Manipulations Audio pour ISN: analyse statistique d'un CD

Christophe Barès

3 février 2014

In [1]:
```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from struct import unpack
```

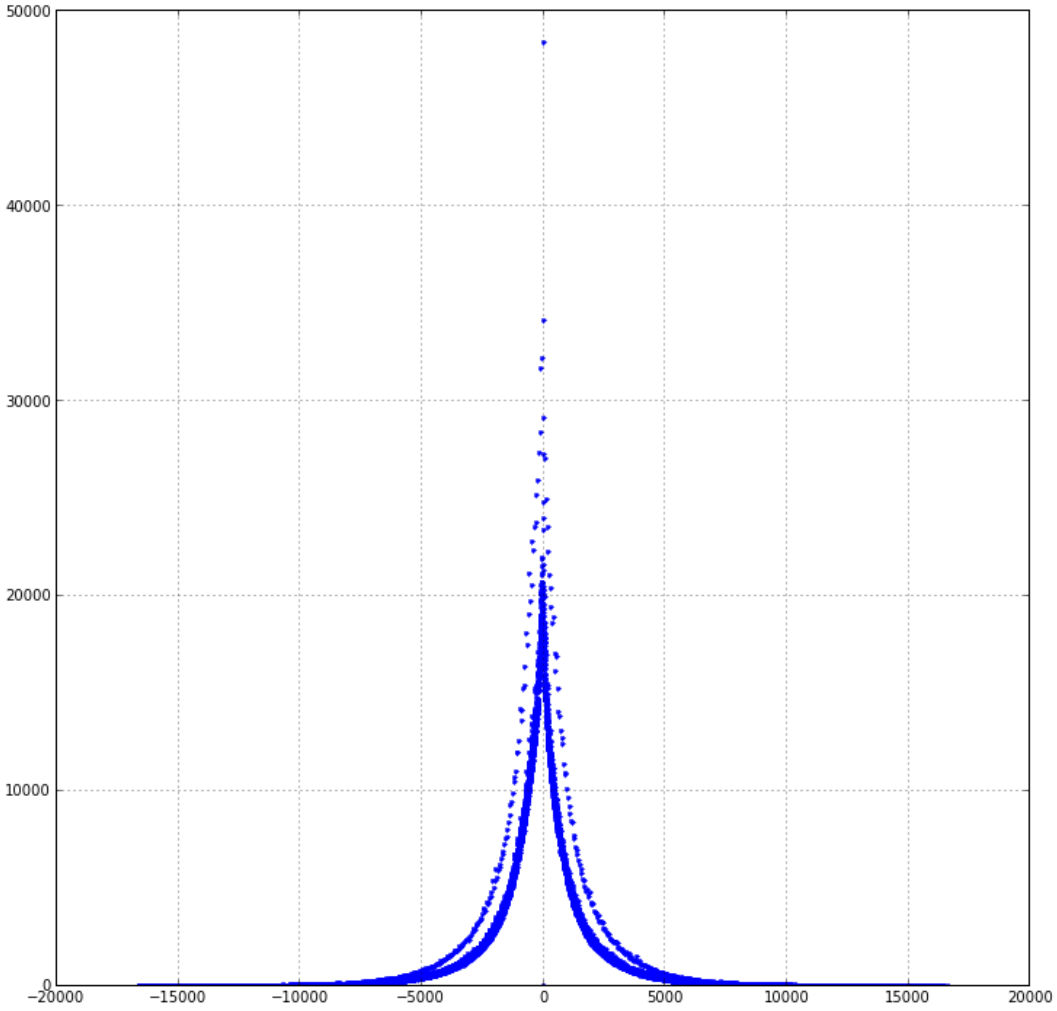
In [5]:
```
#infile = open('sony-bmg/track01.sym9.wav', 'rb')
infile = open('licorne/track01.sym9.wav', 'rb')
infile.seek(44)

from collections import defaultdict
histog = defaultdict(lambda:0)
histod = defaultdict(lambda:0)

fragment = infile.read(4)
while len(fragment)==4:
    g, d = unpack('<2h', fragment)
    histog[g] += 1
    histod[d] += 1
    fragment = infile.read(4)
infile.close()
```

In [6]:
```
histo = histog
print 'Mode: ', np.argmax(histo.itervalues())
print "Nombre d'échantillons: ", sum(histo.itervalues())
print "Moyenne:" , sum(k*v for k,v in histo.iteritems())/float(sum(hist
print "Dynamique:", max(histo.iterkeys()) - min(histo.iterkeys())
```

```
Mode:  0
Nombre d'échantillons:  38684520
Moyenne: -58.7379532175
Dynamique: 33266
```

In [7]:
```
histog[0] = 0
histod[0] = 0
```

In [15]:
```
plt.figure(figsize=(12,12))
x = histo.keys()
x.sort()
y = [histod[i] for i in x]
plt.plot(x,y,'.');
plt.grid();
#plt.axis([0,5000,0,20000])
#plt.axis([-30000,30000,0,10000])
```



Sur cet histogramme, on distingue nettement 2 "traces" : une normale trés dense, et une seconde, beaucoup plus clairsemée. Il y a donc des valeurs d'échantillons qui sont beaucoup plus probables, et même 2 fois plus problables que les autres.

Cela est dû à une manipulation de l'amplitude postérieurement à l'enrgistrement : le signal a été atténué pour en réduire l'amplitude. La fréquences des valeurs plus probables permet de retrouver le coefficient utilisé (ici je crois qu'il vaut environ 0,97).

Les raisons sont inconnues, mais il s'agit souvent de masquer une saturation au moment de l'enregistrement, ce qui ce voit sur un histogramme car les échantillons $+2^{15}$ et -2^{15} sont alors beaucoup trop nombreux.

In [111]:
```
infile = wave.open('licorne/track01.sym9.wav', 'r')
Fs = 44100
n = infile.getnframes()
raw = infile.readframes(12500)
n = 10*Fs
raw = infile.readframes(n)

infile.close()
```

In [112]:
```
datag = pd.Series(struct.unpack("{n}h".format(n=2*n), raw)[0::2], dtype
datad = pd.Series(struct.unpack("{n}h".format(n=2*n), raw)[1::2], dtype
del raw
```

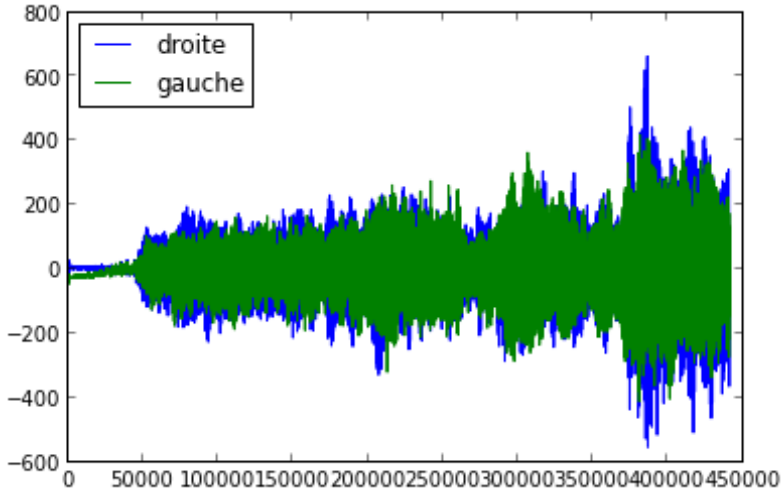
In [113]:
```
df = pd.DataFrame({'gauche':datag, 'droite':datad}, dtype=np.int16)
df = (df - df.mean())
df.describe()
```

Out [113]:

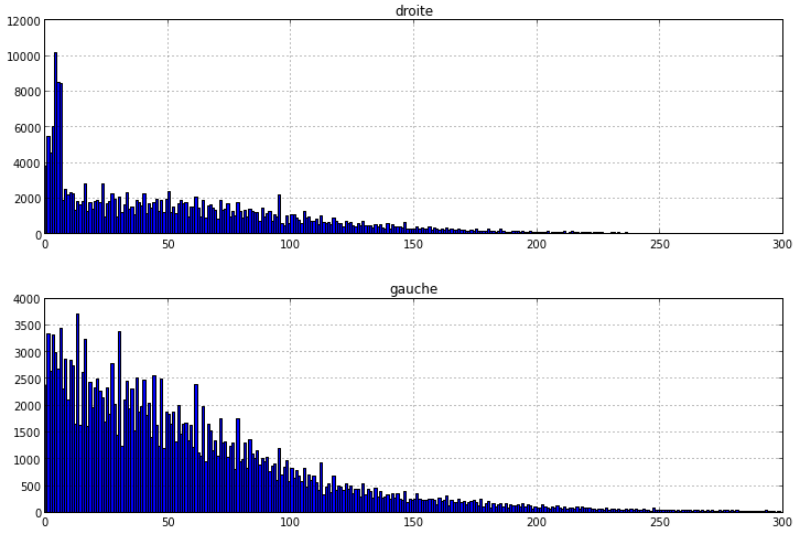
	droite	gauche
count	4.410000e+05	4.410000e+05
mean	-2.512217e-12	7.143587e-12
std	8.751876e+01	7.588831e+01
min	-5.524967e+02	-4.103692e+02
25%	-5.049666e+01	-4.236924e+01
50%	3.503342e+00	-2.369243e+00
75%	5.050334e+01	4.363076e+01
max	6.635033e+02	4.196308e+02

[8 rows x 2 columns]

In [114]:
```
plt.plot(df)
plt.legend(df.columns, 0);
```



In [115]:
```
df.hist(bins=range(0,300), layout=(2,1),figsize=(12,8));
```



Exemple d'histogramme obtenu avec un signal dont l'amplidude à une distribution uniforme, après multiplication par un facteur 1,1.

À cause des erreurs d'arrondi, cela a pour effet de faire disparaître certaines valeurs d'amplitude, qui ne sont donc plus présentent dans l'histogramme.

In [14]:
```
d = 10.
x = range(100)
y = [int(float(i)*(1.1)) for i in x]

#plt.plot(x, x, x, y)
plt.hist(y, bins=x);
plt.axis([0,100,0,3])
1-1/d
```

Out [14]: 0.9