Analisis

September 17, 2015

1 Análisis de los datos obtenidos

Producción del día 20 de Julio de 2015

Los datos del experimento: * Hora de inicio: 09:56 * Hora final : 10:1 * T : 135^oC * $V_{min}tractora$: $1mm/s$ * $V_{max}tractora$: $3mm/s$

Se desea comprobar si el filamento que podemos llegar a extruir con el sistema de la tractora puede llegar a ser bueno como para regularlo.

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

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

```
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 de la muestra
        datos = pd.read_csv('datos.csv')
```

```
In [5]: #Almacenamos en una lista las columnas del fichero con las que vamos a trabajar
        columns = ['Diametro X','VELOCIDAD']
```

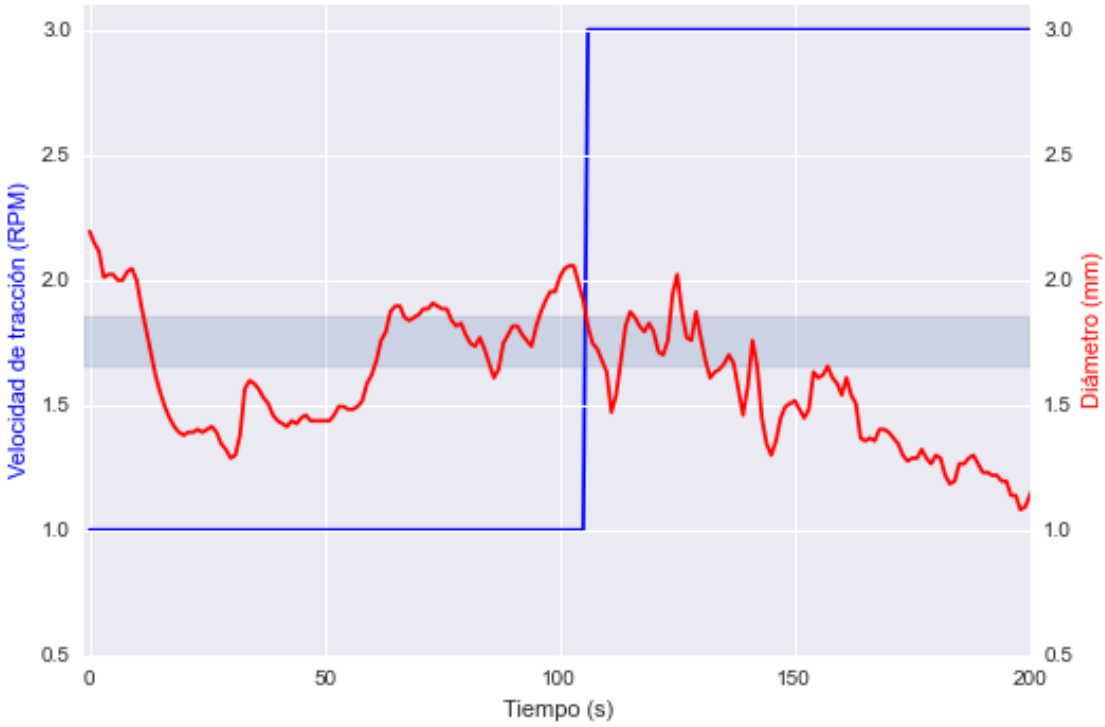
```
In [6]: #Mostramos un resumen de los datos obtenidoss
        datos[columns].describe()
        #datos.describe().loc['mean',['Diametro X [mm]', 'Diametro Y [mm]']]
```

Out[6]:

	Diametro X	VELOCIDAD
count	203.000000	203.000000
mean	1.595262	1.955665
std	0.253779	1.001486
min	1.080699	1.000000
25%	1.396121	1.000000
50%	1.585374	1.000000
75%	1.809036	3.000000
max	2.193277	3.000000

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

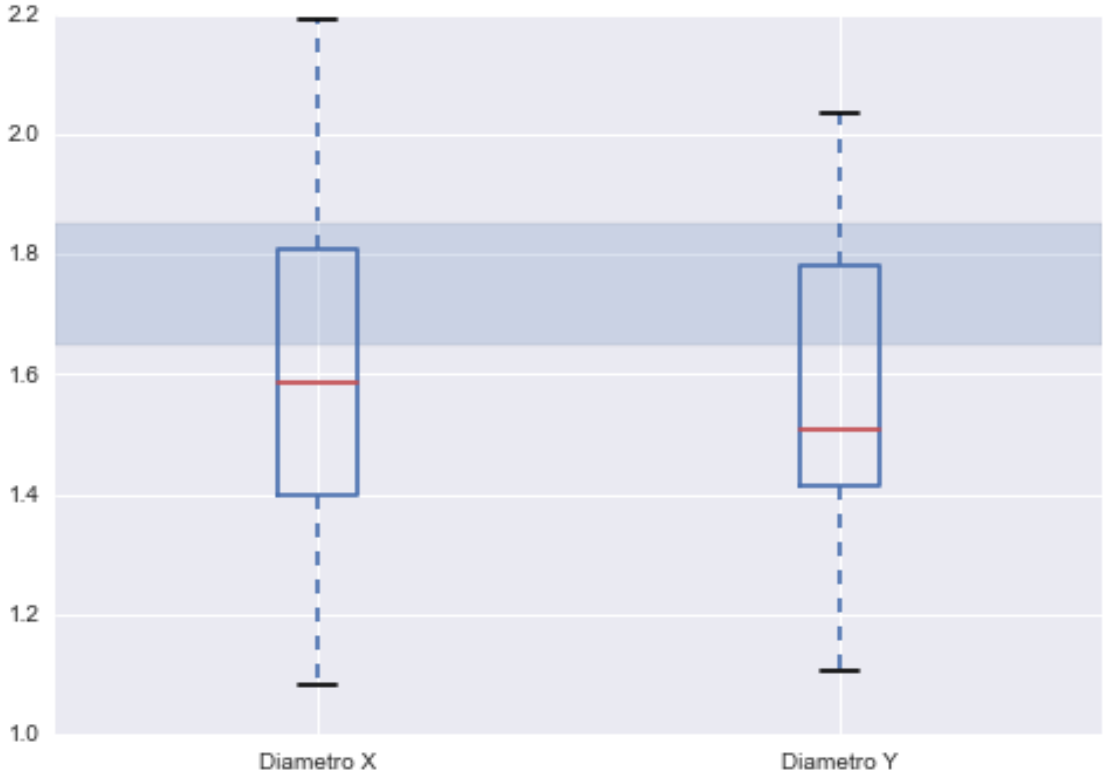
In [26]:
```
#datos.ix[:, "Diametro X":"Diametro Y"].plot(secondary_y=['VELOCIDAD'],figsize=(16,10),ylim=(0
#graf = datos[columns].plot(secondary_y=['VELOCIDAD'],ylim=(1,2.5),figsize=(10,5),title='Relac
#graf.axhspan(1.65,1.85, alpha=0.2)
#datos['RPM TRAC'].plot(secondary_y='RPM TRAC')
fig, ax1 = plt.subplots()

ax1.plot(datos['VELOCIDAD'], 'b-')
ax1.set_xlabel('Tiempo (s)')
ax1.set_ylabel('Velocidad de tracción (RPM)', color='b')
ax1.set_ylim(0.5,3.1)
ax1.set_xlim(-1,200)

ax2 = ax1.twinx()
ax2.plot(datos['Diametro X'], 'r-')
ax2.set_ylabel('Diámetro (mm)', color='r')
ax2.set_ylim(0.5,3.1)
ax2.set_xlim(-1,200)

plt.axhspan(1.65,1.85, alpha=0.2)
plt.figure(figsize=(10,5))
plt.show()
```




```
<matplotlib.figure.Figure at 0x8d06490>
```

In [27]:
```
graf = datos.ix[:, "Diametro X":"Diametro Y"].boxplot(return_type='axes')
graf.axhspan(1.65,1.85, alpha=0.2)
```

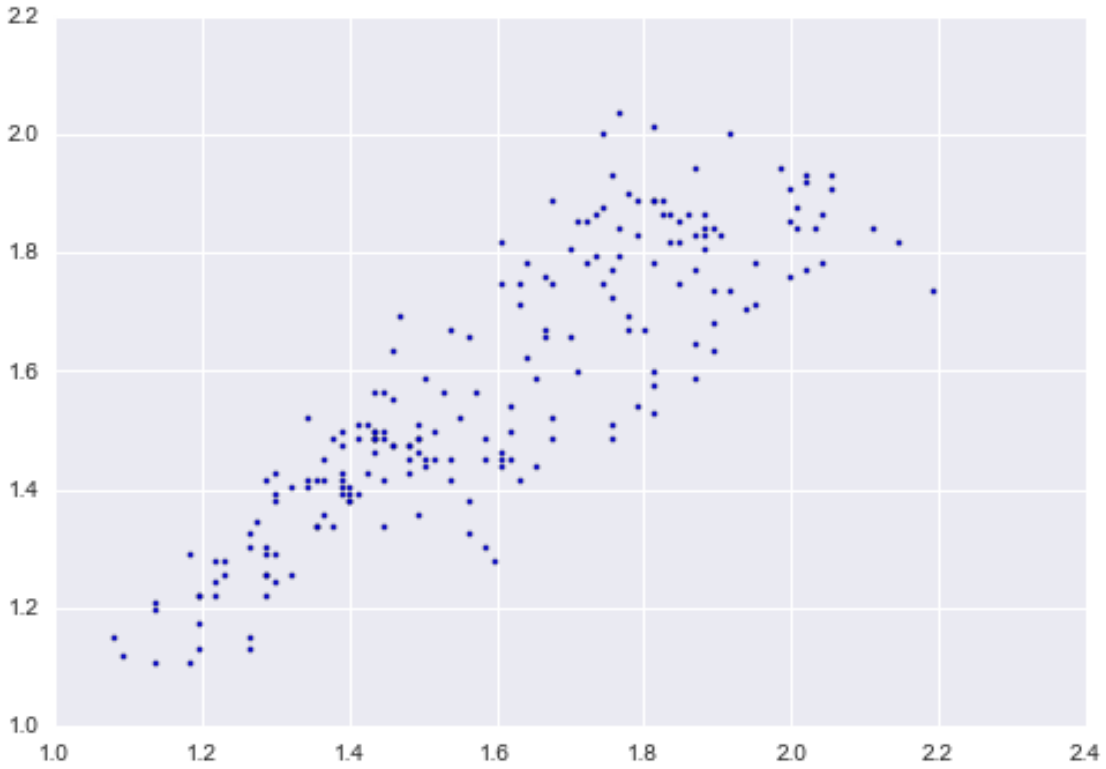
```
Out[27]: <matplotlib.patches.Polygon at 0x869d5b0>
```



Con esta segunda aproximación se ha conseguido estabilizar los datos. Se va a tratar de bajar ese porcentaje. Como cuarta aproximación, vamos a modificar las velocidades de tracción. El rango de velocidades propuesto es de 1.5 a 5.3, manteniendo los incrementos del sistema experto como en el actual ensayo.

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

```
In [9]: plt.scatter(x=datos['Diametro X'], y=datos['Diametro Y'], marker='.')

Out[9]: <matplotlib.collections.PathCollection at 0x8953ad0>
```



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.

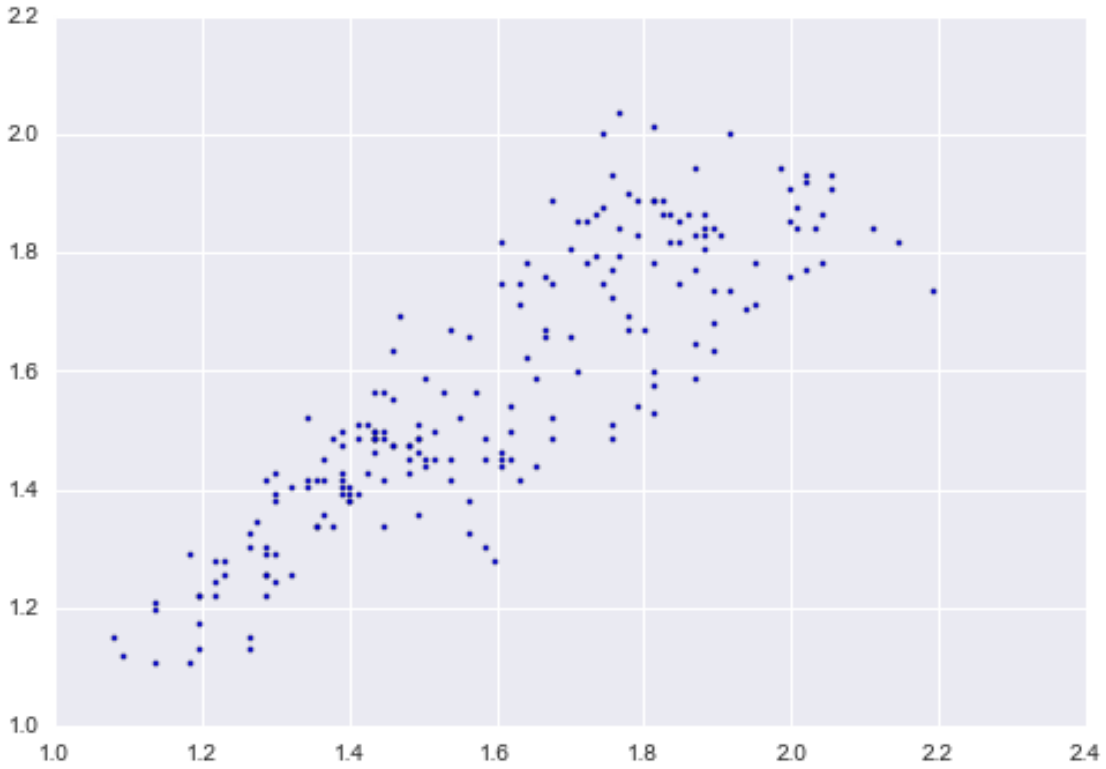
```
In [10]: datos_filtrados = datos[(datos['Diametro X'] >= 0.9) & (datos['Diametro Y'] >= 0.9)]
```

```
In [11]: #datos_filtrados.ix[:, "Diametro X":"Diametro Y"].boxplot(return_type='axes')
```

2.1 Representación de X/Y

```
In [12]: plt.scatter(x=datos_filtrados['Diametro X'], y=datos_filtrados['Diametro Y'], marker='.')
```

```
Out[12]: <matplotlib.collections.PathCollection at 0x899cb90>
```



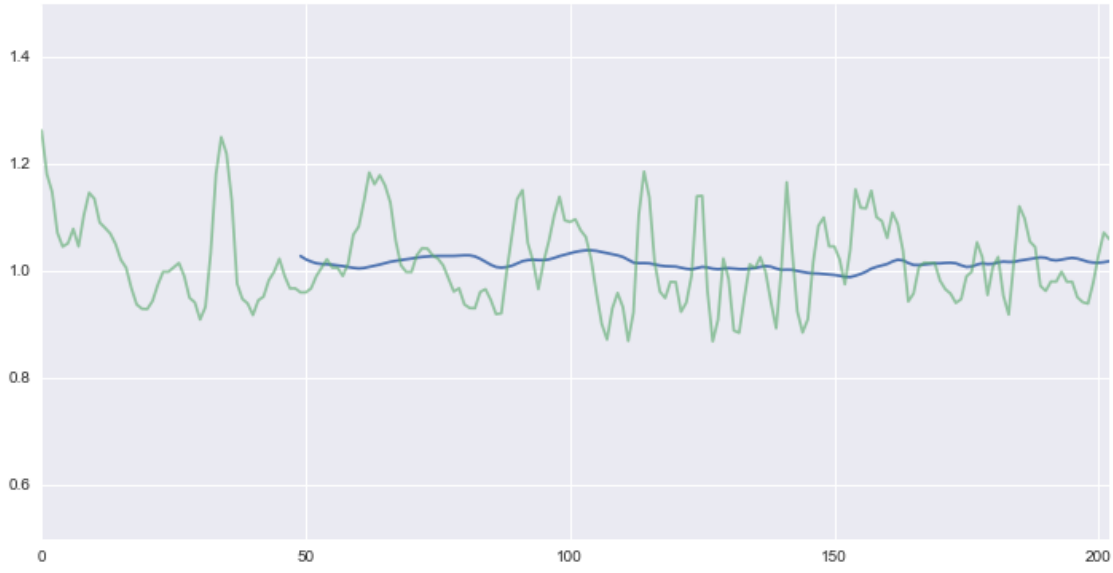
3 Analizamos datos del ratio

```
In [13]: ratio = datos_filtrados['Diametro X']/datos_filtrados['Diametro Y']
         ratio.describe()
```

```
Out[13]: count    203.000000
         mean       1.018532
         std        0.078369
         min        0.869402
         25%        0.960464
         50%        1.007268
         75%        1.066282
         max        1.263562
         dtype: float64
```

```
In [14]: 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[14]: <matplotlib.axes._subplots.AxesSubplot at 0x89bddf0>
```


1.4
1.2
1.0
0.8
0.6
0
50
100
150
200