EUROPYTHON 2016 Bilbao, 17-24 July

Data Formats

for Data Science

Valerio Maggio

Data Scientist and Researcher

Fondazione Bruno Kessler (FBK)

Trento, Italy

@leriomaggio

PyData

About me

- Post Doc Researcher @ FBK
  - Complex Data Analytics Unit (MPBA)
- Interested in *Machine Learning, Text and Data Processing*
  - with "Deep" *divergences* recently
- Fellow Pythonista since 2006
  - scientific Python ecosystem
- **PyData Italy** Chair
  - http://pydata.it

@pydatait

worthwhile mentioning...

EuroSciPy 2016

Erlangen 23 - 27 August

The Program is online: https://www.euroscipy.org/2016/program/

Fees

Registration will open beggining of may.

Tutorials	Academic / Individual	Industry
Early Bird	50 Euro	125 Euro
Regular	100 Euro	250 Euro

Main Conference	Academic / Individual	Industry
Early Bird	50 Euro	125 Euro
Regular	100 Euro	250 Euro

End of early-bird:
Jul 21, 2106
(that's today! 😱)

Data Formats 4 Data Science

- Data **Processing**
  - Q: What's the better way to process data
  - Q⁺: What's the most Pythonic Way to do that? 
- Data **Sharing**
  - Q: What's the best way to share (and to present data)
    - **A:** [Interactive] Charts - Data Visualisation
- **OMG, Bokeh is better than ever!** by *Fabio Pliger* (after this session!)

Jupyter Notebook for Data and Documentation Sharing

1.

Textual Data format

./matrix.txt

```
 1 9.15794193744659423 8e-01 7.71644234657287597 7e-01 4.43449765443801879 9e-01 3.62776070833206176 8e-01 2
 2 9.34689581394195556 6e-01 5.74239075183868408 2e-01 3.73131364583969116 2e-01 3.19647550582885742 2e-01 3
 3 9.49424088001251220 7e-01 8.84977698326110839 8e-01 2.34618917107582092 3e-01 2.39458844065666198 7e-01 9
 4 9.33350801467895507 8e-01 8.45798790454864502 0e-01 4.31081146001815795 9e-01 1.74765363335609436 0e-01 1
 5 9.60948228836059570 3e-01 3.33171516656875610 4e-01 3.58363032341003418 0e-01 2.59227842092514038 1e-01 4
 6 9.79261279106140136 7e-01 9.00877296924591064 5e-01 2.74642467498779296 9e-01 2.82823890447616577 1e-01 1
 7 9.11297857761383056 6e-01 8.60041379928588867 2e-01 3.73763054609298706 1e-01 2.03612133860588073 7e-01 1
 8 9.57156002521514892 6e-01 8.60671520233154296 9e-01 2.63099193572998046 9e-01 2.16055095195770263 7e-01 9
 9 9.32383358478546142 6e-01 8.17140281200408935 5e-01 4.37727779150009155 3e-01 1.50275990366935730 0e-01 1
10 9.35607969760894775 4e-01 7.85106837749481201 2e-01 5.01240551471710205 1e-01 1.45507648587226867 7e-01 1
11 9.09201145172119140 6e-01 7.48335361480712890 6e-01 4.29838418960571289 1e-01 2.54180282354354858 4e-01 1
12 9.50328767299652099 6e-01 8.87313425540924072 3e-01 2.65516847372055053 7e-01 2.21121847629547119 1e-01 1
13 9.23728406429290771 5e-01 8.36317658424377441 4e-01 3.62710148096084594 7e-01 2.36596763134002685 5e-01 1
14 9.56217229366302490 2e-01 9.19413685798645019 5e-01 3.81959676742553710 9e-01 3.11711623287200928 1e-01 1
15 9.46118593215942382 8e-01 8.48429560661315918 0e-01 3.90345662832260131 8e-01 1.66836887598037719 7e-01 1
16 9.46766436100006103 5e-01 8.68262052536010742 2e-01 3.13781559467315673 8e-01 1.82636991143226623 5e-01 9
17 9.39719974994659423 8e-01 8.60964000225067138 7e-01 3.49940776824951171 9e-01 1.61880478262901306 2e-01 1
18 9.22237932682037353 5e-01 8.87685042915344238 4e-01 3.55698913335800170 9e-01 3.47954481840133667 0e-01 8
19 9.41853940486907959 0e-01 8.91886651515960693 4e-01 2.33752176165580749 5e-01 2.46092557907104492 2e-01 9
20 8.90693068504333496 1e-01 8.14490437507629394 5e-01 4.38080459833145141 6e-01 5.20068526268005371 1e-01 1
21 8.54925513267517089 8e-01 7.77565240859985351 6e-01 2.99812227487564086 9e-01 4.50702667236328125 0e-01 1
22 9.36491727828979492 2e-01 8.83662104606628418 0e-01 4.24375027418136596 7e-01 2.40321293473243713 4e-01 9
23 9.40816819667816162 1e-01 4.73922908306121826 2e-01 3.61783891916275024 4e-01 2.82977879047393798 8e-01 4
24 9.31876599788665771 5e-01 7.78179287910461425 8e-01 4.77103233337402343 8e-01 1.84343487024307251 0e-01 8
25 9.61190819740295410 2e-01 7.10161328315734863 3e-01 4.38451111316680908 2e-01 2.05519944429397583 0e-01 1
26 9.41845655441284179 7e-01 7.01128482818603515 6e-01 4.34117734432220459 0e-01 3.87892842292785644 5e-01 4
27 9.14494633674621582 0e-01 3.43847274780273437 5e-01 4.71976578235626220 7e-01 2.63393461704254150 4e-01 4
28 9.46340918540954589 8e-01 3.46242964267730712 9e-01 3.76388847827911377 0e-01 2.53234148025512695 3e-01 4
29 9.57078337669372558 6e-01 4.38870877027511596 7e-01 3.54516685009002685 5e-01 2.83670097589492797 9e-01 4
30 9.44252729415893554 7e-01 4.11161631345748901 4e-01 3.77380192279815673 8e-01 3.20239514112472534 2e-01 5
31 9.47266575488250732 4e-01 2.99052000045776367 2e-01 3.84141653776168823 2e-01 2.76066184043884277 3e-01 4
32 9.21456336975097656 2e-01 4.41649496555328369 1e-01 3.55940699577331543 0e-01 4.30402725934982299 8e-01 4
33 9.51333439982696533 2e-01 2.56332576274871826 2e-01 3.88306945562362670 9e-01 2.68047869205474853 5e-01 4
```

```
f = open('files/textual/matrix.txt')
matrix = []
for line in f.readlines():
    row = [float(x) for x in line.split()]
    matrix.append(row)
f.close()
```

More Pythonic 

```
with open('files/textual/matrix.txt') as f:
    matrix = []
    for line in f.readlines():
        row = [float(x) for x in line.split()]
        matrix.append(row)
```

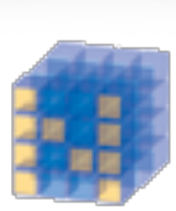
```
# shapes
print('Rows: {} - Cols: {}'.format(len(matrix), len(matrix[0])))
```

```
Rows: 104 - Cols: 768
```

```
with open('files/textual/matrix.txt') as f:
    matrix = []
    for line in f.readlines():
        row = [float(x) for x in line.split()]
        matrix.append(row)
```

```
# shapes
print('Rows: {} - Cols: {}'.format(len(matrix), len(matrix[0])))
```

```
Rows: 104 - Cols: 768
```



Numpy to the rescue

```
import numpy as np
matrix = np.loadtxt('files/textual/matrix.txt')
```

```
# shapes
print('Rows: {} - Cols: {}'.format(*matrix.shape))
```

```
Rows: 104 - Cols: 768
```

```
In [22]: np.loadtxt?
```

```
Signature: np.loadtxt(fname, dtype=<class 'float'>, comments='#', delimiter=None, convert
ers=None, skiprows=0, usecols=None, unpack=False, ndmin=0)
Docstring:
Load data from a text file.

Each row in the text file must have the same number of values.

Parameters
----------
fname : file or str
```

```
In [23]: np.genfromtxt?
```

```
Signature: np.genfromtxt(fname, dtype=<class 'float'>, comments='#', delimiter=None, skip
_header=0, skip_footer=0, converters=None, missing_values=None, filling_values=None, usec
ols=None, names=None, excludelist=None, deletechars=None, replace_space='_', autostrip=Fa
lse, case_sensitive=True, defaultfmt='f%i', unpack=None, usemask=False, loose=True, inval
id_raise=True, max_rows=None)
Docstring:
Load data from a text file, with missing values handled as specified.

Each line past the first `skip_header` lines is split at the `delimiter`
character, and characters following the `comments` character are discarded.
```

csv files

```
!head files/textual/metadata.csv
```

```
FILENAME,DATASET,CLASS,CAMERA,CONF,VARIETY,SOSQ,SOMQ,CAT,FILEPATH
so1_L_e_b_001.jpg,so1,E,NA,B,Lagorai,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/so1/early/so1_L_e_b_001.jpg
so1_L_e_b_002.jpg,so1,E,NA,B,Lagorai,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/so1/early/so1_L_e_b_002.jpg
so1_V_e_b_001.jpg,so1,E,NA,B,Vajolet,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/so1/early/so1_V_e_b_001.jpg
so1_V_e_b_002.jpg,so1,E,NA,B,Vajolet,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/so1/early/so1_V_e_b_002.jpg
so1_V_e_b_003.jpg,so1,E,NA,B,Vajolet,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/so1/early/so1_V_e_b_003.jpg
so1_V_e_b_004.jpg,so1,E,NA,B,Vajolet,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/so1/early/so1_V_e_b_004.jpg
so1_V_e_b_005.jpg,so1,E,NA,B,Vajolet,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/so1/early/so1_V_e_b_005.jpg
so1_L_g_b_001.jpg,so1,G,NA,B,Lagorai,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/so1/good/so1_L_g_b_001.jpg
so1_L_g_b_002.jpg,so1,G,NA,B,Lagorai,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/so1/good/so1_L_g_b_002.jpg
```

```
!head files/textual/metadata.csv
```

```
FILENAME,DATASET,CLASS,CAMERA,CONF,VARIETY,SOSQ,SOMQ,CAT,FILEPATH
so1_L_e_b_001.jpg,so1,E,NA,B,Lagorai,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/so1/early/so1_L_e_b_001.jpg
so1_L_e_b_002.jpg,so1,E,NA,B,Lagorai,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/so1/early/so1_L_e_b_002.jpg
so1_V_e_b_001.jpg,so1,E,NA,B,Vajolet,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/so1/early/so1_V_e_b_001.jpg
so1_V_e_b_002.jpg,so1,E,NA,B,Vajolet,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/so1/early/so1_V_e_b_002.jpg
so1_V_e_b_003.jpg,so1,E,NA,B,Vajolet,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/so1/early/so1_V_e_b_003.jpg
so1_V_e_b_004.jpg,so1,E,NA,B,Vajolet,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/so1/early/so1_V_e_b_004.jpg
so1_V_e_b_005.jpg,so1,E,NA,B,Vajolet,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/so1/early/so1_V_e_b_005.jpg
so1_L_g_b_001.jpg,so1,G,NA,B,Lagorai,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/so1/good/so1_L_g_b_001.jpg
so1_L_g_b_002.jpg,so1,G,NA,B,Lagorai,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/so1/good/so1_L_g_b_002.jpg
```

`csv` Module (in standard library)

```
import csv
```

```
import csv
with open('files/textual/metadata.csv', newline='') as csvfile:
    metadata_reader = csv.reader(csvfile, delimiter=',')
    for row in metadata_reader:
        # store properly
```

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$

```
import pandas as pd

metadata = pd.read_csv('files/textual/metadata.csv')
```

```
metadata.head(8)
```

	FILENAME	DATASET	CLASS	CAMERA	CONF	VARIETY	SOSQ	SOMQ	CAT	FI
0	so1_L_e_b_001.jpg	so1	E	NaN	B	Lagorai	NaN	NaN	NaN	/h
1	so1_L_e_b_002.jpg	so1	E	NaN	B	Lagorai	NaN	NaN	NaN	/h
2	so1_V_e_b_001.jpg	so1	E	NaN	B	Vajolet	NaN	NaN	NaN	/h
3	so1_V_e_b_002.jpg	so1	E	NaN	B	Vajolet	NaN	NaN	NaN	/h
4	so1_V_e_b_003.jpg	so1	E	NaN	B	Vajolet	NaN	NaN	NaN	/h
5	so1_V_e_b_004.jpg	so1	E	NaN	B	Vajolet	NaN	NaN	NaN	/h
6	so1_V_e_b_005.jpg	so1	E	NaN	B	Vajolet	NaN	NaN	NaN	/h
7	so1_L_g_b_001.jpg	so1	G	NaN	B	Lagorai	NaN	NaN	NaN	/h

```
In [29]: pd.read_csv?
```

```
Signature: pd.read_csv(filepath_or_buffer, sep=',', delimiter=None, header='infer',
names=None, index_col=None, usecols=None, squeeze=False, prefix=None, mangle_dupe_cols=Tr
ue, dtype=None, engine=None, converters=None, true_values=None, false_values=None, skipin
itialspace=False, skiprows=None, skipfooter=None, nrows=None, na_values=None, keep_defaul
t_na=True, na_filter=True, verbose=False, skip_blank_lines=True, parse_dates=False, infer
_datetime_format=False, keep_date_col=False, date_parser=None, dayfirst=False, iterator=F
alse, chunksize=None, compression='infer', thousands=None, decimal=b'.', lineterminator=N
one, quotechar='"', quoting=0, escapechar=None, comment=None, encoding=None,
dialect=None, tupleize_cols=False, error_bad_lines=True, warn_bad_lines=True,
skip_footer=0, doublequote=True, delim_whitespace=False, as_recarray=False,
compact_ints=False, use_unsigned=False, low_memory=True, buffer_lines=None, memory_map=Fa
lse, float_precision=None)
Docstring:
Read CSV (comma-separated) file into DataFrame
```

```
pd.read_clipboard
pd.read_csv
pd.read_excel
pd.read_fwf
pd.read_gbq
pd.read_hdf
pd.read_html
pd.read_json
pd.read_msgpack
pd.read_pickle
pd.read_
```

name	HARD CHEESE
user	acquisti@fbk.eu
description	
num_records	145
num_samples	48
num_auto_attributes	8
num_custom_attributes	5
num_wavelengths	331
wavelengths_start	
wavelengths_resolution	

```
!head -n 10 files/textual/collection.csv
```

```
name, HARD CHEESE
user, acquisti@fbk.eu
description,
num_records, 145
num_samples, 48
num_auto_attributes, 8
num_custom_attributes, 5
num_wavelengths, 331
wavelengths_start, 740.0
wavelengths_resolution, 1.0
```

id								device_id	c
int								unicode	N
1								E036D39ADE70A12D	N
2								E036D39ADE70A12D	N
3								E036D39ADE70A12D	N
4								E036D39ADE70A12D	N
5								E036D39ADE70A12D	N
6								E036D39ADE70A12D	N
7								E036D39ADE70A12D	N
8								E036D39ADE70A12D	N
9								E036D39ADE70A12D	N
10								E036D39ADE70A12D	N
11								E036D39ADE70A12D	N
12	9571b870-4b7f-48b0-8aeb-7eda91adfb08	2015-09-07 11:30:20.012000	Goat	A	23	30	2015-09-26 09:00:00	E036D39ADE70A12D	N
13	9571b870-4b7f-48b0-8aeb-7eda91adfb08	2015-09-07 11:30:27.086000	Goat	A	23	30	2015-09-26 09:00:00	E036D39ADE70A12D	N
14	6b057fdb-5ba8-4bf6-afce-63b9f28a9a81	2015-09-07 11:31:52.264000	Cow	A	23	28	2015-09-18 09:00:00	E036D39ADE70A12D	N
15	6b057fdb-5ba8-4bf6-afce-63b9f28a9a81	2015-09-07 11:32:00.425000	Cow	A	23	28	2015-09-18 09:00:00	E036D39ADE70A12D	N
16	6b057fdb-5ba8-4bf6-afce-63b9f28a9a81	2015-09-07 11:32:07.996000	Cow	A	23	28	2015-09-18 09:00:00	E036D39ADE70A12D	N
17	f7fbd198-683e-4ead-8008-b6daacec5eca	2015-09-07 11:33:59.157000	Cow	A	23	28	2015-09-17 09:00:00	E036D39ADE70A12D	N
18	f7fbd198-683e-4ead-8008-b6daacec5eca	2015-09-07 11:34:06.762000	Cow	A	23	28	2015-09-17 09:00:00	E036D39ADE70A12D	N
19	f7fbd198-683e-4ead-8008-b6daacec5eca	2015-09-07 11:34:14.473000	Cow	A	23	28	2015-09-17 09:00:00	E036D39ADE70A12D	N

```
collection = pd.read_csv('files/textual/collection.csv', skiprows=10)
```

```
collection.head()
```

	id	sample_id	sampling_time	Milk Type	Brand	Protein	Fat	Expiration Date	device_id
0	int	unicode	str	unicode	unicode	int	int	unicode	unicode
1	1	fc2dd6d8-11f5-45d3-bf9a-075af1900b72	2015-09-07 11:17:46.514000	Cow	A	30	15	2015-10-04 09:00:00	E036D39ADI
2	2	fc2dd6d8-11f5-45d3-bf9a-075af1900b72	2015-09-07 11:17:58.402000	Cow	A	30	15	2015-10-04 09:00:00	E036D39ADI
3	3	fc2dd6d8-11f5-45d3-bf9a-	2015-09-07 11:18:07.135000	Cow	A	30	15	2015-10-04	E036D39ADI

Textual Data format

- *Be Pythonic*: use context managers (`with`)
- numpy (mostly numerical) and pandas (csv) *to the rescue*
  - `np.loadtxt` and `pd.read_csv`
- (+) Very easy to (re)create and share
  - very easy to process
- (-) Not storage friendly but **highly compressible**!
- (-) No structured information

2.

Binary Data format

Binary format

Integers and floats in *native* and *string* representations

```
# small ints            # medium ints
42    (4 bytes)         123456    (4 bytes)
'42'  (2 bytes)         '123456'  (6 bytes)

# near-int floats       # e-notation floats
12.34    (8 bytes)      42.424242E+42    (8 bytes)
'12.34'  (5 bytes)      '42.424242E+42'  (13 bytes)
```

*

- Space is not the *only* concern (for text). Speed matters!
- Python conversion to `int()` and `float()` are slow
  - costly `atoi()`/`atof()` C functions

* A. Scopatz, K.D. Huff - Effective Computations in Physics - Field Guide to Research in Python, O'Reilly 2015

import pickle

```
import numpy as np
import pickle
```

```
array = np.arange(10000).reshape(10, 1000)
```

```
with open('bin_array.bin', 'wb') as f:
    f.write(pickle.dumps(array))
```

```
print(type(array), array.dtype, array.shape)
```

```
<class 'numpy.ndarray'> int64 (10, 1000)
```

```
a_pickled = pickle.load(open('bin_array.bin', 'rb'))
```

```
print(type(a_pickled), a_pickled.dtype, a_pickled.shape)
```

```
<class 'numpy.ndarray'> int64 (10, 1000)
```

Still, it is often desirable to have something more than a binary chunk of data in a file.

Hierarchical Data Format 5 (a.k.a. **hdf5**)

- Free and open source file format specification
  - HDFGroup - Univ. Illinois Champagne-Urbana
- (+) Works great with both big or tiny datasets
- (+) Storage friendly
  - Allows for Compression
- (+) Dev. Friendly
  - Query DSL + Multiple-language support
  - **Python:** PyTables, hdf5, h5py

```
import h5py
import numpy as np
```

```
f = h5py.File("mytestfile.hdf5", "w")
```

```
dset = f.create_dataset("mydataset", (100,), dtype='i')
```

```
dset.shape
```

```
(100,)
```

```
dset.dtype
```

```
dtype('int3
```

```
type(dset)
```

```
h5py._hl.da
```

```
# Bulk insert
dset[...] = np.arange(100)
```

```
dset[10]
```

```
10
```

```
dset[:100:10]
```

```
array([ 0, 10, 20, 30, 40, 50, 60, 70, 80, 90], dtype=int32)
```

Numpy Arrays tight integration

with PyTables

```
Array
    The files of the filesystem

CArray
    Chunked arrays

EArray
    Extendable arrays

VLArray
    Variable-length arrays

Table
    Structured arrays
```

```
import tables as tb
```

```
f = tb.open_file('mytestfile.hdf5', 'a')
```

```
# tables need descriptions
dt = np.dtype([('id', int), ('name', 'S10')])
knights = np.array([(42, 'Lancelot'), (12, 'Bedivere')], dtype=dt)
f.create_table('/', 'knights', dt)
f.root.knights.append(knights)
```

Accessing the table

Hierarchy and Groups

```
dset.name
```

```
'/mydataset'
```

```
f.name
```

```
'/'
```

```
grp = f.create_group("second_level")
```

```
dset2 =
```

```
dset2.n
```

```
'/secon
```

```
dset3 = f.create_dataset('second_level_2/dset3', (10,), dtype='i')
```

```
dset3.name
```

```
'/second_level_2/dset3'
```

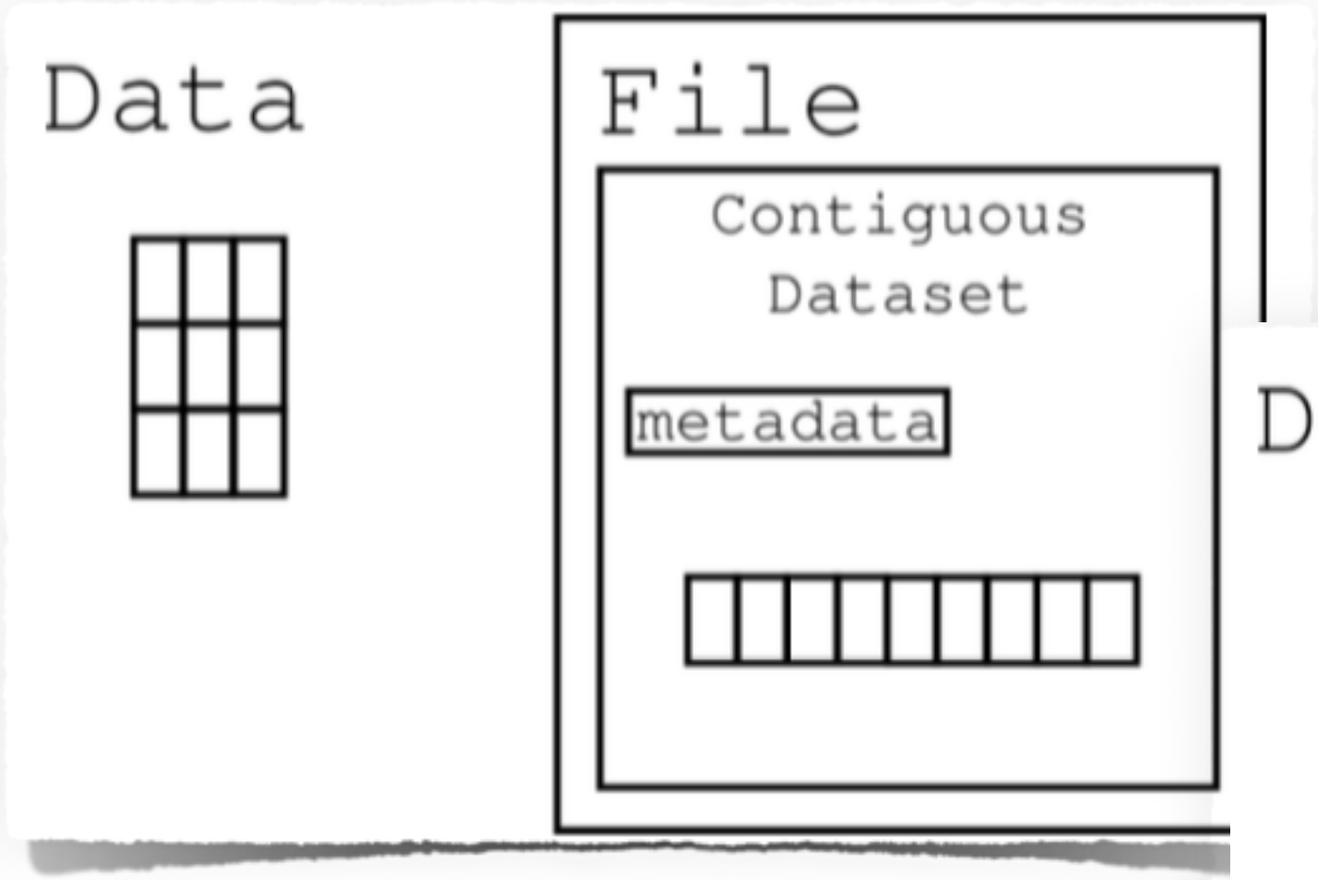
```
dset3_f = f['second_level_2/dset3']
```

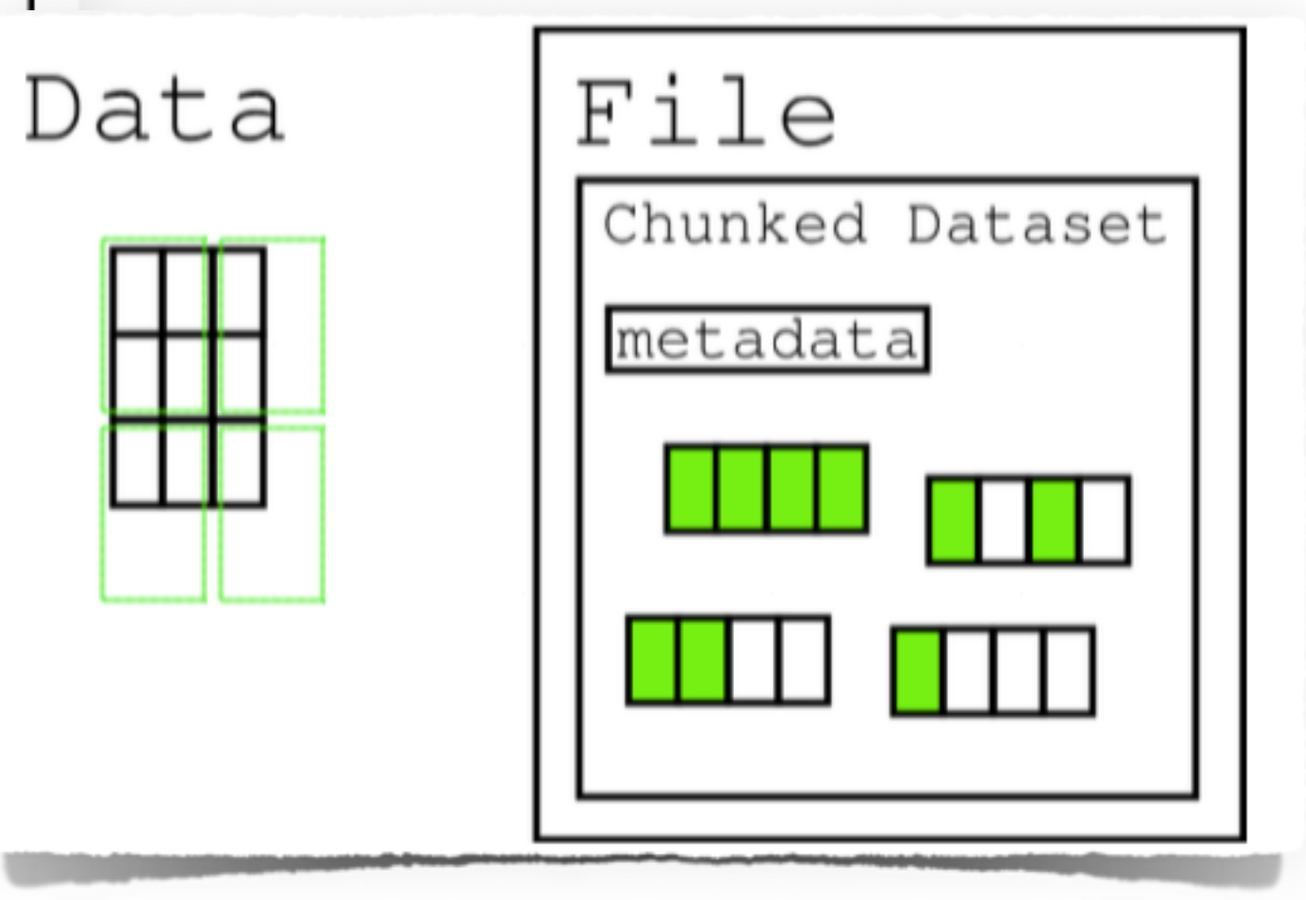
```
dset3 == dset3_f
```

```
True
```

Data Chunking

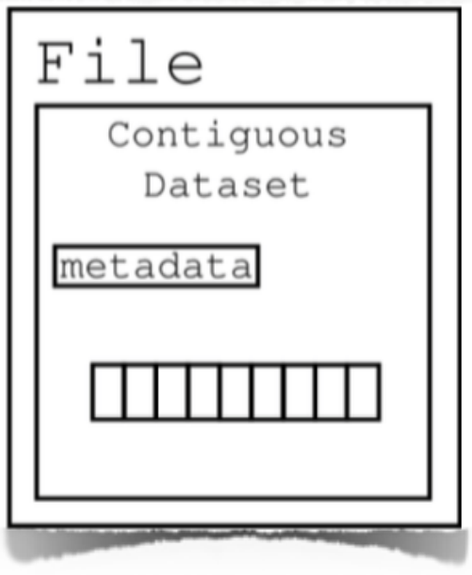
```
dset = f.create_dataset("chunked", (1000, 1000), chunks=(100, 100))
```

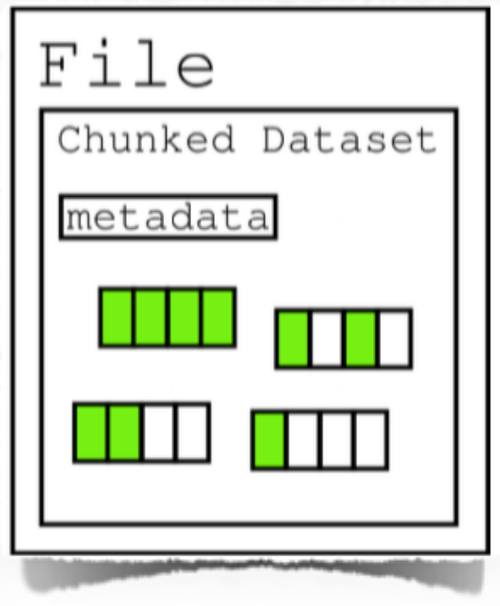




* A. Scopatz, K.D. Huff - Effective Computations in Physics - Field Guide to Research in Python, O'Reilly 2015

Data Chunking





- Small chunks are good for accessing only some of the data at a time.
- Large chunks are good for accessing lots of data at a time.
- Reading and writing chunks may happen in **parallel**

* A. Scopatz, K.D. Huff - Effective Computations in Physics - Field Guide to Research in Python, O'Reilly 2015

Parallel HDF5

MPI (mpi4py) integration

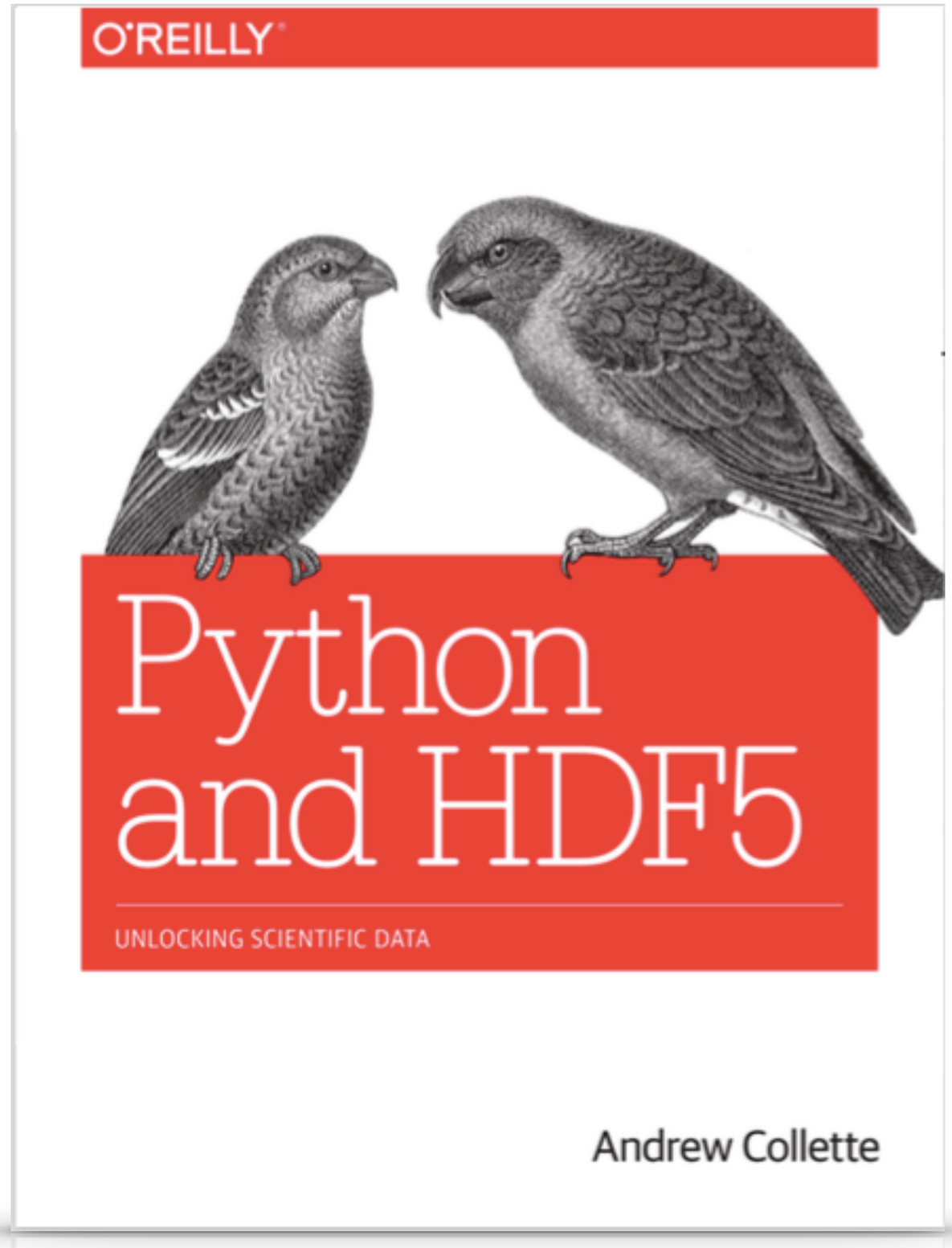
```
from mpi4py import MPI
import h5py

rank = MPI.COMM_WORLD.rank  # The process ID (integer 0-3 for 4-process)

f = h5py.File('parallel_test.hdf5', 'w', driver='mpio',
              comm=MPI.COMM_WORLD)

dset = f.create_dataset('test', (4, 1000), dtype='i')
dset[rank] = np.arange(1000)*rank

f.close()
```



Learn More

- **How to migrate from PostgreSQL to HDF5 and live happily ever after** by *Michele Simionato* @PyData Track on Friday

Data Format

- Data Analysis Framework (and tool) dev. @CERN
  - written in C++;
    - native extension in Python (aka **PyROOT**)
      - ROOT6 also ships a Jupyter Kernel
- Definition of a new Binary Data Format (`.root`)
  - based on the serialisation of C++ Objects

```
MB-Air:~ valerio$ root -l
root [0] new TBrowser()
(class TBrowser*)0x7fc7be267cb0
root [1] (class TFile*)0x7fc7be7b5400
```

ROOT Object Browser

Browser File Edit View Options Tools Help

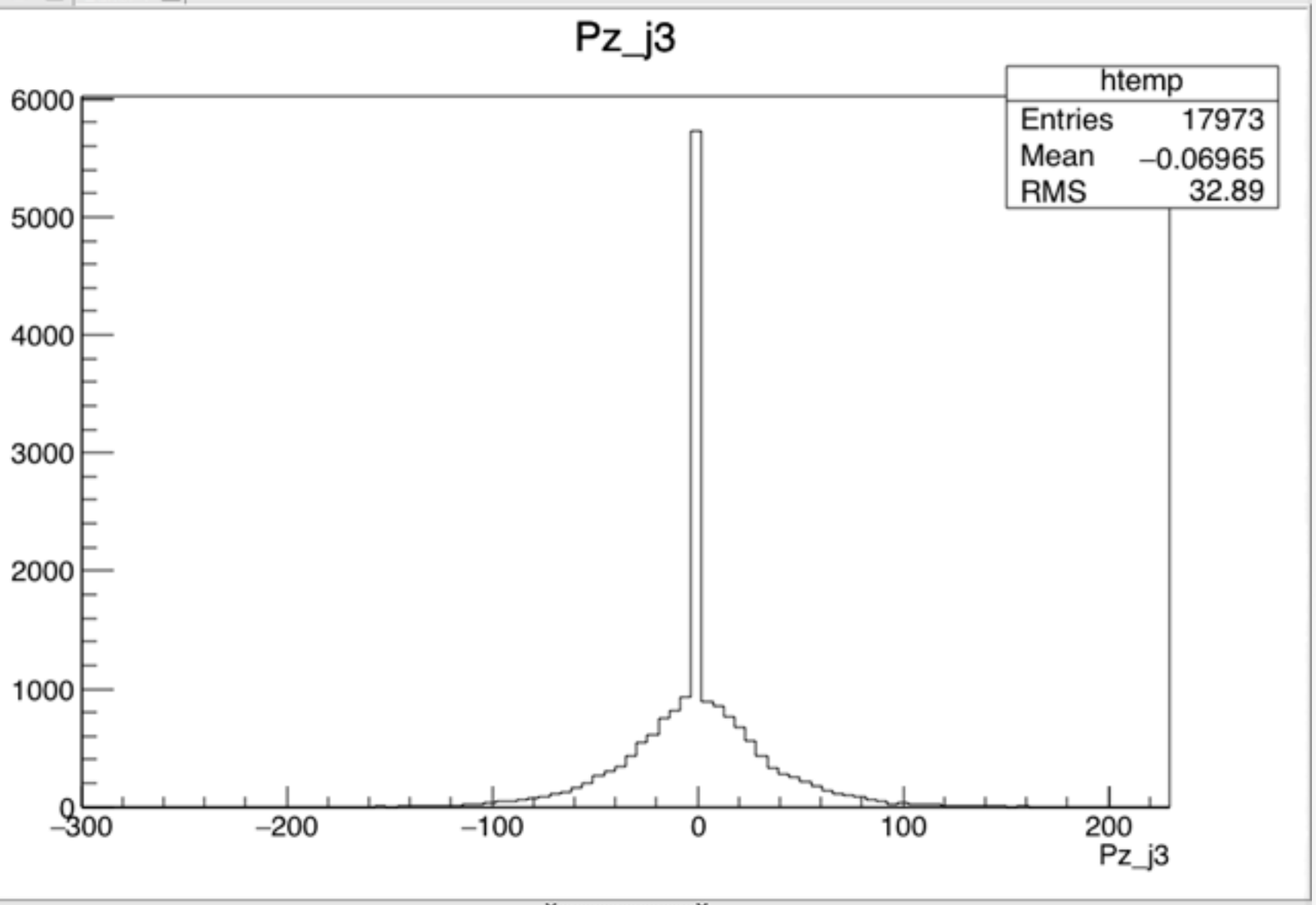
Files

Draw Option:

- Px_j3
- Py_j3
- Pz_j3
- En_j3
- Et_j3
- Jcor_j3
- Emf_j3
- Deta_j3
- Eta_j3
- Phi_j3
- Eta_j3
- Eta_j3
- jpb_j3
- slt_j3
- jetprb3
- Px_j4
- Py_j4
- Pz_j4
- En_j4
- Et_j4
- Jcor_j4
- Emf_j4
- Deta_j4
- Eta_j4
- Phi_j4
- Bnk_j4
- svx_j4
- jpb_j4
- slt_j4
- jetprb4
- ptl_gen
- etal_gen
- phil_gen

Filter: All Files (*.*)

Canvas_1 Editor 1



Command

Command (local):

```
root [0] .x ~/myROOTenv.C
root [1] TFile *tf = new TFile("~/trigger_optimisation/input_file.root")
root [2] TTree *tt = tf->Get("MONTECARLO")
root [3] tt->Draw("HitList_.pm_id_/31:eventNumber_>> h(3853, 0,6306,2070, 0,
2070)","","goff")
               (Long64_t)5931328
root [4] { int arr[3853];
           int count;
           for(int i=0;i<=3853;i++)
           {
               count=0;
               for(int j=0;j<=2070;j++)
                   }
               }
```

C++ style

```
import ROOT
rfile = ROOT.TFile('filepath')
tree = rfile.Get('treename')
hist2d = ROOT.TH2F("name","title",nbinsX,mix,maxX, nbinsY, minY,maxY)
tree.Draw('HitList_.pm_id_/31:eventNumber_ >> hist2d','','goff')
```

rootpy rootpy.github.io/

root_numpy rootpy.github.io/root_numpy/

```
import ROOT
rfile = ROOT.TFile('filepath')
tree = rfile.Get('treename')
hist2d = ROOT.TH2F("name","title",nbinsX,mix,maxX, nbinsY, minY,maxY)
tree.Draw('HitList_.pm_id_/31:eventNumber_ >> hist2d','','goff')
```

```
import rootpy.plotting
from rootpy.io import root_open
root_file = root_open(infile)
rpy_tree = root_file.MONTECARLO
```

```
h2d = rootpy.plotting.Hist2D(type="F")
ret = rpy_tree.Draw('eventNumber_:HitList_.pm_id_/31',
                    hist=h2d, create_hist=False)
```

root_numpy examples

```
import root_numpy as rnp
energies = rnp.root2array(infile, treename="MONTECARLO",
                                  branches = 'neutrino_.E_')
```

Tight integration with PyROOT objects

```
histogram = ROOT.TH1F("en histo","Energies histogram",
                      numberOfEvents, 0, max(energies))

rnp.fill_hist(histogram, energies)

histogram.Draw()
```

root2hdf5 (included in rootpy)

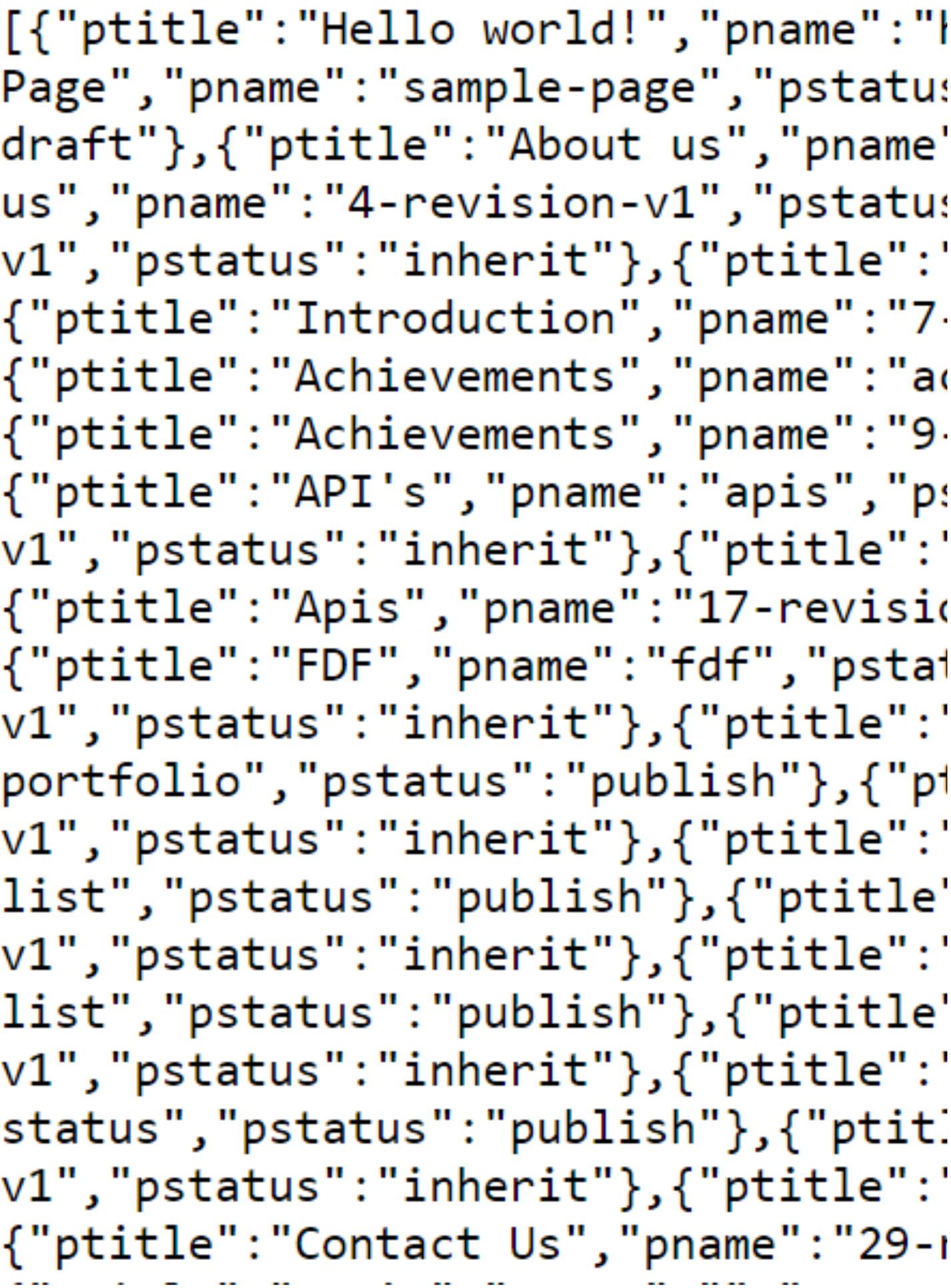
```
$ root2hdf5 -h
[?1034husage: root2hdf5 [-h] [--version] [-n ENTRIES] [-f] [-u] [--ext EXT]
                  [-c {0,1,2,3,4,5,6,7,8,9}] [-l {zlib,lzo,bzip2,blosc}] [-s SELECTION]
                  [--script SCRIPT] [-q] [--no-progress-bar]
                  files [files ...]

Convert ROOT files containing TTrees into HDF5 files containing HDF5 tables

positional arguments:
  files

optional arguments:
  -h, --help            show this help message and exit
  --version             show the version number and exit
  -n ENTRIES, --entries ENTRIES
                        number of entries to read at once (default: 100000)
  -f, --force           overwrite existing output files (default: False)
  -u, --update          update existing output files (default: False)
  --ext EXT             output file extension (default: h5)
  -c {0,1,2,3,4,5,6,7,8,9}, --complevel {0,1,2,3,4,5,6,7,8,9}
                        compression level (default: 5)
  -l {zlib,lzo,bzip2,blosc}, --complib {zlib,lzo,bzip2,blosc}
                        compression algorithm (default: zlib)
  -s SELECTION, --selection SELECTION
                        apply a selection on each tree with a cut expression (default: None)
  --script SCRIPT       Python script containing a function with the same name
                        that will be called on each tree and must return a tree or
                        list of trees that will be converted instead of the
                        original tree (default: None)
  -q, --quiet           suppress all warnings (default: False)
  --no-progress-bar     do not show the progress bar (default: False)
```

http://www.rootpy.org/commands/root2hdf5.html



3.

JSON
Data format


XML!
JSON!

Jupyter Notebook Data Format

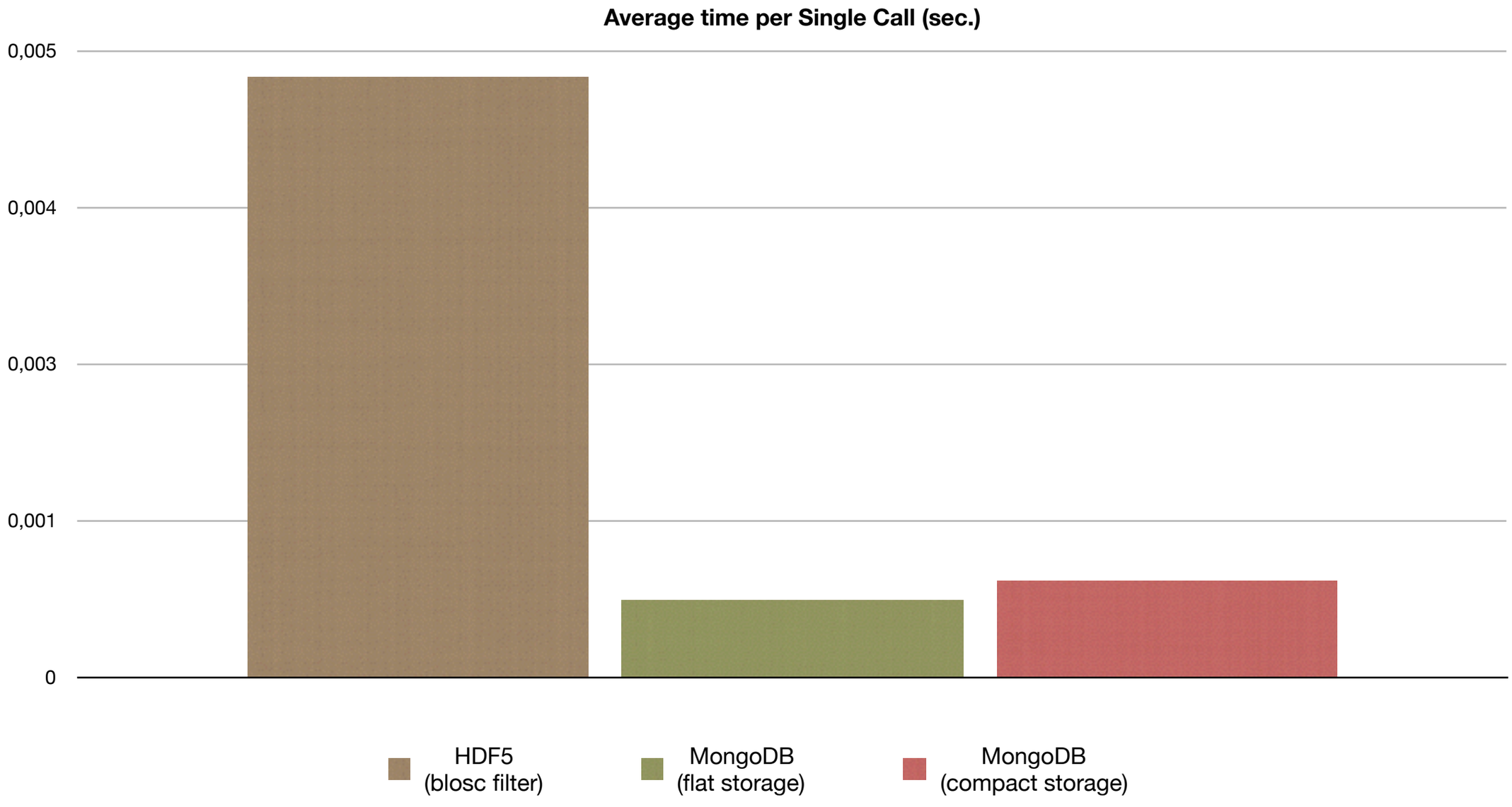
```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Custom Magic Examples"
   ]
  },
  {…},
  {…},
  {…},
  {…},
```

```
  {
   "cell_type": "code",
   "execution_count": 1,
   "metadata": {
    "collapsed": false
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "\"print('This is a line Magic')\""
      ]
     },
     "execution_count": 1,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "%lmagic print('This is a line Magic')"
   ]
  },
```

JSON is the format of choice for Document Oriented DBs (a.k.a. NOSQL DBs)

HDF5 vs MongoDB

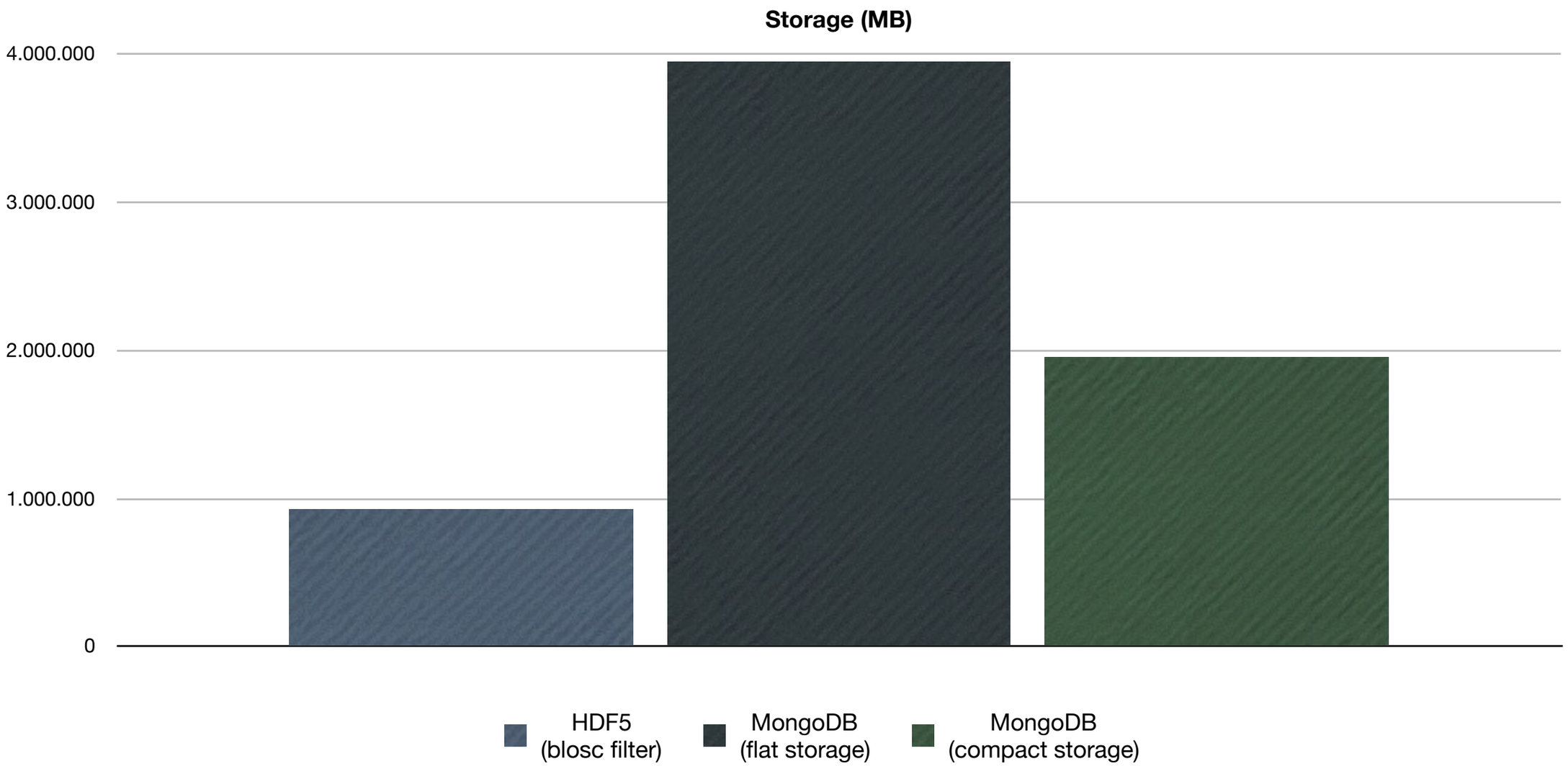
Total Number of Documents	Total Number of Entries	Total Number of Calls
100.000	8.755.882	319.970



HDF5 vs MongoDB

Total Number of Documents	Total Number of Entries	Total Number of Calls
100.000	8.755.882	319.970

Systems	Storage (MB)
HDF5 (blosc filter)	922.528
MongoDB (flat storage)	3.952.148
MongoDB (compact storage)	1.953.125



4.

HDFS
Data format

matthewrocklin.com/blog/work/
2016/02/22/dask-distributed-part-2

HDFS

- **HDFS**: Hadoop Filesystem
  - Distributed Filesystem on top of Hadoop
- Data can be organised in shardes and distributed among several machines (cluster config)
  - (*de facto*) Big Data Data Format
- **Python:** `hdfs3`
  - Native implementation of HDFS in C++
  - No Java along the way!

HDFS + CSV

```
from hdfs3 import HDFileSystem
fs = HDFileSystem()

fs.ls('/user/ubuntu/nyc/', detail=False)
```

```
[u'/user/ubuntu/nyc/yellow_tripdata_2015-01.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-02.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-03.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-04.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-05.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-06.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-07.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-08.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-09.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-10.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-11.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-12.csv']
```

Opening a Single File on the HDFS

```
import pandas as pd

with fs.open('/user/ubuntu/nyc/yellow_tripdata_2015-01.csv') as f:
    df = pd.read_csv(f, nrows=5)
df
```

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance
0	2	2015-01-15 19:05:39	2015-01-15 19:23:42	1	1.59
1	1	2015-01-10 20:33:38	2015-01-10 20:53:28	1	3.30
2	1	2015-01-10 20:33:38	2015-01-10 20:43:41	1	1.80
3	1	2015-01-10 20:33:39	2015-01-10 20:35:31	1	0.50
4	1	2015-01-10 20:33:39	2015-01-10 20:52:58	1	3.00

```
from hdfs3 import HDFileSystem
fs = HDFileSystem()

fs.ls('/user/ubuntu/nyc/', detail=False)
```

```
[u'/user/ubuntu/nyc/yellow_tripdata_2015-01.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-02.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-03.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-04.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-05.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-06.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-07.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-08.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-09.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-10.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-11.csv',
 u'/user/ubuntu/nyc/yellow_tripdata_2015-12.csv']
```

HDFS + CSV

Wildcard opening of CSVs on the HDFS

```
from distributed import Executor, hdfs, progress, wait, s3
e = Executor('cluster.demo.continuum.io:8786')
e
```

```
<Executor: scheduler=cluster.demo.continuum.io:8786 workers=56 threads=56
>
```

```
df = hdfs.read_csv('/user/ubuntu/nyc/yellow_tripdata_2015-*.csv',
                   parse_dates=['tpep_pickup_datetime',
                                'tpep_dropoff_datetime'],
                   header='infer')
df = e.persist(df)
```

```
Setting global dask scheduler to use distributed
```

```
df.columns
```

```
Index([u'VendorID', u'tpep_pickup_datetime', u'tpep_dropoff_datetime',
       u'passenger_count', u'trip_distance', u'pickup_longitude',
       u'pickup_latitude', u'RateCodeID', u'store_and_fwd_flag',
       u'dropoff_longitude', u'dropoff_latitude', u'payment_type',
       u'fare_amount', u'extra', u'mta_tax', u'tip_amount', u'tolls_amoun
t',
       u'improvement_surcharge', u'total_amount'],
      dtype='object')
```

```
df.dtypes
```

```
df2 = df.assign(payment_2=(df.payment_type == 2),
                no_tip=(df.tip_amount == 0))[['no_tip', 'payment_2']]
df2.head()
```

	no_tip	payment_2
0	False	False
1	False	False
2	True	True
3	True	True
4	True	True

Big Data and Columnar DBs

- Big Data World is shifting towards columnar DBs
  - better oriented to OLAP (analytics) rather than OLTP

Group A: Google Bigtable, Apache HBase, Hypertable, Apache Cassandra

Group B: SAP IQ, HP Vertica, Actian Vector, MonetDB, Infobright

	A	B
data model	multi-dimensional mapping	relational data model
column independence	groups of columns are stored together	every columns is stored individually
language	NoSQL	SQL
workload	few reads, more upserts	more reads, few upserts
storage	sparse column-store	dense column-store (positional)

http://dbmsmusings.blogspot.it/2010/03/distinguishing-two-major-types-of_29.html

MonetDB data type	NumPy data type
BOOLEAN	numpy.int8
TINYINT	numpy.int8
SMALLINT	numpy.int16
INTEGER	numpy.int32
BIGINT	numpy.int64
REAL	numpy.float32
FLOAT	numpy.float64
HUGEINT	numpy.float64
STRING	numpy.object


```
CREATE FUNCTION random_floats() RETURNS TABLE(number FLOAT) LANGUAGE PYTHON
    import numpy as np
    values = np.random.rand(1, 30)
    return values
};
```

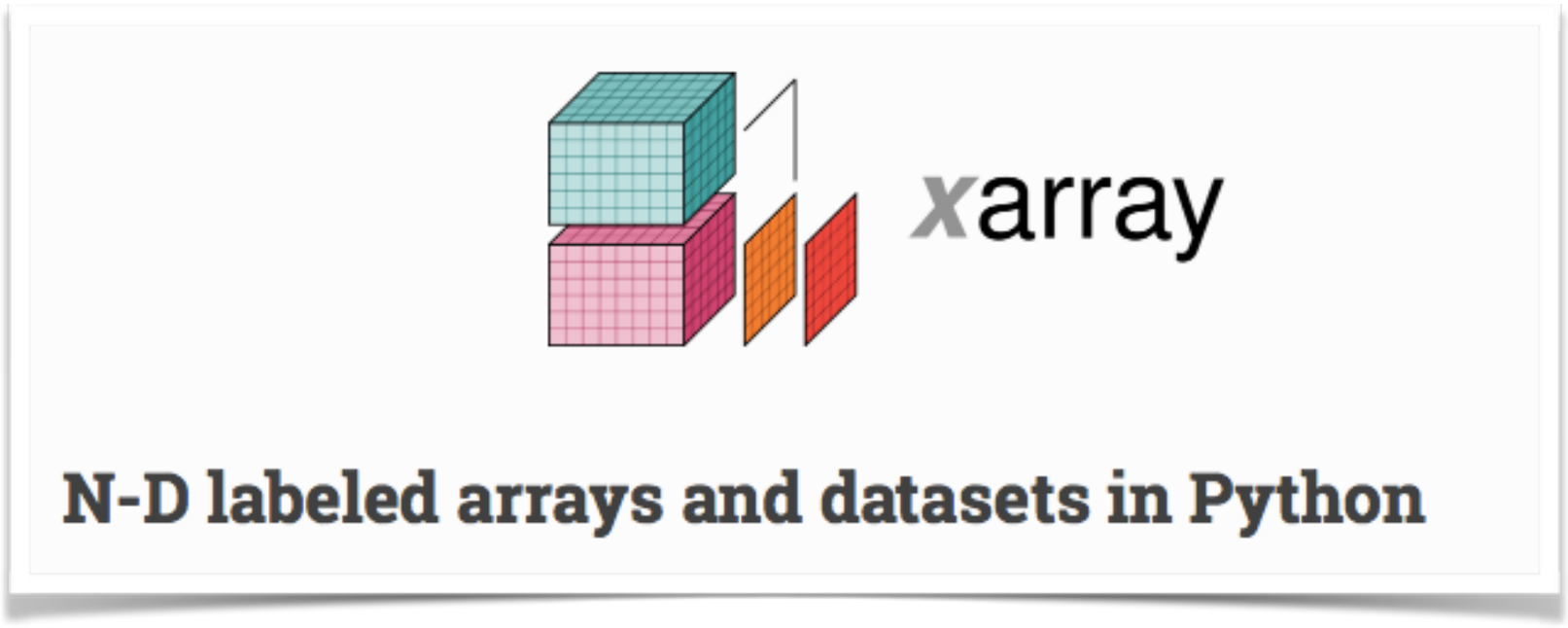
```
CREATE FUNCTION scikit_conf_matrix (y_true INT, y_pred INT)
RETURNS TABLE(col1 INT, col2 INT) LANGUAGE PYTHON
{
    from sklearn.metrics import confusion_matrix
    cfm = confusion_matrix(y_true, y_pred)
    return cfm
};
```

```
CREATE FUNCTION conf_matrix_stats
(c1 INT, c2 INT)
RETURNS TABLE
(accuracy FLOAT, precision FLOAT, sensitivity FLOAT, specificity FLOAT, f1 FLOAT)
LANGUAGE PYTHON
{
    result = dict()
    TP = c2[1]*1.00
    TN = c1[0]*1.00
    FN = c2[0]*1.00
    FP = c1[1]*1.00
    N = TP+TN+FP+FN
    accuracy = (TP + TN)/N
    precision = TP / (TP + FP)
    sensitivity = TP / (TP + FN)
    specificity = TN / (TN + FP)
    F1 = 2*TP / (2*TP + FP + FN)
    result['accuracy'] = accuracy
    result['precision'] = precision
    result['sensitivity'] = sensitivity
    result['specificity'] = specificity
    result['f1'] = F1
    return result
};
```

- **In-Database analytics with python and MonetDB** by *G. Emireni* @PyData Italy 2016

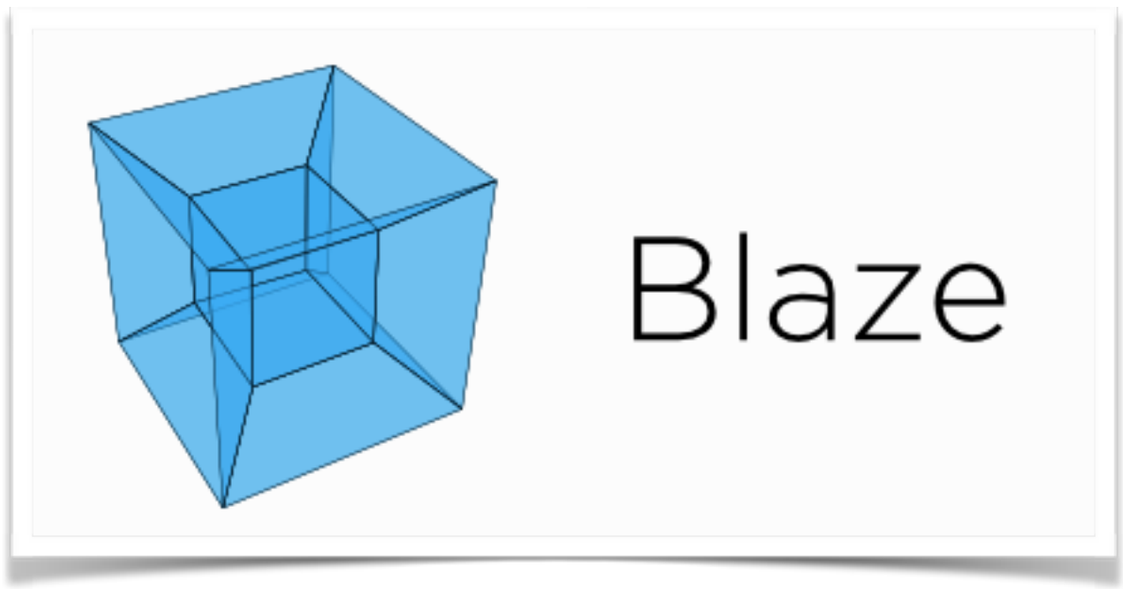
```
SELECT * FROM conf_matrix_stats (
    (SELECT * FROM scikit_conf_matrix (
        (SELECT a.target*1.00 AS y_true,
          b.prediction*1.00 AS y_pred
) );
FROM promodata_preproc a
INNER JOIN predicted b ON a.id = b.id))
```


A format
has no
name



http://xarray.pydata.org/en/stable/index.html

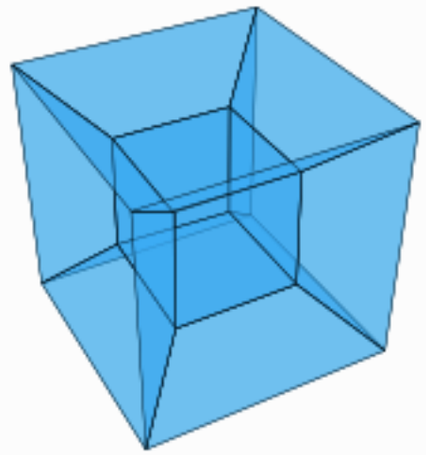
http://blaze.pydata.org



```python
import numpy as np
import pandas as pd
from blaze import data, by, join, merge, concat

# construct a DataFrame
df = pd.DataFrame({
    'name': ['Alice', 'Bob', 'Joe', 'Bob'],
    'amount': [100, 200, 300, 400],
    'id': [1, 2, 3, 4],
})

# put the `df` DataFrame into a Blaze Data obje
df = data(df)
```



Blaze

Out-of-Core Processing

```
>>> from blaze import data, by
>>> t = data('sqlite:///%s::iris' % example('iris.db'))
>>> t.peek()
   sepal_length  sepal_width  petal_length  petal_width      species
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
5           5.4          3.9           1.7          0.4  Iris-setosa
6           4.6          3.4           1.4          0.3  Iris-setosa
7           5.0          3.4           1.5          0.2  Iris-setosa
8           4.4          2.9           1.4          0.2  Iris-setosa
9           4.9          3.1           1.5          0.1  Iris-setosa
...
>>> by(t.species, max=t.petal_length.max(), min=t.petal_length.min())
          species  max  min
0     Iris-setosa  1.9  1.0
1 Iris-versicolor  5.1  3.0
2  Iris-virginica  6.9  4.5
```


```
In [7]: xr.DataArray(pd.Series(range(3), index=list('abc'), name='foo'))
Out[7]:
<xarray.DataArray 'foo' (dim_0: 3)>
array([0, 1, 2])
Coordinates:
  * dim_0    (dim_0) object 'a' 'b' 'c'
```

```
In [4]: xr.DataArray(np.random.randn(2, 3))
Out[4]:
<xarray.DataArray (dim_0: 2, dim_1: 3)>
array([[-1.344,  0.845,  1.076],
       [-0.109,  1.644, -1.469]])
Coordinates:
  * dim_0    (dim_0) int64 0 1
  * dim_1    (dim_1) int64 0 1 2

In [5]: data = xr.DataArray(np.random.randn(2, 3), [('x', ['a', 'b']), ('y', [-2, 0, 2])])

In [6]: data
Out[6]:
<xarray.DataArray (x: 2, y: 3)>
array([[ 0.357, -0.675, -1.777],
       [-0.969, -1.295,  0.414]])
Coordinates:
  * x        (x) |S1 'a' 'b'
  * y        (y) int64 -2 0 2
```

*Complicated **data** require complicated **formats***

*Complicated formats require good **tools***

OPeNDAP: http://goo.gl/fMehjh

```
import this
```

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Thanks a lot for your kind attention


@leriomaggio


vmaggio@fbk.com


+ValerioMaggio


it.linkedin.com/in/valeriomaggio