Budapest BI FORUM 2016

DATA FORMATS FOR DATA SCIENCE

Remastered

Valerio Maggio

@leriomaggio

Data Scientist and Researcher

Fondazione Bruno Kessler (FBK)

Trento, Italy

WhoAmI

- 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

kidding, that's me!-)

FLORENCE 2016

DATA FORMATS FOR DATA SCIENCE

- Data **Processing**
  - Q: What's the better way to process (my) 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

JUPYTER NOTEBOOK FOR DATA SHARING AND DOCUMENTATION

#1

DATA THAT YOU <u>CAN</u> READ

Human Readable Formats

DOES YOUR DATA HAS A STRUCTURE OR NOT?

DATA FORMATS THAT YOU CAN READ

Unstructured Data

9.609482288360595703e-01	3.331715166568756104e-01	3.583630323410034180e-01	2.592278420925140З
9.792612791061401367e-01	9.008772969245910645e-01	2.746424674987792969e-01	[illegible]
9.112978577613830566e-01	8.600413799285888672e-01	3.737630546092987061e-01	[illegible]
9.571560025215148926e-01	8.606715202331542969e-01	2.630991935729980469e-01	[illegible]
9.323833584785461426e-01	8.171402812004089355e-01	4.377277791500091553e-01	1.502759903669357З
9.356079697608947754e-01	7.851068377494812012e-01	5.012405514717102051e-01	1.455076485872268б
9.092011451721191406e-01	7.483353614807128906e-01	4.298384189605712891e-01	2.541802823543548
9.503287672996520996e-01	8.873134255409240723e-01	2.655168473720550537e-01	2.211218476295471
9.237284064292907715e-01	8.363176584243774414e-01	3.627101480960845947e-01	2.365967631340026
9.562172293663024902e-01	9.194136857986450195e-01	3.819596767425537109e-01	3.117111623287200
9.461185932159423828e-01	8.484295606613159180e-01	3.903456628322601318e-01	1.668368875980377
9.467664361000061035e-01	8.682620525360107422e-01	3.137815594673156738e-01	1.826369911432266
9.397199749946594238e-01	8.609640002250671387e-01	3.499407768249511719e-01	1.618804782629013
9.222379326820373535e-01	8.876875042915344238e-01	3.556989133358001709e-01	3.479544818401336
9.418539404869079590e-01	8.918866515159606934e-01	2.337521761655807495e-01	2.460925579071044
8.906930685043334961e-01	8.144904375076293945e-01	4.380804598331451416e-01	5.200685262680053
8.549255132675170898e-01	7.775652408599853516e-01	2.998122274875640869e-01	4.507026672363281
9.364917278289794922e-01	8.836621046066284180e-01	4.243750274181365967e-01	2.403212934732437
9.408168196678161621e-01	4.739229083061218262e-01	3.617838919162750244e-01	2.829778790473937
9.318765997886657715e-01	7.781792879104614258e-01	4.771032333374023438e-01	1.843434870243072
9.611908197402954102e-01	7.101613283157348633e-01	4.384511113166809082e-01	2.055199444293975
9.418456554412841797e-01	7.011284828186035156e-01	4.341177344322204590e-01	3.878928422927856
9.144946336746215820e-01	3.438472747802734375e-01	4.719765782356262207e-01	2.633934617042541
9.463409185409545898e-01	3.462429642677307129e-01	3.763888478279113770e-01	2.532341480255126

```
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
```

```python
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)
```

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

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

Numpy to the rescue

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

```python
# 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

Structured Data

```
FILENAME,DATASET,CLASS,CAMERA,CONF,VARIETY,SOSQ,SOMQ,CAT,FILEPATH
sol_L_e_b_001.jpg,sol,E,NA,B,Lagorai,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/sol/early/sol_L_e_b_001.jpg
sol_L_e_b_002.jpg,sol,E,NA,B,Lagorai,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/sol/early/sol_L_e_b_002.jpg
sol_V_e_b_001.jpg,sol,E,NA,B,Vajolet,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/sol/early/sol_V_e_b_001.jpg
sol_V_e_b_002.jpg,sol,E,NA,B,Vajolet,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/sol/early/sol_V_e_b_002.jpg
sol_V_e_b_003.jpg,sol,E,NA,B,Vajolet,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/sol/early/sol_V_e_b_003.jpg
sol_V_e_b_004.jpg,sol,E,NA,B,Vajolet,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/sol/early/sol_V_e_b_004.jpg
sol_V_e_b_005.jpg,sol,E,NA,B,Vajolet,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/sol/early/sol_V_e_b_005.jpg
sol_L_g_b_001.jpg,sol,G,NA,B,Lagorai,NA,NA,NA,/home/webvalley/deepLearnin
g/data/images/datasets_new/sol/good/sol_L_g_b_001.jpg
sol_L_g_b_002.jpg,sol,G,NA,B,Lagorai,NA,NA,NA,/home/webvalley/deepLearnin
```

```
!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

pandas

```
import pandas as pd

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

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

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

```
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
```

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	sample_id									emperature	location	outlier	spectrum_74
int	unicode									oat	str	str	float
1	fc2dd6d8-11f5-45d3-bf9a-075af1900									1.67	None	no	0.62537066
2	fc2dd6d8-11f5-45d3-bf9a-075af1900									1.57	None	no	0.60968911
3	fc2dd6d8-11f5-45d3-bf9a-075af1900									1.51	None	no	0.62250119
4	d1b88976-acaf-4c3f-849e-c78ae51af									1.05	None	no	0.75176557
5	d1b88976-acaf-4c3f-849e-c78ae51af									1.01	None	no	0.76443552
6	d1b88976-acaf-4c3f-849e-c78ae51af									1.01	None	no	0.77390102
7	be407a0d-5b7d-4241-aaae-3d2259c									1.25	None	no	0.76951867
8	be407a0d-5b7d-4241-aaae-3d2259c									0.39	None	no	0.64316374
9	be407a0d-5b7d-4241-aaae-3d2259c									0.25	None	no	0.74909751
10	be407a0d-5b7d-4241-aaae-3d2259c									0.19	None	no	0.67673417
11	9571b870-4b7f-48b0-8aeb-7eda91adfb08	2015-09-07 11:30:07.199000	Goat	A	23	30	2015-09-26 09:00:00	E036D39ADE70A12D	None	20.95	None	no	0.52303851
12	9571b870-4b7f-48b0-8aeb-7eda91adfb08	2015-09-07 11:30:20.012000	Goat	A	23	30	2015-09-26 09:00:00	E036D39ADE70A12D	None	20.75	None	no	0.62668754
13	9571b870-4b7f-48b0-8aeb-7eda91adfb08	2015-09-07 11:30:27.086000	Goat	A	23	30	2015-09-26 09:00:00	E036D39ADE70A12D	None	20.61	None	no	0.57163047
14	6b057fdb-5ba8-4bf6-afce-63b9f28a9a81	2015-09-07 11:31:52.264000	Cow	A	23	28	2015-09-18 09:00:00	E036D39ADE70A12D	None	20.25	None	no	0.66914171
15	6b057fdb-5ba8-4bf6-afce-63b9f28a9a81	2015-09-07 11:32:00.425000	Cow	A	23	28	2015-09-18 09:00:00	E036D39ADE70A12D	None	20.15	None	no	0.72110082
16	6b057fdb-5ba8-4bf6-afce-63b9f28a9a81	2015-09-07 11:32:07.996000	Cow	A	23	28	2015-09-18 09:00:00	E036D39ADE70A12D	None	20.05	None	no	0.72678361
17	f7fbd198-683e-4ead-8008-b6daacec5eca	2015-09-07 11:33:59.157000	Cow	A	23	28	2015-09-17 09:00:00	E036D39ADE70A12D	None	20.61	None	no	0.63983926
18	f7fbd198-683e-4ead-8008-b6daacec5eca	2015-09-07 11:34:06.762000	Cow	A	23	28	2015-09-17 09:00:00	E036D39ADE70A12D	None	20.55	None	no	0.59454983
19	f7fbd198-683e-4ead-8008-b6daacec5eca	2015-09-07 11:34:14.473000	Cow	A	23	28	2015-09-17 09:00:00	E036D39ADE70A12D	None	20.55	None	no	0.65788239

```
!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
```

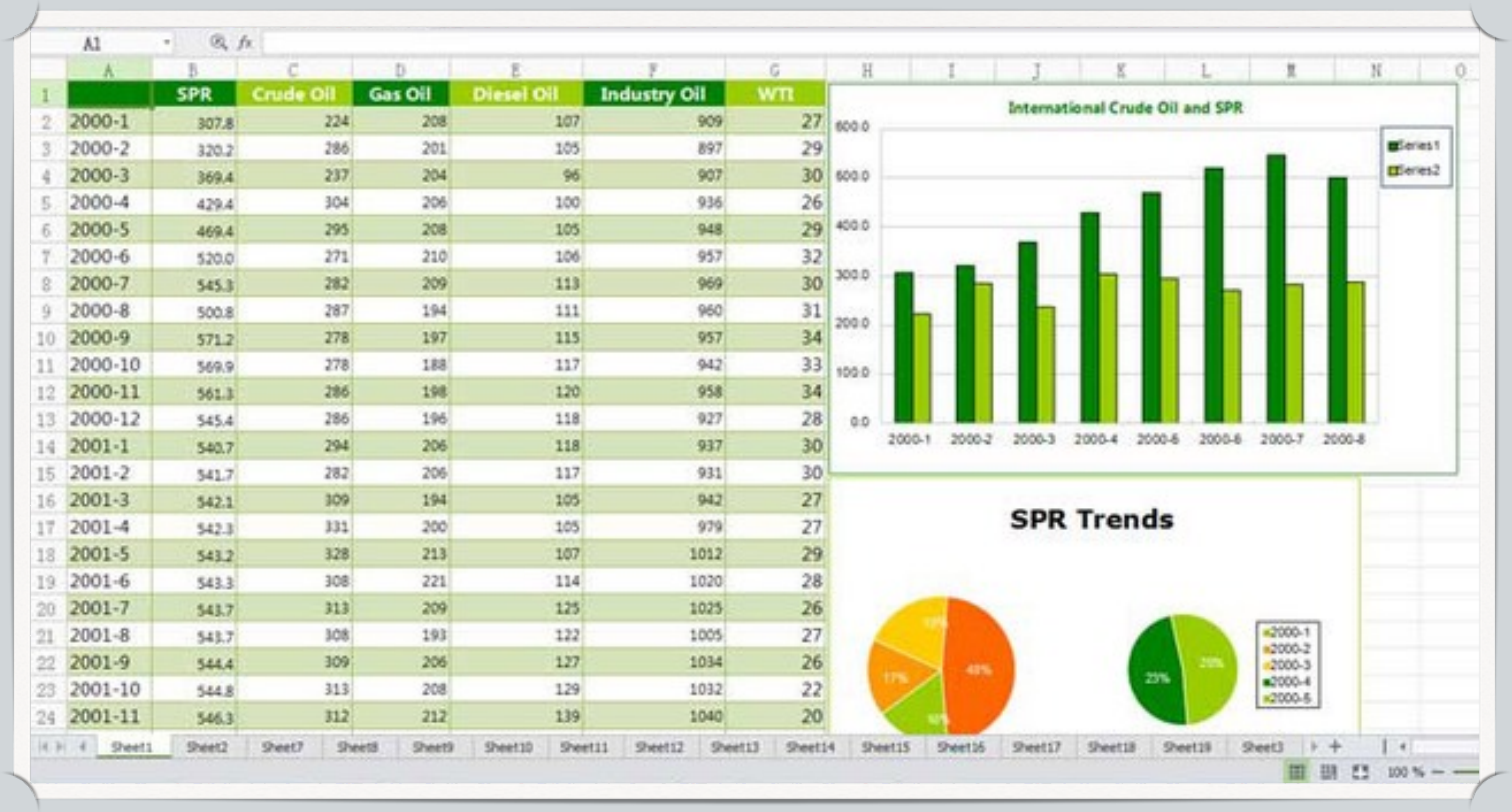
```
collection = pd.read_csv('files/textual/collection.csv', skiprows=10)
```

```
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

SPREADSH~~I~~EETS

XSL(X)



	SPR	Crude Oil	Gas Oil	Diesel Oil	Industry Oil	WTI
2000-1	307.8	224	208	107	909	27
2000-2	320.2	286	201	105	897	29
2000-3	369.4	237	204	96	907	30
2000-4	429.4	304	206	100	936	26
2000-5	469.4	295	208	105	948	29
2000-6	520.0	271	210	106	957	32
2000-7	545.3	282	209	113	969	30
2000-8	500.8	287	194	111	960	31
2000-9	571.2	278	197	115	957	34
2000-10	569.9	278	188	117	942	33
2000-11	561.3	286	198	120	958	34
2000-12	545.4	286	196	118	927	28
2001-1	540.7	294	206	118	937	30
2001-2	541.7	282	206	117	931	30
2001-3	542.1	309	194	105	942	27
2001-4	542.3	331	200	105	979	27
2001-5	543.2	328	213	107	1012	29
2001-6	543.3	308	221	114	1020	28
2001-7	543.7	313	209	125	1025	26
2001-8	543.7	308	193	122	1005	27
2001-9	544.4	309	206	127	1034	26
2001-10	544.8	313	208	129	1032	22
2001-11	546.3	312	212	139	1040	20

```
In [2]: pd.read_excel?
```

```
Signature: pd.read_excel(io, sheetname=0, header=0, skiprows=None, skip_footer=0,
index_col=None, names=None, parse_cols=None, parse_dates=False, date_parser=None,
na_values=None, thousands=None, convert_float=True, has_index_names=None, converters=None
, engine=None, squeeze=False, **kwds)
Docstring:
Read an Excel table into a pandas DataFrame
```

xlsxwriter.readthedocs.io

```
worksheet.data_validation('B25', {'validate': 'integer',
                                  'criteria': 'between',
                                  'minimum': 1,
                                  'maximum': 100,
                                  'input_title': 'Enter an integer:',
                                  'input_message': 'between 1 and 100'})
```

```
import xlsxwriter

# Create a workbook and add a worksheet.
workbook = xlsxwriter.Workbook('Expenses01.xlsx')
worksheet = workbook.add_worksheet()

# Some data we want to write to the worksheet.
expenses = (
    ['Rent', 1000],
    ['Gas',   100],
    ['Food',  300],
    ['Gym',    50],
)

# Start from the first cell. Rows and columns are zero indexed.
row = 0
col = 0

# Iterate over the data and write it out row by row.
for item, cost in (expenses):
    worksheet.write(row, col,     item)
    worksheet.write(row, col + 1, cost)
    row += 1

# Write a total using a formula.
worksheet.write(row, 0, 'Total')
worksheet.write(row, 1, '=SUM(B1:B4)')

workbook.close()
```

Structured Data++

Analyse DBs from many angles

- Normalisation (No Duplicates) & Fixed Structure
- Relational Databases
- SQL: Structured Query Language
  - Many different dialects!
- ORM is the way!

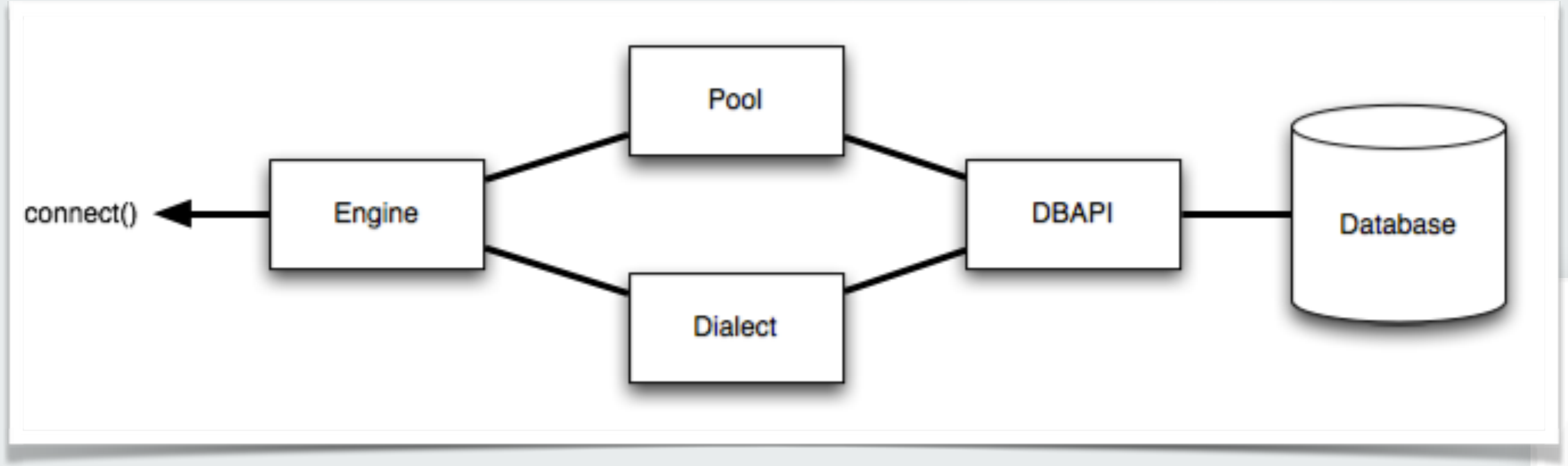
1. INFORMATION ARCHITECTURE

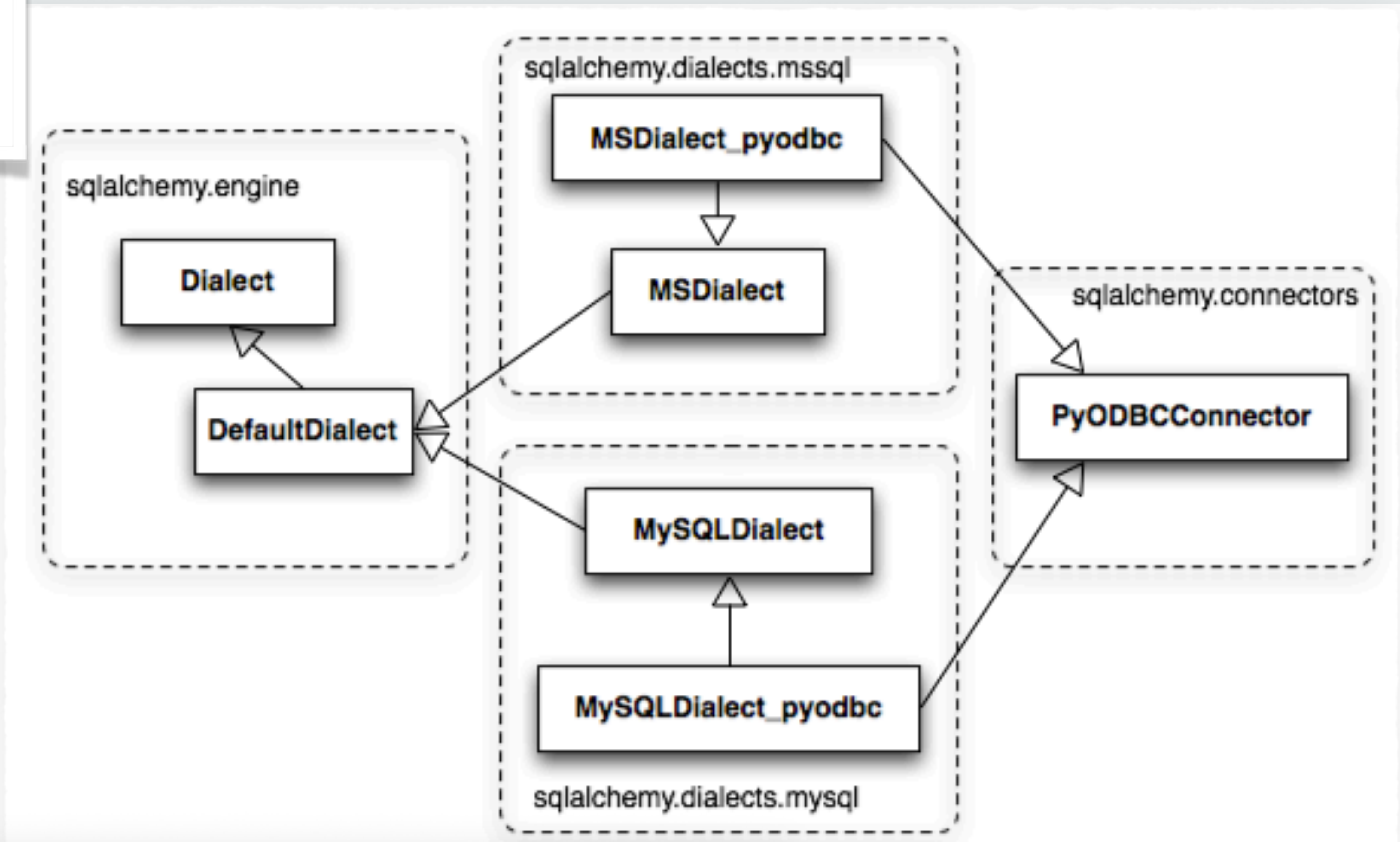

aLCHEMY Live
dIRE sTRAITS
THIS IS A RECORDING OF EXCERPTS OF ONE DIRE STRAITS PERFORMANCE,
IT CONTAINS NO RE-RECORDING OR OVERDUBS OF ANY KIND

SQL

ALCHEMY

SQLAlchemy






```python
# sqlite://<nohostname>/<path>
# where <path> is relative:
engine = create_engine('sqlite:///foo.db')
```

```python
# default
engine = create_engine('postgresql://scott:tiger@localhost/mydatabase')

# psycopg2
engine = create_engine('postgresql+psycopg2://scott:tiger@locahost/mydatabase')

# pg8000
engine = create_engine('postgresql+pg8000://scott:tiger@localhost/mydatabase')
```

SQLAlchemy

```python
from sqlalchemy.ext.declarative import declarative_base

Base = declarative_base()  # This will be the Declarative Base Class
```

```python
from sqlalchemy import Column, Integer, String

class User(Base):
    __tablename__ = 'users'

    id = Column(Integer, primary_key=True)
    name = Column(String)
    fullname = Column(String)
    password = Column(String)

    def __repr__(self):
        return "<User(name='{}', fullname='{}', password='{}')>".format(
                                self.name, self.fullname, self.password)
```

```python
session.add_all([
   User(name='anakin', fullname='Anakin Skywalker', password='iamyourfather'),
   User(name='obiwan', fullname='Obi-Wan Kenobi', password='usetheforce'),
   User(name='luke', fullname='Luke Skywalker', password='lastjedi'),
   User(name='leia', fullname='Leia Organa', password='iloveu'),
   User(name='solo', fullname='Han Solo', password='iknow'),
   User(name='chuby', fullname='Chubaka', password='uuuoouuu'),])
session.commit()
```

- Your data requires a **flexible** (not fixed) structure
- a.k.a. **NO-SQL** (*databases*)
- **JSON**-based data format
- e.g. **MongoDB**

pymongo

2. FLEXIBILITY

itle":"Hello world!","pname":"hello-world","pstatus":"publish"},{"ptitle":"Sample
","pname":"sample-page","pstatus":"trash"},{"ptitle":"Auto Draft","p
"},{"ptitle":"About us","pname":"about-us","pstatus":"publish"},{"pt
pname":"4-revision-v1","pstatus":"inherit"},{"ptitle":"About us","pname":"4-revisio
pstatus":"inherit"},{"ptitle":"Introduction","pname":"introduction","pstatus":"publ
itle":"Introduction","pname":"7-revision-v1","pstatus":"inherit"},
itle":"Achievements","pname":"achievements","pstatus":"publish"},
itle":"Achievements","pname":"9-revision-v1","pstatus":"inherit"},
itle":"API's","pname":"apis","pstatus":"publish"},{"ptitle":"API's","pname":"11-revi
pstatus":"inherit"},{"ptitle":"Apis","pname":"apis-2","pstatus":"publish"},
itle":"Apis","pname":"17-revision-v1","pstatus":"inherit"},
itle":"FDF","pname":"fdf","pstatus":"publish"},{"ptitle":"FDF","pname":"19-revision-
pstatus":"inherit"},{"ptitle":"Product Portfolio","pname":"product-
folio","pstatus":"publish"},{"ptitle":"Product Portfolio","pname":"21-revision-
pstatus":"inherit"},{"ptitle":"Intermediate Products List","pname":"intermediate-pr
","pstatus":"publish"},{"ptitle":"Intermediate Products List","pname":"23-revision-
pstatus":"inherit"},{"ptitle":"Impurity Standard List","pname":"impurity-standard-
","pstatus":"publish"},{"ptitle":"Impurity Standard List","pname":"25-revision-
pstatus":"inherit"},{"ptitle":"Regulatory Status","pname":"regulatory-
us","pstatus":"publish"},{"ptitle":"Regulatory Status","pname":"27-revision-
pstatus":"inherit"},{"ptitle":"Contact Us","pname":"contact-us","pstatus":"publish"
itle":"Contact Us","pname":"29-revision-v1","pstatus":"inherit"},

JSON

Jupyter Notebook Data Format

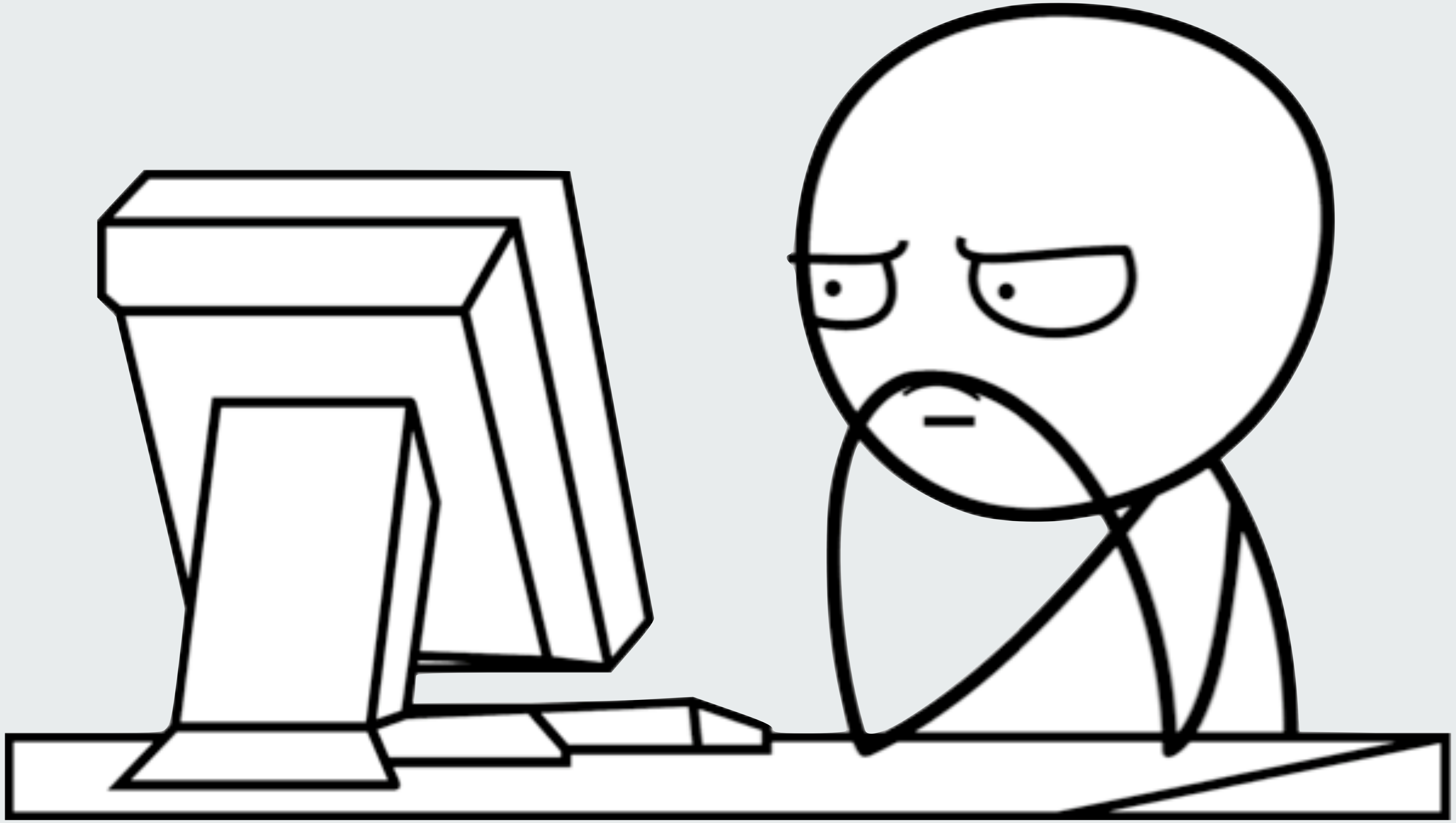
```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Custom Magic Examples"
   ]
  },
  {…},
  {…},
  {…},
  {…},
```

```
  {
   "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')"
   ]
  },
```

- Your data requires a **flexible**(ish) structure
- But you want to validate your data
- XML-based data format

2.5 FLEXIBILITY AND *validation*

Use XML, they said.



It will be fun, they said.

- Normalisation (No Duplicates) & Fixed Structure
- Relational Databases
- (Super effective) in-DB Analytics
- **Column**-oriented DB

3 STRUCTURE AND *speed*

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

monetdb

```
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
};
```

```
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))
```

#2

DATA THAT YOU CANNOT READ

unless..

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

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

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

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

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

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

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

```python
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
- (+) Works great with both big or tiny datasets
- (+) Storage friendly
  - Allows for Compression
- (+) Dev. Friendly
  - Query DSL + Multiple-language support
  - **Python:** PyTables, hdf5, h5py

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

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

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

```python
dset.shape
```

```
(100,)
```

```python
dset.dtype
```

```
dtype('int32')
```

```python
type(dset)
```

```
h5py._hl.dataset.Dataset
```

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

```python
dset[10]
```

```
10
```

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

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

NUMPY ARRAYS TIGHT INTEGRATION

with PyTables

```
import tables as tb
```

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

`Array`
: The files of the filesystem

`CArray`
: Chunked arrays

`EArray`
: Extendable arrays

`VLArray`
: Variable-length arrays

`Table`
: Structured arrays

```
# 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 = grp.create_dataset("new_dataset", (50,), dtype='f')
```

```
dset2.name
```

```
'/second_level/new_dataset'
```

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

```
dset3.name
```

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

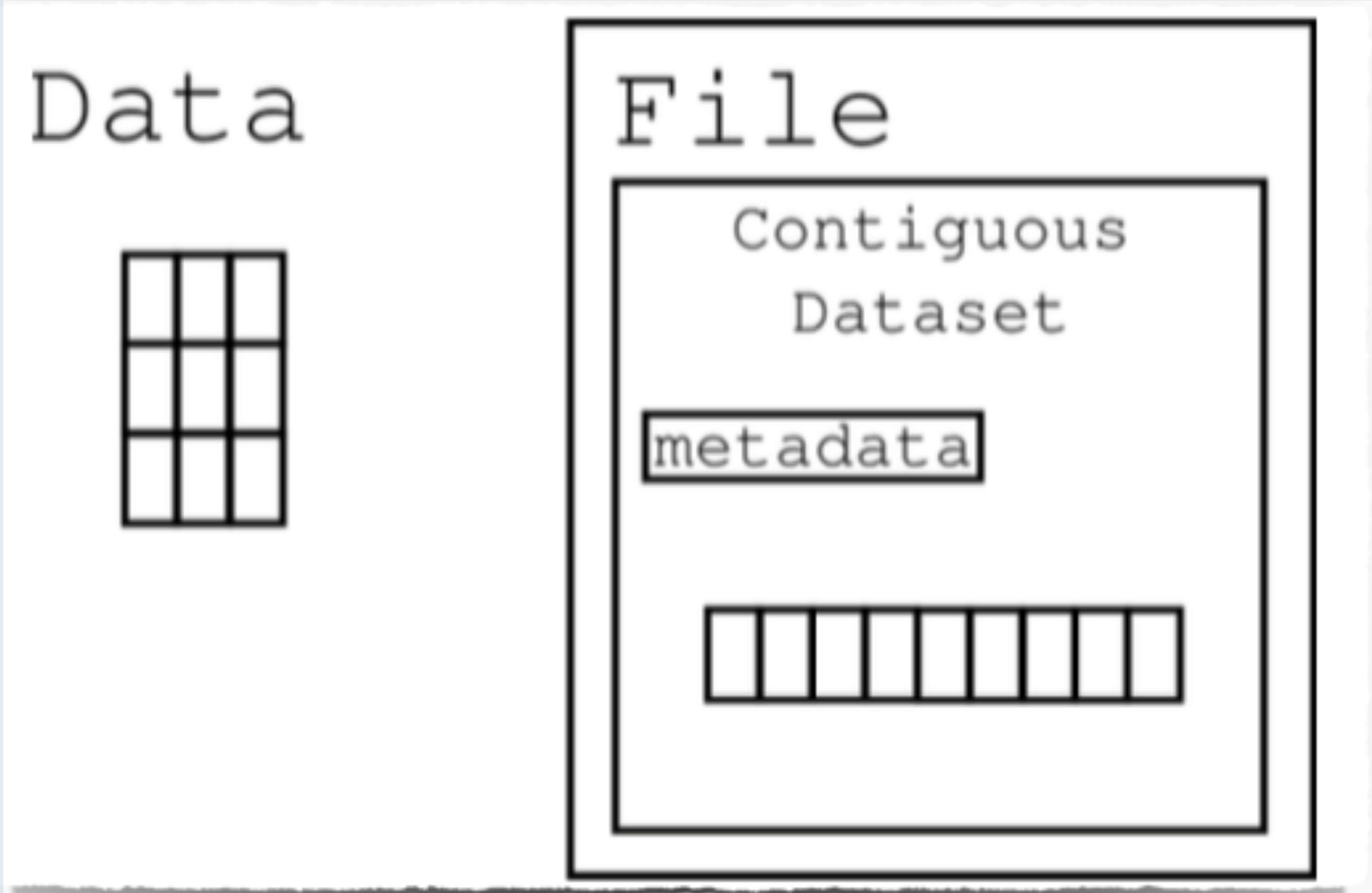
```
dset3_f = f['second_level_2/dset3']
```

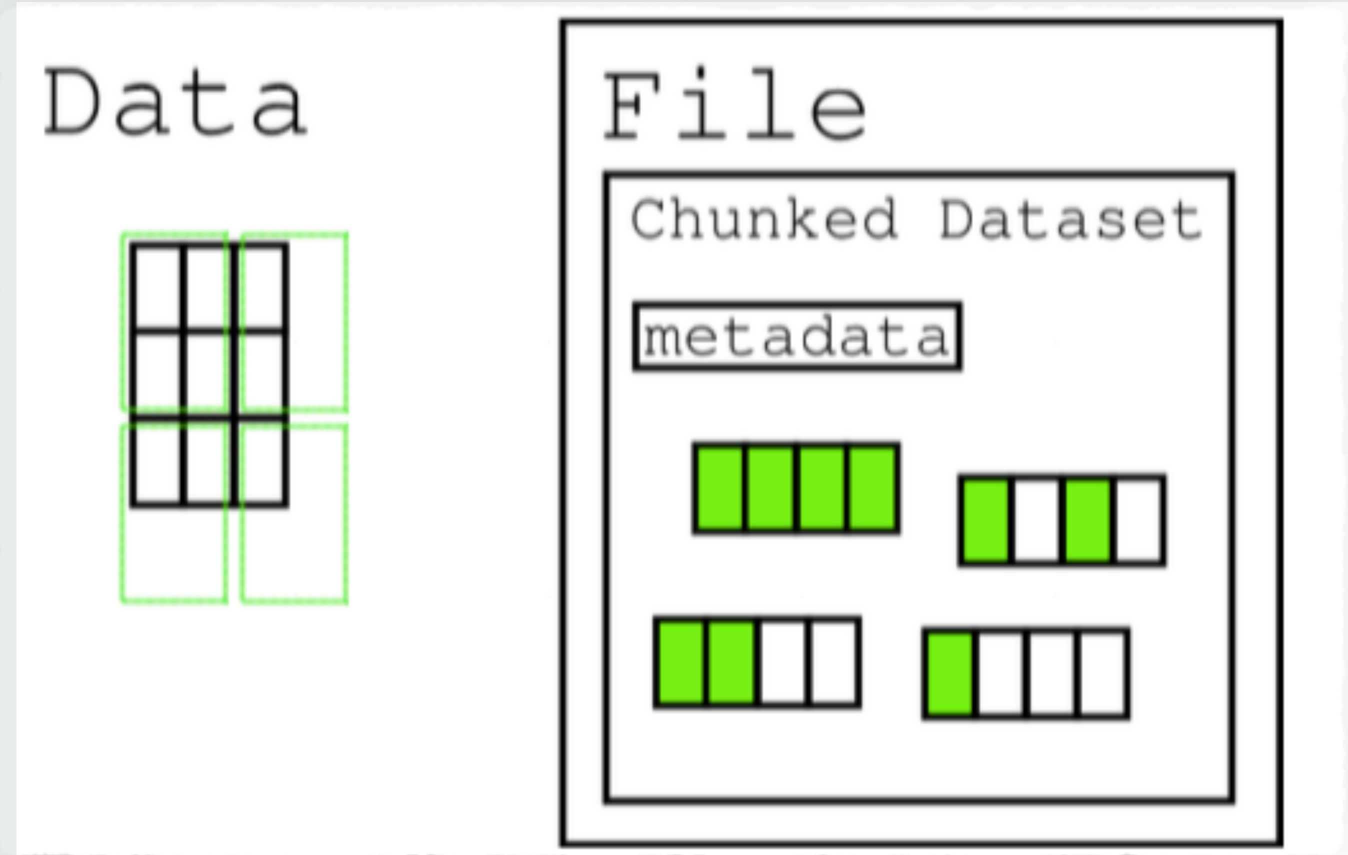
```
dset3 == dset3_f
```

```
True
```

DATA CHUNKING

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

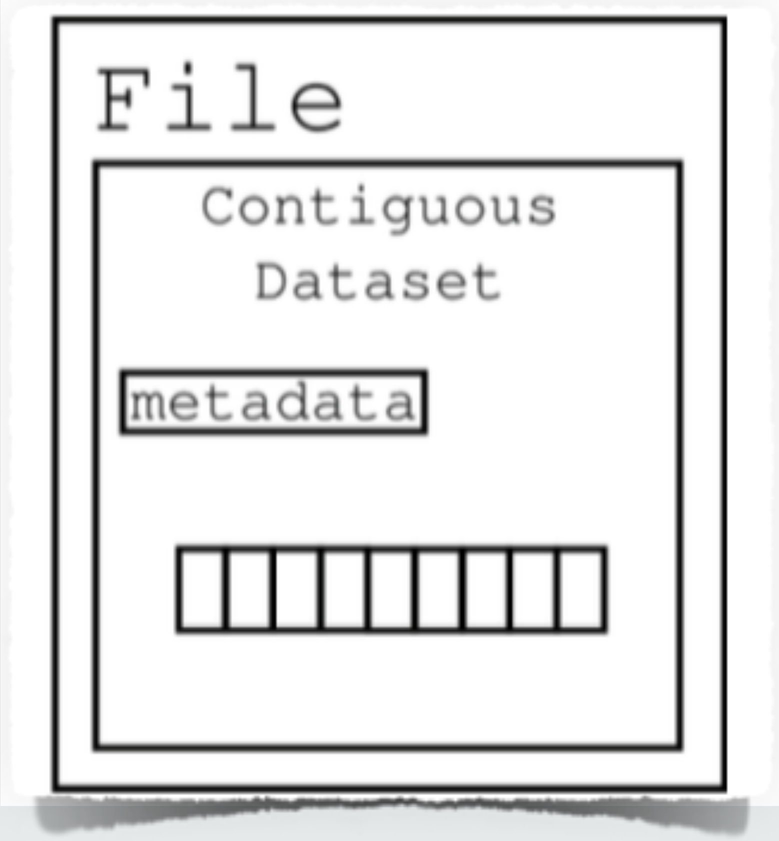


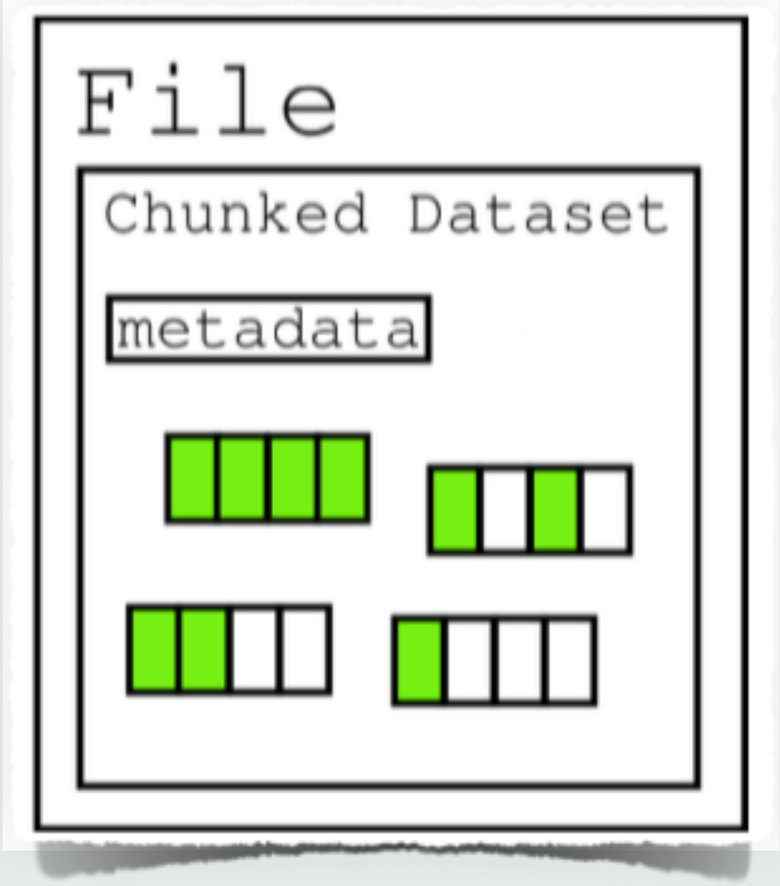


* A. Scopatz, K.D. Huff - Effective Computations in Physics - Field Guide to Research in Python, O'Reilly 2015

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

MPI (mpi4py) integration

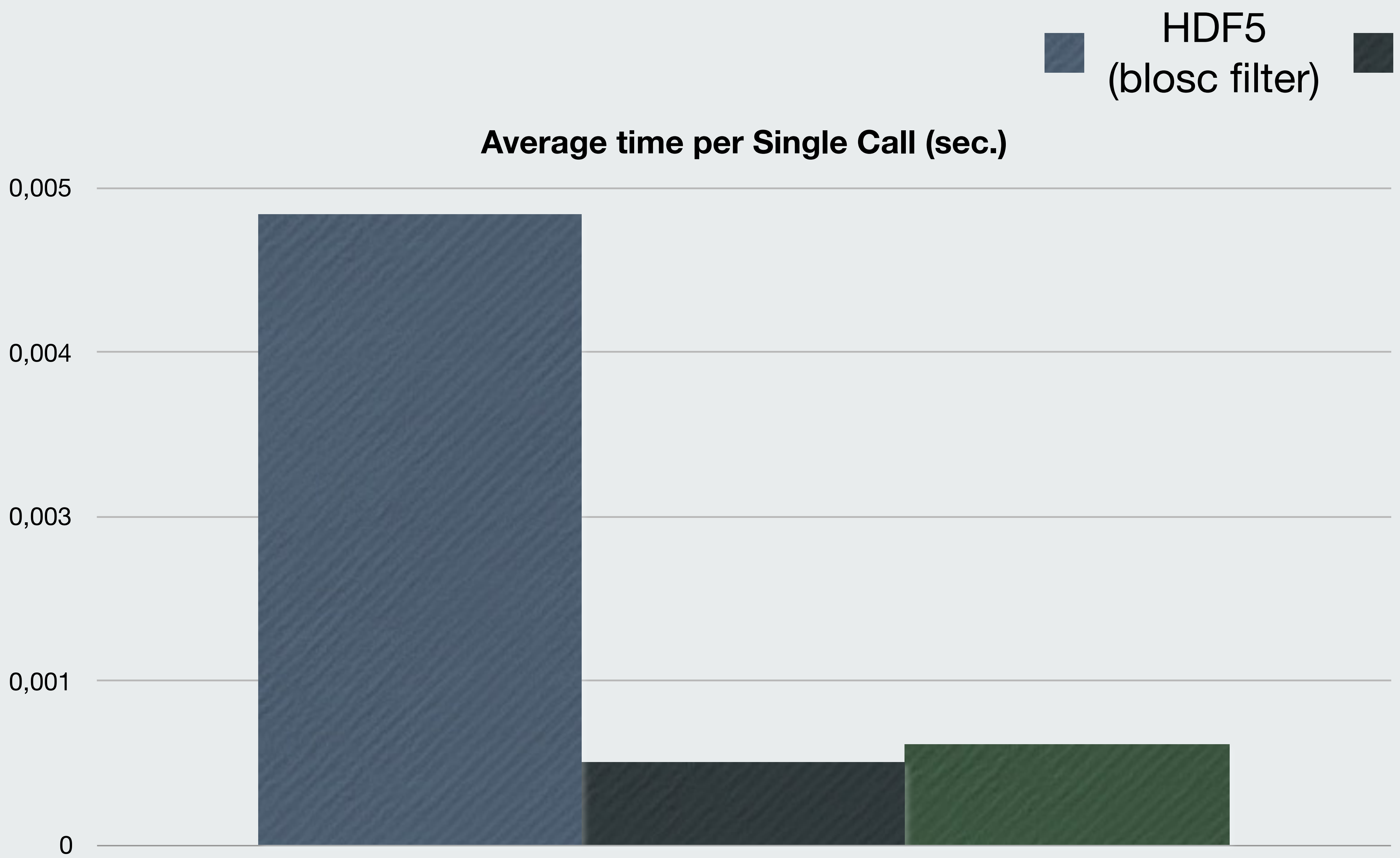
```
from mpi4py import MPI
import h5py

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()
```

PARALLEL HDF5

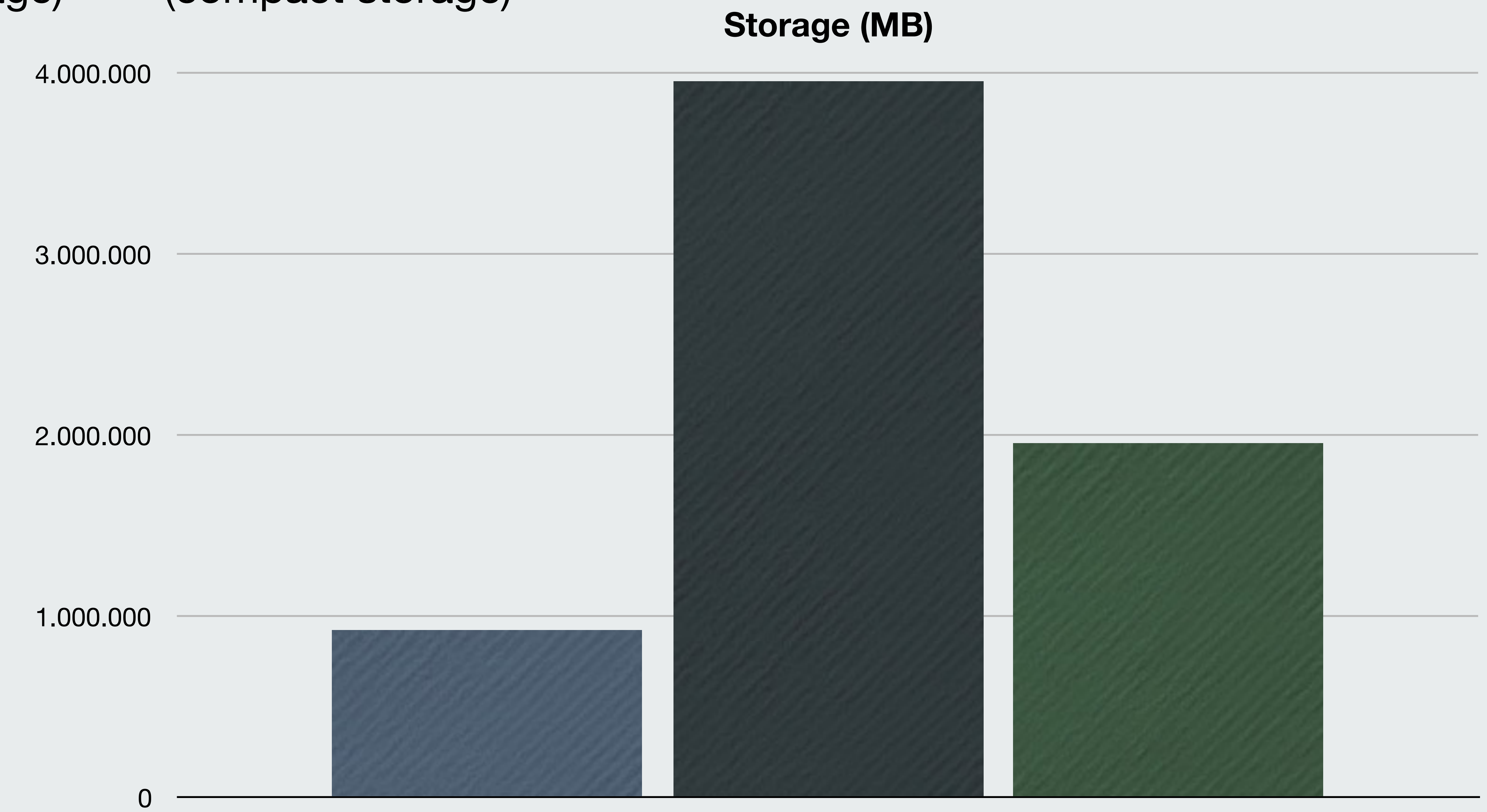
HDF5 VS MONGODB

Total Number of Documents	Total Number of Entries
100.000	8.755.882

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



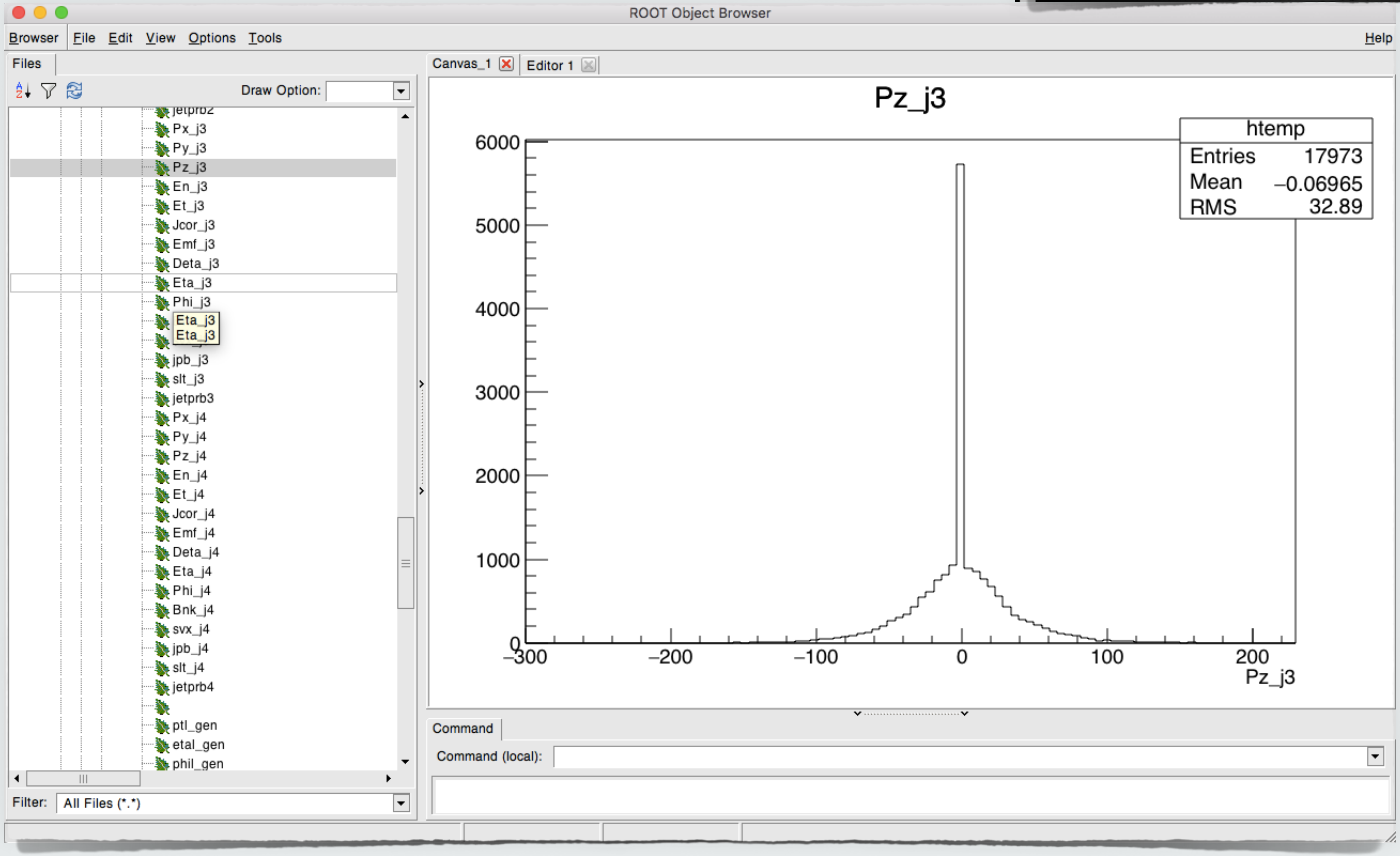
Query Time



Storage Space

- 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 [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++)
               {
               int content = h->GetBinContent(i,j);
               if(content!=0)
                   count+=1;
               }
               arr[i]=count;

           }
      }
```

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/

```python
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')
```

```python
import rootpy.plotting
from rootpy.io import root_open
```

```python
root_file = root_open(infile)
rpy_tree = root_file.MONTECARLO
```

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

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

Tight integration with PyROOT objects

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

rnp.fill_hist(histogram, energies)

histogram.Draw()
```

```
$ 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

MULTIDIMENSIONAL LABELED ARRAY

xarray

N-D labeled arrays and datasets in Python

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

when Pandas is not enough!

```
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
```

#3

DATA IN MULTIPLE FORMATS

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

HDFS

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

```python
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']
```

Wildcard opening of CSVs on the HDFS

```python
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
>
```

```python
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
```

```
VendorID                          int64
tpep_pickup_datetime     datetime64[ns]
tpep_dropoff_datetime    datetime64[ns]
passenger_count                   int64
trip_distance                   float64
pickup_longitude
pickup_latitude
RateCodeID
store_and_fwd_flag
dropoff_longitude
```

```
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

Out-of-Core Processing

```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 object
df = data(df)
```

Blaze

```
>>> 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
```

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

Complicated formats require good ***tools***

```
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!
```

OPeNDAP: http://goo.gl/fMehjh

PyData

Thanks a lot for your kind attention

@leriomaggio

vmaggio@fbk.com

+ValerioMaggio

it.linkedin.com/in/valeriomaggio