AnomalyDetection_1_ExploringData

September 29, 2020

1 AnomalyDetection_1_ExploringData

The first part of the project (separating significant movements from non-significant ones) has been complete, with the following condition having been found:

$$M = \begin{cases} -1 & g_i g_i > p\left(\frac{1}{\dot{\theta}_i \dot{\theta}_i}\right) \\ 1 & g_i g_i \leq p\left(\frac{1}{\dot{\theta}_i \dot{\theta}_i}\right) \end{cases}$$

$$p(x_i) = C_i x_i$$

$$C_i = \begin{pmatrix} 0.7741697399557282 \\ -0.15839741967042406 \\ 0.09528795099596377 \\ -0.004279871380772796 \end{pmatrix} \text{ and } x_i = \begin{pmatrix} x^4 \\ x^2 \\ x \\ 1 \end{pmatrix}$$

On the assumption that this is a good model (ideally given more resources and time, more elaborate testing would have been carried out), the goal now is to find anomalies in time series of the significant movements.

1.1 Libraries and Configuration

[1]:
```
""" Libraries """

#file / system libraries
import os
import datetime as dt

# mathematical

import numpy as np

# data exploration

import pandas as pd
```

```
# data visualization

import matplotlib.pyplot as plt

""" Configuration """

# pandas

pd.set_option('display.max_columns', None)
```

1.2 Functions

[2]:
```
def polynomial(x):
    """ takes an array and returns it after our polynomial function has been␣
 ↪applied to it"""
    C = [0.7741697399557282,-0.15839741967042406,0.09528795099596377,-0.
 ↪0042798713807727796]
    y = C[0]*np.power(x,4)+C[1]*np.power(x,2)+C[2]*x+C[3]
    return y

def directory_to_df(paths, exclude = [None], filetype = '.csv',ignore_index =␣
 ↪True, exception = '_repet'):
    """ concatenates all files in a directory into a dataframe
    components:
    path: path to the directory (must end with /)
    exclude: array of directories to excludes from the treatment
    filetype: a string of the file extension (must include .)
    ignore_index: boolean that tells pandas to ignore the index or not
    exception: takes a string. Any time a filename includes this string it is␣
 ↪treated differently (for cases when you have
    more than one )
    """
    filenames = []
    file_column = []
    frames = []
    test_index = 1

    for path in paths:
        for filename in os.listdir(path):
            print(path)
            if filetype in filename and filename not in exclude:
                if exception in filename:
                    curr_df = pd.read_csv(path+filename)
                    curr_df = special_treatment(curr_df)
                else:
                    curr_df = pd.read_csv(path+filename)
                frames.append(curr_df)
                filenames.append(filename.replace(filetype,''))
                for i in range(curr_df.shape[0]):
                    file_column.append(test_index)
                test_index+=1

    df = pd.concat(frames,ignore_index = ignore_index)
    df['files'] = file_column
    return df, filenames


def special_treatment(df):
    """ performs a custom operation on a dataframe
    components:
    df: dataframe to play on
    """
    columns = df.columns.values.tolist()
    columns.remove('date')
    df.drop('gyrZ',inplace = True, axis = 1)
    df.columns = columns
    df.reset_index(inplace = True)
    df.rename(columns= {'index':'date'},inplace = True)
    return df
```

1.3 Data

[18]:
```
base = '/Users/yousefnami/KinKeepers/ProjectAI/Kin-Keepers/Data/{}'
paths = [base.format('Rohan/'),base.format('Ignacio/')]
frames = []

for index,path in enumerate(paths):
    frames.append(directory_to_df([path]))
    frames[index][0]['accTotal'] =  np.sqrt(np.
 ↪power(frames[index][0][['accX','accY','accZ']],2).sum(axis = 1))
    frames[index][0]['gyrTotal'] =  np.sqrt(np.
 ↪power(frames[index][0][['gyrX','gyrY','gyrZ']],2).sum(axis = 1))




df_rohan = frames[0][0]
df_ignacio = frames[1][0]



dfs = []
```

```
dfs.append(df_rohan)
dfs.append(df_ignacio)
names = ["rohan's data", "ignacio's data"]

for index,df in enumerate(dfs):
    dfs[index] = df[df.accTotal > polynomial(1/df.gyrTotal)]
    dfs[index].to_csv('/Users/yousefnami/KinKeepers/ProjectAI/Kin-Keepers/Data/
 ↪{}_filtered.csv'.format(names[index][:-7]))

df_rohan = dfs[0]
df_ignacio = dfs[1]
```

```
/Users/yousefnami/KinKeepers/ProjectAI/Kin-Keepers/Data/Rohan/
/Users/yousefnami/KinKeepers/ProjectAI/Kin-Keepers/Data/Rohan/
/Users/yousefnami/KinKeepers/ProjectAI/Kin-Keepers/Data/Rohan/
/Users/yousefnami/KinKeepers/ProjectAI/Kin-Keepers/Data/Rohan/
/Users/yousefnami/KinKeepers/ProjectAI/Kin-Keepers/Data/Ignacio/
/Users/yousefnami/KinKeepers/ProjectAI/Kin-Keepers/Data/Ignacio/
/Users/yousefnami/KinKeepers/ProjectAI/Kin-Keepers/Data/Ignacio/
/Users/yousefnami/KinKeepers/ProjectAI/Kin-Keepers/Data/Ignacio/
/Users/yousefnami/KinKeepers/ProjectAI/Kin-Keepers/Data/Ignacio/
/Users/yousefnami/KinKeepers/ProjectAI/Kin-Keepers/Data/Ignacio/
```

[4]:
```
df_rohan
```

[4]:
```
                      date  accX  accY  accZ    gyrX   gyrY   gyrZ  files  \
220    2020-09-14 19:19:26  0.01  0.02  0.00    3.62   1.04   1.38      1
319    2020-09-14 19:20:39  0.09  0.16  0.14   36.11  25.84  67.85      1
320    2020-09-14 19:20:40  0.09  0.16  0.09   22.98  15.43  16.45      1
321    2020-09-14 19:20:41  0.05  0.07  0.09   22.98  15.43  16.45      1
322    2020-09-14 19:20:42  0.12  0.07  0.07   29.44  39.83  27.27      1
…                      …    …    …    …    …      …      …      …
31311  2020-09-09 18:04:12  0.03 -0.03  0.05   -3.88   3.52  -6.43      4
31312  2020-09-09 18:04:13  0.01  0.01  0.00   -2.91  -1.91  -2.85      4
31313  2020-09-09 18:04:13  0.00  0.03 -0.05 -122.00   5.52  -0.07      4
31314  2020-09-09 18:04:14 -0.10  0.05  0.20  -26.89  38.96  29.60      4
31315  2020-09-09 18:04:15  0.14 -0.09  0.10   36.23 -63.69 -18.68      4

       accTotal   gyrTotal
220    0.022361   4.011284
319    0.230868  81.087978
320    0.204450  32.198879
321    0.124499  32.198879
322    0.155563  56.540210
…           …          …
31311  0.065574   8.293956
31312  0.014142   4.498744
```

```
31313  0.058310  122.124835
31314  0.229129   55.831118
31315  0.194165   75.617269

[1478 rows x 10 columns]
```

[5]:
```
df_ignacio
```

[5]:
```
                     date  accX  accY  accZ   gyrX   gyrY   gyrZ  files  \
0     2020-09-13 17:09:25  0.02  0.12  0.03   1.47   3.32   2.22      1
1     2020-09-13 17:09:26  0.02  0.12  0.03   1.47   3.32   2.22      1
2     2020-09-13 17:09:27  0.01  0.01  0.00   7.43   6.82  10.10      1
12    2020-09-13 17:09:34  0.01  0.01  0.00   6.64   7.07  12.45      1
13    2020-09-13 17:09:34  0.01  0.01  0.00   4.12   3.61   5.81      1
…                     …    …     …     …     …      …      …      …
46380 2020-09-19 23:39:58  0.04  0.05  0.03   7.75   5.83   4.42      4
46384 2020-09-19 23:40:01  0.00  0.01  0.02   4.95   4.24   1.71      4
46385 2020-09-19 23:40:02  0.01  0.01  0.02   4.95   4.24   1.71      4
46386 2020-09-19 23:40:03  0.05  0.07  0.04  14.41  12.23  20.73      4
46387 2020-09-19 23:40:03  0.05  0.07  0.04   1.55   1.27   0.74      4

       accTotal   gyrTotal
0      0.125300   4.255784
1      0.125300   4.255784
2      0.014142  14.273307
12     0.014142  15.782173
13     0.014142   7.985149
…           …          …
46380  0.070711  10.657758
46384  0.022361   6.738264
46385  0.024495   6.738264
46386  0.094868  28.052699
46387  0.094868   2.136118

[2623 rows x 10 columns]
```

[6]:
```
for df in dfs:
    df.date = pd.to_datetime(df.date)
    times = []

    # this is good, but you must apply it for EACH day
    for index,time in enumerate(df.date.values):
        if index == 0:
            times.append((time - time)/np.timedelta64(1, 's'))
        else:
            times.append((time - df.date.values[0])/np.timedelta64(1, 's'))
    df['times'] = times
```

```
print('time',type(time))
print('value',type(df.date.values[0]))
df_ignacio.head()
```

```
time <class 'numpy.datetime64'>
value <class 'numpy.datetime64'>
```

```
/Users/yousefnami/python_environments/KinKeepers_AI/lib/python3.7/site-
packages/pandas/core/generic.py:5159: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
/Users/yousefnami/python_environments/KinKeepers_AI/lib/python3.7/site-
packages/ipykernel_launcher.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  # This is added back by InteractiveShellApp.init_path()
```

[6]:
```
                  date  accX  accY  accZ  gyrX  gyrY   gyrZ  files  accTotal  \
0  2020-09-13 17:09:25  0.02  0.12  0.03  1.47  3.32   2.22      1  0.125300
1  2020-09-13 17:09:26  0.02  0.12  0.03  1.47  3.32   2.22      1  0.125300
2  2020-09-13 17:09:27  0.01  0.01  0.00  7.43  6.82  10.10      1  0.014142
12 2020-09-13 17:09:34  0.01  0.01  0.00  6.64  7.07  12.45      1  0.014142
13 2020-09-13 17:09:34  0.01  0.01  0.00  4.12  3.61   5.81      1  0.014142

     gyrTotal  times
0    4.255784    0.0
1    4.255784    1.0
2   14.273307    2.0
12  15.782173    9.0
13   7.985149    9.0
```

[15]:
```
class seasonality():
    """ takes in a dataframe, outputting it with two extra columns: seasonality␣
↪(but column name = seasonality
    inputted) and times, where 'times' is a plottable version of date with␣
↪reference to a prespecified start time
    (day_start)
    Components:
    df: the dataframe, must have the dates column as 'date' and in np.
↪datetime64 timeformat
```

```
    seasonality (optional): defaults to 'day'. This is the criteria for␣
↪splitting the data
    day_start (optional): this signifies what is the 'start time' of the day (i.
↪e. the 0 point on the x axis). Defaults
    for midnight.
    time_delta (optional): this defines the units for the time delta between␣
↪data points. Defaults to seconds.
    EDIT THIS MSG
    NEED TO FIX THIS
    """
    def __init__(self,df,seasonality='day',day_start = '00:00:00', time_delta =␣
↪'s'):

        if seasonality not in ['hour','day','month','year']:
            raise ValueError("you can only input the following for seasonality:␣
↪'day', 'month', or 'year'")
        self.df = df
        self.seasonality = 'seasonality_{}'.format(seasonality)
        try:
            self.day_start = dt.datetime.strptime(day_start,'%H:%M:%S')
        except:
            raise ValueError('Please enter your day_start in the correct format:
↪ "HH:MM:SS". "{}" is not acceptable'\
                                    .format(day_start))
        self.time_delta = time_delta

    def find_seasonal_trends(self):
        if 'hour' in self.seasonality:
            self.df[self.seasonality] = self.df.date.dt.hour
        elif 'day' in self.seasonality:
            self.df[self.seasonality] = self.df.date.dt.day
        elif 'month' in self.seasonality:
            self.df[self.seasonality] = self.df.date.dt.month
        else:
            self.df[self.seasonality] = self.df.date.dt.year

        self.create_times()


        return self.df

    def create_times(self):
        times = []
        for season in self.df[self.seasonality].unique():
            temp_dates = self.df.date[self.df[self.seasonality] == season].
↪values
```

```
            date = dt.datetime.strptime(str(temp_dates[0])[:-3], '%Y-%m-%dT%H:
↪%M:%S.%f')
            # 'date' is wrong: this will not work for when you have a lower␣
↪order seasonality.
            # it needs to adapt such that it starts recording when the␣
↪beginning of the year
            start_day = dt.datetime(date.year,
                                    date.month,
                                    date.day,
                                    self.day_start.hour,
                                    self.day_start.minute,
                                    self.day_start.second)
            start_day = np.datetime64(start_day)

            for index, date in enumerate(temp_dates):
                times.append((date - start_day)/np.timedelta64(1, self.
↪time_delta))
        self.df['times'] = times
```

[8]:
```
df_temp = df_ignacio
#df_temp.date = pd.to_datetime(df_temp.date)

myObj = seasonality(df_temp,time_delta = 's')


df_temp = myObj.find_seasonal_trends()

#df_ignacio = find_seasonal_trends(df_ignacio,seasonality = 'month')
#df_ignacio.date.dt.day
#df_ignacio.head()
df_temp.head()
```

```
/Users/yousefnami/python_environments/KinKeepers_AI/lib/python3.7/site-
packages/ipykernel_launcher.py:30: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/Users/yousefnami/python_environments/KinKeepers_AI/lib/python3.7/site-
packages/ipykernel_launcher.py:58: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
[8]:                   date  accX  accY  accZ  gyrX  gyrY   gyrZ  files  accTotal  \
     0  2020-09-13 17:09:25  0.02  0.12  0.03  1.47  3.32   2.22      1  0.125300
     1  2020-09-13 17:09:26  0.02  0.12  0.03  1.47  3.32   2.22      1  0.125300
     2  2020-09-13 17:09:27  0.01  0.01  0.00  7.43  6.82  10.10      1  0.014142
     12 2020-09-13 17:09:34  0.01  0.01  0.00  6.64  7.07  12.45      1  0.014142
     13 2020-09-13 17:09:34  0.01  0.01  0.00  4.12  3.61   5.81      1  0.014142

          gyrTotal    times  seasonality_day
     0    4.255784  61765.0               13
     1    4.255784  61766.0               13
     2   14.273307  61767.0               13
     12  15.782173  61774.0               13
     13   7.985149  61774.0               13
```

```
[9]: #df.date = pd.to_datetime(df_temp.date)
     for index,df in enumerate(dfs):
         seasonal = seasonality(df)
         dfs[index] = seasonal.find_seasonal_trends()

     df_rohan = dfs[0]
     df_ignacio = dfs[1]
```

```
/Users/yousefnami/python_environments/KinKeepers_AI/lib/python3.7/site-
packages/ipykernel_launcher.py:30: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/Users/yousefnami/python_environments/KinKeepers_AI/lib/python3.7/site-
packages/ipykernel_launcher.py:58: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/Users/yousefnami/python_environments/KinKeepers_AI/lib/python3.7/site-
packages/ipykernel_launcher.py:30: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/Users/yousefnami/python_environments/KinKeepers_AI/lib/python3.7/site-
packages/ipykernel_launcher.py:58: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[10]:
```
df_rohan.head()
```

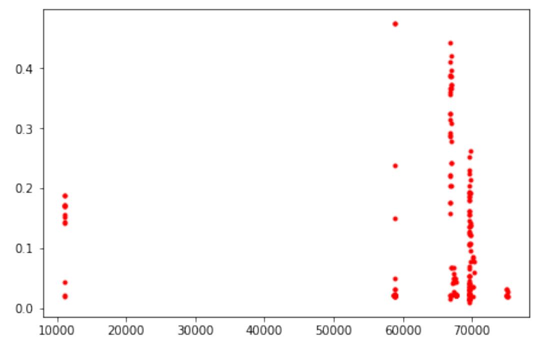
[10]:
```
                   date  accX  accY  accZ   gyrX   gyrY   gyrZ  files  \
220 2020-09-14 19:19:26  0.01  0.02  0.00   3.62   1.04   1.38      1
319 2020-09-14 19:20:39  0.09  0.16  0.14  36.11  25.84  67.85      1
320 2020-09-14 19:20:40  0.09  0.16  0.09  22.98  15.43  16.45      1
321 2020-09-14 19:20:41  0.05  0.07  0.09  22.98  15.43  16.45      1
322 2020-09-14 19:20:42  0.12  0.07  0.07  29.44  39.83  27.27      1

     accTotal   gyrTotal    times  seasonality_day
220  0.022361   4.011284  69566.0               14
319  0.230868  81.087978  69639.0               14
320  0.204450  32.198879  69640.0               14
321  0.124499  32.198879  69641.0               14
322  0.155563  56.540210  69642.0               14
```

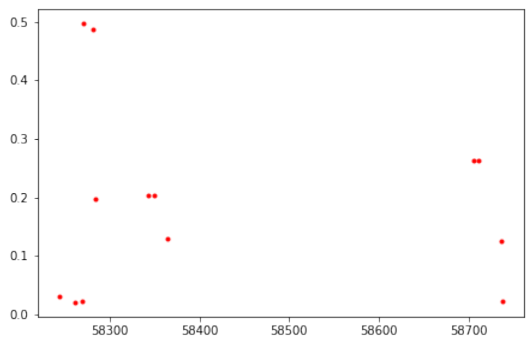
[11]:
```
df_ignacio.head()
```

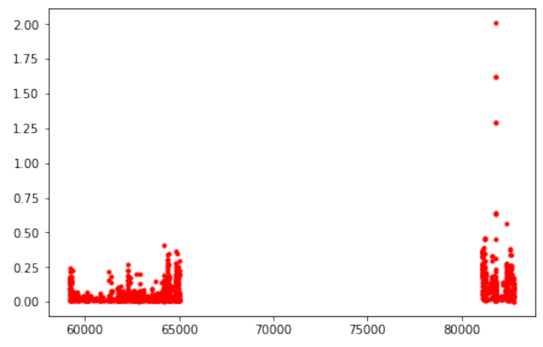
[11]:
```
                  date  accX  accY  accZ  gyrX  gyrY   gyrZ  files  accTotal  \
0  2020-09-13 17:09:25  0.02  0.12  0.03  1.47  3.32   2.22      1  0.125300
1  2020-09-13 17:09:26  0.02  0.12  0.03  1.47  3.32   2.22      1  0.125300
2  2020-09-13 17:09:27  0.01  0.01  0.00  7.43  6.82  10.10      1  0.014142
12 2020-09-13 17:09:34  0.01  0.01  0.00  6.64  7.07  12.45      1  0.014142
13 2020-09-13 17:09:34  0.01  0.01  0.00  4.12  3.61   5.81      1  0.014142

     gyrTotal    times  seasonality_day
0    4.255784  61765.0               13
1    4.255784  61766.0               13
2   14.273307  61767.0               13
12  15.782173  61774.0               13
13   7.985149  61774.0               13
```

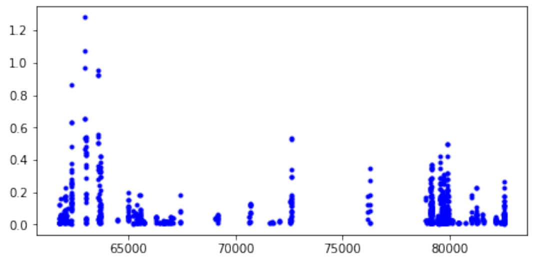
[12]:
```
colors = ['r','b']
i = 1
for df,color in zip(dfs,colors):
    fig = plt.figure(figsize = (16,16))
    for season in df.seasonality_day.unique():
        df_temp = df[df.seasonality_day == season]
        fig.add_subplot(len(df.seasonality_day.unique()),len(dfs),i)
        print(i)
        plt.plot(df_temp.times,df_temp.accTotal,'{}.'.format(color))
        i+=1
```

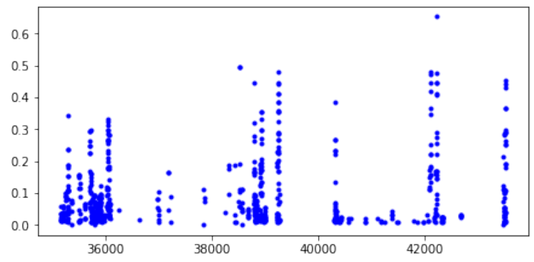
```
#plt.plot(df.times,df.accTotal,'{}.'.format(color))
#print(df.times.max())
```

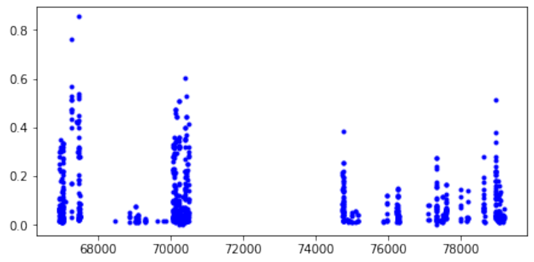


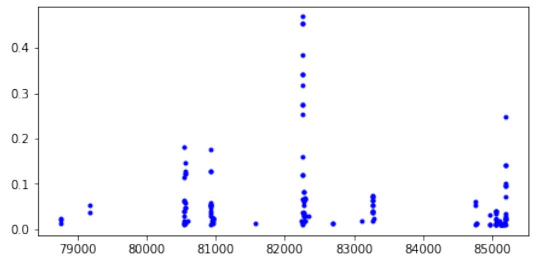




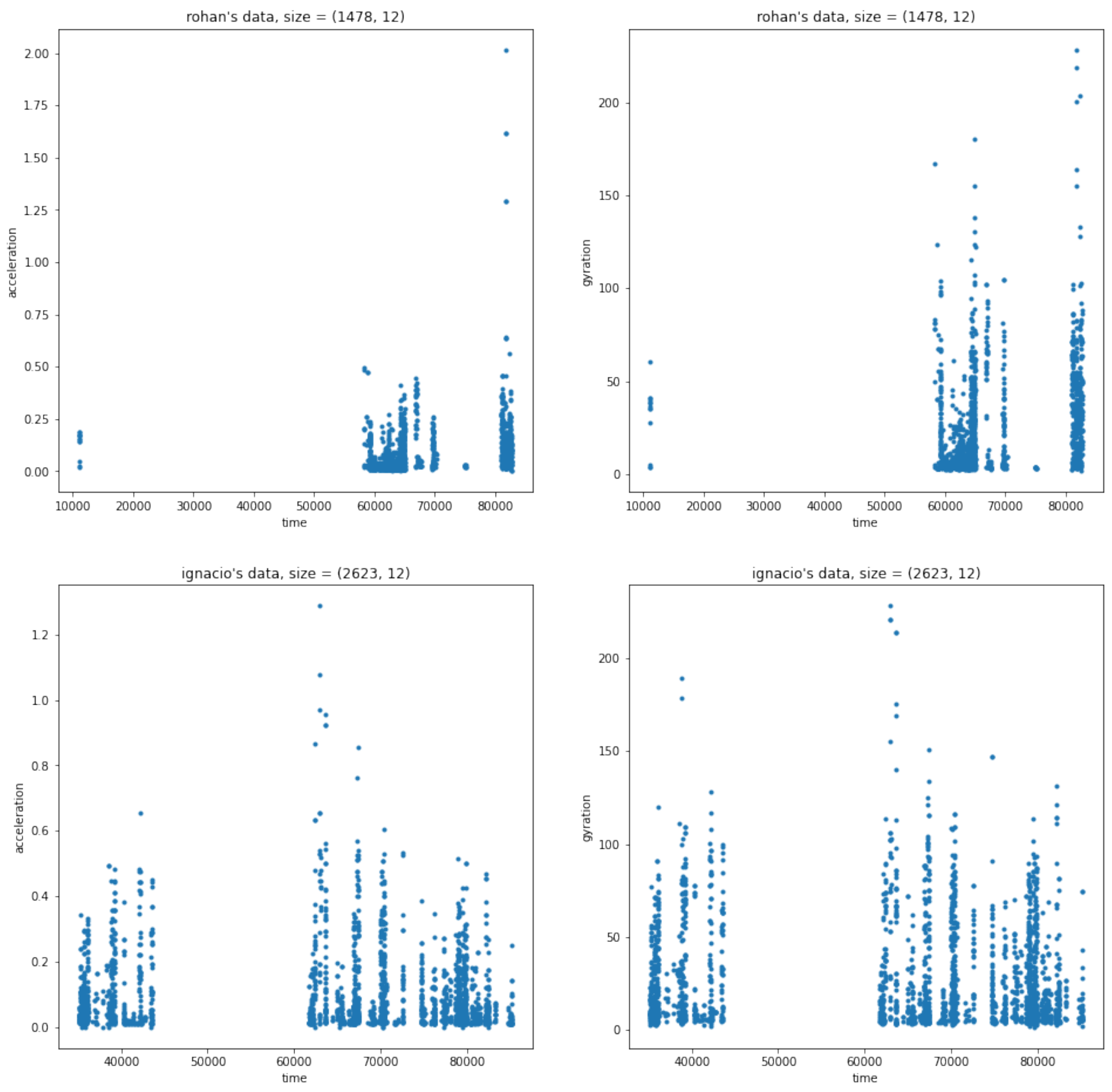








[14]:
```
fig = plt.figure(figsize = (16,16))
i = 1
for df,name in zip(dfs,names):
    fig.add_subplot(2,2,i)
    i+=1
    plt.plot(df.times,df.accTotal,'.')
    plt.title("{}, size = {}".format(name,df.shape))
    plt.xlabel('time')
    plt.ylabel('acceleration')

    fig.add_subplot(2,2,i)
    i+=1
    plt.plot(df.times,df.gyrTotal,'.')
    plt.title("{}, size = {}".format(name,df.shape))
    plt.xlabel('time')
    plt.ylabel('gyration')
```


rohan's data, size = (1478, 12)
acceleration
time
rohan's data, size = (1478, 12)
gyration
time
ignacio's data, size = (2623, 12)
acceleration
time
ignacio's data, size = (2623, 12)
gyration
time

[ ]: