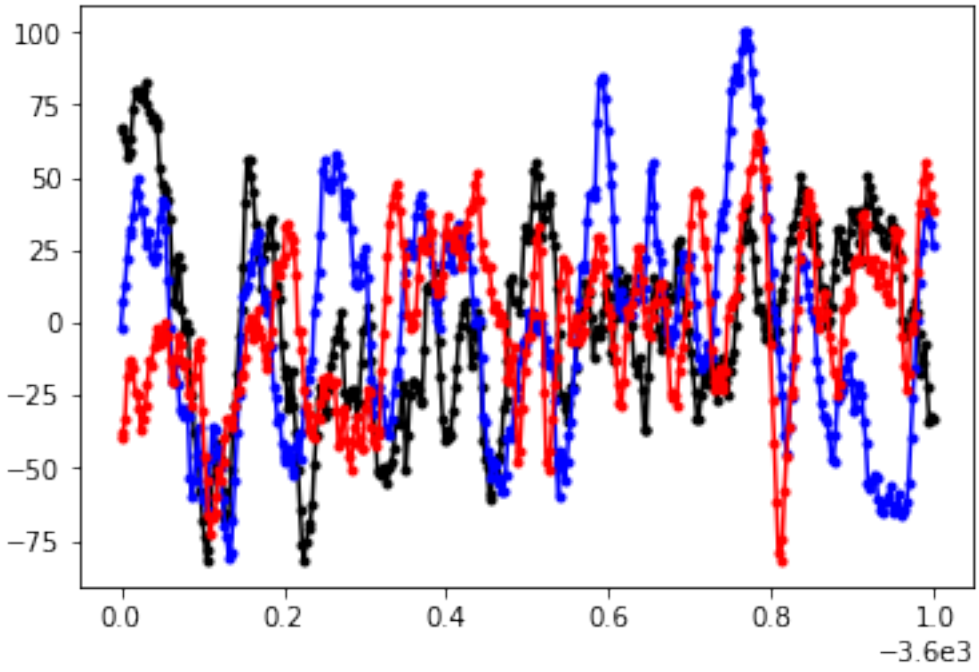
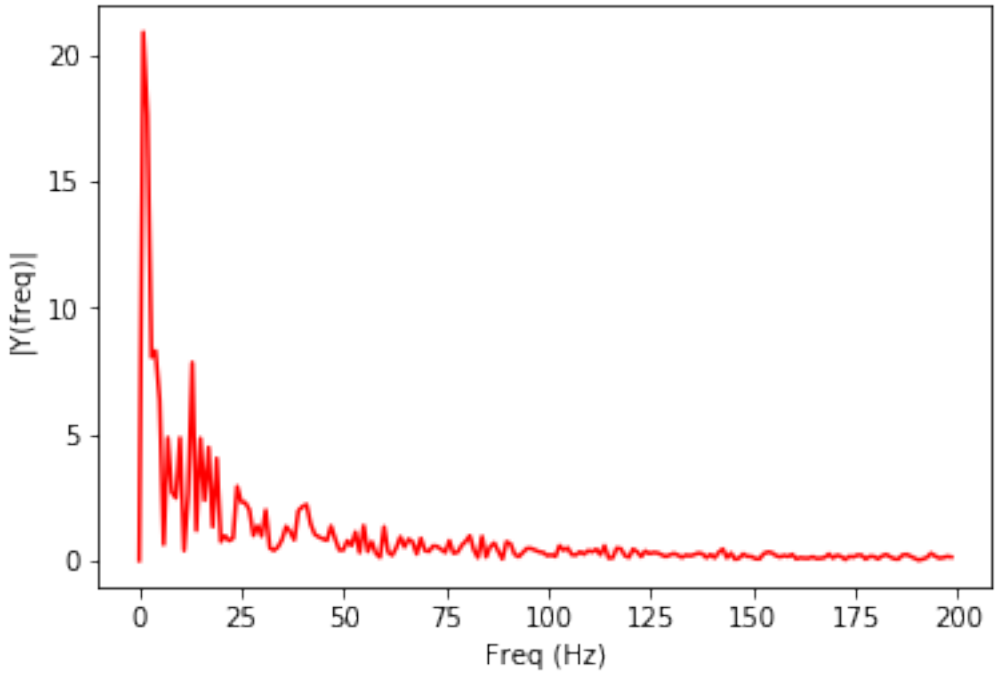
Epilepsy

May 22, 2018

```
In [1]: import seizure_detection_simple

        data = seizure_detection_simple.doload('Dog_1', False, False)

Reading from clips/Dog_1/ 596 596


In [2]: data

Out[2]: <class 'pandas.core.panel.Panel'>
        Dimensions: 596 (items) x 400 (major_axis) x 17 (minor_axis)
        Items axis: ictal_1 to interictal_99
        Major_axis axis: 0 to 399
        Minor_axis axis: 0 to time

In [3]: data.keys()

Out[3]: Index(['ictal_1', 'ictal_10', 'ictal_100', 'ictal_101', 'ictal_102',
               'ictal_103', 'ictal_104', 'ictal_105', 'ictal_106', 'ictal_107',
               ...
               'interictal_90', 'interictal_91', 'interictal_92', 'interictal_93',
               'interictal_94', 'interictal_95', 'interictal_96', 'interictal_97',
               'interictal_98', 'interictal_99'],
              dtype='object', length=596)

In [4]: data['ictal_1'].keys()

Out[4]: Index([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 'time'], dtype='object')

In [5]: seizure_detection_simple.plot(data)
```



In [6]: seizure_detection_simple.plotSpectrum(data['ictal_1'][0], 399)



In [7]:
```
import numpy as np

def fft(time_data):
    return np.log10(np.absolute(np.fft.rfft(time_data, axis=1)[:,1:48]))
```

0.1 Step 1.3

In [8]:
```
# For each segment in data.keys(), take data[segment], transpose it, and call fft() on i
#X=np.zeros((data.shape[0],(data.shape[2])*47))
#y=np.chararray((data.shape[0],1), itemsize=10)

X=[]
Y=[]
for key in data.keys():
    transp_data=data[key].T
    freq_data=fft(transp_data)
    freq_data=np.reshape(freq_data, (np.product(freq_data.shape),))
    if key[0:5]=='ictal':

        y='ictal'

    else:
        y='interictal'

    #data_label=list(np.hstack((freq_data, y)))
    #X.append(data_label)
    X.append(freq_data)
    Y.append(y)

X=np.vstack(X)
#Y=np.vstack(Y)

# Take the result (all channels by all 47 frequencies) and convert it into a 1D array.
# This is a feature row in 2D array X.
# Set the class label for the same row in 1D array y, dependong on if the segment name
# includes 'interictal' (this is a non-match) vs 'ictal' (this is a match)
```

In [9]:
```
# Create X_train, X_test, y_train, y_test with
# random_state = 42 and test_size = 0.3
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=42
```

0.2 Step 1.4 Best Result

In [10]:
```
## Produce your best output here!
 from sklearn.ensemble import RandomForestClassifier
 from sklearn.ensemble import AdaBoostClassifier
 from sklearn.ensemble import BaggingClassifier
```

```
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn import preprocessing
classifier_results=[]
################################################
min_max_scaler = preprocessing.MinMaxScaler()
X_scaled_train= min_max_scaler.fit_transform(X_train)
X_scaled_test= min_max_scaler.fit_transform(X_test)

clf3 = SVC(random_state=42)
classifier = BaggingClassifier(clf3,n_estimators=31,random_state=314)
classifier.fit(X_scaled_train, y_train)
y_pred_test = classifier.predict(X_scaled_test)
test_score1 = classifier.score(X_scaled_test, y_test)
classifier_results.append({'Classifier': 'Bag-SVM', 'Score': test_score1})

classifier = RandomForestClassifier(n_estimators=31,random_state=314)
classifier.fit(X_scaled_train, y_train)
y_pred_test = classifier.predict(X_scaled_test)
test_score = classifier.score(X_scaled_test, y_test)
classifier_results.append({'Classifier': 'RandomForest', 'Score': test_score})

############################################
scaler =StandardScaler().fit(X_train)
X_scaled_train=scaler.transform(X_train)
X_scaled_test=scaler.transform(X_test)


classifier = SVC(random_state=42)
classifier.fit(X_scaled_train, y_train)
y_pred_test = classifier.predict(X_scaled_test)
test_score = classifier.score(X_scaled_test, y_test)
classifier_results.append({'Classifier': 'SVM', 'Score': test_score})

############################################
X_scaled_train= preprocessing.scale(X_train)
X_scaled_test = preprocessing.scale(X_test)

classifier = LogisticRegression(penalty='l1',random_state=42,solver='liblinear')
classifier.fit(X_scaled_train, y_train)
y_pred_test = classifier.predict(X_scaled_test )
test_score = classifier.score(X_scaled_test , y_test)
classifier_results.append({'Classifier': 'LogReg-L1','Score': test_score})
classifier_results
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/preprocessing/data.py:181: UserWarning: Numerical
  warnings.warn("Numerical issues were encountered "
```

```
Out[10]: [{'Classifier': 'Bag-SVM', 'Score': 1.0},
          {'Classifier': 'RandomForest', 'Score': 1.0},
          {'Classifier': 'SVM', 'Score': 1.0},
          {'Classifier': 'LogReg-L1', 'Score': 1.0}]

In [11]: test_score1

Out[11]: 1.0
```