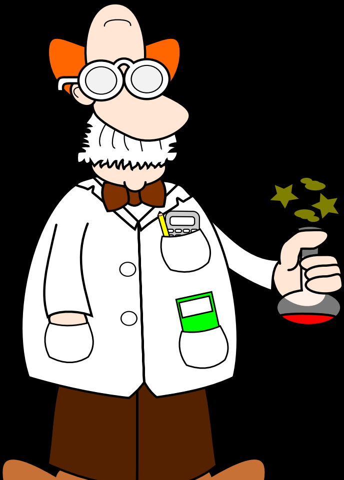
Scientist meets web dev:
how Python became the language of data

Gaël Varoquaux

Inria



Scientist meets web dev:
how Python became the language of data

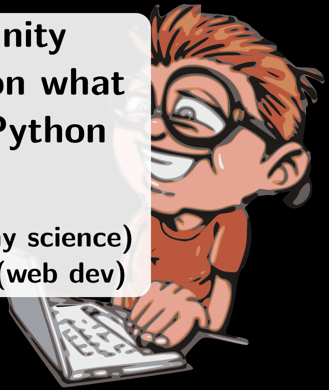
Gaël Varoquaux

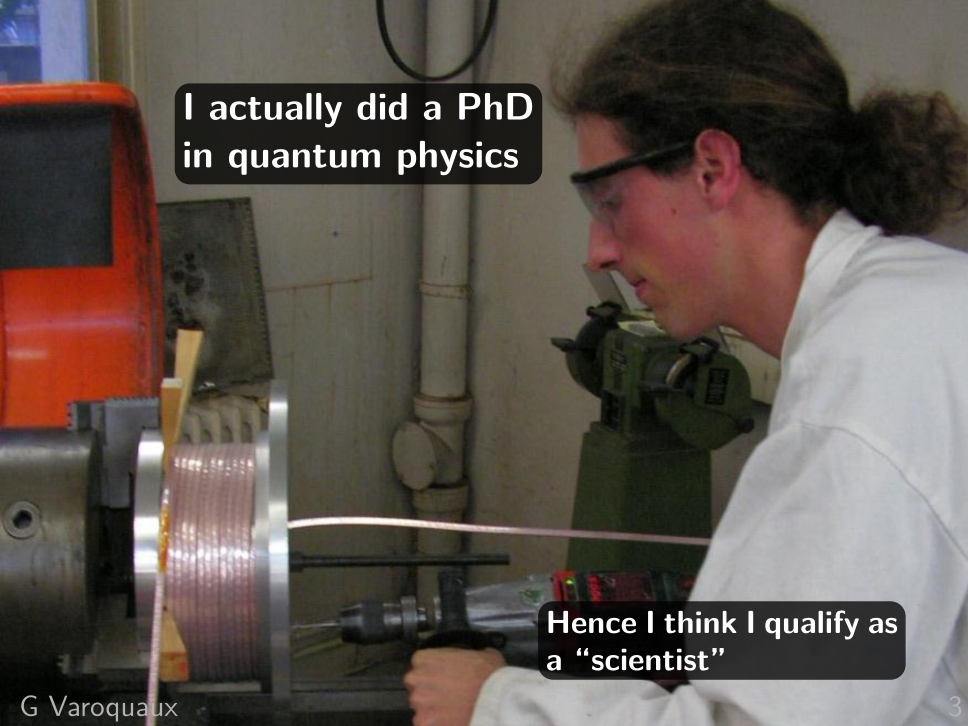
Inria

Very diverse community

This talk: a reflection on what we have in common, Python

I am talking about
things you don't understand (my science)
and things I don't understand (web dev)





I actually did a PhD in quantum physics

Hence I think I qualify as a “scientist”

I now do computer science for neuroscience

Try to link neural activity to thoughts and cognition

I now do computer science for neuroscience

Try to link neural activity to thoughts and cognition

We attack it as a machine learning problem

Python software: `nilearn`

On the way, we created
a machine-learning library:

scikit-learn

Huge success.

Cool.

Data science is THE thing.

Data science with Python is hot

Huge success.

Cool.

Data science is THE thing.

Python is the go-to language

How did it happen?

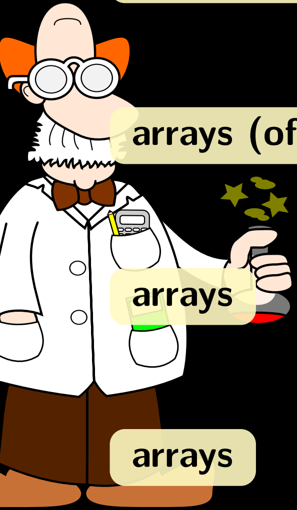
We built scikit-learn, others pandas, etc...,
but these were built on solid foundations

1 Scientists come from Jupiter

And web devs from Saturn?

And sysadmins from Neptune?

1 We're different



numbers (in arrays)

arrays (of numbers)

arrays

arrays

strings

databases

object-oriented programming

flow control

A bit of a culture gap

1 Let's do something together: sort EuroPython site

205 talks:

How OpenStack makes Python better (and vice-versa)

Introduction to aiohttp

So you think your Python startup is worth $10 million...

SQLAlchemy as the backbone of a Data Science company

Learn Python The Fun Way

Scaling Microservices with Crossbar.io

If you can read this you don't need glasses

Let's find some common topics with data science

1 Let's do something together: sort EuroPython site

205 talks:

How OpenStack makes Python better (and vice-versa)

Introduction to aiohttp

So you think your Python startup is worth $10 million...

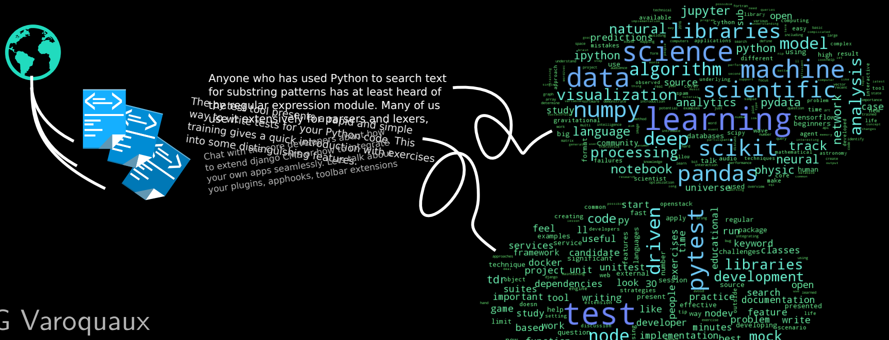
SQLAlchemy as the backbone of a Data Science company

Learn Python The Fun Way

Scaling Microservices with Crossbar.io

If you can read this you don't need glasses

Let's find some common topics with data science



205 talks:

How OpenStack makes Python better (and vice-versa)

Introduction to aiohttp

So you think your Python startup is worth $10 million...

SQLAlchemy as the backbone of a Data Science company

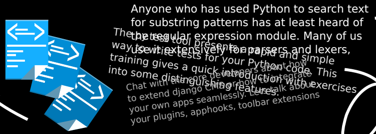
Learn Python The Fun Way

Scaling Microservices with Crossbar.io

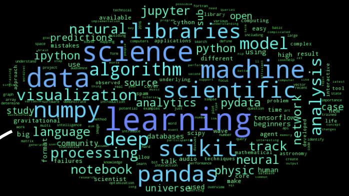
If you can read this you don't need glasses

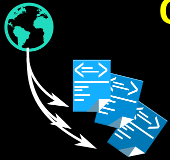
Let's find some common topics with data science

import urllib2, bs4



import sklearn, wordcloud



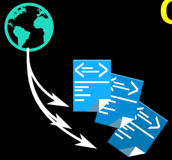


Crawl

- the schedule to get a list of titles and URLs
- talk pages to retrieve abstract and tags

`bs4`: beautiful soup, matchings on the DOM tree

1 Let's do something together: sort EuroPython site

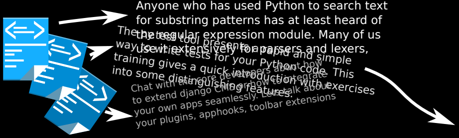


Crawl

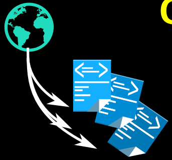
- the schedule to get a list of titles and URLs
- talk pages to retrieve abstract and tags

`bs4`: beautiful soup, matchings on the DOM tree

Vectorize



Term	Freq
a	20
can	10
code	4
is	14
module	3
profiling	2
performance	1
Python	9
the	18

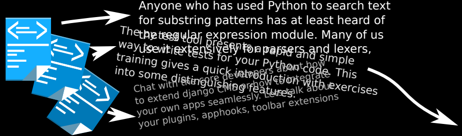


Crawl

- the schedule to get a list of titles and URLs
- talk pages to retrieve abstract and tags

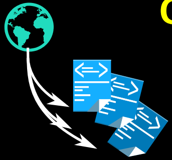
bs4: beautiful soup, matchings on the DOM tree

Vectorize



Term	Freq	All docs
a	20	1321
can	10	540
code	4	208
is	14	964
module	3	123
profiling	2	7
performance	1	6
Python	9	191
the	18	1450

1 Let's do something together: sort EuroPython site

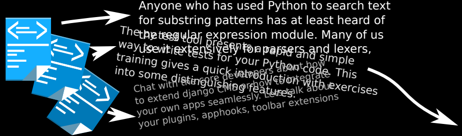


Crawl

- the schedule to get a list of titles and URLs
- talk pages to retrieve abstract and tags

bs4: beautiful soup, matchings on the DOM tree

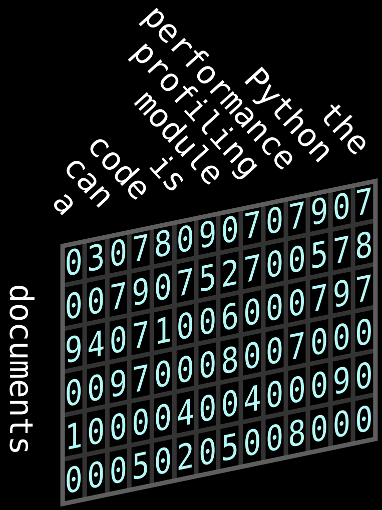
Vectorize



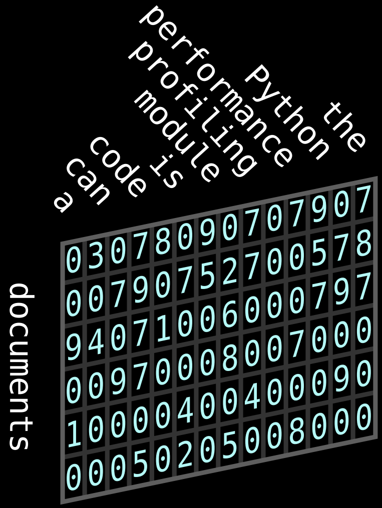
Term	Freq	All docs	Ratio
a	20	1321	.015
can	10	540	.018
code	4	208	.019
is	14	964	.014
module	3	123	.023
profiling	2	7	.286
performance	1	6	.167
Python	9	191	.047
the	18	1450	.012

TF-IDF in scikit-learn

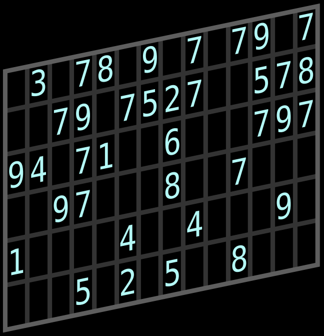
`sklearn.feature_extraction.text.TfidfVectorizer`



Term-document matrix

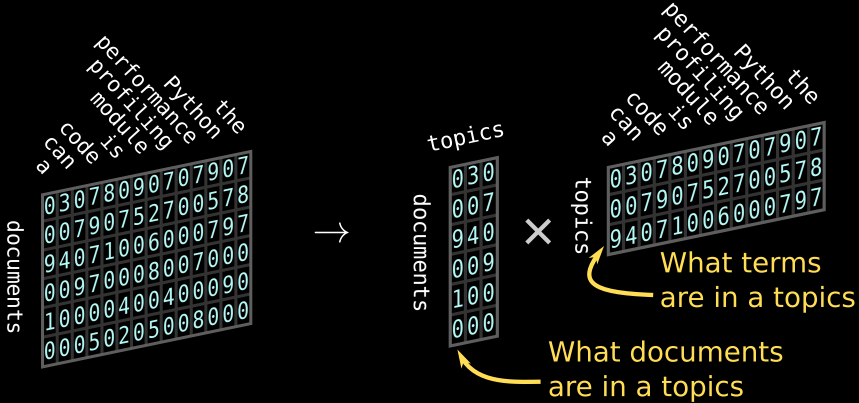


Term-document matrix



Can be a sparse matrix

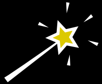
1 Let's do something together: sort EuroPython site



A matrix factorization

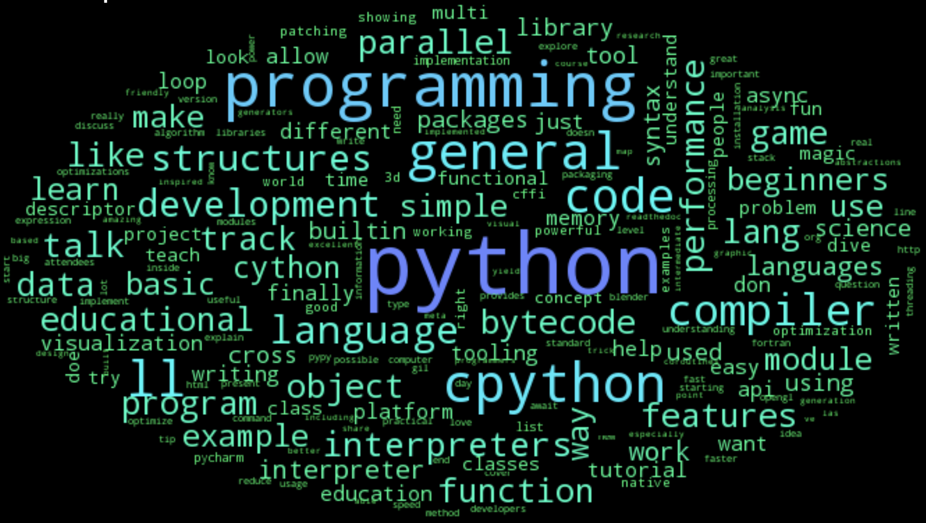
Often with non-negative constraints

`sklearn.decompositions.NMF`



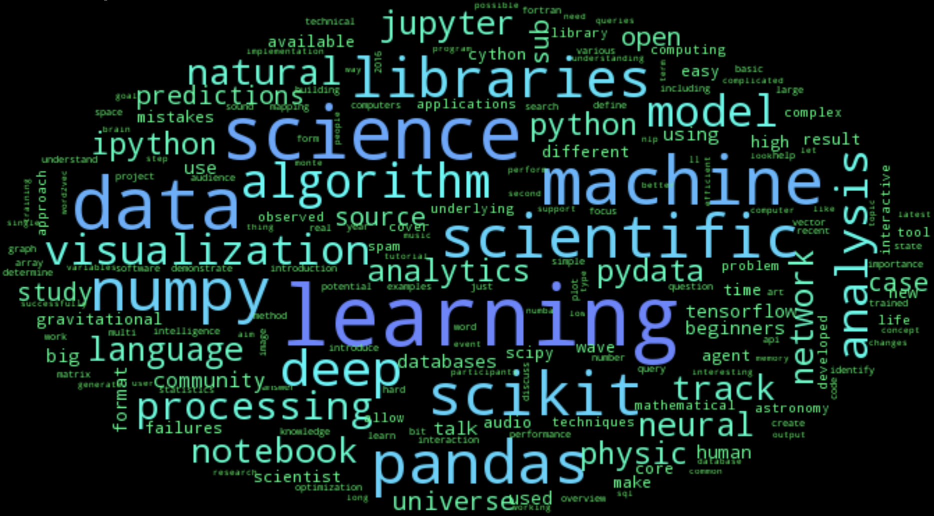
EuroPyton abstracts

Topic 1



EuroPyton abstracts

Topic 2



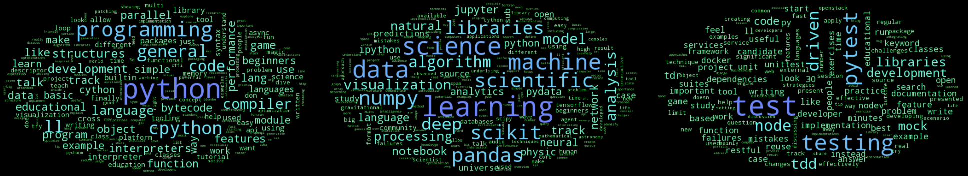
EuroPyton abstracts

Topic 3

EuroPyton abstracts

G Varoquaux

EuroPyton abstracts



Add one of Python’s great templating engine

... get a usable website

https://gaelvaroquaux.github.io/my_topics/ep16

Want to try it?

```
$ pip install scikit-learn
```

Want to try it?

```
$ pip install scikit-learn
...
ImportError: Numerical Python (NumPy) is not installed.
scikit-learn requires NumPy >= 1.6.1
```

Want to try it?

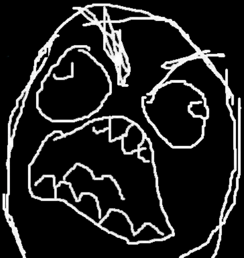
```
$ pip install scikit-learn
...
ImportError: Numerical Python (NumPy) is not installed.
scikit-learn requires NumPy >= 1.6.1
```

```
C:> pip install numpy
```

Want to try it?

```
$ pip install scikit-learn
...
ImportError: Numerical Python (NumPy) is not installed.
scikit-learn requires NumPy >= 1.6.1
```

```
C:> pip install numpy
...
error:  Unable to find vcvarsall.bat
```



1 We're different

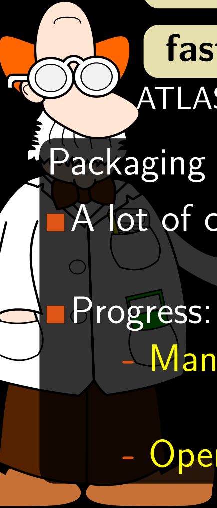
libfortran.so.3 ??

Well

you're kidding me

fast linear algebra

ATLAS (Fortran) 70x faster



libfortran.so.3 ??

Well

you’re kidding me

fast linear algebra

ATLAS (Fortran) 70x faster

Packaging is a major roadblock for scientific Python

- A lot of compiled code + shared libraries
  $\Rightarrow$ library + ABI compatibility issues
- Progress:
  - Manylinux wheels: PEP 513, RT. McGibbon, NJ. Smith
    rely on a conservative core set of libs
  - Openblas: pure-C, fast linear algebra

1 We're different

But working together gives us awesome things

Text mining $\Rightarrow$ intelligent interfaces

2 The scientist's view of code

Numerics versus control flow

Numerics versus databases

Numerics versus strings

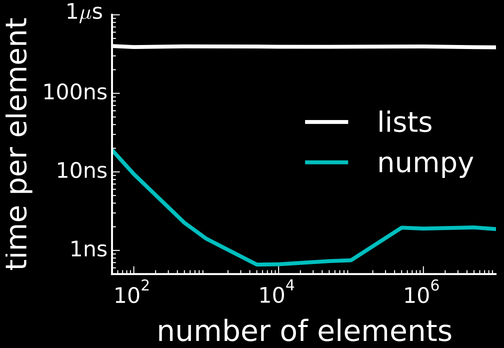
Numerics versus the world

100 000 term frequency vs inverse doc frequency:

```
In [*]: %timeit [t * i for t, i in izip(tf, idf)]
100 loops, best of 3: 6.2 ms per loop
```

The numpy style:

```
In [*]: %timeit tf * idf
1000 loops, best of 3: 74.2 µs per loop
```



100 000 term frequency vs inverse doc frequency:

```
In [*]: %timeit [t * i for t, i in izip(tf, idf)]
100 loops, best of 3: 6.2 ms per loop
```

The numpy style:

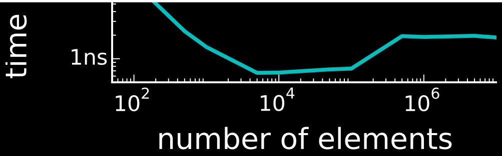
```
In [*]: %timeit tf * idf
1000 loops, best of 3: 74.2 µs per loop
```

Array computing can be more readable

`tf * idf`

vs

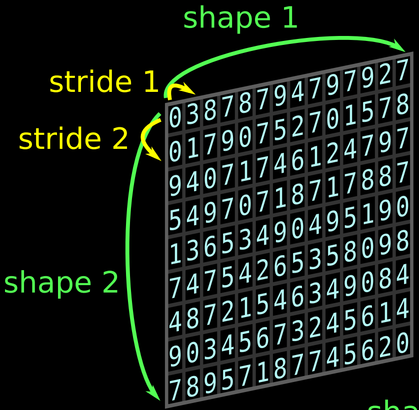
`[t * i for t, i in izip(tf, idf)]`



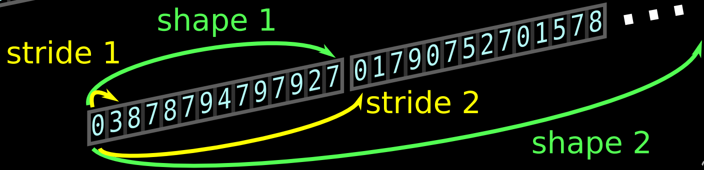
2 arrays are nothing but pointers

A numpy array =

- memory address
- data type
- shape
- strides



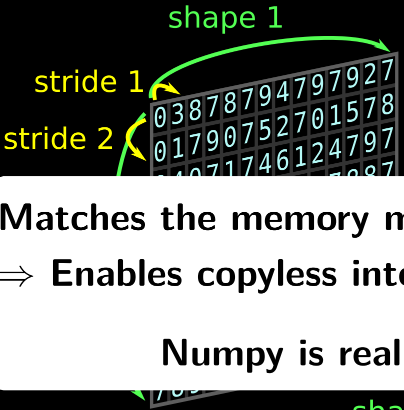
Represents any regular data in a structured way: how to access elements via pointer arythmetics (computing offsets)



2 arrays are nothing but pointers

A numpy array =

- memory address
- data type
- shape
- strides

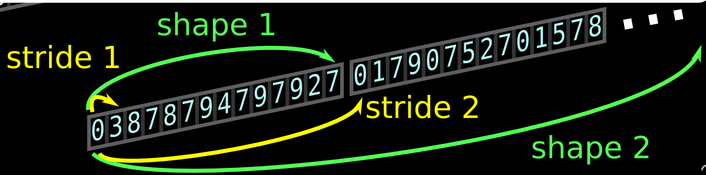


Represents any regular data in a structured way:

Matches the memory model of numerical libraries

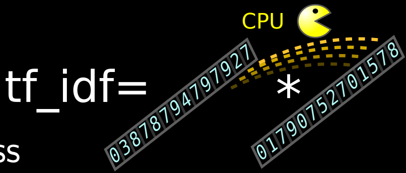
⇒ Enables copyless interactions

Numpy is really a memory model

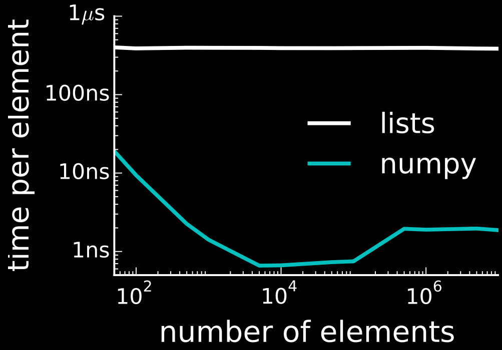


2 Array computing is fast

tf_idf = tf * idf

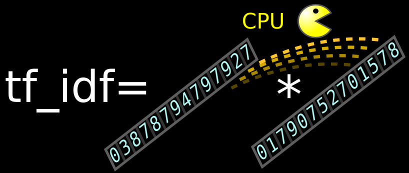


- No type checking
- Direct sequential memory access
- Vector operations (SIMD)

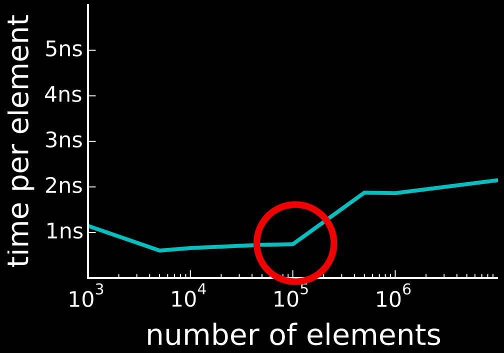


2 Array computing is limited by CPU starvation

tf_idf = tf * idf

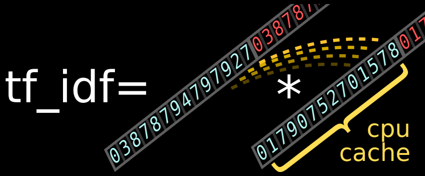


2x slowdown passed a certain size

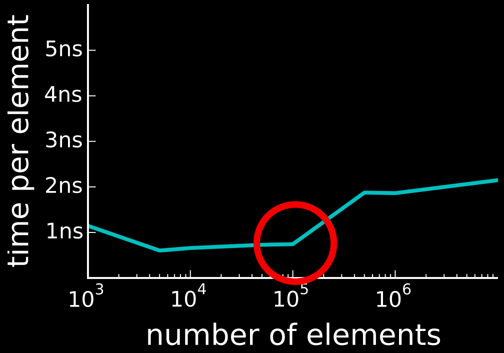


2 Array computing is limited by CPU starvation

$\mathrm{tf\_idf} = \mathrm{tf} * \mathrm{idf}$



2x slowdown passed a certain size

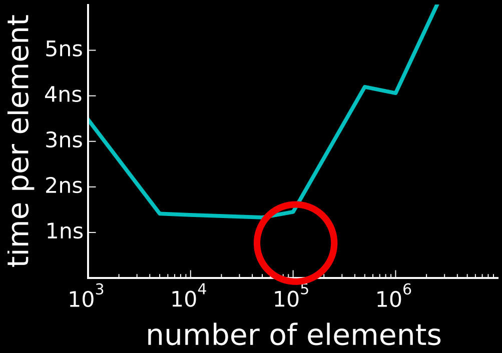


10^5 $\sim$ size of the CPU cache

Memory is much slower than CPU

tf_idf = tf * idf - 1

It gets worse for complex expressions



Memory is much slower than CPU

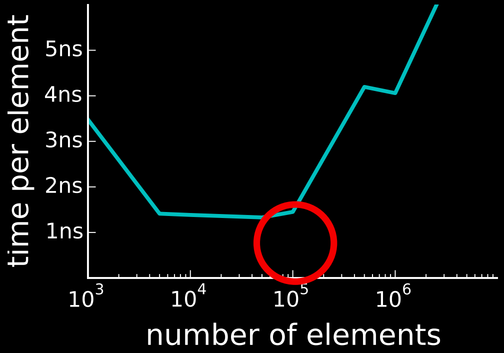
2 Array computing is limited by CPU starvation

tf_idf = tf * idf - 1

What's going on:

1. tmp ← tf * idf
2. tf_idf ← tmp - 1

Big temporary: Moving data in & out of cache



Memory is much slower than CPU

2 Array computing is limited by CPU starvation

tf_idf = tf * idf - 1

What's going on:

1. tmp ← tf * idf
2. tf_idf ← tmp - 1

Big temporary: Moving data in & out of cache

```
In [*]: %timeit tf * idf
10000 loops, best of 3: 74.2 µs per loop

In [*]: %timeit tf * idf - 1
1000 loops, best of 3: 418 µs per loop
```

2 Array computing is limited by CPU starvation

```
tf_idf = tf * idf - 1
```

What's going on:

1. tmp ← tf * idf
2. tf_idf ← tmp - 1

Big temporary: Moving data in & out of cache

In [*]: %timeit **tf * idf**
10000 loops, best of 3: **74.2** μs per loop

In [*]: %timeit **tf * idf - 1**
1000 loops, best of 3: **418** μs per loop

In-place operations: reuse the allocation

In [*]: %timeit **tmp = tf * idf; tmp -= 1**
10000 loops, best of 3: **112** μs per loop

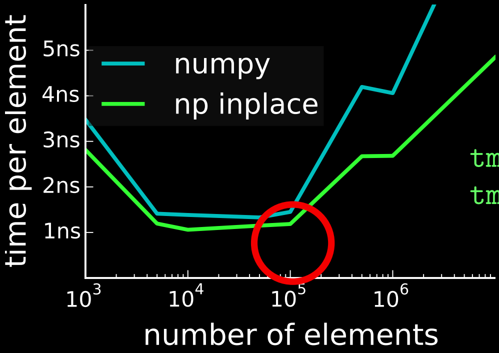
2 Array computing is limited by CPU starvation

tf_idf = tf * idf - 1

What's going on:

1. tmp ← tf * idf
2. tf_idf ← tmp - 1

Big temporary: Moving data in & out of cache



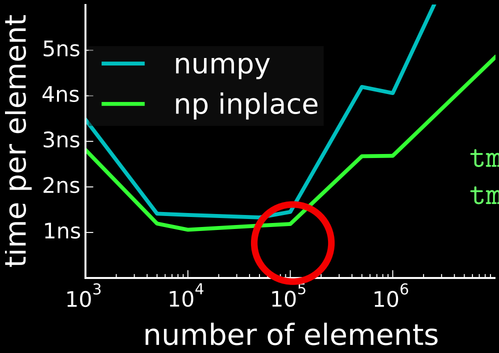

```
tmp = tf * idf
tmp -= 1
```

A compilation problem:

```
tf_idf = tf * idf - 1
```
⇝
```
tf_idf = tf * idf
tf_idf -= 1
```

1. tmp ← tf * idf
2. tf_idf ← tmp - 1

Big temporary: Moving data in & out of cache



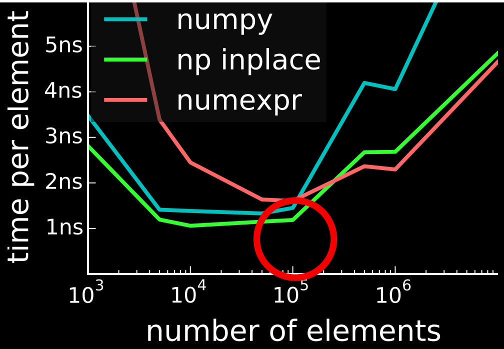

```
tmp = tf * idf
tmp -= 1
```

A compilation problem:

- **Removing/reusing temporaries**
- **Operating on "chunks" that fit in cache**

■ **Addressed by numexpr, with string expressions** ☹

```
numexpr.evaluate('tf * idf - 1', locals())
```

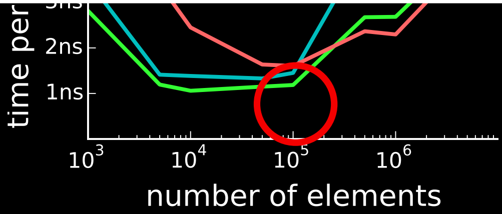


A compilation problem:

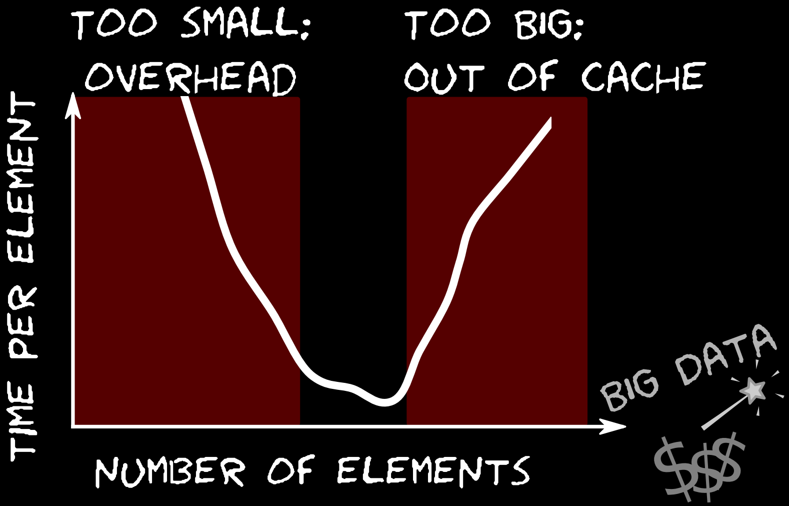
- Removing/reusing temporaries
- Operating on "chunks" that fit in cache

- Addressed by numexpr, with string expressions ☹
- Addressed by numba, with bytecode inspection ☹
- lazyarray

Similar problem to pagination with SQL queries



tf_idf = tf * idf


TOO SMALL:
OVERHEAD
TOO BIG:
OUT OF CACHE
TIME PER ELEMENT
NUMBER OF ELEMENTS
BIG DATA
$$$

What if there is an if

```
tf_idf = tf / idf
tf_idf[idf == 0] = 0
```

Suppose the we are looking at ages in a population:

```
ages[gender == 'male'].mean()
        - ages[gender == 'female'].mean()
```

2 Numerics versus control flow

What if there is an if

```
tf_idf = tf / idf
tf_idf[idf == 0] = 0
```

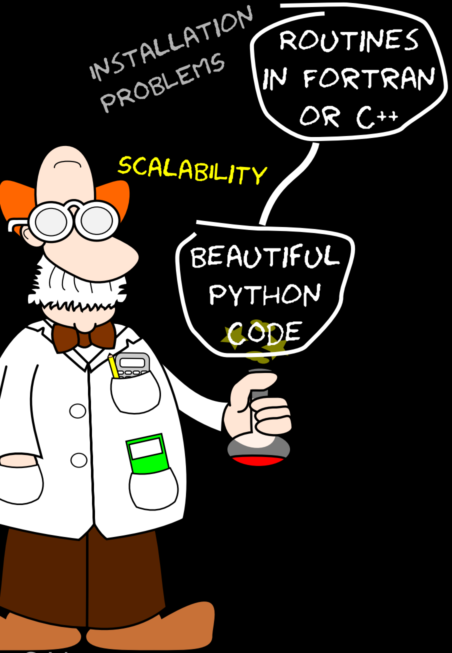
Suppose the we are looking at ages in a population:

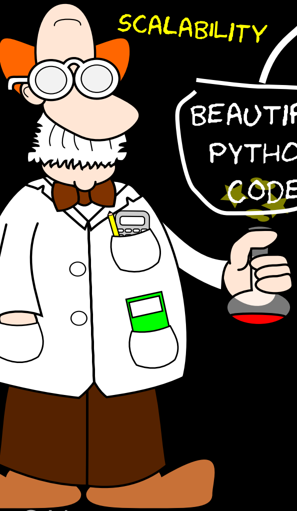
```
ages[gender == 'male'].mean()
        - ages[gender == 'female'].mean()
```

This is really starting to be looking like databases

pandas: something in between arrays and an in-memory database

Great for queries, less great for numerics.


INSTALLATION PROBLEMS
ROUTINES IN FORTRAN OR C++
SCALABILITY
BEAUTIFUL PYTHON CODE



Numpy is the scientist's equivalent to an ORM

Gives speed
with non-Python code

numerics vs databases

numerics efficient on regularly spaced data

But numpy creates cache misses for big arrays

$\Rightarrow$ Need to remove temporaries and chunk data

numerics vs databases

numerics efficient on regularly spaced data

But numpy creates cache misses for big arrays

$\Rightarrow$ Need to remove temporaries and chunk data

selection and grouping efficient with indexes or trees

$\Rightarrow$ Need to group queries

Compilation

numerics vs databases

numerics efficient on regularly spaced data

But numpy creates cache misses for big arrays

$\Rightarrow$ Need to remove temporaries and chunk data

selection and grouping efficient with indexes or trees

$\Rightarrow$ Need to group queries

Compilation is unpythonic

A computation & query language? `numexpr`

I hate domain-specific languages (SQL)

Numpy is very expressive

numerics vs databases

numerics efficient on regularly spaced data

But numpy creates cache misses for big arrays

$\Rightarrow$ Need to remove temporaries and chunk data

selection and grouping efficient with indexes or trees

$\Rightarrow$ Need to group queries

Compilation is unpythonic

A computation & query language? `numexpr`

I hate domain-specific languages (SQL)

Numpy is very expressive

PonyORM: Compiling Python to optimized SQL

Datascience with SQL: Ibis & Blaze

numerics vs databases

numerics efficient on regularly spaced data

But numpy creates cache misses for big arrays

⇒ Need to remove temporaries and chunk data

selection and grouping efficient with indexes or trees

⇒ Need to group queries

Spark: java-world "big data" rising star

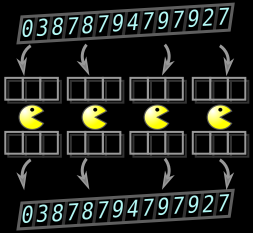
combines distributed store

+ computing model

We (scikit-learn) are faster when data fits in RAM

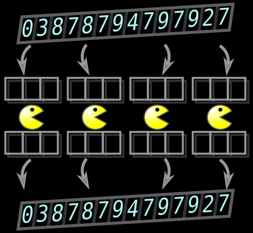
Operations on chunks

- Machine learning, data mining = numerics



Operations on chunks

- Machine learning, data mining = numerics

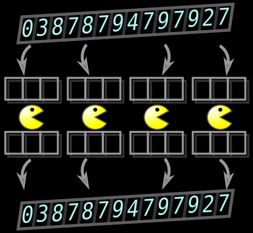


ETL (extract, transform, & load)

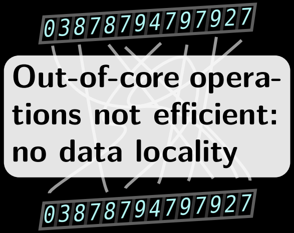
Multivariate statistics

Operations on chunks, or algorithms on chunks

- Machine learning, data mining = numerics

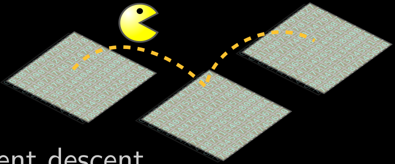


ETL (extract, transform, & load)



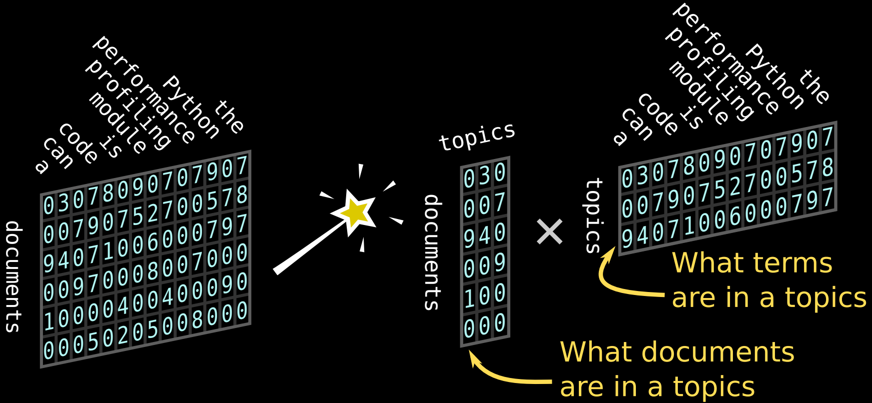
Multivariate statistics

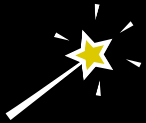
- On-line **aglorithms** (streaming)



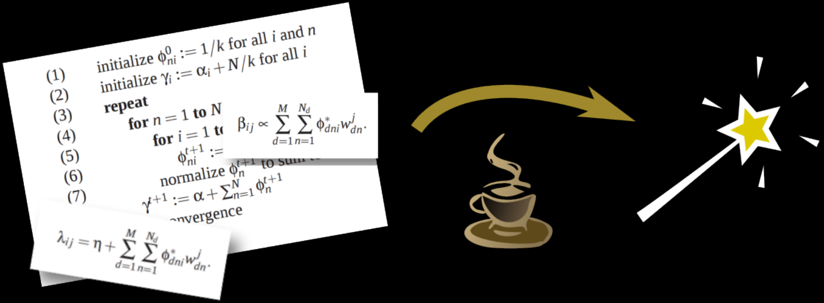
eg stochastic gradient descent
As in deep learning

Making the data-science magic happens



from sklearn import 

Making the data-science magic happens

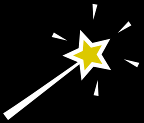


Turning applied maths papers to robust code

High-level, readable, simple syntax reduces cognitive load

Thanks

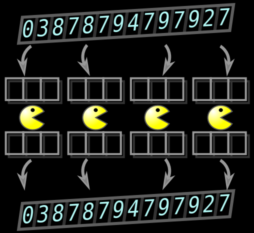
`from sklearn import`

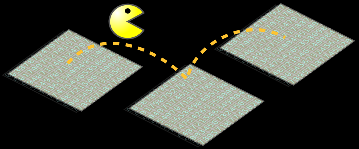


3 Beyond numerics

Make #PyData great (again)

3 Data/computation flow is crucial





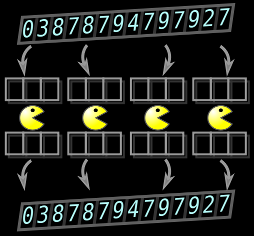
Data-flow engines are everywhere

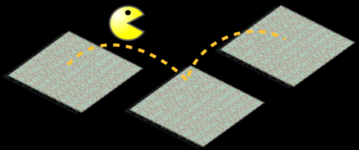
`dask`
- pure-Python
- dynamic scheduler
- static compiler
- parallel & distributed

`theano`
- expression analysis
- pure-Python

`tensorflow`
- C library
- distributed

3 Data/computation flow is crucial





Data-flow engines are everywhere

Python should shine there:

reflexivity + metaprogramming + async

"Python is the best numerical language out there because it's not a numerical language." – Nathaniel Smith

API challenging:
For algorithm design: no framework / inversion of control

Distributed computation & Run-time analysis

Reflexivity is central

- Debugging
  - Interactive work
    - Code analysis
      - Persistence
        - Parallel computing

3 Ingredients for future data flows

Distributed computation & Run-time analysis

Reflexivity is central

- Debugging
- Interactive work
- Code analysis
- Persistence
- Parallel computing

Pickle

- distribute code and data without data model
- serialize intermediate results
- deep of hash of any data structure `joblib.hash`

Very limited (eg no lambda #19272)
⇒ variants: dill, cloudpickle

Distributed computation & Run-time analysis

joblib:

- **Simple parallel syntax:**

```
Parallel(n_jobs=2)(delayed(sqrt)(i) for i in range(10))
```

- **Fast persistence:**

```
joblib.dump(anything, 'filename.pkl.gz')
```

- **Primitive for out of core:**

```
pointer = mem.cache(f).call_and_shelves(big_data)
```

- **Non-invasive syntax / paradigm**
- **Fast on big numpy arrays**
- **Soon backend system (job broker and persistence)**

Gets job managment into algorithms (eg in scikit-learn)

3 The Python VM is great

The simplicity of the VM is our strength

- Software Transactional Memory... would be nice

  But, I want to use foreign memory

  Java gained `jmalloc` for foreign memory

- Better garbage collection

  Yes but, I easily plug into reference counting

A strength of Python is its clear C API

$\Rightarrow$ Easy foreign functionality

3 The Python VM is great

The simplicity of the VM is our strength

`Cython`: **the best of C and Python**

- **Add types for speed** (**numpy arrays as** `float*`)
- **Call C to bind external libraries: surprisingly easy**

no pointer arithmetics 🙂

An adaptation layer between Python VM and C

A strength of Python is its clear C API

$\Rightarrow$ Easy foreign functionality

4 Working together

4 Scikit-learn is easy machine learning

As easy as py

```python
from sklearn import svm
classifier = svm.SVC()
classifier.fit(X_train, Y_train)
Y_test = classifier.predict(X_test)
```

People love the encapsulation

`classifier` is a semi black box

- The power of a simple object-oriented API
- Documentation-driven development

As easy as py

```
from sklearn import svm
classifier = svm.SVC()
classifier.fit(X_train, Y_train)
Y_test = classifier.predict(X_test)
```

People love the encapsulation

`classifier` is a semi black box

- The power of a simple object-oriented API
- Documentation-driven development

High-level, readable, simple API reduces cognitive load

PyData loves Python in return

4 Difference is richness

We all do different things

We can all benefit from others

though we don't know how

4 Difference is richness, but requires outreach

We all do different things

We can all benefit from others

though we don't know how

Being didactic outside one's community is crucial

- Avoiding jargon — take that machine learning ☹
- Prioritizing information
- "Simple is better than complex"

Students learning numerics don't care about unicode

Build documentation upon very simple examples

Think stackoverflow

Sphinx + Sphinx-gallery

Scientist ❤ web dev: Python is the language for data

- Python language & VM is perfect to manipulate
  low-level constructs
  with high-level wordings
  Connects to other paradigms, eg C

@GaelVaroquaux

Scientist ❤ web dev: Python is the language for data

- Python language & VM is perfect to manipulate
  low-level constructs
  with high-level wordings

- Dynamism and reflexivity
  ⇒ meta-programming and debugging

@GaelVaroquaux

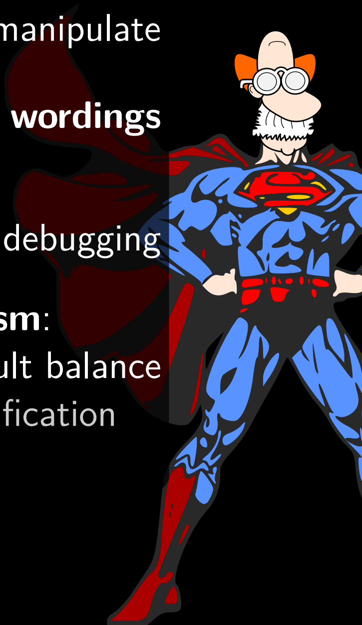
Scientist ❤ web dev: Python is the language for data

- Python language & VM is perfect to manipulate
  low-level constructs
  with high-level wordings

- Dynamism and reflexivity
  ⇒ meta-programming and debugging

- Needs for **compilation** and **dynamism**:
  a difficult balance

  PEP 509: guards on run-time modification
  PEP 510: function specicalization



@GaelVaroquaux

Scientist ❤ web dev: Python is the language for data

- Python language & VM is perfect to manipulate
  low-level constructs
  with high-level wordings

- Dynamism and reflexivity
  $\Rightarrow$ meta-programming and debugging

- Needs for **compilation** and **dynamism**

- Pydata will use DB and concurrency from web

- PyData can give knowledge engineering + AI

@GaelVaroquaux