ensayo2

September 18, 2015

1 Análisis de los datos obtenidos

Uso de ipython para el análsis y muestra de los datos obtenidos durante la producción.Se implementa un regulador experto. Los datos analizados son del día 12 de Agosto del 2015

Los datos del experimento: * Hora de inicio: 11:05 * Hora final : 11:35 * Filamento extruido: 435cm * T : 150^oC * $V_{min}tractora$: 1.5mm/s * $V_{max}tractora$: 3.4mm/s * Los incrementos de velocidades en las reglas del sistema experto son distintas: * En el caso 5 se pasa de un incremento de velocidad de +1 a un incremento de +2.

```
In [20]: #Importamos las librerías utilizadas
         import numpy as np
         import pandas as pd
         import seaborn as sns
```

```
In [21]: #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 [22]: #Abrimos el fichero csv con los datos de la muestra
         datos = pd.read_csv('ensayo2.CSV')
```

```
In [23]: %pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib

WARNING: pylab import has clobbered these variables: ['box']
`%matplotlib` prevents importing * from pylab and numpy
```

```
In [24]: #Almacenamos en una lista las columnas del fichero con las que vamos a trabajar
         columns = ['Diametro X','Diametro Y', 'RPM TRAC']
```

```
In [25]: #Mostramos un resumen de los datos obtenidoss
         datos[columns].describe()
         #datos.describe().loc['mean',['Diametro X [mm]', 'Diametro Y [mm]']]
```

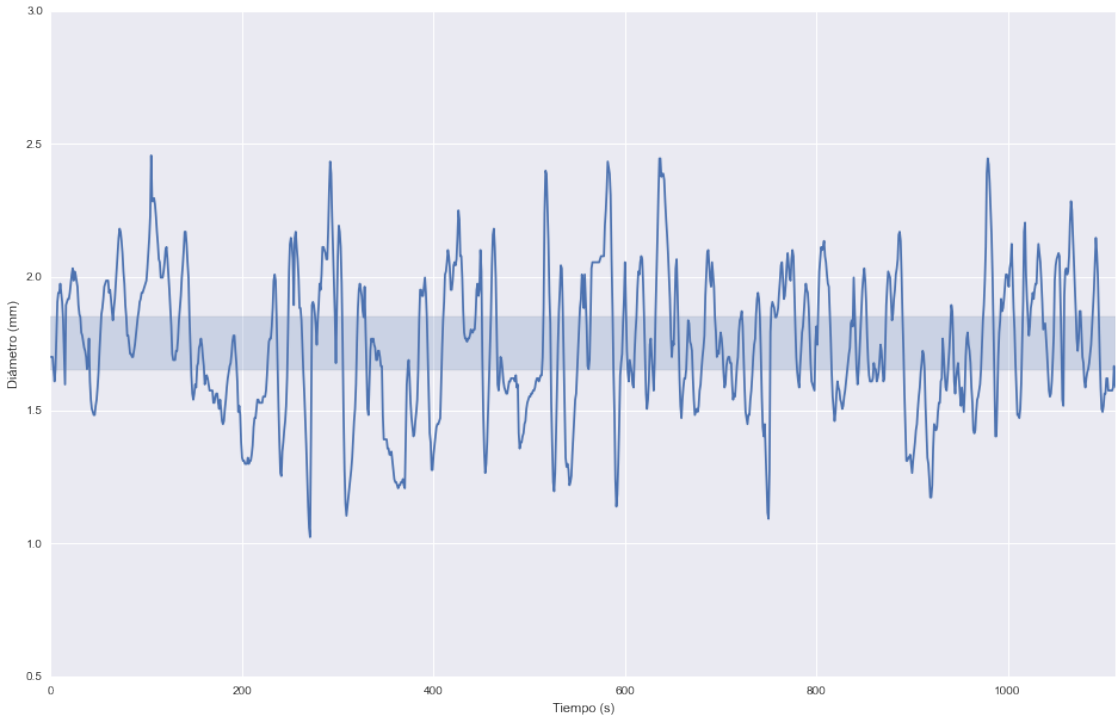
```
Out[25]:          Diametro X   Diametro Y     RPM TRAC
         count  1114.000000  1114.000000  1114.000000
         mean      1.748681     1.728351     2.530507
         std       0.266034     0.269547     0.871920
         min       1.023350     1.080687     1.497500
```

```
25%         1.565302      1.551901      1.497500
50%         1.734482      1.712803      2.610000
75%         1.952410      1.928298      3.500000
max         2.457085      2.528808      3.500000
```

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

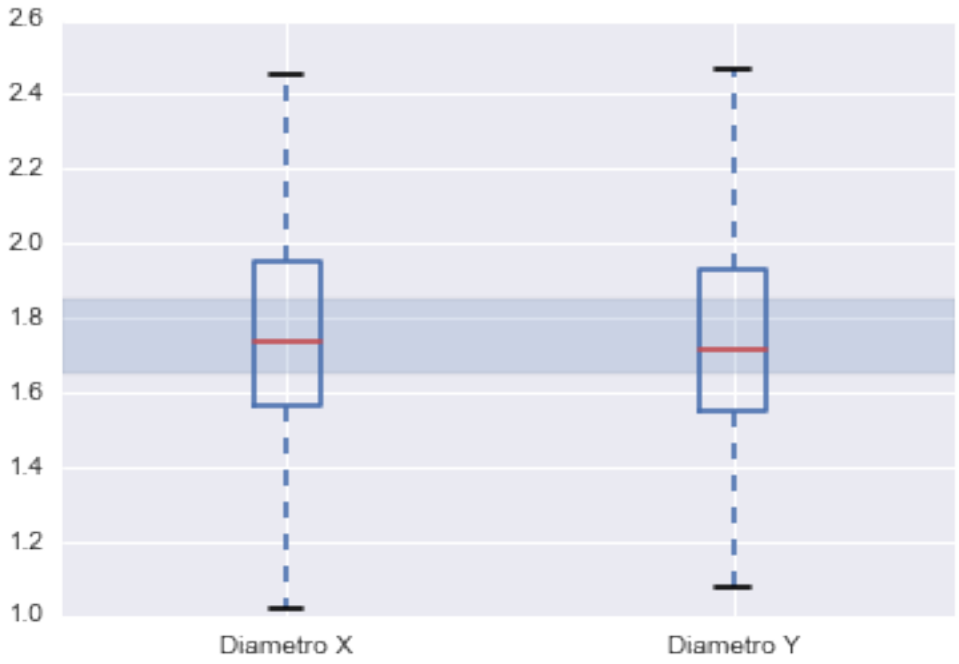
In [26]:
```
graf=datos.ix[:, "Diametro X"].plot(figsize=(16,10),ylim=(0.5,3))
graf.axhspan(1.65,1.85, alpha=0.2)
graf.set_xlabel('Tiempo (s)')
graf.set_ylabel('Diámetro (mm)')
#datos['RPM TRAC'].plot(secondary_y='RPM TRAC')
```

Out[26]: <matplotlib.text.Text at 0x95b5a70>



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

Out[27]: <matplotlib.patches.Polygon at 0x95eb610>

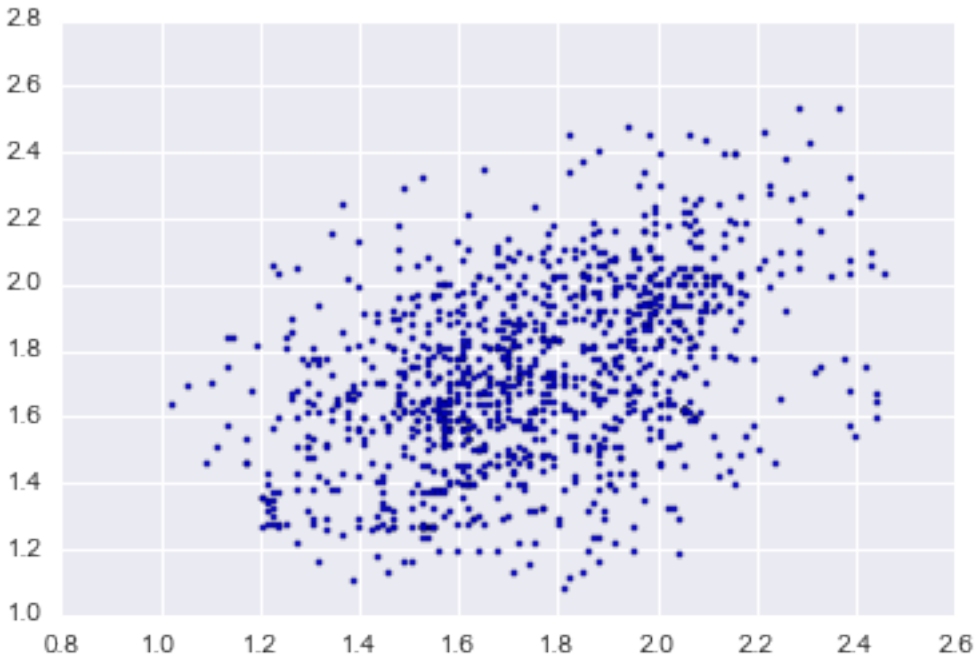


Con esta segunda aproximación se ha conseguido estabilizar los datos. Se va a tratar de bajar ese porcentaje. Como segunda aproximación, vamos a modificar los incrementos en los que el diámetro se encuentra entre $1.80mm$ y $1.70mm$, en ambos sentidos. (casos 3 a 6)

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

```
In [28]: plt.scatter(x=datos['Diametro X'], y=datos['Diametro Y'], marker='.')
```

```
Out[28]: <matplotlib.collections.PathCollection at 0x9aa58d0>
```



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.

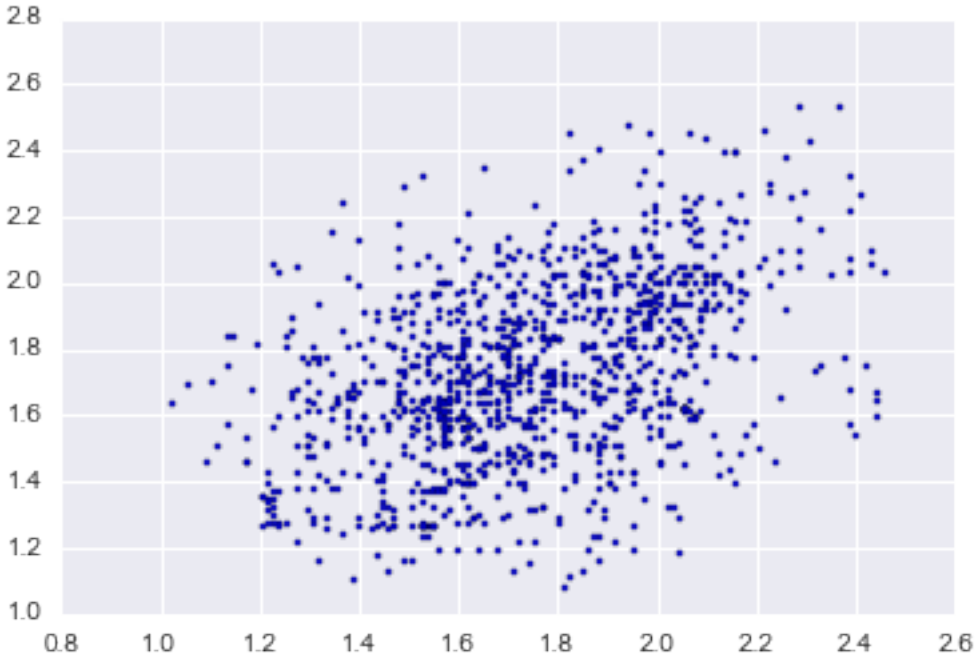
```
In [29]: datos_filtrados = datos[(datos['Diametro X'] >= 0.9) & (datos['Diametro Y'] >= 0.9)]
```

```
In [30]: #datos_filtrados.ix[:, "Diametro X":"Diametro Y"].boxplot(return_type='axes')
```

2.1 Representación de X/Y

```
In [31]: plt.scatter(x=datos_filtrados['Diametro X'], y=datos_filtrados['Diametro Y'], marker='.')
```

```
Out[31]: <matplotlib.collections.PathCollection at 0x9ce8c50>
```



3 Analizamos datos del ratio

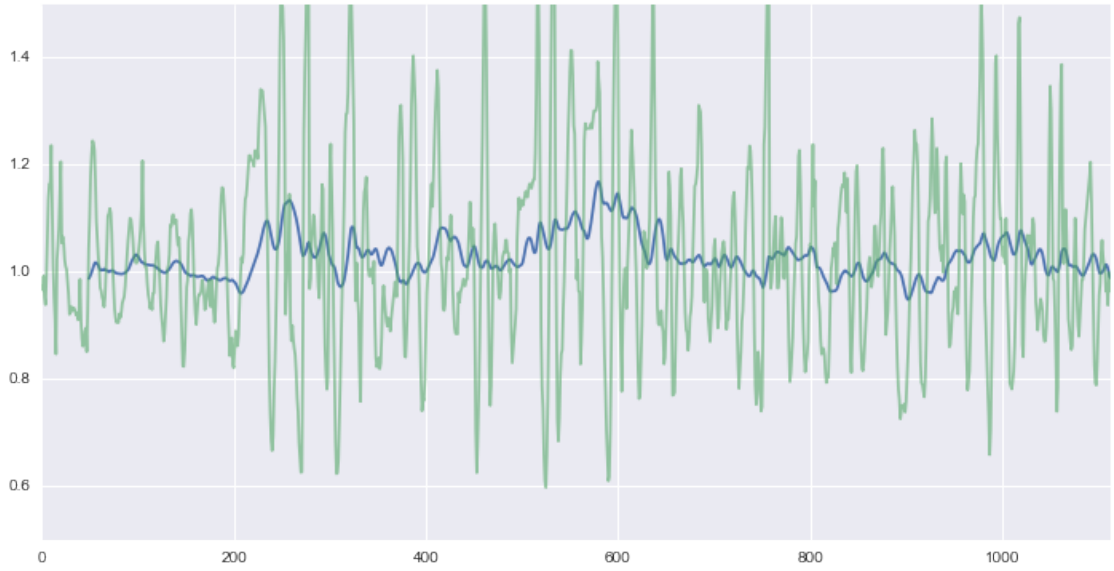
```
In [32]: ratio = datos_filtrados['Diametro X']/datos_filtrados['Diametro Y']
         ratio.describe()
```

```
Out[32]: count    1114.000000
         mean        1.027120
         std         0.177696
         min         0.597692
         25%         0.912259
```

```
50%         1.006821
75%         1.123901
max         1.726313
dtype: float64
```

```
In [33]: 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[33]: <matplotlib.axes._subplots.AxesSubplot at 0x9cf7170>
```



4 Límites de calidad

Calculamos el número de veces que traspasamos unos límites de calidad. $Th^+ = 1.85$ and $Th^- = 1.65$

```
In [34]: Th_u = 1.85
         Th_d = 1.65
```

```
In [35]: data_violations = datos[(datos['Diametro X'] > Th_u) | (datos['Diametro X'] < Th_d) |
                                (datos['Diametro Y'] > Th_u) | (datos['Diametro Y'] < Th_d)]
```

```
In [36]: data_violations.describe()
```

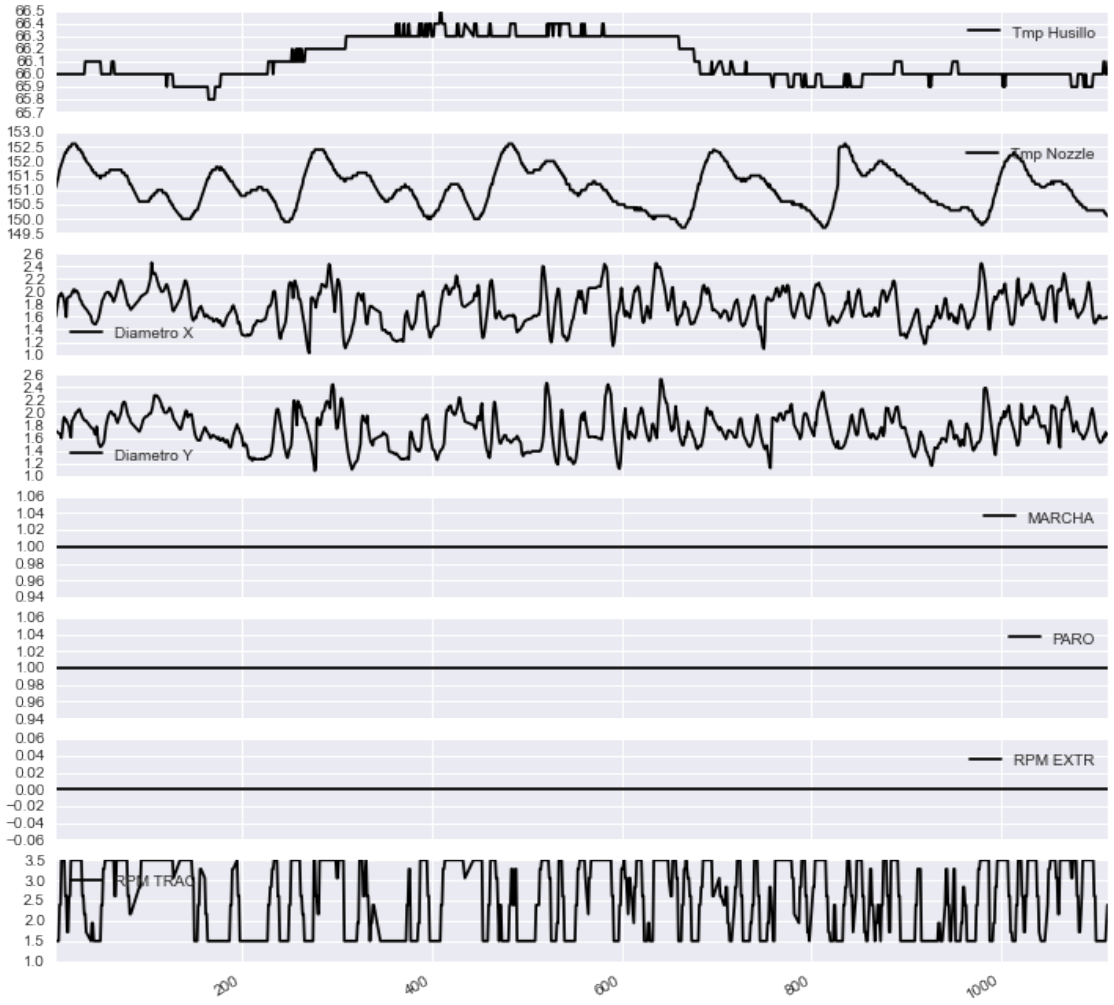
```
Out[36]:        Tmp Husillo   Tmp Nozzle   Diametro X   Diametro Y MARCHA  PARO  \
         count  1020.000000  1020.000000  1020.000000  1020.000000   1020  1020
         mean     66.109804   151.083725     1.749832     1.726640      1     1
         std       0.159864     0.725026     0.277519     0.281122      0     0
         min      65.800000   149.700000     1.023350     1.080687   True  True
         25%      66.000000   150.500000     1.550964     1.528915      1     1
         50%      66.000000   151.100000     1.734482     1.701310      1     1
         75%      66.300000   151.600000     1.975350     1.942664      1     1
```

```
max      66.500000   152.600000     2.457085     2.528808   True   True

       RPM EXTR     RPM TRAC
count      1020  1020.000000
mean          0     2.551975
std           0     0.878147
min           0     1.497500
25%           0     1.497500
50%           0     2.610000
75%           0     3.500000
max           0     3.500000
```

In [37]:
```
data_violations.plot(subplots=True, figsize=(12,12))
```

Out[37]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x09D14DB0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x09D6D890>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x09D9C190>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x09DB9EB0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x09DE7690>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x09E07D30>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x09E36630>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x09E55CB0>], dtype=object)
```



In [ ]: