -10.1939760133644, the Temperature: 10.5216587260785, the row is: 598

The water level: -10.2119567730395, the Temperature: 10.8844563878947, the row is: 2199

The water level: -9.8411652681218, the Temperature: 2.07317374340667, the row is: 2959

The water level: -9.08461406356181, the Temperature: 1.39630567837468, the row is: 2969

The water level: -8.68034346724605, the Temperature: -1.51614060916072, the row is: 3199

The water level: -8.75525517694182, the Temperature:1.06486125085496, the row is: 3299

-----------------------------------------------------

The mean of the residual = 3.812267548032659e-14

The std of the residual = 12.456866906148734

The water level: -11.3933741762572, the Temperature: 18.4827137833432, the row is: 1419

The water level: -10.1807982237015, the Temperature: 8.58909008979396, the row is: 2426

-----------------------------------------------------

The mean of the residual = -1.220861124667725e-14

The std of the residual = 12.42769063549792

-----------------------------------------------------

no more outliers

```
In [503]: print('Alpha=',W_Matrix[0],' micro-strain/min')
          print('Beta=',W_Matrix[1],'micro-strain')
          print('Gamma=',W_Matrix[3],'micro-strains/Celsius')
          print('Delta=',W_Matrix[2],'micro-strains/meter')
```

$\hat{\alpha}$= -3.99452445236156e-05 micro-strain/min

$\hat{\beta}$= 12.8355141371718 micro-strain

$\hat{\gamma}$= 1.5261812447776726 micro-strains/Celsius

$\hat{\delta}$= -0.958624064677513 micro-strains/meter

(f)

```
In [550]: q_3000 = W_Matrix[0]*2000*144+W_Matrix[1]
          print('q_3000 =',q_3000, 'micro-strain')
          q_5y = W_Matrix[0]*5*365*24*60+W_Matrix[1]
          print('q_5y=',q_5y, 'micro-strain')
```

Lih-Shyan (Chris) Fang
lihshyaf

q_3000 = -4.415079381818739 micro-strain

q_5y= -92.14058847089 micro-strain (g)

(g)

```
In [555]: W_Matrix
          print('Alpha=',W_Matrix[0],' micro-strain/min')
          print('Beta=',W_Matrix[1],'micro-strain')
          print('Delta=',W_Matrix[2],'micro-strains/meter')
          q_3000 = W_Matrix[0]*2999*144+W_Matrix[1]
          print('q_3000 =',q_3000, 'micro-strain')
          q_5y = W_Matrix[0]*5*365*24*60+W_Matrix[1]
          print('q_5y=',q_5y, 'micro-strain')
```

$\hat{\alpha}$= -8.044421308508439e-05 micro-strain/min

$\hat{\beta}$= -37.65036760882104 micro-strain

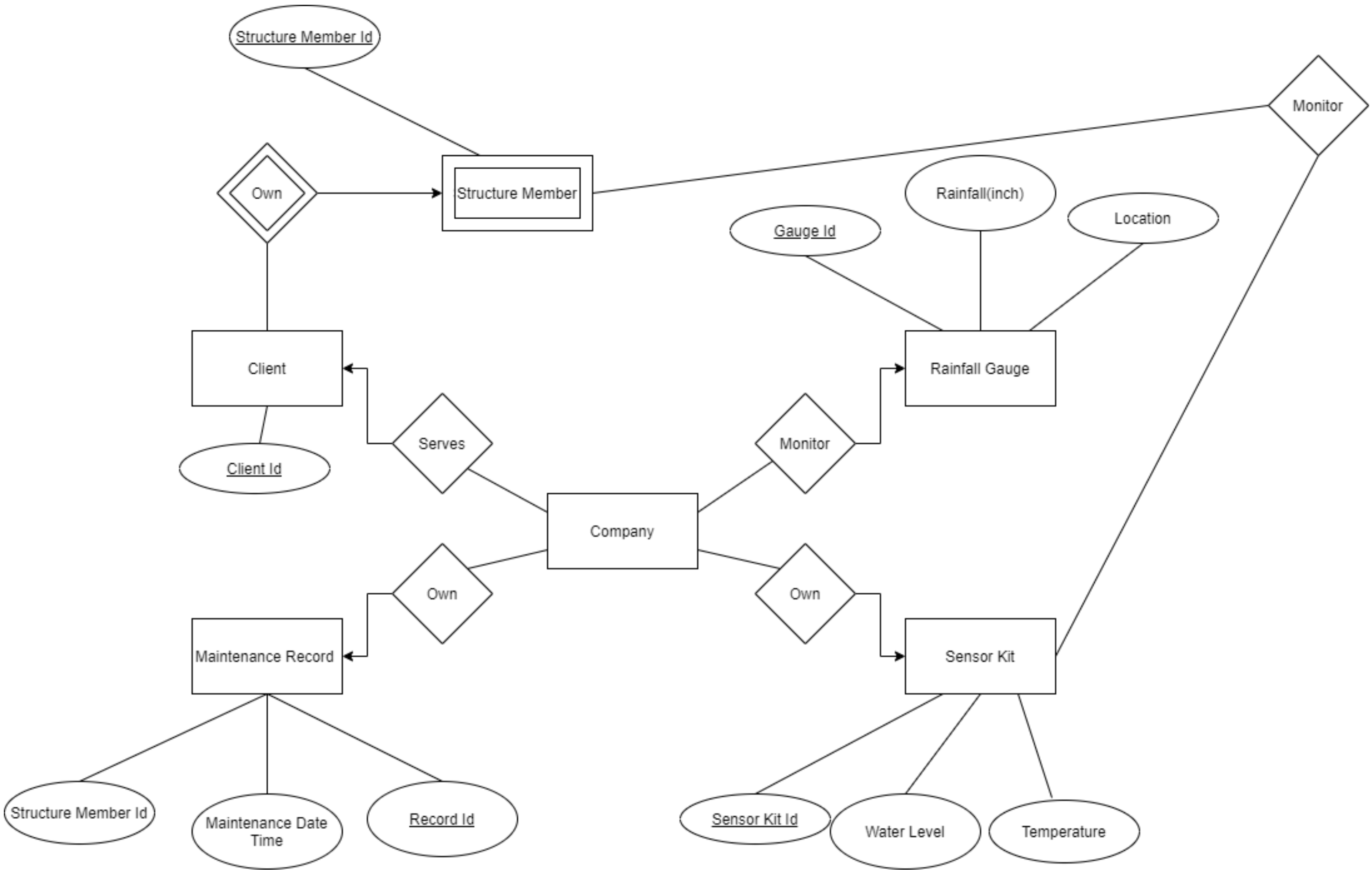
$\hat{\delta}$= -9.409107442878295 micro-strains/meter

q_3000 = -72.39068369489326 micro-strain

q_5y= -249.05775959642278 micro-strain

Removing the Temperature record, the estimated y of our linear model has a drastic difference, and the predictions are more inaccurate. Especially when we are predicting future values of y. The further into the future we are predicting, the larger the noise will be, therefore the larger the inaccuracy will be.

3. Wet Databases (15%)



Lih-Shyan (Chris) Fang
lihshyaf

4. Set Theory (10%)

Because:

$$\frac{1}{10} baseball\ players = \frac{1}{6} Dominicans$$

Therefore, the baseball players are a larger group.

$$\frac{Baseball\ players}{Dominicans} = \frac{5}{3}$$