

Data Formats for Data Science

Valerio Maggio

Data Scientist and Researcher

Fondazione Bruno Kessler (FBK)

Trento, Italy



@leriomaggio



About me

kidding, that's 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 beginning of May.


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

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

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

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

Adi 7. di Gennaio 1610 Giove si vedeva col Canone di
3. stelle fffe così ² ³ ⁴ ⁵ ⁶ ⁷ ⁸ ⁹ ¹⁰ ¹¹ ¹² ¹³ ¹⁴ ¹⁵ ¹⁶ ¹⁷ ¹⁸ ¹⁹ ²⁰ ²¹ ²² ²³ ²⁴ ²⁵ ²⁶ ²⁷ ²⁸ ²⁹ ³⁰ ³¹ ³² ³³ ³⁴ ³⁵ ³⁶ ³⁷ ³⁸ ³⁹ ⁴⁰ ⁴¹ ⁴² ⁴³ ⁴⁴ ⁴⁵ ⁴⁶ ⁴⁷ ⁴⁸ ⁴⁹ ⁵⁰ ⁵¹ ⁵² ⁵³ ⁵⁴ ⁵⁵ ⁵⁶ ⁵⁷ ⁵⁸ ⁵⁹ ⁶⁰ ⁶¹ ⁶² ⁶³ ⁶⁴ ⁶⁵ ⁶⁶ ⁶⁷ ⁶⁸ ⁶⁹ ⁷⁰ ⁷¹ ⁷² ⁷³ ⁷⁴ ⁷⁵ ⁷⁶ ⁷⁷ ⁷⁸ ⁷⁹ ⁸⁰ ⁸¹ ⁸² ⁸³ ⁸⁴ ⁸⁵ ⁸⁶ ⁸⁷ ⁸⁸ ⁸⁹ ⁹⁰ ⁹¹ ⁹² ⁹³ ⁹⁴ ⁹⁵ ⁹⁶ ⁹⁷ ⁹⁸ ⁹⁹ ¹⁰⁰ ¹⁰¹ ¹⁰² ¹⁰³ ¹⁰⁴ ¹⁰⁵ ¹⁰⁶ ¹⁰⁷ ¹⁰⁸ ¹⁰⁹ ¹¹⁰ ¹¹¹ ¹¹² ¹¹³ ¹¹⁴ ¹¹⁵ ¹¹⁶ ¹¹⁷ ¹¹⁸ ¹¹⁹ ¹²⁰ ¹²¹ ¹²² ¹²³ ¹²⁴ ¹²⁵ ¹²⁶ ¹²⁷ ¹²⁸ ¹²⁹ ¹³⁰ ¹³¹ ¹³² ¹³³ ¹³⁴ ¹³⁵ ¹³⁶ ¹³⁷ ¹³⁸ ¹³⁹ ¹⁴⁰ ¹⁴¹ ¹⁴² ¹⁴³ ¹⁴⁴ ¹⁴⁵ ¹⁴⁶ ¹⁴⁷ ¹⁴⁸ ¹⁴⁹ ¹⁵⁰ ¹⁵¹ ¹⁵² ¹⁵³ ¹⁵⁴ ¹⁵⁵ ¹⁵⁶ ¹⁵⁷ ¹⁵⁸ ¹⁵⁹ ¹⁶⁰ ¹⁶¹ ¹⁶² ¹⁶³ ¹⁶⁴ ¹⁶⁵ ¹⁶⁶ ¹⁶⁷ ¹⁶⁸ ¹⁶⁹ ¹⁷⁰ ¹⁷¹ ¹⁷² ¹⁷³ ¹⁷⁴ ¹⁷⁵ ¹⁷⁶ ¹⁷⁷ ¹⁷⁸ ¹⁷⁹ ¹⁸⁰ ¹⁸¹ ¹⁸² ¹⁸³ ¹⁸⁴ ¹⁸⁵ ¹⁸⁶ ¹⁸⁷ ¹⁸⁸ ¹⁸⁹ ¹⁹⁰ ¹⁹¹ ¹⁹² ¹⁹³ ¹⁹⁴ ¹⁹⁵ ¹⁹⁶ ¹⁹⁷ ¹⁹⁸ ¹⁹⁹ ²⁰⁰ ²⁰¹ ²⁰² ²⁰³ ²⁰⁴ ²⁰⁵ ²⁰⁶ ²⁰⁷ ²⁰⁸ ²⁰⁹ ²¹⁰ ²¹¹ ²¹² ²¹³ ²¹⁴ ²¹⁵ ²¹⁶ ²¹⁷ ²¹⁸ ²¹⁹ ²²⁰ ²²¹ ²²² ²²³ ²²⁴ ²²⁵ ²²⁶ ²²⁷ ²²⁸ ²²⁹ ²³⁰ ²³¹ ²³² ²³³ ²³⁴ ²³⁵ ²³⁶ ²³⁷ ²³⁸ ²³⁹ ²⁴⁰ ²⁴¹ ²⁴² ²⁴³ ²⁴⁴ ²⁴⁵ ²⁴⁶ ²⁴⁷ ²⁴⁸ ²⁴⁹ ²⁵⁰ ²⁵¹ ²⁵² ²⁵³ ²⁵⁴ ²⁵⁵ ²⁵⁶ ²⁵⁷ ²⁵⁸ ²⁵⁹ ²⁶⁰ ²⁶¹ ²⁶² ²⁶³ ²⁶⁴ ²⁶⁵ ²⁶⁶ ²⁶⁷ ²⁶⁸ ²⁶⁹ ²⁷⁰ ²⁷¹ ²⁷² ²⁷³ ²⁷⁴ ²⁷⁵ ²⁷⁶ ²⁷⁷ ²⁷⁸ ²⁷⁹ ²⁸⁰ ²⁸¹ ²⁸² ²⁸³ ²⁸⁴ ²⁸⁵ ²⁸⁶ ²⁸⁷ ²⁸⁸ ²⁸⁹ ²⁹⁰ ²⁹¹ ²⁹² ²⁹³ ²⁹⁴ ²⁹⁵ ²⁹⁶ ²⁹⁷ ²⁹⁸ ²⁹⁹ ³⁰⁰ ³⁰¹ ³⁰² ³⁰³ ³⁰⁴ ³⁰⁵ ³⁰⁶ ³⁰⁷ ³⁰⁸ ³⁰⁹ ³¹⁰ ³¹¹ ³¹² ³¹³ ³¹⁴ ³¹⁵ ³¹⁶ ³¹⁷ ³¹⁸ ³¹⁹ ³²⁰ ³²¹ ³²² ³²³ ³²⁴ ³²⁵ ³²⁶ ³²⁷ ³²⁸ ³²⁹ ³³⁰ ³³¹ ³³² ³³³ ³³⁴ ³³⁵ ³³⁶ ³³⁷ ³³⁸ ³³⁹ ³⁴⁰ ³⁴¹ ³⁴² ³⁴³ ³⁴⁴ ³⁴⁵ ³⁴⁶ ³⁴⁷ ³⁴⁸ ³⁴⁹ ³⁵⁰ ³⁵¹ ³⁵² ³⁵³ ³⁵⁴ ³⁵⁵ ³⁵⁶ ³⁵⁷ ³⁵⁸ ³⁵⁹ ³⁶⁰ ³⁶¹ ³⁶² ³⁶³ ³⁶⁴ ³⁶⁵ ³⁶⁶ ³⁶⁷ ³⁶⁸ ³⁶⁹ ³⁷⁰ ³⁷¹ ³⁷² ³⁷³ ³⁷⁴ ³⁷⁵ ³⁷⁶ ³⁷⁷ ³⁷⁸ ³⁷⁹ ³⁸⁰ ³⁸¹ ³⁸² ³⁸³ ³⁸⁴ ³⁸⁵ ³⁸⁶ ³⁸⁷ ³⁸⁸ ³⁸⁹ ³⁹⁰ ³⁹¹ ³⁹² ³⁹³ ³⁹⁴ ³⁹⁵ ³⁹⁶ ³⁹⁷ ³⁹⁸ ³⁹⁹ ⁴⁰⁰ ⁴⁰¹ ⁴⁰² ⁴⁰³ ⁴⁰⁴ ⁴⁰⁵ ⁴⁰⁶ ⁴⁰⁷ ⁴⁰⁸ ⁴⁰⁹ ⁴¹⁰ ⁴¹¹ ⁴¹² ⁴¹³ ⁴¹⁴ ⁴¹⁵ ⁴¹⁶ ⁴¹⁷ ⁴¹⁸ ⁴¹⁹ ⁴²⁰ ⁴²¹ ⁴²² ⁴²³ ⁴²⁴ ⁴²⁵ ⁴²⁶ ⁴²⁷ ⁴²⁸ ⁴²⁹ ⁴³⁰ ⁴³¹ ⁴³² ⁴³³ ⁴³⁴ ⁴³⁵ ⁴³⁶ ⁴³⁷ ⁴³⁸ ⁴³⁹ ⁴⁴⁰ ⁴⁴¹ ⁴⁴² ⁴⁴³ ⁴⁴⁴ ⁴⁴⁵ ⁴⁴⁶ ⁴⁴⁷ ⁴⁴⁸ ⁴⁴⁹ ⁴⁵⁰ ⁴⁵¹ ⁴⁵² ⁴⁵³ ⁴⁵⁴ ⁴⁵⁵ ⁴⁵⁶ ⁴⁵⁷ ⁴⁵⁸ ⁴⁵⁹ ⁴⁶⁰ ⁴⁶¹ ⁴⁶² ⁴⁶³ ⁴⁶⁴ ⁴⁶⁵ ⁴⁶⁶ ⁴⁶⁷ ⁴⁶⁸ ⁴⁶⁹ ⁴⁷⁰ ⁴⁷¹ ⁴⁷² ⁴⁷³ ⁴⁷⁴ ⁴⁷⁵ ⁴⁷⁶ ⁴⁷⁷ ⁴⁷⁸ ⁴⁷⁹ ⁴⁸⁰ ⁴⁸¹ ⁴⁸² ⁴⁸³ ⁴⁸⁴ ⁴⁸⁵ ⁴⁸⁶ ⁴⁸⁷ ⁴⁸⁸ ⁴⁸⁹ ⁴⁹⁰ ⁴⁹¹ ⁴⁹² ⁴⁹³ ⁴⁹⁴ ⁴⁹⁵ ⁴⁹⁶ ⁴⁹⁷ ⁴⁹⁸ ⁴⁹⁹ ⁵⁰⁰ ⁵⁰¹ ⁵⁰² ⁵⁰³ ⁵⁰⁴ ⁵⁰⁵ ⁵⁰⁶ ⁵⁰⁷ ⁵⁰⁸ ⁵⁰⁹ ⁵¹⁰ ⁵¹¹ ⁵¹² ⁵¹³ ⁵¹⁴ ⁵¹⁵ ⁵¹⁶ ⁵¹⁷ ⁵¹⁸ ⁵¹⁹ ⁵²⁰ ⁵²¹ ⁵²² ⁵²³ ⁵²⁴ ⁵²⁵ ⁵²⁶ ⁵²⁷ ⁵²⁸ ⁵²⁹ ⁵³⁰ ⁵³¹ ⁵³² ⁵³³ ⁵³⁴ ⁵³⁵ ⁵³⁶ ⁵³⁷ ⁵³⁸ ⁵³⁹ ⁵⁴⁰ ⁵⁴¹ ⁵⁴² ⁵⁴³ ⁵⁴⁴ ⁵⁴⁵ ⁵⁴⁶ ⁵⁴⁷ ⁵⁴⁸ ⁵⁴⁹ ⁵⁵⁰ ⁵⁵¹ ⁵⁵² ⁵⁵³ ⁵⁵⁴ ⁵⁵⁵ ⁵⁵⁶ ⁵⁵⁷ ⁵⁵⁸ ⁵⁵⁹ ⁵⁶⁰ ⁵⁶¹ ⁵⁶² ⁵⁶³ ⁵⁶⁴ ⁵⁶⁵ ⁵⁶⁶ ⁵⁶⁷ ⁵⁶⁸ ⁵⁶⁹ ⁵⁷⁰ ⁵⁷¹ ⁵⁷² ⁵⁷³ ⁵⁷⁴ ⁵⁷⁵ ⁵⁷⁶ ⁵⁷⁷ ⁵⁷⁸ ⁵⁷⁹ ⁵⁸⁰ ⁵⁸¹ ⁵⁸² ⁵⁸³ ⁵⁸⁴ ⁵⁸⁵ ⁵⁸⁶ ⁵⁸⁷ ⁵⁸⁸ ⁵⁸⁹ ⁵⁹⁰ ⁵⁹¹ ⁵⁹² ⁵⁹³ ⁵⁹⁴ ⁵⁹⁵ ⁵⁹⁶ ⁵⁹⁷ ⁵⁹⁸ ⁵⁹⁹ ⁶⁰⁰ ⁶⁰¹ ⁶⁰² ⁶⁰³ ⁶⁰⁴ ⁶⁰⁵ ⁶⁰⁶ ⁶⁰⁷ ⁶⁰⁸ ⁶⁰⁹ ⁶¹⁰ ⁶¹¹ ⁶¹² ⁶¹³ ⁶¹⁴ ⁶¹⁵ ⁶¹⁶ ⁶¹⁷ ⁶¹⁸ ⁶¹⁹ ⁶²⁰ ⁶²¹ ⁶²² ⁶²³ ⁶²⁴ ⁶²⁵ ⁶²⁶ ⁶²⁷ ⁶²⁸ ⁶²⁹ ⁶³⁰ ⁶³¹ ⁶³² ⁶³³ ⁶³⁴ ⁶³⁵ ⁶³⁶ ⁶³⁷ ⁶³⁸ ⁶³⁹ ⁶⁴⁰ ⁶⁴¹ ⁶⁴² ⁶⁴³ ⁶⁴⁴ ⁶⁴⁵ ⁶⁴⁶ ⁶⁴⁷ ⁶⁴⁸ ⁶⁴⁹ ⁶⁵⁰ ⁶⁵¹ ⁶⁵² ⁶⁵³ ⁶⁵⁴ ⁶⁵⁵ ⁶⁵⁶ ⁶⁵⁷ ⁶⁵⁸ ⁶⁵⁹ ⁶⁶⁰ ⁶⁶¹ ⁶⁶² ⁶⁶³ ⁶⁶⁴ ⁶⁶⁵ ⁶⁶⁶ ⁶⁶⁷ ⁶⁶⁸ ⁶⁶⁹ ⁶⁷⁰ ⁶⁷¹ ⁶⁷² ⁶⁷³ ⁶⁷⁴ ⁶⁷⁵ ⁶⁷⁶ ⁶⁷⁷ ⁶⁷⁸ ⁶⁷⁹ ⁶⁸⁰ ⁶⁸¹ ⁶⁸² ⁶⁸³ ⁶⁸⁴ ⁶⁸⁵ ⁶⁸⁶ ⁶⁸⁷ ⁶⁸⁸ ⁶⁸⁹ ⁶⁹⁰ ⁶⁹¹ ⁶⁹² ⁶⁹³ ⁶⁹⁴ ⁶⁹⁵ ⁶⁹⁶ ⁶⁹⁷ ⁶⁹⁸ ⁶⁹⁹ ⁷⁰⁰ ⁷⁰¹ ⁷⁰² ⁷⁰³ ⁷⁰⁴ ⁷⁰⁵ ⁷⁰⁶ ⁷⁰⁷ ⁷⁰⁸ ⁷⁰⁹ ⁷¹⁰ ⁷¹¹ ⁷¹² ⁷¹³ ⁷¹⁴ ⁷¹⁵ ⁷¹⁶ ⁷¹⁷ ⁷¹⁸ ⁷¹⁹ ⁷²⁰ ⁷²¹ ⁷²² ⁷²³ ⁷²⁴ ⁷²⁵ ⁷²⁶ ⁷²⁷ ⁷²⁸ ⁷²⁹ ⁷³⁰ ⁷³¹ ⁷³² ⁷³³ ⁷³⁴ ⁷³⁵ ⁷³⁶ ⁷³⁷ ⁷³⁸ ⁷³⁹ ⁷⁴⁰ ⁷⁴¹ ⁷⁴² ⁷⁴³ ⁷⁴⁴ ⁷⁴⁵ ⁷⁴⁶ ⁷⁴⁷ ⁷⁴⁸ ⁷⁴⁹ ⁷⁵⁰ ⁷⁵¹ ⁷⁵² ⁷⁵³ ⁷⁵⁴ ⁷⁵⁵ ⁷⁵⁶ ⁷⁵⁷ ⁷⁵⁸ ⁷⁵⁹ ⁷⁶⁰ ⁷⁶¹ ⁷⁶² ⁷⁶³ ⁷⁶⁴ ⁷⁶⁵ ⁷⁶⁶ ⁷⁶⁷ ⁷⁶⁸ ⁷⁶⁹ ⁷⁷⁰ ⁷⁷¹ ⁷⁷² ⁷⁷³ ⁷⁷⁴ ⁷⁷⁵ ⁷⁷⁶ ⁷⁷⁷ ⁷⁷⁸ ⁷⁷⁹ ⁷⁸⁰ ⁷⁸¹ ⁷⁸² ⁷⁸³ ⁷⁸⁴ ⁷⁸⁵ ⁷⁸⁶ ⁷⁸⁷ ⁷⁸⁸ ⁷⁸⁹ ⁷⁹⁰ ⁷⁹¹ ⁷⁹² ⁷⁹³ ⁷⁹⁴ ⁷⁹⁵ ⁷⁹⁶ ⁷⁹⁷ ⁷⁹⁸ ⁷⁹⁹ ⁸⁰⁰ ⁸⁰¹ ⁸⁰² ⁸⁰³ ⁸⁰⁴ ⁸⁰⁵ ⁸⁰⁶ ⁸⁰⁷ ⁸⁰⁸ ⁸⁰⁹ ⁸¹⁰ ⁸¹¹ ⁸¹² ⁸¹³ ⁸¹⁴ ⁸¹⁵ ⁸¹⁶ ⁸¹⁷ ⁸¹⁸ ⁸¹⁹ ⁸²⁰ ⁸²¹ ⁸²² ⁸²³ ⁸²⁴ ⁸²⁵ ⁸²⁶ ⁸²⁷ ⁸²⁸ ⁸²⁹ ⁸³⁰ ⁸³¹ ⁸³² ⁸³³ ⁸³⁴ ⁸³⁵ ⁸³⁶ ⁸³⁷ ⁸³⁸ ⁸³⁹ ⁸⁴⁰ ⁸⁴¹ ⁸⁴² ⁸⁴³ ⁸⁴⁴ ⁸⁴⁵ ⁸⁴⁶ ⁸⁴⁷ ⁸⁴⁸ ⁸⁴⁹ ⁸⁵⁰ ⁸⁵¹ ⁸⁵² ⁸⁵³ ⁸⁵⁴ ⁸⁵⁵ ⁸⁵⁶ ⁸⁵⁷ ⁸⁵⁸ ⁸⁵⁹ ⁸⁶⁰ ⁸⁶¹ ⁸⁶² ⁸⁶³ ⁸⁶⁴ ⁸⁶⁵ ⁸⁶⁶ ⁸⁶⁷ ⁸⁶⁸ ⁸⁶⁹ ⁸⁷⁰ ⁸⁷¹ ⁸⁷² ⁸⁷³ ⁸⁷⁴ ⁸⁷⁵ ⁸⁷⁶ ⁸⁷⁷ ⁸⁷⁸ ⁸⁷⁹ ⁸⁸⁰ ⁸⁸¹ ⁸⁸² ⁸⁸³ ⁸⁸⁴ ⁸⁸⁵ ⁸⁸⁶ ⁸⁸⁷ ⁸⁸⁸ ⁸⁸⁹ ⁸⁹⁰ ⁸⁹¹ ⁸⁹² ⁸⁹³ ⁸⁹⁴ ⁸⁹⁵ ⁸⁹⁶ ⁸⁹⁷ ⁸⁹⁸ ⁸⁹⁹ ⁹⁰⁰ ⁹⁰¹ ⁹⁰² ⁹⁰³ ⁹⁰⁴ ⁹⁰⁵ ⁹⁰⁶ ⁹⁰⁷ ⁹⁰⁸ ⁹⁰⁹ ⁹¹⁰ ⁹¹¹ ⁹¹² ⁹¹³ ⁹¹⁴ ⁹¹⁵ ⁹¹⁶ ⁹¹⁷ ⁹¹⁸ ⁹¹⁹ ⁹²⁰ ⁹²¹ ⁹²² ⁹²³ ⁹²⁴ ⁹²⁵ ⁹²⁶ ⁹²⁷ ⁹²⁸ ⁹²⁹ ⁹³⁰ ⁹³¹ ⁹³² ⁹³³ ⁹³⁴ ⁹³⁵ ⁹³⁶ ⁹³⁷ ⁹³⁸ ⁹³⁹ ⁹⁴⁰ ⁹⁴¹ ⁹⁴² ⁹⁴³ ⁹⁴⁴ ⁹⁴⁵ ⁹⁴⁶ ⁹⁴⁷ ⁹⁴⁸ ⁹⁴⁹ ⁹⁵⁰ ⁹⁵¹ ⁹⁵² ⁹⁵³ ⁹⁵⁴ ⁹⁵⁵ ⁹⁵⁶ ⁹⁵⁷ ⁹⁵⁸ ⁹⁵⁹ ⁹⁶⁰ ⁹⁶¹ ⁹⁶² ⁹⁶³ ⁹⁶⁴ ⁹⁶⁵ ⁹⁶⁶ ⁹⁶⁷ ⁹⁶⁸ ⁹⁶⁹ ⁹⁷⁰ ⁹⁷¹ ⁹⁷² ⁹⁷³ ⁹⁷⁴ ⁹⁷⁵ ⁹⁷⁶ ⁹⁷⁷ ⁹⁷⁸ ⁹⁷⁹ ⁹⁸⁰ ⁹⁸¹ ⁹⁸² ⁹⁸³ ⁹⁸⁴ ⁹⁸⁵ ⁹⁸⁶ ⁹⁸⁷ ⁹⁸⁸ ⁹⁸⁹ ⁹⁹⁰ ⁹⁹¹ ⁹⁹² ⁹⁹³ ⁹⁹⁴ ⁹⁹⁵ ⁹⁹⁶ ⁹⁹⁷ ⁹⁹⁸ ⁹⁹⁹ ¹⁰⁰⁰ ¹⁰⁰¹ ¹⁰⁰² ¹⁰⁰³ ¹⁰⁰⁴ ¹⁰⁰⁵ ¹⁰⁰⁶ ¹⁰⁰⁷ ¹⁰⁰⁸ ¹⁰⁰⁹ ¹⁰¹⁰ ¹⁰¹¹ ¹⁰¹² ¹⁰¹³ ¹⁰¹⁴ ¹⁰¹⁵ ¹⁰¹⁶ ¹⁰¹⁷ ¹⁰¹⁸ ¹⁰¹⁹ ¹⁰²⁰ ¹⁰²¹ ¹⁰²² ¹⁰²³ ¹⁰²⁴ ¹⁰²⁵ ¹⁰²⁶ ¹⁰²⁷ ¹⁰²⁸ ¹⁰²⁹ ¹⁰³⁰ ¹⁰³¹ ¹⁰³² ¹⁰³³ ¹⁰³⁴ ¹⁰³⁵ ¹⁰³⁶ ¹⁰³⁷ ¹⁰³⁸ ¹⁰³⁹ ¹⁰⁴⁰ ¹⁰⁴¹ ¹⁰⁴² ¹⁰⁴³ ¹⁰⁴⁴ ¹⁰⁴⁵ ¹⁰⁴⁶ ¹⁰⁴⁷ ¹⁰⁴⁸ ¹⁰⁴⁹ ¹⁰⁵⁰ ¹⁰⁵¹ ¹⁰⁵² ¹⁰⁵³ ¹⁰⁵⁴ ¹⁰⁵⁵ ¹⁰⁵⁶ ¹⁰⁵⁷ ¹⁰⁵⁸ ¹⁰⁵⁹ ¹⁰⁶⁰ ¹⁰⁶¹ ¹⁰⁶² ¹⁰⁶³ ¹⁰⁶⁴ ¹⁰⁶⁵ ¹⁰⁶⁶ ¹⁰⁶⁷ ¹⁰⁶⁸ ¹⁰⁶⁹ ¹⁰⁷⁰ ¹⁰⁷¹ ¹⁰⁷² ¹⁰⁷³ ¹⁰⁷⁴ ¹⁰⁷⁵ ¹⁰⁷⁶ ¹⁰⁷⁷ ¹⁰⁷⁸ ¹⁰⁷⁹ ¹⁰⁸⁰ ¹⁰⁸¹ ¹⁰⁸² ¹⁰⁸³ ¹⁰⁸⁴ ¹⁰⁸⁵ ¹⁰⁸⁶ ¹⁰⁸⁷ ¹⁰⁸⁸ ¹⁰⁸⁹ ¹⁰⁹⁰ ¹⁰⁹¹ ¹⁰⁹² ¹⁰⁹³ ¹⁰⁹⁴ ¹⁰⁹⁵ ¹⁰⁹⁶ ¹⁰⁹⁷ ¹⁰⁹⁸ ¹⁰⁹⁹ ¹¹⁰⁰ ¹¹⁰¹ ¹¹⁰² ¹¹⁰³ ¹¹⁰⁴ ¹¹⁰⁵ ¹¹⁰⁶ ¹¹⁰⁷ ¹¹⁰⁸ ¹¹⁰⁹ ¹¹¹⁰ ¹¹¹¹ ¹¹¹² ¹¹¹³ ¹¹¹⁴ ¹¹¹⁵ ¹¹¹⁶ ¹¹¹⁷ ¹¹¹⁸ ¹¹¹⁹ ¹¹²⁰ ¹¹²¹ ¹¹²² ¹¹²³ ¹¹²⁴ ¹¹²⁵ ¹¹²⁶ ¹¹²⁷ ¹¹²⁸ ¹¹²⁹ ¹¹³⁰ ¹¹³¹ ¹¹³² ¹¹³³ ¹¹³⁴ ¹¹³⁵ ¹¹³⁶ ¹¹³⁷ ¹¹³⁸ ¹¹³⁹ ¹¹⁴⁰ ¹¹⁴¹ ¹¹⁴² ¹¹⁴³ ¹¹⁴⁴ ¹¹⁴⁵ ¹¹⁴⁶ ¹¹⁴⁷ ¹¹⁴⁸ ¹¹⁴⁹ ¹¹⁵⁰ ¹¹⁵¹ ¹¹⁵² ¹¹⁵³ ¹¹⁵⁴ ¹¹⁵⁵ ¹¹⁵⁶ ¹¹⁵⁷ ¹¹⁵⁸ ¹¹⁵⁹ ¹¹⁶⁰ ¹¹⁶¹ ¹¹⁶² ¹¹⁶³ ¹¹⁶⁴ ¹¹⁶⁵ ¹¹⁶⁶ ¹¹⁶⁷ ¹¹⁶⁸ ¹¹⁶⁹ ¹¹⁷⁰ ¹¹⁷¹ ¹¹⁷² ¹¹⁷³ ¹¹⁷⁴ ¹¹⁷⁵ ¹¹⁷⁶ ¹¹⁷⁷ ¹¹⁷⁸ ¹¹⁷⁹ ¹¹⁸⁰ ¹¹⁸¹ ¹¹⁸² ¹¹⁸³ ¹¹⁸⁴ ¹¹⁸⁵ ¹¹⁸⁶ ¹¹⁸⁷ ¹¹⁸⁸ ¹¹⁸⁹ ¹¹⁹⁰ ¹¹⁹¹ ¹¹⁹² ¹¹⁹³ ¹¹⁹⁴ ¹¹⁹⁵ ¹¹⁹⁶ ¹¹⁹⁷ ¹¹⁹⁸ ¹¹⁹⁹ ¹²⁰⁰ ¹²⁰¹ ¹²⁰² ¹²⁰³ ¹²⁰⁴ ¹²⁰⁵ ¹²⁰⁶ ¹²⁰⁷ ¹²⁰⁸ ¹²⁰⁹ ¹²¹⁰ ¹²¹¹ ¹²¹² ¹²¹³ ¹²¹⁴ ¹²¹⁵ ¹²¹⁶ ¹²¹⁷ ¹²¹⁸ ¹²¹⁹ ¹²²⁰ ¹²²¹ ¹²²² ¹²²³ ¹²²⁴ ¹²²⁵ ¹²²⁶ ¹²²⁷ ¹²²⁸ ¹²²⁹ ¹²³⁰ ¹²³¹ ¹²³² ¹²³³ ¹²³⁴ ¹²³⁵ ¹²³⁶ ¹²³⁷ ¹²³⁸ ¹²³⁹ ¹²⁴⁰ ¹²⁴¹ ¹²⁴² ¹²⁴³ ¹²⁴⁴ ¹²⁴⁵ ¹²⁴⁶ ¹²⁴⁷ ¹²⁴⁸ ¹²⁴⁹ ¹²⁵⁰ ¹²⁵¹ ¹²⁵² ¹²⁵³ ¹²⁵⁴ ¹²⁵⁵ ¹²⁵⁶ ¹²⁵⁷ ¹²⁵⁸ ¹²⁵⁹ ¹²⁶⁰ ¹²⁶¹ ¹²⁶² ¹²⁶³ ¹²⁶⁴ ¹²⁶⁵ ¹²⁶⁶ ¹²⁶⁷ ¹²⁶⁸ ¹²⁶⁹ ¹²⁷⁰ ¹²⁷¹ ¹²⁷² ¹²⁷³ ¹²⁷⁴ ¹²⁷⁵ ¹²⁷⁶ ¹²⁷⁷ ¹²⁷⁸ ¹²⁷⁹ ¹²⁸⁰ ¹²⁸¹ ¹²⁸² ¹²⁸³ ¹²⁸⁴ ¹²⁸⁵ ¹²⁸⁶ ¹²⁸⁷ ¹²⁸⁸ ¹²⁸⁹ ¹²⁹⁰ ¹²⁹¹ ¹²⁹² ¹²⁹³ ¹²⁹⁴ ¹²⁹⁵ ¹²⁹⁶ ¹²⁹⁷ ¹²⁹⁸ ¹²⁹⁹ ¹³⁰⁰ ¹³⁰¹ ¹³⁰² ¹³⁰³ ¹³⁰⁴ ¹³⁰⁵ ¹³⁰⁶ ¹³⁰⁷ ¹³⁰⁸ ¹³⁰⁹ ¹³¹⁰ ¹³¹¹ ¹³¹² ¹³¹³ ¹³¹⁴ ¹³¹⁵ ¹³¹⁶ ¹³¹⁷ ¹³¹⁸ ¹³¹⁹ ¹³²⁰ ¹³²¹ ¹³²² ¹³²³ ¹³²⁴ ¹³²⁵ ¹³²⁶ ¹³²⁷ ¹³²⁸ ¹³²⁹ ¹³³⁰ ¹³³¹ ¹³³² ¹³³³ ¹³³⁴ ¹³³⁵ ¹³³⁶ ¹³³

./matrix.txt

1	9.157941937446594238e-01	7.716442346572875977e-01	4.434497654438018799e-01	3.627760708332061768e-01	2.912405514717102051e-01
2	9.346895813941955566e-01	5.742390751838684082e-01	3.731313645839691162e-01	3.196475505828857422e-01	3.455076485872268677e-01
3	9.494240880012512207e-01	8.849776983261108398e-01	2.346189171075820923e-01	2.394588440656661987e-01	9.092011451721191406e-01
4	9.333508014678955078e-01	8.457987904548645020e-01	4.310811460018157959e-01	1.747653633356094360e-01	1.502759903669357300e-01
5	9.609482288360595703e-01	3.331715166568756104e-01	3.583630323410034180e-01	2.592278420925140381e-01	4.298384189605712891e-01
6	9.792612791061401367e-01	9.008772969245910645e-01	2.746424674987792969e-01	2.828238904476165771e-01	1.455076485872268677e-01
7	9.112978577613830566e-01	8.600413799285888672e-01	3.737630546092987061e-01	2.036121338605880737e-01	1.541802823543548584e-01
8	9.571560025215148926e-01	8.606715202331542969e-01	2.630991935729980469e-01	2.160550951957702637e-01	9.503287672996520996e-01
9	9.323833584785461426e-01	8.171402812004089355e-01	4.377277791500091553e-01	1.502759903669357300e-01	1.237284064292907715e-01
10	9.356079697608947754e-01	7.851068377494812012e-01	5.012405514717102051e-01	1.455076485872268677e-01	1.562172293663024902e-01
11	9.092011451721191406e-01	7.483353614807128906e-01	4.298384189605712891e-01	2.541802823543548584e-01	1.461185932159423828e-01
12	9.503287672996520996e-01	8.873134255409240723e-01	2.655168473720550537e-01	2.211218476295471191e-01	1.467664361000061035e-01
13	9.237284064292907715e-01	8.363176584243774414e-01	3.627101480960845947e-01	2.365967631340026855e-01	1.397199749946594238e-01
14	9.562172293663024902e-01	9.194136857986450195e-01	3.819596767425537109e-01	3.117111623287200928e-01	1.222379326820373535e-01
15	9.461185932159423828e-01	8.484295606613159180e-01	3.903456628322601318e-01	1.668368875980377197e-01	1.418539404869079590e-01
16	9.467664361000061035e-01	8.682620525360107422e-01	3.137815594673156738e-01	1.826369911432266235e-01	8.906930685043334961e-01
17	9.397199749946594238e-01	8.609640002250671387e-01	3.499407768249511719e-01	1.618804782629013062e-01	1.549255132675170898e-01
18	9.222379326820373535e-01	8.876875042915344238e-01	3.556989133358001709e-01	3.479544818401336670e-01	9.364917278289794922e-01
19	9.418539404869079590e-01	8.918866515159606934e-01	2.337521761655807495e-01	2.460925579071044922e-01	9.408168196678161621e-01
20	8.906930685043334961e-01	8.144904375076293945e-01	4.380804598331451416e-01	5.200685262680053711e-01	9.318765997886657715e-01
21	8.549255132675170898e-01	7.775652408599853516e-01	2.998122274875640869e-01	4.507026672363281250e-01	9.611908197402954102e-01
22	9.364917278289794922e-01	8.836621046066284180e-01	4.243750274181365967e-01	2.403212934732437134e-01	9.418456554412841797e-01
23	9.408168196678161621e-01	4.739229083061218262e-01	3.617838919162750244e-01	2.829778790473937988e-01	9.144946336746215820e-01
24	9.318765997886657715e-01	7.781792879104614258e-01	4.771032333374023438e-01	1.843434870243072510e-01	9.463409185409545898e-01
25	9.611908197402954102e-01	7.101613283157348633e-01	4.384511113166809082e-01	2.055199444293975830e-01	9.570783376693725586e-01
26	9.418456554412841797e-01	7.011284828186035156e-01	4.341177344322204590e-01	3.878928422927856445e-01	9.442527294158935547e-01
27	9.144946336746215820e-01	3.438472747802734375e-01	4.719765782356262207e-01	2.633934617042541504e-01	9.472667574882507324e-01
28	9.463409185409545898e-01	3.462429642677307129e-01	3.763888478279113770e-01	2.532341480255126953e-01	9.214563369750976562e-01
29	9.570783376693725586e-01	4.388708770275115967e-01	3.545166850090026855e-01	2.836700975894927979e-01	9.513333439826965332e-01
30	9.442527294158935547e-01	4.111616313457489014e-01	3.773801922798156738e-01	3.202395141124725342e-01	
31	9.472667574882507324e-01	2.990520000457763672e-01	3.841416537761688232e-01	2.760661840438842773e-01	
32	9.214563369750976562e-01	4.416494965553283691e-01	3.559406995773315430e-01	4.304027259349822998e-01	
33	9.513333439826965332e-01	2.563325762748718262e-01	3.883069455623626709e-01	2.680478692054748535e-01	


```
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)
```

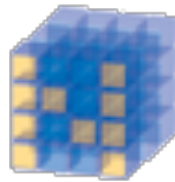
```
# shapes
print('Rows: {} - Cols: {}'.format(len(matrix), len(matrix[0])))
```

```
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, converters=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, usecols=None, names=None, excludelist=None, deletechars=None, replace_space='_', autostrip=False, case_sensitive=True, defaultfmt='f%i', unpack=None, usemask=False, loose=True, invalid_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
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
g/data/images/datasets_new/sol/good/sol_L_g_b_002.jpg
```



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

```
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
g/data/images/datasets_new/sol/good/sol_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	FILE
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=True, dtype=None, engine=None, converters=None, true_values=None, false_values=None, skipinitialspace=False, skiprows=None, skipfooter=None, nrows=None, na_values=None, keep_default_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=False, chunksize=None, compression='infer', thousands=None, decimal=b'.', lineterminator=None, 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=False, 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											
id									device_id		
int									unicode		
1									E036D39ADE70A12D		
2									E036D39ADE70A12D		
3									E036D39ADE70A12D		
4									E036D39ADE70A12D		
5									E036D39ADE70A12D		
6									E036D39ADE70A12D		
7									E036D39ADE70A12D		
8									E036D39ADE70A12D		
9									E036D39ADE70A12D		
10									E036D39ADE70A12D		
11									E036D39ADE70A12D		
12	9571b870-4b7f-48b0-8aeb-7eda91adfb08	2015-09-07 11:30:20.012000	Goat	A	23	30	2015-09-26 09:00:00	E036D39ADE70A12D			
13	9571b870-4b7f-48b0-8aeb-7eda91adfb08	2015-09-07 11:30:27.086000	Goat	A	23	30	2015-09-26 09:00:00	E036D39ADE70A12D			
14	6b057fdb-5ba8-4bf6-afce-63b9f28a9a81	2015-09-07 11:31:52.264000	Cow	A	23	28	2015-09-18 09:00:00	E036D39ADE70A12D			
15	6b057fdb-5ba8-4bf6-afce-63b9f28a9a81	2015-09-07 11:32:00.425000	Cow	A	23	28	2015-09-18 09:00:00	E036D39ADE70A12D			
16	6b057fdb-5ba8-4bf6-afce-63b9f28a9a81	2015-09-07 11:32:07.996000	Cow	A	23	28	2015-09-18 09:00:00	E036D39ADE70A12D			
17	f7fbd198-683e-4ead-8008-b6daacec5eca	2015-09-07 11:33:59.157000	Cow	A	23	28	2015-09-17 09:00:00	E036D39ADE70A12D			
18	f7fbd198-683e-4ead-8008-b6daacec5eca	2015-09-07 11:34:06.762000	Cow	A	23	28	2015-09-17 09:00:00	E036D39ADE70A12D			
19	f7fbd198-683e-4ead-8008-b6daacec5eca	2015-09-07 11:34:14.473000	Cow	A	23	28	2015-09-17 09:00:00	E036D39ADE70A12D			

```
!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

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

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('int32') # Bulk insert
dset[...] = np.arange(100)
```

```
type(dset)
```

```
h5py._hl.data
```

```
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

```
import tables as tb
```

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

Array
The files of the filesystem

CArray
Chunked arrays

EArray
Extendable arrays

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

```
dset2.name dset3.name
```

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

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

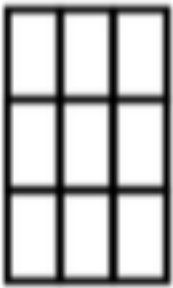
```
dset3 == dset3_f
```

```
True
```

Data Chunking

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

Data



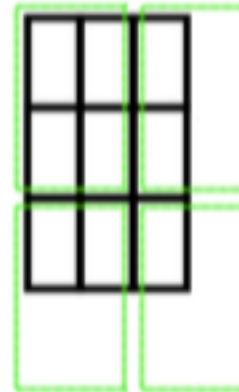
File

Contiguous
Dataset

metadata



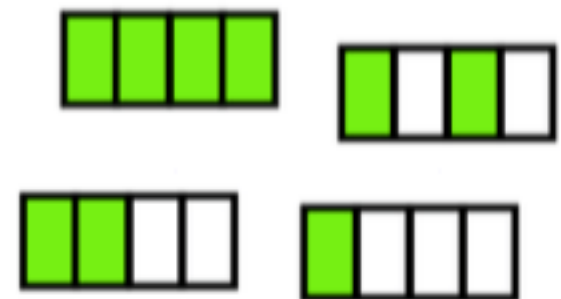
Data



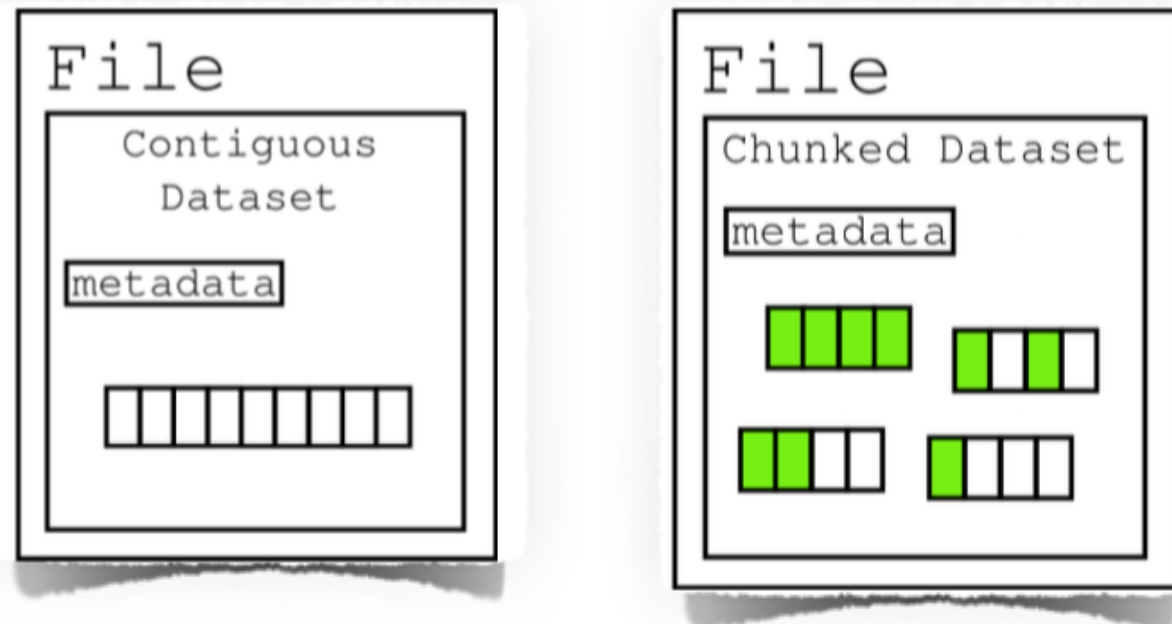
File

Chunked Dataset

metadata



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**

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()
```

O'REILLY®



Python and HDF5

UNLOCKING SCIENTIFIC DATA

Andrew Collette

Andrew Collette

Learn More

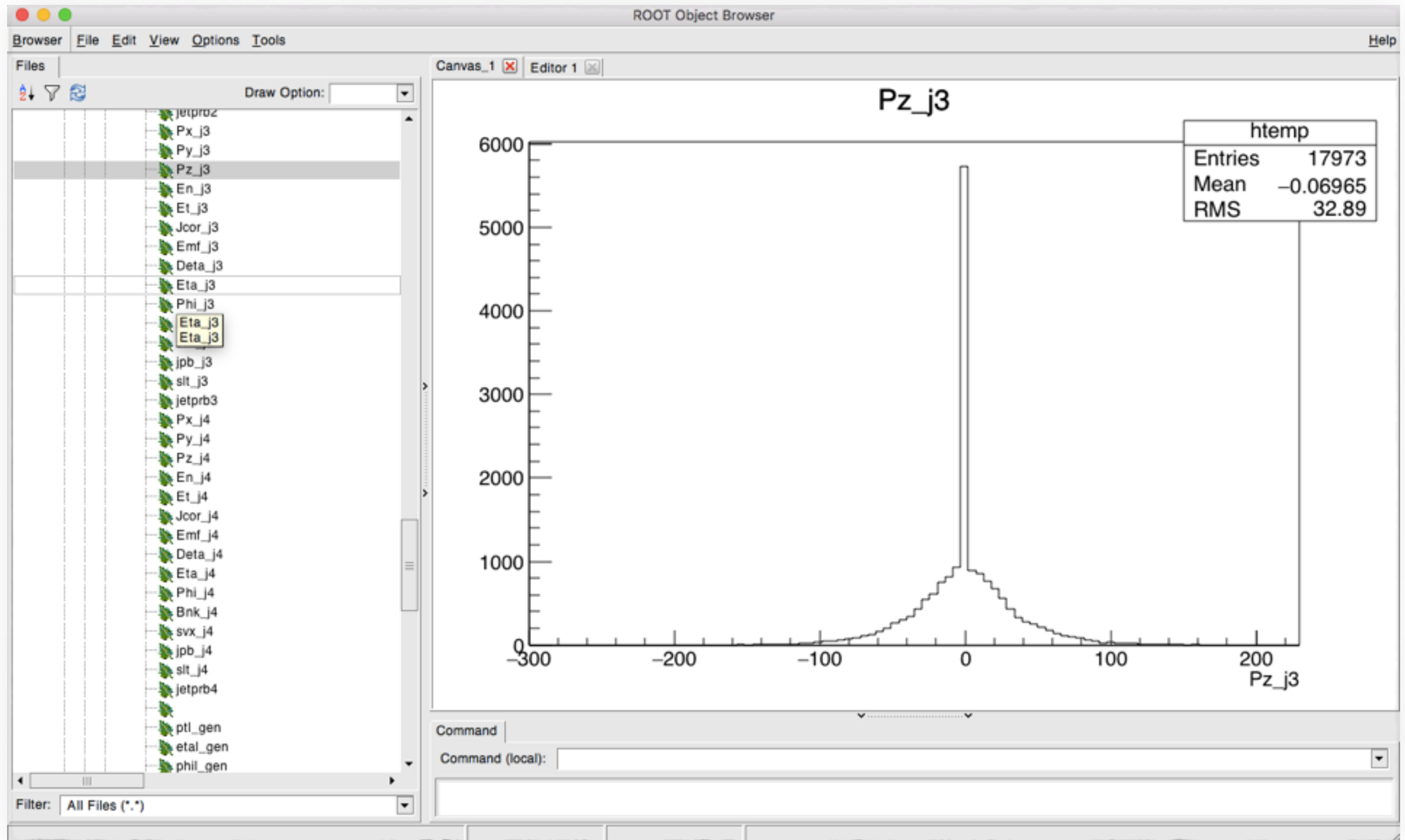
- How to migrate from PostgreSQL to HDF5 and live happily ever after by *Michele Simionato* @PyData Track on Friday



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 [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, minx, 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>

```
[{"ptitle": "Hello world!", "pname": "1-  
Page", "pname": "sample-page", "pstatus":  
draft"}, {"ptitle": "About us", "pname":  
us", "pname": "4-revision-v1", "pstatus":  
v1", "pstatus": "inherit"}, {"ptitle": "  
{"ptitle": "Introduction", "pname": "7-  
{"ptitle": "Achievements", "pname": "ac  
{"ptitle": "Achievements", "pname": "9-  
{"ptitle": "API's", "pname": "apis", "ps  
v1", "pstatus": "inherit"}, {"ptitle": "  
{"ptitle": "Apis", "pname": "17-revisio  
{"ptitle": "FDF", "pname": "fdf", "pstat  
v1", "pstatus": "inherit"}, {"ptitle": "  
portfolio", "pstatus": "publish"}, {"pt  
v1", "pstatus": "inherit"}, {"ptitle": "  
list", "pstatus": "publish"}, {"ptitle": "  
v1", "pstatus": "inherit"}, {"ptitle": "  
list", "pstatus": "publish"}, {"ptitle": "  
v1", "pstatus": "inherit"}, {"ptitle": "  
status", "pstatus": "publish"}, {"ptit:  
v1", "pstatus": "inherit"}, {"ptitle": "  
{"ptitle": "Contact Us", "pname": "29-1
```

3.

JSON

Data format



XML!

JSON!

Jupyter Notebook Data Format

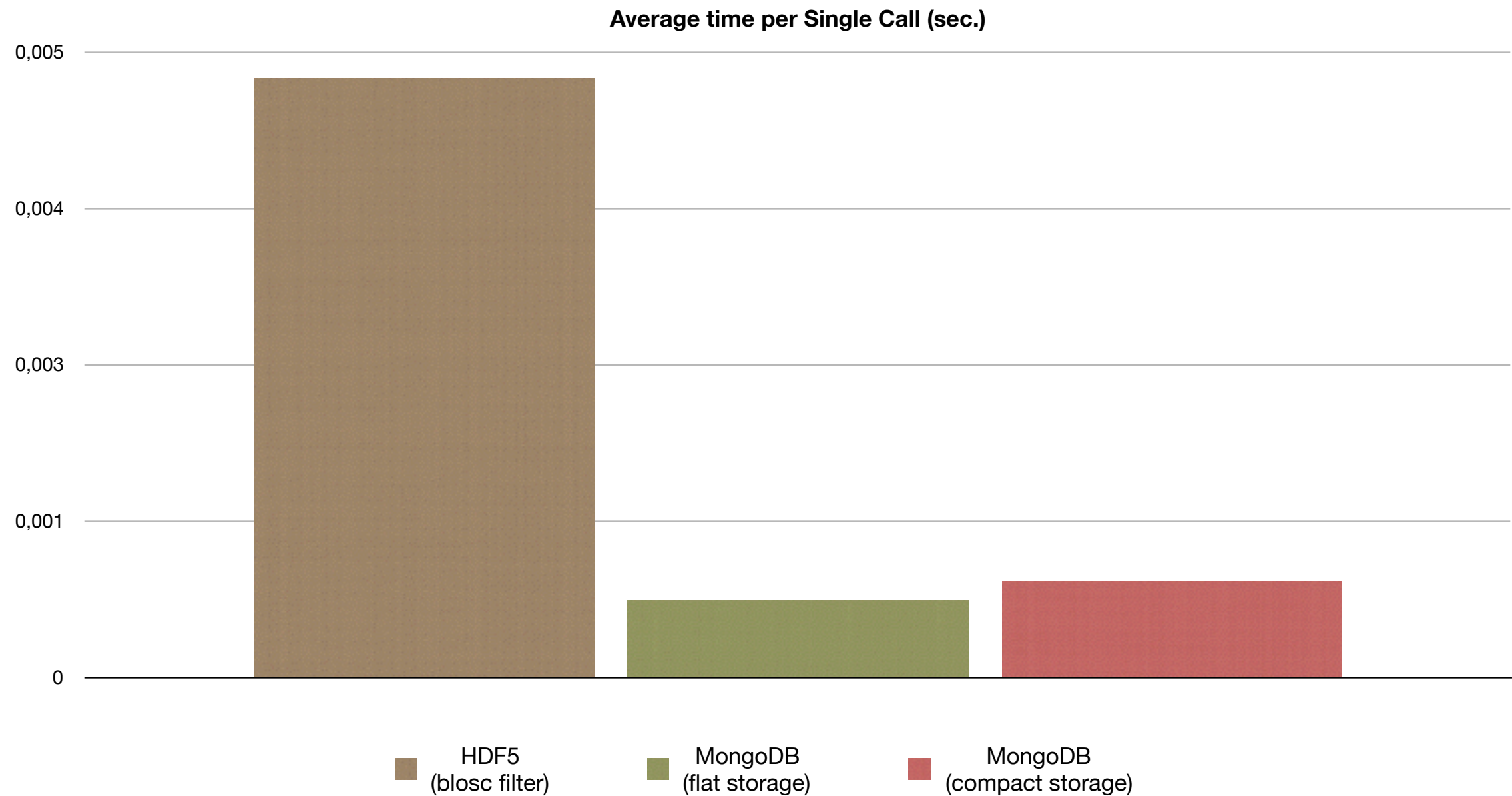
```
{
  "cells": [
    {
      "cell_type": "markdown",
      "metadata": {},
      "source": [
        "# Custom Magic Examples"
      ]
    },
    {},
    {},
    {},
    {},
  ]
}
```

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

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

HDF5 vs MongoDB

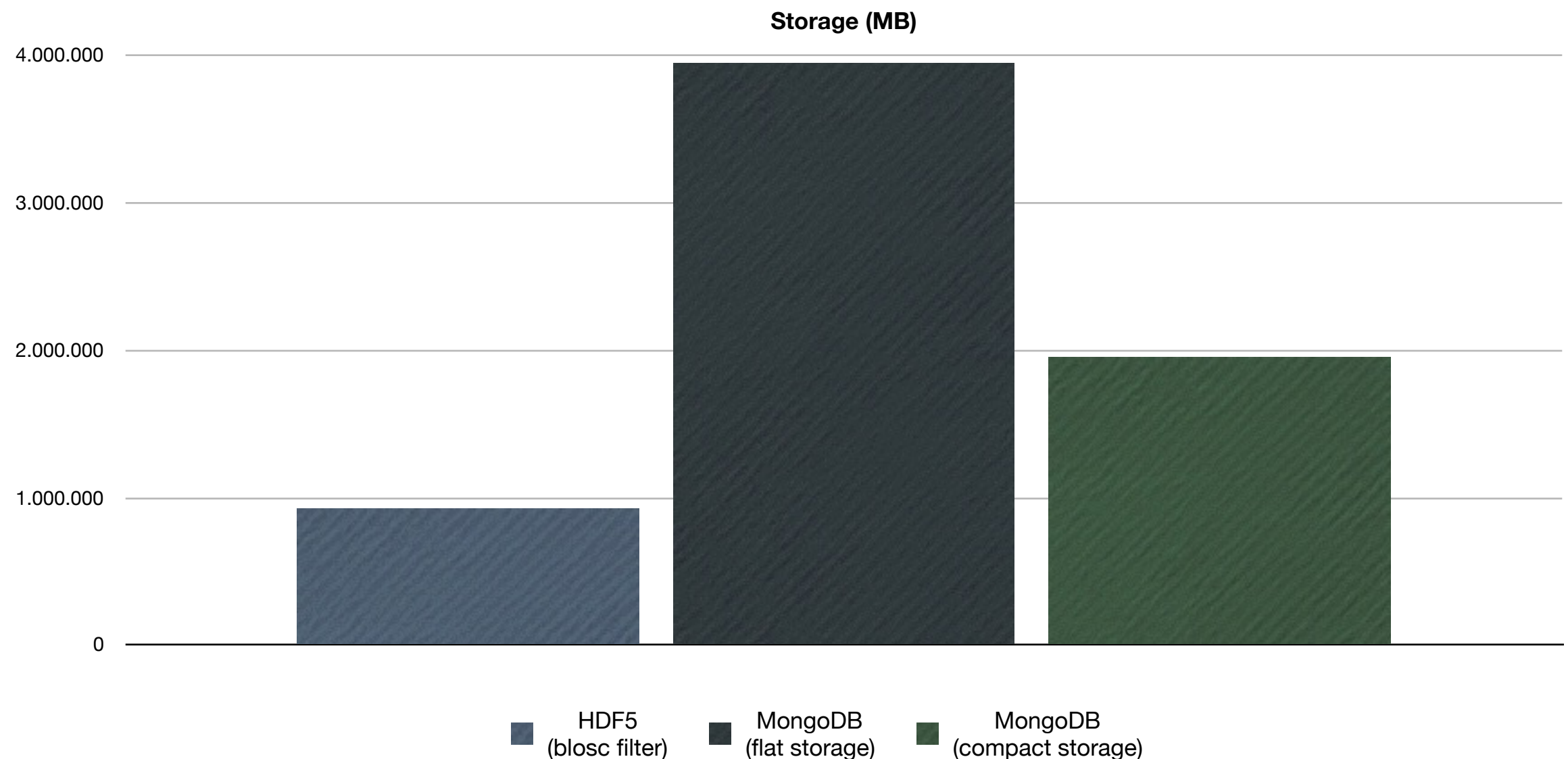
Total Number of Documents	Total Number of Entries	Total Number of Calls
100.000	8.755.882	319.970



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](http://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 sharded 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!


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

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

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

HDFS + CSV

Opening a Single File on the HDFS

```
import pandas as pd

with fs.open('/user/ubuntunyc/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 HDFSFilesystem
fs = HDFSFilesystem()

fs.ls('/user/ubuntunyc/', detail=False)

[u'/user/ubuntunyc/yellow_tripdata_2015-01.csv',
 u'/user/ubuntunyc/yellow_tripdata_2015-02.csv',
 u'/user/ubuntunyc/yellow_tripdata_2015-03.csv',
 u'/user/ubuntunyc/yellow_tripdata_2015-04.csv',
 u'/user/ubuntunyc/yellow_tripdata_2015-05.csv',
 u'/user/ubuntunyc/yellow_tripdata_2015-06.csv',
 u'/user/ubuntunyc/yellow_tripdata_2015-07.csv',
 u'/user/ubuntunyc/yellow_tripdata_2015-08.csv',
 u'/user/ubuntunyc/yellow_tripdata_2015-09.csv',
 u'/user/ubuntunyc/yellow_tripdata_2015-10.csv',
 u'/user/ubuntunyc/yellow_tripdata_2015-11.csv',
 u'/user/ubuntunyc/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/ubuntunyc/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
};
```

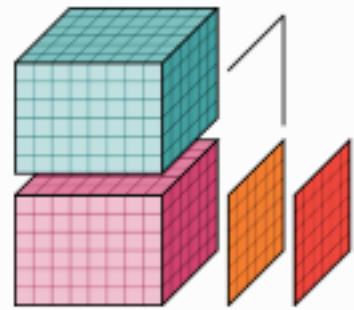
```
CREATE FUNCTION conf_matrix_stats
(c1 INT, c2 INT)
RETURNS TABLE
(accuracy FLOAT, precision FLOAT, sensitivity FLOAT, specificity FLOAT, f1 FLOAT)
LANGUAGE PYTHON
{
    result = dict()
    TP = c2[1]*1.00
    TN = c1[0]*1.00
    FN = c2[0]*1.00
    FP = c1[1]*1.00
    N = TP+TN+FP+FN
    accuracy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

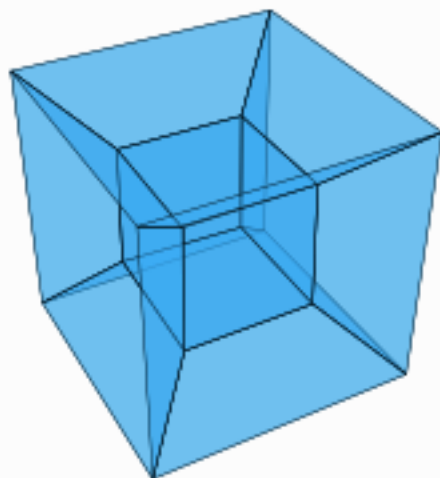


xarray

N-D labeled arrays and datasets in Python

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

<http://blaze.pydata.org>

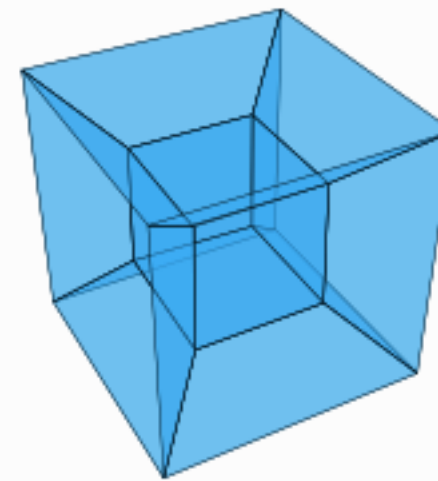


Blaze


```
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],
})

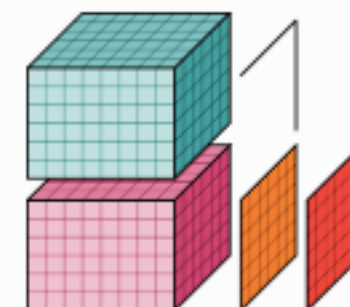
# put the `df` DataFrame into a Blaze Data object
df = data(df)
```



Blaze

Out-of-Core Processing

```
>>> from blaze import data, by
>>> t = data('sqlite:///iris::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
```

xarray

```
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