Processing of triggerlessly acquired detector's data

PREPROCESSING

Load and prepare the dataset inside a Pandas' DataFrame.

In [6]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from os import listdir
from os.path import isfile, join

%matplotlib inline
```

Inside each directory there are several files related to different test. Now we want to merge all of them into the same DataFrame.

In [5]:
```
# Load the entire dataset inside the directory
# List all files inside the directory
# Run000260, Run000261, Run000333
directory = "/data/Run000260/"
file_names = [file for file in listdir(directory) if isfile(j
oin(directory, file))]

# Create dataframe by appending the data from each file
data = pd.read_csv(directory + file_names[0])
for i in range(1, len(file_names)):
    data = data.append(pd.read_csv(directory + file_names[i
]))
data.shape
```

```
---------------------------------------------------------------
--------------
KeyboardInterrupt                          Traceback (most rec
ent call last)
<ipython-input-5-008d3b4fff0a> in <module>
      8 data = pd.read_csv(directory + file_names[0])
      9 for i in range(1, len(file_names)):
---> 10     data = data.append(pd.read_csv(directory + file_n
ames[i]))
     11 data.shape

/usr/lib64/python3.6/site-packages/pandas/io/parsers.py in pa
rser_f(filepath_or_buffer, sep, delimiter, header, names, ind
ex_col, usecols, squeeze, prefix, mangle_dupe_cols, dtype, en
gine, converters, true_values, false_values, skipinitialspac
e, skiprows, nrows, na_values, keep_default_na, na_filter, ve
rbose, skip_blank_lines, parse_dates, infer_datetime_format,
 keep_date_col, date_parser, dayfirst, iterator, chunksize, c
ompression, thousands, decimal, lineterminator, quotechar, qu
oting, escapechar, comment, encoding, dialect, tupleize_cols,
error_bad_lines, warn_bad_lines, skipfooter, doublequote, del
im_whitespace, low_memory, memory_map, float_precision)
    676                         skip_blank_lines=skip_blank_line
    679
    680     parser_f.__name__ = name

/usr/lib64/python3.6/site-packages/pandas/io/parsers.py in _r
ead(filepath_or_buffer, kwds)
    444
    445     try:
--> 446         data = parser.read(nrows)
    447     finally:
    448         parser.close()

/usr/lib64/python3.6/site-packages/pandas/io/parsers.py in re
ad(self, nrows)
   1034                 raise ValueError('skipfooter not supp
orted for iteration')
   1035
-> 1036         ret = self._engine.read(nrows)
   1037
   1038         # May alter columns / col_dict

/usr/lib64/python3.6/site-packages/pandas/io/parsers.py in re
ad(self, nrows)
   1846     def read(self, nrows=None):
   1847         try:
-> 1848             data = self._reader.read(nrows)
   1849         except StopIteration:
   1850             if self._first_chunk:

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader.r
ead()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._
read_low_memory()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._
read_rows()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._
convert_column_data()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._
convert_tokens()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._
convert_with_dtype()

/usr/lib64/python3.6/site-packages/pandas/core/dtypes/common.
py in is_integer_dtype(arr_or_dtype)
    809
    810
--> 811 def is_integer_dtype(arr_or_dtype):
    812     """
    813     Check whether the provided array or dtype is of a
n integer dtype.

KeyboardInterrupt:
```

In [7]:

```
## TEST ONLY
```

```
# Load the dataset
data = pd.read_csv(directory + file_name)
```

In [8]:

```
# Useful constants
Tmax = 390 # ns
L = 42 # mm
Vd = L/(2*Tmax) # mm/ns
pos_offset = 21 # mm

# Add column of time (ns)
# There is a problem with the precision of the measures, so w
e drop the orbit
# Real time: data['TIME_NS'] = data["ORBIT_CNT"]*3564*25 + da
ta["BX_COUNTER"]*25 + data["TDC_MEAS"]*25/30
data['TIME_NS'] = data["BX_COUNTER"]*25 + data["TDC_MEAS"]*25
/30

# Show first 5 rows
data.head(5)
```

Out[8]:

	HEAD	FPGA	TDC_CHANNEL	ORBIT_CNT	BX_COUNTER	TDC_MEAS	T
0	1	1	116	1897414884	1533	24	3
1	1	1	71	1897414887	1650	21	4
2	1	1	67	1897414914	980	8	2
3	1	1	70	1897414922	1287	8	3
4	1	0	57	1897414922	2162	22	5

To compute the constant t_0, which is different for every event, we can use the following relation:

$$T_{MAX} = \frac{t_1 + t_3}{2} + t_2$$

where $t_1 = t_{R_1} - t_0$, $t_2 = t_{R_2} - t_0$ and $t_3 = t_{R_3} - t_0$. Then the relation become:

$$T_{MAX} = \frac{t_{R_1} - t_0 + t_{R_3} - t_0}{2} + t_{R_2} - t_0$$

from which we get:

$$t_0 = \frac{t_{R_1} + t_{R_3} + 2t_{R_2} - 2T_{MAX}}{4}$$

Finally we notice that t_{R_1}, t_{R_2}, t_{R_3} are the times recorded by each cell, which are already available in our dataset.

Before processing the dataset, we have to create some missing columns, in fact the DataFrame with the events must contain the following information:

- CHAMBER, which is the Detector number [1-4];
- LAYER, which is the layer of the cell [1-4];
- CELL, which is in the number of the cell [1-16];
- POSTION, which is the position where a particle traverses the cell [0-21] (in mm).

To get the layer we can compute the remainder of the TDC_CHANNEL with 4 (total number of layers), and then we have to remap the values in the following way:

REMAINDER	LAYER
0	1
1	4
2	2
3	3

In [9]:

```
# To get the layer we must get the remainder of the TDC_CHANN
EL with 4
# Then we must reoder the result as described above
data['LAYER'] = data['TDC_CHANNEL'] % 4

# Map 1 --> 4
data.loc[data['LAYER'] == 1,'LAYER'] = 4

# Map 0 -> 1
data.loc[data['LAYER'] == 0,'LAYER'] = 1

# Check the correctness
data.head(5)
```

Out[9]:

	HEAD	FPGA	TDC_CHANNEL	ORBIT_CNT	BX_COUNTER	TDC_MEAS	T
0	1	1	116	1897414884	1533	24	3
1	1	1	71	1897414887	1650	21	4
2	1	1	67	1897414914	980	8	2
3	1	1	70	1897414922	1287	8	3
4	1	0	57	1897414922	2162	22	5

Column of CHAMBER

Create the column for the chamber according to the following rules:

- Detector 1 → FPGA 0, TDC_CHANNEL in [1-64]
- Detector 2 → FPGA 0, TDC_CHANNEL in [65-128]
- Detector 3 → FPGA 1, TDC_CHANNEL in [1-64]
- Detector 4 → FPGA 1, TDC_CHANNEL in [65-128]

In [10]:

```
# Create column for chamber
# Before create empty column
data['CHAMBER'] = 0

# Detector 1
# Select all rows with FPGA = 0 and TDC_CHANNEL <= 64
data.loc[(data['FPGA'] == 0) & (data['TDC_CHANNEL'] <= 64),'C
HAMBER'] = 1
```

```
data['TDC_CHANNEL'] <= 128),'CHAMBER'] = 2

# Detector 3
# Select all rows with FPGA = 1 and TDC_CHANNEL <= 64
data.loc[(data['FPGA'] == 1) & (data['TDC_CHANNEL'] <= 64),
'CHAMBER'] = 3

# Detector 4
# Select all rows with FPGA = 0 and  64 < TDC_CHANNEL <= 128
data.loc[(data['FPGA'] == 1) & (data['TDC_CHANNEL'] > 64) & (
data['TDC_CHANNEL'] <= 128),'CHAMBER'] = 4

# Check the correctness
data.head(5)
```

Out[10]:

	HEAD	FPGA	TDC_CHANNEL	ORBIT_CNT	BX_COUNTER	TDC_MEAS	T
0	1	1	116	1897414884	1533	24	3
1	1	1	71	1897414887	1650	21	4
2	1	1	67	1897414914	980	8	2
3	1	1	70	1897414922	1287	8	3
4	1	0	57	1897414922	2162	22	5

Column of CELL

This conlumn contains the values from 1 to 16. These values can be obtained as follows:

$$\lceil \frac{N_{CHANNEL}\%64}{4} \rceil$$

In [11]:

```
# Create column for chamber
data['CELL'] = ((data['TDC_CHANNEL']%64)/4).apply(np.ceil).as
type(int)

# Check the correctness
data.head(5)
```

Out[11]:

	HEAD	FPGA	TDC_CHANNEL	ORBIT_CNT	BX_COUNTER	TDC_MEAS	T
0	1	1	116	1897414884	1533	24	3
1	1	1	71	1897414887	1650	21	4
2	1	1	67	1897414914	980	8	2
3	1	1	70	1897414922	1287	8	3
4	1	0	57	1897414922	2162	22	5

PART 1

The dataset is ready to be processed, so we can start detecting the events through the trigger 139.

```
# Search all the orbit with the trigger 139
orbit = data.loc[data['TDC_CHANNEL'] == 139,'ORBIT_CNT']
list_orbit = orbit.values.tolist()
events = data.loc[data['ORBIT_CNT'].isin(list_orbit)]

# Sort data
events = events.sort_values(by = ['ORBIT_CNT','TDC_CHANNEL'])
events.head(5)
```

Out[12]:

	HEAD	FPGA	TDC_CHANNEL	ORBIT_CNT	BX_COUNTER	TDC_MEAS
5	1	0	24	1897414934	2014	13
6	1	0	26	1897414934	2014	20
7	1	0	26	1897414934	2026	13
10	1	0	27	1897414934	2024	11
11	1	0	29	1897414934	2026	9

Computation of t_0

To compute t_0 we have to apply the Talete's equation to the cell alignment inside our dataset. We will limit our search to the following patterns inside the 'LAYER' column:

- 1, 2, 3
- 2, 3, 4

This process can be easily generalized to other patterns.

In [14]:

```
# Remove the trigger 139
events = events[events['TDC_CHANNEL']<129]

# Make three shifted copy of the LAYER column nd of TIME_NS
events['LAYER_1'] = events['LAYER'].shift(-1)
events['LAYER_2'] = events['LAYER'].shift(-2)
events['TIME_NS_1'] = events['TIME_NS'].shift(-1)
events['TIME_NS_2'] = events['TIME_NS'].shift(-2)

# Search pattern to get the real t0
mask_pattern_1 = (events['LAYER']==1) & (events['LAYER_1']==2
) & (events['LAYER_2']==3)
mask_pattern_2 = (events['LAYER']==2) & (events['LAYER_1']==3
) & (events['LAYER_2']==4)
mask_pattern = mask_pattern_1 | mask_pattern_2

# Search pattern 1-2-3 or 2-3-4 to apply the Talete's Theorem
for t0
events.loc[mask_pattern, 't0'] = (events['TIME_NS'] + events[
'TIME_NS_2'] + 2*events['TIME_NS_1'] - 2*Tmax)/4

# Populate values of adiacent cell (according to the choosen
 pattern)
events = events.fillna(0)
events['t0'] = events['t0'] + events['t0'].shift(1) + events[
't0'].shift(2)
```

```
events.head(5)
```

Out[14]:

	HEAD	FPGA	TDC_CHANNEL	ORBIT_CNT	BX_COUNTER	TDC_MEAS
5	1	0	24	1897414934	2014	13
6	1	0	26	1897414934	2014	20
7	1	0	26	1897414934	2026	13
10	1	0	27	1897414934	2024	11
11	1	0	29	1897414934	2026	9

Column POSITION

Only at this point we can create the column with the position, thanks to t_0.

In [15]:

```
# Compute the position
events.loc[events['t0']!=0,'POSITION'] = (events['TIME_NS'] - events['t0'])*Vd
events = events.fillna(0)

events.head(5)
```

Out[15]:

	HEAD	FPGA	TDC_CHANNEL	ORBIT_CNT	BX_COUNTER	TDC_MEAS
5	1	0	24	1897414934	2014	13
6	1	0	26	1897414934	2014	20
7	1	0	26	1897414934	2026	13
10	1	0	27	1897414934	2024	11
11	1	0	29	1897414934	2026	9

In [64]:

```
# Map obit values to a range of int
grouped_orbit = events.groupby('ORBIT_CNT')
# Search all orbits
orbits = list(grouped_orbit.groups.keys())
# Create increasing number list for the events
event_number = np.arange(1, len(orbits)+1)
# Create the map
event_map = dict(zip(orbits, event_number))
# Map values
orbit_to_map = events['ORBIT_CNT']
orbit_mapped = orbit_to_map.map(event_map)
events['EVENT_NUMBER'] = orbit_mapped
events.head(5)
```

Out[64]:

	HEAD	FPGA	TDC_CHANNEL	ORBIT_CNT	BX_COUNTER	TDC_MEAS
5	1	0	24	1897414934	2014	13
6	1	0	26	1897414934	2014	20
7	1	0	26	1897414934	2026	13

In [65]:
```
# Remove unexpected values of the position
events.loc[(events['POSITION']<0) | (events['POSITION']>=21), 'POSITION'] = 0
```

In [66]:
```
# Final DataFrame
events_final = events[['EVENT_NUMBER','CHAMBER','LAYER','CELL','POSITION']]
events_final.set_index(['EVENT_NUMBER','CHAMBER','LAYER'], inplace=True)
events_final.sort_index(inplace=True)
events_final.head(5)
```

Out[66]:

			CELL	POSITION
EVENT_NUMBER	**CHAMBER**	**LAYER**		
1	**1**	**1**	6	0.000000
		2	7	0.000000
		2	7	11.935897
		3	7	9.153846
		4	8	11.756410

In [67]:
```
events_pattern = events[(events['t0']!=0) & (events['POSITION']!=0)]
events_pattern
```

Out[67]:

	HEAD	FPGA	TDC_CHANNEL	ORBIT_CNT	BX_COUNTER	TDC_M
7	1	0	26	1897414934	2026	13
10	1	0	27	1897414934	2024	11
11	1	0	29	1897414934	2026	9
139	1	1	41	1897415674	1195	9
229	1	1	10	1897416153	2931	4
243	1	0	86	1897416210	2941	28
342	1	0	74	1897416591	542	12
338	1	0	75	1897416591	535	10
339	1	0	81	1897416591	535	16
365	1	0	4	1897416718	2248	20
366	1	0	9	1897416718	2252	6
395	1	0	40	1897416923	3020	4
394	1	0	45	1897416923	3015	24
424	1	1	110	1897417046	1468	22

440	1	1	116	1897417046	1473	24
426	1	1	118	1897417046	1468	13
433	1	1	119	1897417046	1469	0
428	1	1	122	1897417046	1468	26
434	1	1	123	1897417046	1469	21
438	1	1	125	1897417046	1468	15
458	1	0	100	1897417112	2150	21
460	1	0	105	1897417112	2148	13
596	1	0	120	1897417876	3267	22
601	1	0	68	1897417883	229	7
720	1	1	116	1897418327	1533	18
724	1	1	121	1897418327	1541	24
761	1	0	20	1897418583	2539	17
760	1	0	22	1897418583	2535	5
...	...	...	...	...	...	...
1308947	1	1	34	1920953258	3427	3
1308948	1	1	91	1920953258	3427	16
1308938	1	1	93	1920953258	3419	21
1309038	1	1	69	1920953339	1379	15
1309047	1	0	124	1920953339	1378	18
1309037	1	0	126	1920953339	1375	13
1309040	1	0	127	1920953339	1374	4
1309098	1	1	14	1920953380	2300	29
1309155	1	1	112	1920953426	645	15
1309158	1	1	114	1920953426	651	3
1309256	1	0	80	1920953522	2260	10
1309277	1	1	108	1920953535	2106	23
1309284	1	1	16	1920953548	3453	27
1309285	1	1	78	1920953548	3461	23
1309282	1	1	79	1920953548	3452	6
1309507	1	1	47	1920953732	3246	2
1309506	1	1	101	1920953732	3245	29
1309513	1	1	80	1920953734	3324	19
1309514	1	1	82	1920953734	3338	8

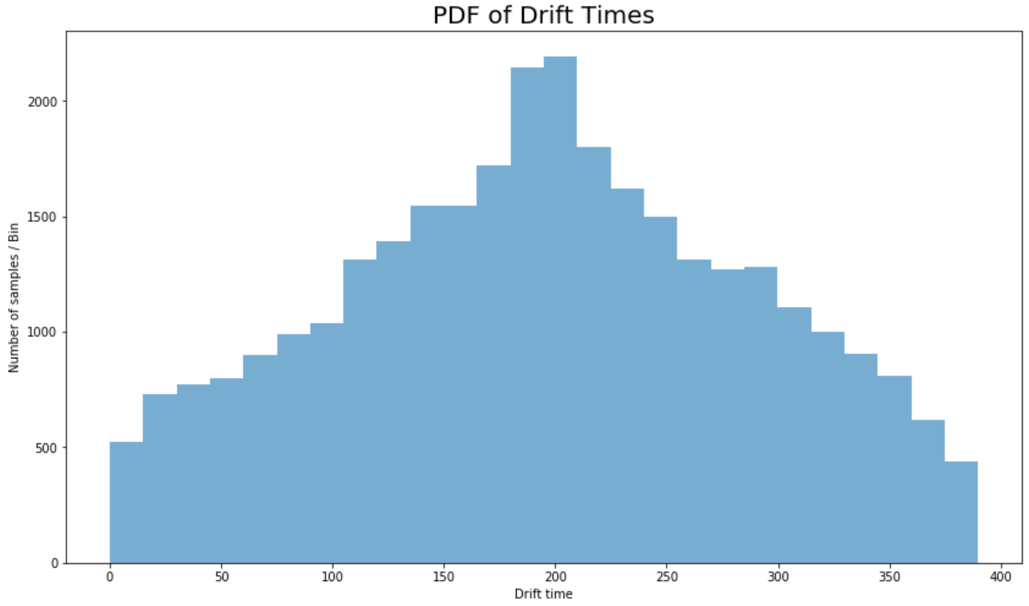
1309892	1	1	93	1920953944	1828	13
1309971	1	0	87	1920953985	2424	2
1309968	1	0	113	1920953985	2413	26
1309992	1	0	11	1920953995	1840	15
1309989	1	0	69	1920953995	1833	5
1310069	1	0	4	1920954050	259	24
1310575	1	1	48	1920954509	1134	27
1310578	1	1	50	1920954509	1146	25
1310576	1	1	111	1920954509	1134	22

31279 rows × 18 columns

PLOT DISTRIBUTION OF DRIFT TIMES

```
In [92]: # Get all drift times
         drift_times = events_pattern['TIME_NS']-events_pattern['t0']

         # Plot PDF
         figure = plt.figure(figsize=(14,8))
         ax = figure.add_subplot(111)
         number_bins = 26
         y, edges, bins = ax.hist(drift_times, bins = number_bins, lab
         el='PDF', alpha=0.6)
         ax.set_ylabel("Number of samples / Bin")
         ax.set_xlabel("Drift time")
         ax.set_title("PDF of Drift Times", fontsize=20)
```

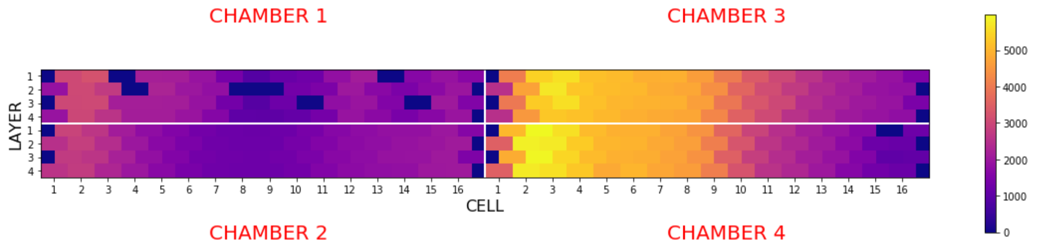
```
Out[92]: Text(0.5, 1.0, 'PDF of Drift Times')
```



PLOT OF THE DATAFRAME

```
# Rebuild the events' dataframe
orbit = data.loc[data['TDC_CHANNEL']==139,'ORBIT_CNT']
list_orbit = orbit.values.tolist()
events = data.loc[data['ORBIT_CNT'].isin(list_orbit)]
events.head(100)


# Remove unreal hits
events = events[events['TDC_CHANNEL']<129]

# Build the hit matrix
raw_mat = np.zeros((16, 32))
rows = np.array((events['CHAMBER']-1)*4+events['LAYER']-1)
columns = np.array((events['CELL']-1)*2).astype(int)

for i in range(len(rows)):
    raw_mat[rows[i], columns[i]] = raw_mat[rows[i], columns[i
]]+1
    raw_mat[rows[i], columns[i]+1] = raw_mat[rows[i], columns
[i]+1]+1

# Reshape the hit matrix
final_mat = np.zeros((8, 66))

# Chamber 1,2
for i in range(8):
    if (i%2 == 0):
        final_mat[i, 1:33] = raw_mat[i, :32]
    else:
        final_mat[i, :32] = raw_mat[i, :32]
# Chamber 3,4
for i in range(8, 16):
    if (i%2 == 0):
        final_mat[i-8, 34:66] = raw_mat[i, :32]
    else:
        final_mat[i-8, 33:65] = raw_mat[i, :32]


# Showing the resutls
plt.figure(figsize=(20,4))
ax = plt.imshow(final_mat, cmap='plasma')

plt.annotate('CHAMBER 1',xy=(0.5, 0.5), xytext=(12,-4), fonts
ize=20, color='red')
plt.annotate('CHAMBER 2',xy=(0.5, 0.5), xytext=(12,12), fonts
ize=20, color='red')
plt.annotate('CHAMBER 3',xy=(0.5, 0.5), xytext=(46,-4), fonts
ize=20, color='red')
plt.annotate('CHAMBER 4',xy=(0.5, 0.5), xytext=(46,12), fonts
ize=20, color='red')

plt.axvline(x=32.5, color='white', linewidth=2)
plt.axhline(y=3.5, color='white', linewidth=2)

plt.xticks(np.concatenate((np.arange(0.5, 32, 2 ), np.arange(
33.5, 65, 2))),
           ['1','2','3','4','5','6','7','8','9','10','11','1
2','13','14','15','16',
'3','4'])
plt.ylabel('LAYER', fontsize=16)
plt.xlabel('CELL', fontsize=16)
plt.colorbar()
```

```
Out[12]: <matplotlib.colorbar.Colorbar at 0x7fa9e47aa668>
```



CODICE VECCHIO (NON CANCELLARE)

```
In [10]: ## continuiamo col procedimento vecchio per la seconda parte

events_final = pd.DataFrame(columns=['ORBIT','CHAMBER','LAYE
R','CELL'])

gr_events = events.groupby('ORBIT_CNT')

c = 0
for orb, gr in gr_events:
    events_final.loc[c] = [orb, np.array(gr['CHAMBER']), np.a
rray(gr['LAYER']), np.array(gr['CELL'])]
    c+=1
events_final.head()
```

```
-------------------------------------------------------------
--------------
KeyboardInterrupt                         Traceback (most rec
ent call last)
<ipython-input-10-e609961db0bd> in <module>
      5 c = 0
      6 for orb, gr in gr_events:
----> 7     events_final.loc[c] = [orb, np.array(gr['CHAMBER'
]), np.array(gr['LAYER']), np.array(gr['CELL'])]
      8     c+=1
      9 events_final.head()

/usr/lib64/python3.6/site-packages/pandas/core/indexing.py in
__setitem__(self, key, value)
    187             key = com._apply_if_callable(key, self.ob
j)
    188         indexer = self._get_setitem_indexer(key)
--> 189         self._setitem_with_indexer(indexer, value)
    190
    191     def _validate_key(self, key, axis):

/usr/lib64/python3.6/site-packages/pandas/core/indexing.py in
_setitem_with_indexer(self, indexer, value)
    449                                        name=indexer)
    450
--> 451                         self.obj._data = self.obj.append(
    453                         return self.obj

/usr/lib64/python3.6/site-packages/pandas/core/frame.py in ap
pend(self, other, ignore_index, verify_integrity, sort)
   6209         return concat(to_concat, ignore_index=ignore_
index,
   6210                       verify_integrity=verify_integri
ty,
-> 6211                       sort=sort)
   6212
   6213     def join(self, other, on=None, how='left', lsuffi
x='', rsuffix='',

/usr/lib64/python3.6/site-packages/pandas/core/reshape/conca
t.py in concat(objs, axis, join, join_axes, ignore_index, key
s, levels, names, verify_integrity, sort, copy)
    224                        verify_integrity=verify_integr
ity,
    225                        copy=copy, sort=sort)
--> 226     return op.get_result()
    227
    228

/usr/lib64/python3.6/site-packages/pandas/core/reshape/conca
t.py in get_result(self)
    421             new_data = concatenate_block_managers(
    422                 mgrs_indexers, self.new_axes, concat_
axis=self.axis,
--> 423                 copy=self.copy)
    424             if not self.copy:
    425                 new_data._consolidate_inplace()

/usr/lib64/python3.6/site-packages/pandas/core/internals.py i
n concatenate_block_managers(mgrs_indexers, axes, concat_axi
s, copy)
   5416         elif is_uniform_join_units(join_units):
   5417             b = join_units[0].block.concat_same_type(
-> 5418                 [ju.block for ju in join_units], plac
ement=placement)
   5419         else:
   5420             b = make_block(

/usr/lib64/python3.6/site-packages/pandas/core/internals.py i
n concat_same_type(self, to_concat, placement)
    366         """
    367         values = self._concatenator([blk.values for b
lk in to_concat],
--> 368                                     axis=self.ndim -
 1)
    369         return self.make_block_same_class(
    370             values, placement=placement or slice(0, l
en(values), 1))

KeyboardInterrupt:
```

In [47]:

```
## INIZIO SEZZIONE TEST

grouped_orbit = data.groupby('ORBIT_CNT')
triggered_orbit = grouped_orbit.filter(lambda x: x['TDC_CHANN
e orbit che hanno il trigger 139
## Ci mette molto anche usando filter (meno di 5 minuti però,
credo...)
triggered_orbit.head()
## Le orbit sembrano già essere in ordine crescente
```

Out[47]:

	HEAD	FPGA	TDC_CHANNEL	ORBIT_CNT	BX_COUNTER	TDC_MEAS	T
5	1	0	24	1897414934	2014	13	5
6	1	0	26	1897414934	2014	20	5
7	1	0	26	1897414934	2026	13	5
8	1	0	139	1897414934	2028	0	5
9	1	0	33	1897414934	2026	14	5

In [49]:

```
## Lets costruiamo questo fucking dataframe
events = pd.DataFrame(columns=['ORBIT','CHAMBER','LAYER','CEL
L'])

## Eliminiamo le hit corrispondenti ai trigger
triggered_orbit = triggered_orbit[triggered_orbit['TDC_CHANNE
L']<137]
gr_event = triggered_orbit.groupby('ORBIT_CNT')

## Ocio che ci mette molto tempo
c = 0
for orb, gr in gr_event:
    events.loc[c] = [orb, np.array(gr['CHAMBER']), np.array(g
r['LAYER']), np.array(gr['CELL'])]
    c+=1
events.head()
## FUCK YEAH!
```

Out[49]:

	ORBIT	CHAMBER	LAYER	CELL
0	1897414934	[1, 1, 1, 1, 1, 1]	[1, 2, 2, 4, 3, 4]	[3, 4, 4, 5, 4, 4]
1	1897415301	[2, 2, 2, 2]	[2, 3, 1, 4]	[11, 11, 11, 12]
2	1897415425	[3, 3, 3, 3]	[3, 1, 2, 4]	[4, 4, 5, 5]
3	1897415544	[4, 4, 4, 4]	[1, 3, 4, 2]	[12, 12, 12, 12]
4	1897415674	[3, 3, 3, 3, 3]	[1, 4, 4, 2, 3]	[5, 6, 3, 5, 5]

In [ ]:

```
## Adesso voglio provare a visualizzare i risultati: ogni cha
mber sarà una matrice 32x132
## --> 4 righe di altezza, 16 di larghezza
## Comincio da una matrice nulla, sommo 1 alla cella in cui h
o l'hit
## Devo ricordarmi che ho gli strati sfalsati, con riferiment
o all'immagine
## --> Conviene riempire la matrice normalmente lasciando lib
era l'ultima "mezza cella", e poi shiftare i risultati
mat_ch1 = mat_ch2 = mat_ch3 = mat_ch4 = np.zeros((32,132))
```

```
events = pd.DataFrame(columns=['ORBIT','CHAMBER','LAYER','CEL
L','POSITION'])

# Sort data according their orbit and their cell
data_sorted = data.sort_values(by = ['ORBIT_CNT','TDC_CHANNE
L'])


# VERY SLOW
pattern = [1,2,3,4]
data_sorted.rolling(len(pattern)).apply(lambda x: all(np.equa
l(x, pattern)))
matched = matched.sum(axis = 1).astype(bool)
print(matched)
```

Out[10]:

	HEAD	FPGA	CELL	ORBIT_CNT	BX_COUNTER	TDC_MEAS	TIME_NS
0	1	1	116	1897414884	1533	24	38345.000
1	1	1	71	1897414887	1650	21	41267.500
2	1	1	67	1897414914	980	8	24506.666
4	1	0	57	1897414922	2162	22	54068.333
3	1	1	70	1897414922	1287	8	32181.666
5	1	0	24	1897414934	2014	13	50360.833
6	1	0	26	1897414934	2014	20	50366.666
7	1	0	26	1897414934	2026	13	50660.833
10	1	0	27	1897414934	2024	11	50609.166
11	1	0	29	1897414934	2026	9	50657.500
9	1	0	33	1897414934	2026	14	50661.666
8	1	0	139	1897414934	2028	0	50700.000
12	1	0	98	1897414940	973	4	24328.333
13	1	0	111	1897414952	2765	28	69148.333
14	1	0	119	1897414956	1736	20	43416.666
15	1	0	69	1897414964	559	13	13985.833
16	1	0	128	1897414976	2010	29	50274.166
17	1	0	128	1897414976	2020	13	50510.833
18	1	0	116	1897414996	3493	24	87345.000
19	1	0	116	1897414996	3503	10	87583.333

In [8]:

```
# Now I can group hits according to the orbit
grouped_orbit = data.groupby(['ORBIT_CNT'])

index = 0
# At this point all the hits are grouped according to their o
rbit,
# so we have to distinguish which of them form an event
```

```
    flag = group['CELL'] == 139;
    # If there are more than 5 hits and there is the trigger
    if group.shape[0] >= 5 and flag.any():
        # Sort group by cell number
        group.sort_values(by = 'CELL')
        index += 1
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-8-33256438bd25> in <module>
      5 # At this point all the hits are grouped according to their orbit,
      6 # so we have to distinguish which of them form an event
----> 7 for key, group in grouped_orbit: # For every group
      8     # Check if there is the trigger inside the hits group
      9     flag = group['CELL'] == 139;

/usr/lib64/python3.6/site-packages/pandas/core/groupby/groupby.py in get_iterator(self, data, axis)
   2226         splitter = self._get_splitter(data, axis=axis)
   2227         keys = self._get_group_keys()
-> 2228         for key, (i, group) in zip(keys, splitter):
   2229             yield key, group
   2230 

/usr/lib64/python3.6/site-packages/pandas/core/groupby/groupby.py in __iter__(self)
   5053             #     raise AssertionError('Start %s must be less than end %s'
   5054             #                          % (str(start), str(end)))
-> 5055             yield i, self._chop(sdata, slice(start, end))
   5056 
   5057     def _get_sorted_data(self):

/usr/lib64/python3.6/site-packages/pandas/core/groupby/groupby.py in _chop(self, sdata, slice_obj)
   5092     def _chop(self, sdata, slice_obj):
   5093         if self.axis == 0:
-> 5094             return sdata.iloc[slice_obj]
   5095         else:
   5096             return sdata._slice(slice_obj, axis=1)  # .loc[:, slice_obj]

/usr/lib64/python3.6/site-packages/pandas/core/indexing.py in __getitem__(self, key)
   1476 
   1477             maybe_callable = com._apply_if_callable(key, self.obj)
-> 1478             return self._getitem_axis(maybe_callable, axis=axis)
   1479 
   1480     def _is_scalar_access(self, key):
```

```
   2078
   2079         if isinstance(key, slice):
-> 2080             return self._get_slice_axis(key, axis=axi
s)
   2081 
   2082         if isinstance(key, list):

/usr/lib64/python3.6/site-packages/pandas/core/indexing.py in
_get_slice_axis(self, slice_obj, axis)
   2048         slice_obj = self._convert_slice_indexer(slice
_obj, axis)
   2049         if isinstance(slice_obj, slice):
-> 2050             return self._slice(slice_obj, axis=axis,
 kind='iloc')
   2051         else:
   2052             return self.obj._take(slice_obj, axis=axi
s)

/usr/lib64/python3.6/site-packages/pandas/core/indexing.py in
_slice(self, obj, axis, kind)
    148         if axis is None:
    149             axis = self.axis
--> 150         return self.obj._slice(obj, axis=axis, kind=k
ind)
    151 
    152     def _get_setitem_indexer(self, key):

/usr/lib64/python3.6/site-packages/pandas/core/generic.py in
_slice(self, slobj, axis, kind)
   2588         """
   2589         axis = self._get_block_manager_axis(axis)
-> 2590         result = self._constructor(self._data.get_sli
ce(slobj, axis=axis))
   2591         result = result.__finalize__(self)
   2592 

/usr/lib64/python3.6/site-packages/pandas/core/internals.py i
n get_slice(self, slobj, axis)
   3882         new_axes[axis] = new_axes[axis][slobj]
   3883 
-> 3884         bm = self.__class__(new_blocks, new_axes, do_
integrity_check=False)
   3885         bm._consolidate_inplace()
   3886         return bm

/usr/lib64/python3.6/site-packages/pandas/core/internals.py i
n __init__(self, blocks, axes, do_integrity_check)
   3284         self._consolidate_check()
   3285 
-> 3286         self._rebuild_blknos_and_blklocs()
   3287 
   3288     def make_empty(self, axes=None):

/usr/lib64/python3.6/site-packages/pandas/core/internals.py i
n _rebuild_blknos_and_blklocs(self)
   3375             new_blklocs[rl.indexer] = np.arange(len(r
l))
   3376 
-> 3377         if (new_blknos == -1).any():
```

```
/usr/lib64/python3.6/site-packages/numpy/core/_methods.py in
_any(a, axis, dtype, out, keepdims)
     41
     42 def _any(a, axis=None, dtype=None, out=None, keepdims
=False):
---> 43     return umr_any(a, axis, dtype, out, keepdims)
     44
     45 def _all(a, axis=None, dtype=None, out=None, keepdims
=False):

KeyboardInterrupt:
```