analisis

September 17, 2015

1 Análisis de los datos obtenidos

Uso de ipython para el análsis y muestra de los datos obtenidos durante la producción. La regulación del diámetro se hace mediante el control del filawinder. Los datos analizados son del día 16 de Junio del 2015

Los datos del experimento: * Hora de inicio: 11:50 * Hora final : 12:20 * T : 150^oC

```
In [2]: #Importamos las librerías utilizadas
        import numpy as np
        import pandas as pd
        import seaborn as sns
```

```
In [3]: #Mostramos las versiones usadas de cada librerías
        print ("Numpy v{}".format(np.__version__))
        print ("Pandas v{}".format(pd.__version__))
        print ("Seaborn v{}".format(sns.__version__))
```

```
Numpy v1.9.2
Pandas v0.16.2
Seaborn v0.6.0
```

```
In [4]: #Abrimos el fichero csv con los datos 2de la muestra
        datos = pd.read_csv('ensayo1.CSV')
```

```
In [5]: %pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
In [6]: #Almacenamos en una lista las columnas del fichero con las que vamos a trabajar
        columns = ['Diametro X','Diametro Y']
```

```
In [7]: #Mostramos un resumen de los datos obtenidoss
        datos[columns].describe()
        #datos.describe().loc['mean',['Diametro X [mm]', 'Diametro Y [mm]']]
```

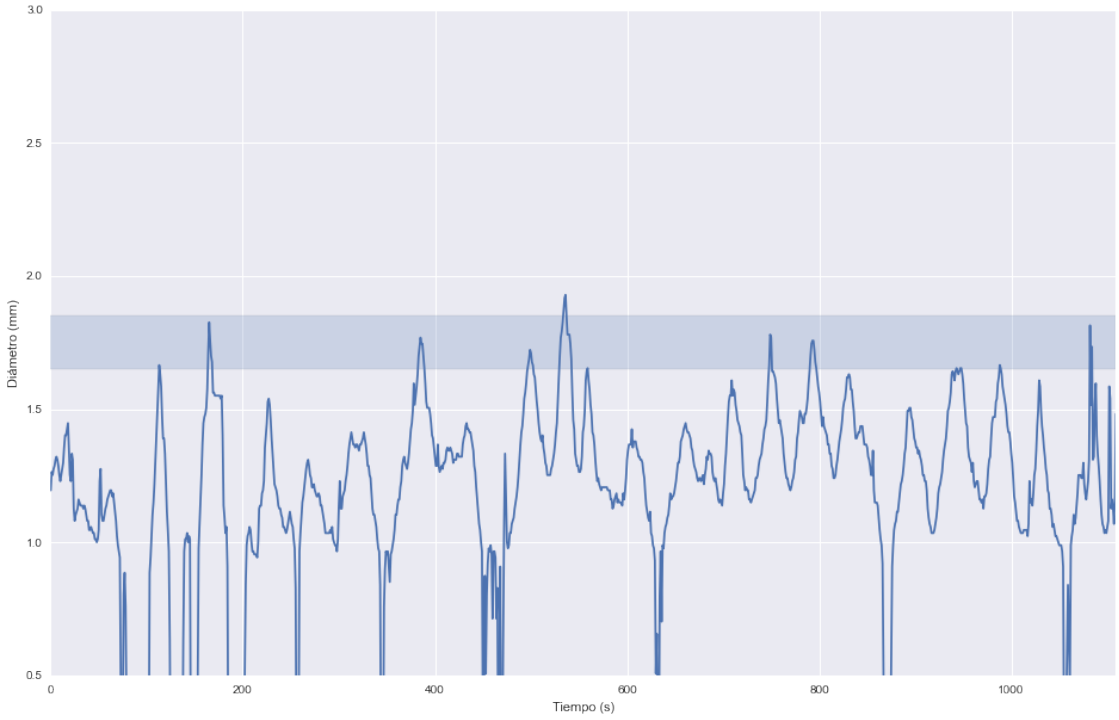
```
Out[7]:         Diametro X   Diametro Y
        count  1110.000000  1110.000000
        mean      1.176870     0.913499
        std       0.393488     0.510295
        min       0.014000     0.000342
        25%       1.069229     0.761755
        50%       1.241277     1.092180
        75%       1.401856     1.241589
        max       1.929470     1.747282
```

Representamos ambos diámetro y la velocidad de la tractora en la misma gráfica

In [20]:
```
graf=datos.ix[:, "Diametro X"].plot(figsize=(16,10),ylim=(0.5,3))
graf.axhspan(1.65,1.85, alpha=0.2)
graf.set_xlabel('Tiempo (s)')
graf.set_ylabel('Diámetro (mm)')
```

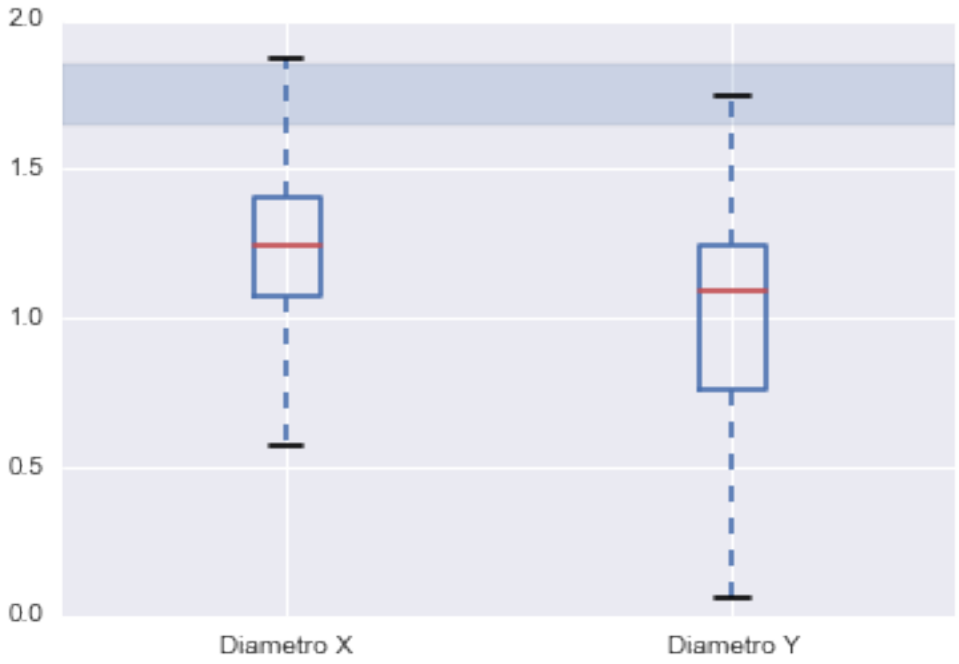
```
#datos['RPM TRAC'].plot(secondary_y='RPM TRAC')
```

Out[20]: <matplotlib.text.Text at 0x940c250>



In [9]:
```
box = datos.ix[:, "Diametro X":"Diametro Y"].boxplot(return_type='axes')
box.axhspan(1.65,1.85, alpha=0.2)
```

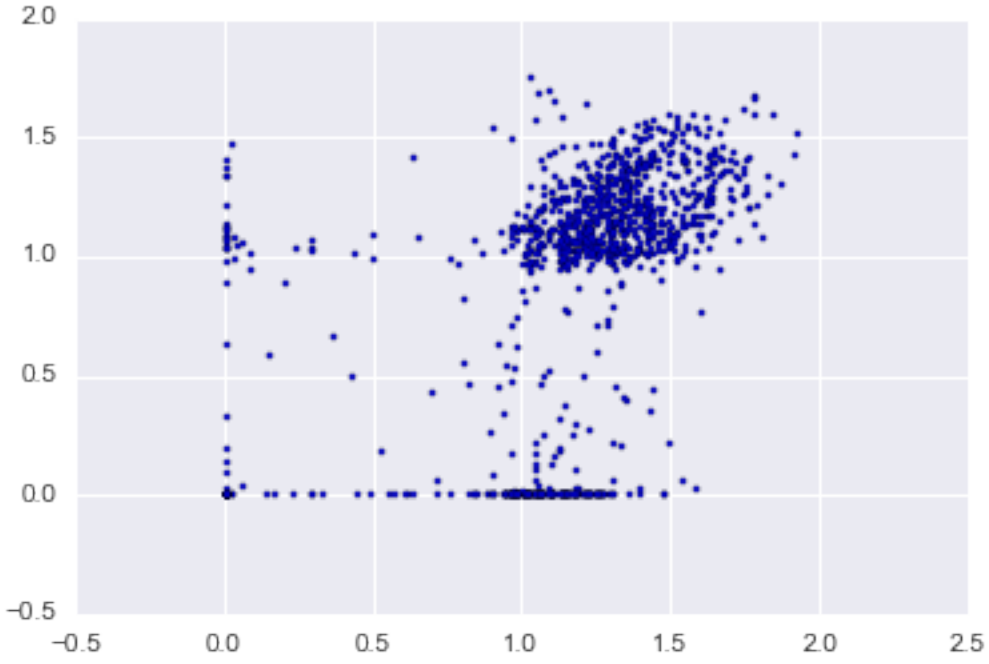
Out[9]: <matplotlib.patches.Polygon at 0x849efb0>



Comparativa de Diametro X frente a Diametro Y para ver el ratio del filamento

```
In [10]: plt.scatter(x=datos['Diametro X'], y=datos['Diametro Y'], marker='.')
```

```
Out[10]: <matplotlib.collections.PathCollection at 0x8899cb0>
```



2 Filtrado de datos

Las muestras tomadas $d_x >= 0.9$ or $d_y >= 0.9$ las asumimos como error del sensor, por ello las filtramos de las muestras tomadas.

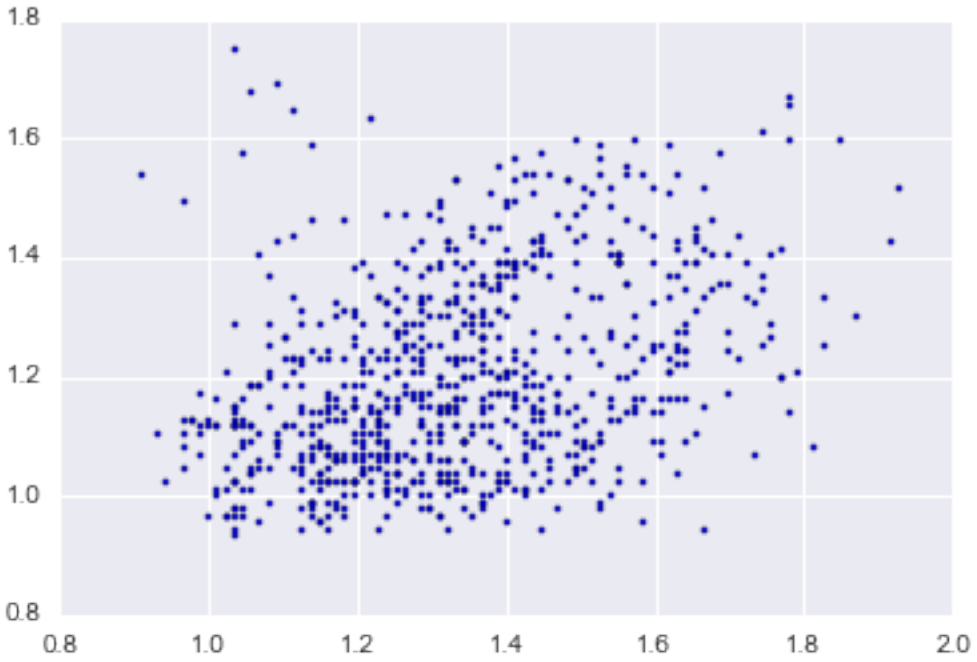
```
In [11]: datos_filtrados = datos[(datos['Diametro X'] >= 0.9) & (datos['Diametro Y'] >= 0.9)]
```

```
In [12]: #datos_filtrados.ix[:, "Diametro X":"Diametro Y"].boxplot(return_type='axes')
```

2.1 Representación de X/Y

```
In [13]: plt.scatter(x=datos_filtrados['Diametro X'], y=datos_filtrados['Diametro Y'], marker='.')
```

```
Out[13]: <matplotlib.collections.PathCollection at 0x88de710>
```



3 Analizamos datos del ratio

```
In [14]: ratio = datos_filtrados['Diametro X']/datos_filtrados['Diametro Y']
         ratio.describe()
```

```
Out[14]: count    781.000000
         mean       1.123596
         std        0.172839
         min        0.589877
         25%        1.007567
         50%        1.107309
         75%        1.238627
         max        1.766776
         dtype: float64
```

In [15]:
```
rolling_mean = pd.rolling_mean(ratio, 50)
rolling_std = pd.rolling_std(ratio, 50)
rolling_mean.plot(figsize=(12,6))
# plt.fill_between(ratio, y1=rolling_mean+rolling_std, y2=rolling_mean-rolling_std, alpha=0.5)
ratio.plot(figsize=(12,6), alpha=0.6, ylim=(0.5,1.5))
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x8931650>
```