*“Silence is a true friend who never betrays”**

Thomas Kober
@tttthomasssss

PyData Edinburgh
7th Feb 2019 (1549566000)

*) Unless you write code

Whats this all about?

Whats this all about?

- This talk is based on a true story

Whats this all about?

- This talk is based on a true story
- A story about SCARY BEHAVIOUR in `scipy.sparse` matrices and `numpy` arrays

Whats this all about?

- This talk is based on a true story
- A story about SCARY BEHAVIOUR in `scipy.sparse` matrices and `numpy` arrays
- A story about **co-occurrence matrices**

Co-occurrence Matrices

Co-occurrence Matrices

- Count how often 2 items co-occur, or, more abstractly, a measure of association between two items

Co-occurrence Matrices

- Count how often 2 items co-occur, or, more abstractly, a measure of association between two items
- For example, lets suppose a semi-imaginary Spotify history:

Co-occurrence Matrices

- Count how often 2 items co-occur, or, more abstractly, a measure of association between two items

- For example, lets suppose a semi-imaginary Spotify history:

Bad Weed
Arab Strap
Voodoo Jürgens
Miles Davis
Bill Evans
Muddy Waters

Co-occurrence Matrices

- Count how often 2 items co-occur, or, more abstractly, a measure of association between two items

- For example, lets suppose a semi-imaginary Spotify history:

	Nick
Bad Weed	0
Arab Strap	0
Voodoo Jürgens	0
Miles Davis	7
Bill Evans	12
Muddy Waters	4

Co-occurrence Matrices

- Count how often 2 items co-occur, or, more abstractly, a measure of association between two items

- For example, lets suppose a semi-imaginary Spotify history:

	Nick	Thomas
Bad Weed	0	8
Arab Strap	0	5
Voodoo Jürgens	0	6
Miles Davis	7	0
Bill Evans	12	0
Muddy Waters	4	0

Co-occurrence Matrices

- Count how often 2 items co-occur, or, more abstractly, a measure of association between two items

- For example, lets suppose a semi-imaginary Spotify history:

	Nick	Thomas	Sam
Bad Weed	0	8	0
Arab Strap	0	5	6
Voodoo Jürgens	0	6	0
Miles Davis	7	0	0
Bill Evans	12	0	4
Muddy Waters	4	0	2

Co-occurrence Matrices

- Count how often 2 items co-occur, or, more abstractly, a measure of association between two items

- For example, lets suppose a semi-imaginary Spotify history:

	Nick	Thomas	Sam
Bad Weed	0	8	0
Arab Strap	0	5	6
Voodoo Jürgens	0	6	0
Miles Davis	7	0	0
Bill Evans	12	0	4
Muddy Waters	4	0	2

Co-occurrence Matrices

- Count how often 2 items co-occur, or, more abstractly, a measure of association between two items

- For example, lets suppose a semi-imaginary Spotify history:

	Nick	Thomas	Sam
Bad Weed	0	8	0
Arab Strap	0	5	6
Voodoo Jürgens	0	6	0
Miles Davis	7	0	0
Bill Evans	12	0	4
Muddy Waters	4	0	2

Co-occurrence Matrices

- Count how often 2 items co-occur, or, more abstractly, a measure of association between two items

- For example, lets suppose a semi-imaginary Spotify history:

	Nick	Thomas	Sam
Bad Weed	0	8	0
Arab Strap	0	5	6
Voodoo Jürgens	0	6	0
Miles Davis	7	0	0
Bill Evans	12	0	4
Muddy Waters	4	0	2

Co-occurrence Matrices

- Count how often 2 items co-occur, or, more abstractly, a measure of association between two items

- For example, lets suppose a semi-imaginary Spotify history:

	Nick	Thomas	Sam
Bad Weed	0	8	0
Arab Strap	0	5	6
Voodoo Jürgens	0	6	0
Miles Davis	7	0	0
Bill Evans	12	0	4
Muddy Waters	4	0	2

Co-occurrence Matrices

- Count how often 2 items co-occur, or, more abstractly, a measure of association between two items

- For example, lets suppose a semi-imaginary Spotify history:

	Nick	Thomas	Sam	Martina
Bad Weed	0	8	0	4
Arab Strap	0	5	6	0
Voodoo Jürgens	0	6	0	0
Miles Davis	7	0	0	6
Bill Evans	12	0	4	1
Muddy Waters	4	0	2	0

Co-occurrence Matrices

- Count how often 2 items co-occur, or, more abstractly, a measure of association between two items

- For example, lets suppose a semi-imaginary Spotify history:

	Nick	Thomas	Sam	Martina
Bad Weed	0	8	0	4
Arab Strap	0	5	6	0
Voodoo Jürgens	0	6	0	0
Miles Davis	7	0	0	6
Bill Evans	12	0	4	1
Muddy Waters	4	0	2	0

Co-occurrence Matrices

- Count how often 2 items co-occur, or, more abstractly, a measure of association between two items

- For example, lets suppose a semi-imaginary Spotify history:

	Nick	Thomas	Sam	Martina
Bad Weed	0	8	0	4
Arab Strap	0	5	6	0
Voodoo Jürgens	0	6	0	0
Miles Davis	7	0	0	6
Bill Evans	12	0	4	1
Muddy Waters	4	0	2	0

Co-occurrence Matrices

- Count how often 2 items co-occur, or, more abstractly, a measure of association between two items

- For example, lets suppose a semi-imaginary Spotify history:

	Nick	Thomas	Sam	Martina
Bad Weed	0	8	0	4
Arab Strap	0	5	6	0
Voodoo Jürgens	0	6	0	0
Miles Davis	7	0	0	6
Bill Evans	12	0	4	1
Muddy Waters	4	0	2	0

Co-occurrence Matrices

- Count how often 2 items co-occur, or, more abstractly, a measure of association between two items

- For example, lets suppose a semi-imaginary Spotify history:

	Nick	Thomas	Sam	Martina
Bad Weed	0	8	0	4
Arab Strap	0	5	6	0
Voodoo Jürgens	0	6	0	0
Miles Davis	7	0	0	6
Bill Evans	12	0	4	1
Muddy Waters	4	0	2	0

Co-occurrence Matrices

- Count how often 2 items co-occur, or, more abstractly, a measure of association between two items

- For example, lets suppose a semi-imaginary Spotify history:

	Nick	Thomas	Sam	Martina
Bad Weed	0	8	0	4
Arab Strap	0	5	6	0
Voodoo Jürgens	0	6	0	0
Miles Davis	7	0	0	6
Bill Evans	12	0	4	1
Muddy Waters	4	0	2	0

- So co-occurrence matrices can be very useful for **recommender systems**

Co-occurrence Matrices

Co-occurrence Matrices

- Another use-case, **distributional semantics**, i.e. modelling the meaning of a word

Co-occurrence Matrices

- Another use-case, **distributional semantics**, i.e. modelling the meaning of a word
- For example, lets suppose we have counted how often every word in some text collection has co-occurred with every other word

Co-occurrence Matrices

- Another use-case, **distributional semantics**, i.e. modelling the meaning of a word
- For example, lets suppose we have counted how often every word in some text collection has co-occurred with every other word

	eat	restaurant	play	italian	walk

Co-occurrence Matrices

- Another use-case, **distributional semantics**, i.e. modelling the meaning of a word
- For example, lets suppose we have counted how often every word in some text collection has co-occurred with every other word

	eat	restaurant	play	italian	walk
pizza	32	8	0	14	0

Co-occurrence Matrices

- Another use-case, **distributional semantics**, i.e. modelling the meaning of a word
- For example, lets suppose we have counted how often every word in some text collection has co-occurred with every other word

	eat	restaurant	play	italian	walk
pizza	32	8	0	14	0
lasagne	24	15	0	18	0

Co-occurrence Matrices

- Another use-case, **distributional semantics**, i.e. modelling the meaning of a word
- For example, lets suppose we have counted how often every word in some text collection has co-occurred with every other word

	eat	restaurant	play	italian	walk
pizza	32	8	0	14	0
lasagne	24	15	0	18	0
cat	0	0	17	0	4

Co-occurrence Matrices

- Another use-case, **distributional semantics**, i.e. modelling the meaning of a word
- For example, lets suppose we have counted how often every word in some text collection has co-occurred with every other word

	eat	restaurant	play	italian	walk
pizza	32	8	0	14	0
lasagne	24	15	0	18	0
cat	0	0	17	0	4
dog	0	2	25	0	23

Co-occurrence Matrices

- Another use-case, **distributional semantics**, i.e. modelling the meaning of a word
- For example, lets suppose we have counted how often every word in some text collection has co-occurred with every other word

	eat	restaurant	play	italian	walk
pizza	32	8	0	14	0
lasagne	24	15	0	18	0
cat	0	0	17	0	4
dog	0	2	25	0	23
broomstick	0	0	1	0	0

Co-occurrence Matrices

- Another use-case, **distributional semantics**, i.e. modelling the meaning of a word
- For example, lets suppose we have counted how often every word in some text collection has co-occurred with every other word

	eat	restaurant	play	italian	walk
pizza	32	8	0	14	0
lasagne	24	15	0	18	0
cat	0	0	17	0	4
dog	0	2	25	0	23
broomstick	0	0	1	0	0

Co-occurrence Matrices

- Another use-case, **distributional semantics**, i.e. modelling the meaning of a word
- For example, lets suppose we have counted how often every word in some text collection has co-occurred with every other word

	eat	restaurant	play	italian	walk
pizza	32	8	0	14	0
lasagne	24	15	0	18	0
cat	0	0	17	0	4
dog	0	2	25	0	23
broomstick	0	0	1	0	0

Co-occurrence Matrices

- Another use-case, **distributional semantics**, i.e. modelling the meaning of a word
- For example, lets suppose we have counted how often every word in some text collection has co-occurred with every other word

	eat	restaurant	play	italian	walk
pizza	32	8	0	14	0
lasagne	24	15	0	18	0
cat	0	0	17	0	4
dog	0	2	25	0	23
broomstick	0	0	1	0	0

Co-occurrence Matrices

- Another use-case, **distributional semantics**, i.e. modelling the meaning of a word
- For example, lets suppose we have counted how often every word in some text collection has co-occurred with every other word

	eat	restaurant	play	italian	walk
pizza	32	8	0	14	0
lasagne	24	15	0	18	0
cat	0	0	17	0	4
dog	0	2	25	0	23
broomstick	0	0	1	0	0

Co-occurrence Matrices

Co-occurrence Matrices

- Typically very **high-dimensional**

Co-occurrence Matrices

- Typically very **high-dimensional**
  - *Number of users* **x** *Number of songs*

Co-occurrence Matrices

- Typically very **high-dimensional**
  - *Number of users* **x** *Number of songs*
  - *Number of unique words* **x** *Number of unique words*

Co-occurrence Matrices

- Typically very **high-dimensional**
  - *Number of users* **x** *Number of songs*
  - *Number of unique words* **x** *Number of unique words*
- And very **sparse**

Co-occurrence Matrices

- Typically very **high-dimensional**
  - *Number of users* **x** *Number of songs*
  - *Number of unique words* **x** *Number of unique words*
- And very **sparse**
  - Any user doesn’t listen to most songs

Co-occurrence Matrices

- Typically very **high-dimensional**
  - *Number of users* **x** *Number of songs*
  - *Number of unique words* **x** *Number of unique words*
- And very **sparse**
  - Any user doesn’t listen to most songs
  - Any word doesn’t co-occur with most other words

Co-occurrence Matrices

- Typically very **high-dimensional**
  - *Number of users* **x** *Number of songs*
  - *Number of unique words* **x** *Number of unique words*
- And very **sparse**
  - Any user doesn't listen to most songs
  - Any word doesn't co-occur with most other words
- So when we're creating them ourselves from data, we make use of **`numpy & scipy`** (because we like python and data)

scipy.sparse

scipy.sparse

```
In [195]: x = np.full((3,12), 255, dtype=np.uint8)
In [196]: x
Out[196]:
array([[255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255]], dtype=uint8)
```

scipy.sparse

```
In [195]: x = np.full((3,12), 255, dtype=np.uint8)
In [196]: x
Out[196]:
array([[255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255]], dtype=uint8)
```

```
In [197]: x.sum(axis=1)
Out[197]: array([3060, 3060, 3060], dtype=uint64)
```

scipy.sparse

```
In [195]: x = np.full((3,12), 255, dtype=np.uint8)
In [196]: x
Out[196]:
array([[255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255]], dtype=uint8)
```

numpy automatically upcasted the dtype

```
In [197]: x.sum(axis=1)
Out[197]: array([3060, 3060, 3060], dtype=uint64)
```

scipy.sparse

```
In [195]: x = np.full((3,12), 255, dtype=np.uint8)
In [196]: x
Out[196]:
array([[255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255]], dtype=uint8)
```

numpy automatically upcasted the dtype

```
In [197]: x.sum(axis=1)
Out[197]: array([3060, 3060, 3060], dtype=uint64)
```

```
In [198]: from scipy import sparse
In [199]: xs = sparse.coo_matrix(x)
```

scipy.sparse

```
In [195]: x = np.full((3,12), 255, dtype=np.uint8)
In [196]: x
Out[196]:
array([[255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255]], dtype=uint8)
```

numpy automatically upcasted the dtype

```
In [197]: x.sum(axis=1)
Out[197]: array([3060, 3060, 3060], dtype=uint64)
```

```
In [198]: from scipy import sparse
In [199]: xs = sparse.coo_matrix(x)
```

```
In [201]: xs.sum(axis=1)
Out[201]:
matrix([[244],
        [244],
        [244]], dtype=uint8)
```

scipy.sparse

```
In [195]: x = np.full((3,12), 255, dtype=np.uint8)
In [196]: x
Out[196]:
array([[255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255]], dtype=uint8)
```

numpy automatically upcasted the dtype

```
In [197]: x.sum(axis=1)
Out[197]: array([3060, 3060, 3060], dtype=uint64)
```

```
In [198]: from scipy import sparse
In [199]: xs = sparse.coo_matrix(x)
```

```
In [201]: xs.sum(axis=1)
Out[201]:
matrix([[244],
        [244],
        [244]], dtype=uint8)
```

scipy didn't

scipy.sparse

```
In [195]: x = np.full((3,12), 255, dtype=np.uint8)
In [196]: x
Out[196]:
array([[255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255]], dtype=uint8)
```

numpy automatically upcasted the dtype

```
In [197]: x.sum(axis=1)
Out[197]: array([3060, 3060, 3060], dtype=uint64)
```

This does not look right!!!

```
In [198]: from scipy import sparse
In [199]: xs = sparse.coo_matrix(x)
```

```
In [201]: xs.sum(axis=1)
Out[201]:
matrix([[244],
        [244],
        [244]], dtype=uint8)
```

scipy didn't

scipy.sparse

```
In [195]: x = np.full((3,12), 255, dtype=np.uint8)
In [196]: x
Out[196]:
array([[255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255]], dtype=uint8)
```

numpy automatically upcasted the dtype

```
In [197]: x.sum(axis=1)
Out[197]: array([3060, 3060, 3060], dtype=uint64)
```

This does not look right!!!

```
In [198]: from scipy import sparse
In [199]: xs = sparse.coo_matrix(x)
```

```
In [201]: xs.sum(axis=1)
Out[201]:
matrix([[244],
        [244],
        [244]], dtype=uint8)
```

scipy didn't

- Did you hear that loud bang and crash from the error?

scipy.sparse

```
In [195]: x = np.full((3,12), 255, dtype=np.uint8)
In [196]: x
Out[196]:
array([[255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255]], dtype=uint8)
```

numpy automatically upcasted the dtype

This does not look right!!!

```
In [197]: x.sum(axis=1)
Out[197]: array([3060, 3060, 3060], dtype=uint64)
```

```
In [198]: from scipy import sparse
In [199]: xs = sparse.coo_matrix(x)
```

```
In [201]: xs.sum(axis=1)
Out[201]:
matrix([[244],
        [244],
        [244]], dtype=uint8)
```

scipy didn't

- Did you hear that loud bang and crash from the error?
  - No, me neither.

scipy.sparse

```
In [195]: x = np.full((3,12), 255, dtype=np.uint8)
In [196]: x
Out[196]:
array([[255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255]], dtype=uint8)
```

numpy automatically upcasted the dtype

This does not look right!!!

```
In [197]: x.sum(axis=1)
Out[197]: array([3060, 3060, 3060], dtype=uint64)
```

```
In [198]: from scipy import sparse
In [199]: xs = sparse.coo_matrix(x)
```

```
In [201]: xs.sum(axis=1)
Out[201]:
matrix([[244],
        [244],
        [244]], dtype=uint8)
```

scipy didn't

- Did you hear that loud bang and crash from the error?
  - No, me neither.
  - Thats because this was a SILENT OVERFLOW

scipy.sparse

```
In [195]: x = np.full((3,12), 255, dtype=np.uint8)
In [196]: x
Out[196]:
array([[255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255]
```

numpy automatically upcasted the dtype

This does not look right!!!

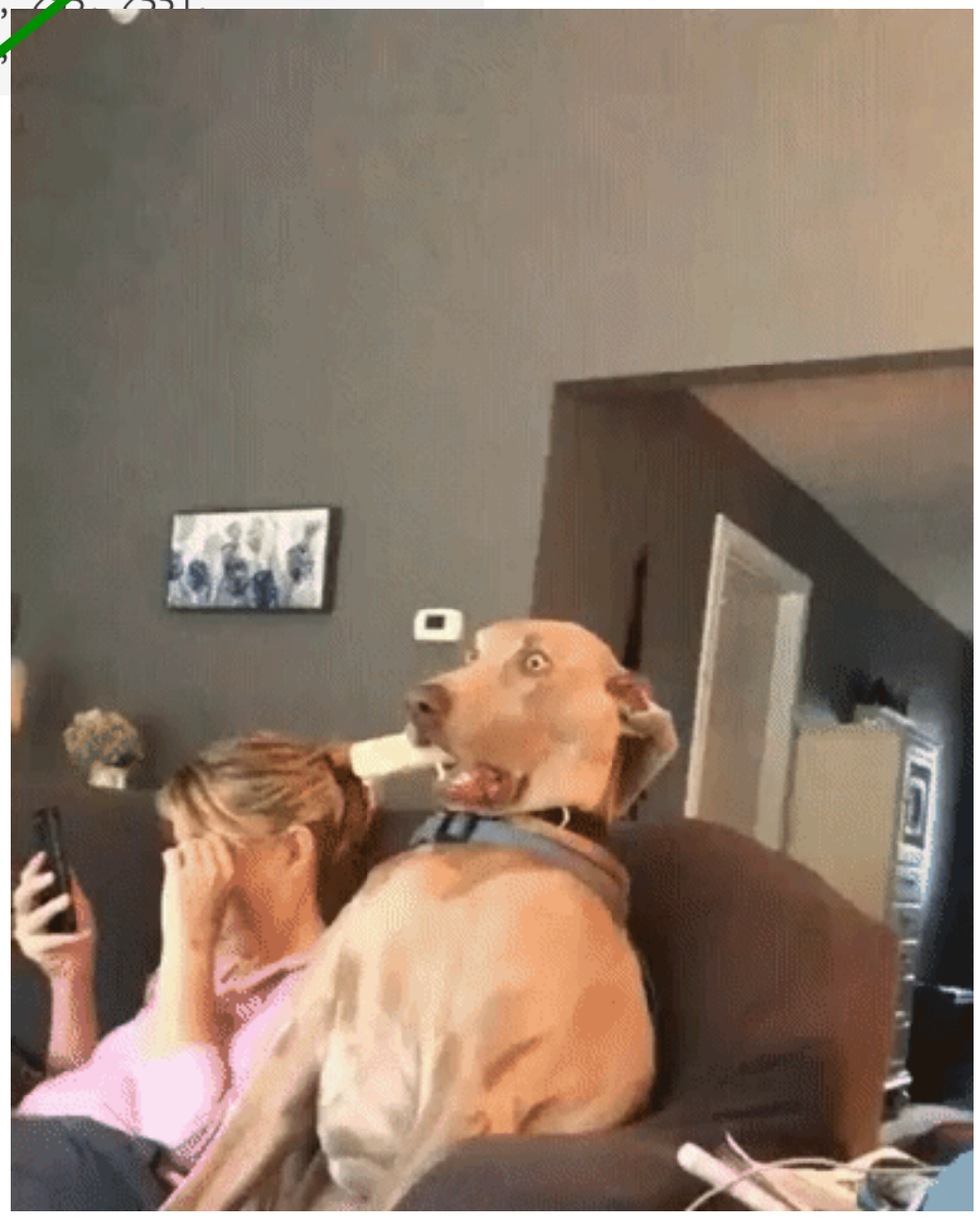
```
In [197]: x.sum(axis=1)
Out[197]: array([3060, 3060, 3060], dtype=uint64)
```

```
In [198]: from scipy import sparse
In [199]: xs = sparse.coo_matrix(x)
```

```
In [201]: xs.sum(axis=1)
Out[201]:
matrix([[244],
        [244],
        [244]], dtype=uint8)
```

scipy didn't

- Did you hear that loud bang and crash from the error?
  - No, me neither.
  - Thats because this was a SILENT OVERFLOW



There is good news and there is bad news

There is good news and there is bad news

- The **good** news

There is good news and there is bad news

- The **good** news
  - That bug has been **fixed** with `scipy v.0.18.0` (ca. 2016)

There is good news and there is bad news

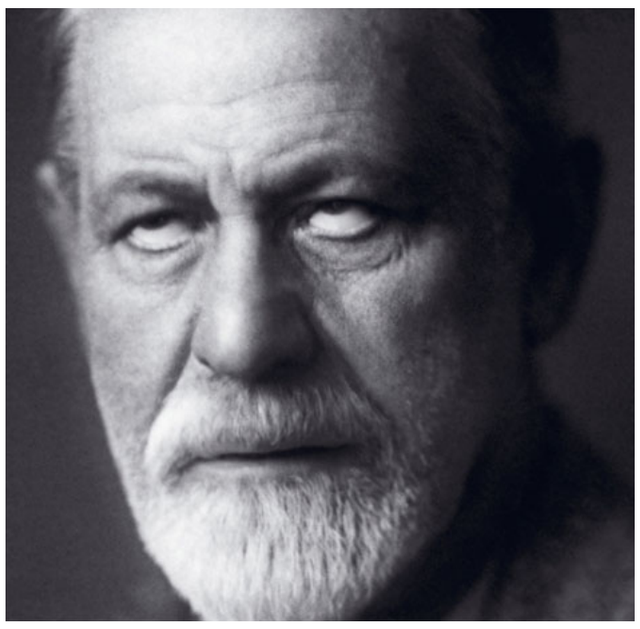
- The good news
  - That bug has been **fixed** with `scipy v.0.18.0` (ca. 2016)
- THE BAD NEWS

There is good news and there is bad news

- The good news
  - That bug has been **fixed** with `scipy v.0.18.0` (ca. 2016)
- THE BAD NEWS
  - There is another one of these…

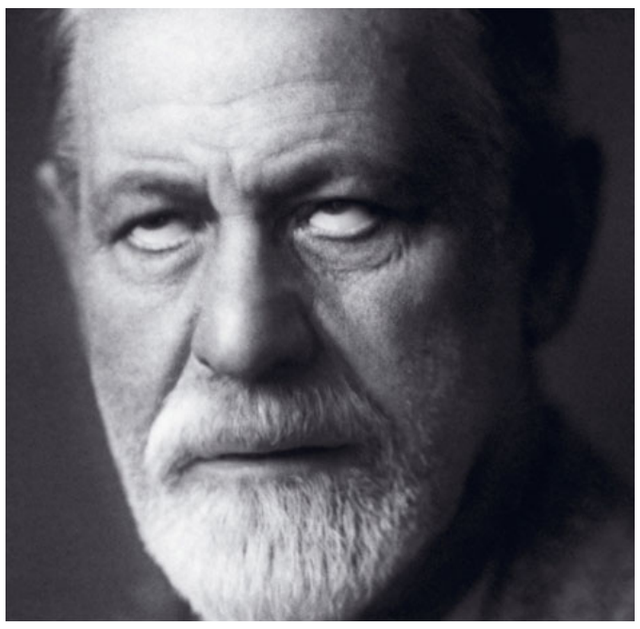
There is good news and there is bad news

- The good news
  - That bug has been **fixed** with `scipy v.0.18.0` (ca. 2016)
- THE BAD NEWS
  - There is another one of these…



There is good news and there is bad news

- The **good** news
  - That bug has been **fixed** with `scipy v.0.18.0` (ca. 2016)
- THE BAD NEWS
  - There is another one of these…



- Lets Live Demo the buggy code

So whats the deal with this?

So whats the deal with this?

- The major problem is

So whats the deal with this?

- The major problem is
    - SILENCE - there is no warning or error anywhere that tells you that bad stuff happened

So whats the deal with this?

- The major problem is
  - SILENCE - there is no warning or error anywhere that tells you that bad stuff happened
  - The bug leads to **inconsistent data**, even though the code - arguably - is correct

So whats the deal with this?

- The major problem is
  - SILENCE - there is no warning or error anywhere that tells you that bad stuff happened
  - The bug leads to **inconsistent data**, even though the code - arguably - is correct
  - This is very bad if you care about the correctness of your data

So whats the deal with this?

- The major problem is
  - SILENCE - there is no warning or error anywhere that tells you that bad stuff happened
  - The bug leads to **inconsistent data**, even though the code - arguably - is correct
  - This is very bad if you care about the correctness of your data
  - Also, this is a **known problem** (at least in numpy, going back to 2012 - see e.g. https://github.com/numpy/numpy/issues/593)

So whats the deal with this?

- The major problem is
  - SILENCE - there is no warning or error anywhere that tells you that bad stuff happened
  - The bug leads to **inconsistent data**, even though the code - arguably - is correct
  - This is very bad if you care about the correctness of your data
  - Also, this is a **known problem** (at least in numpy, going back to 2012 - see e.g. https://github.com/numpy/numpy/issues/593)
  - The reason why this hasn't been addressed yet is performance (see https://github.com/numpy/numpy/issues/8987#issuecomment-327378779)

So whats the deal with this?

- The major problem is
    - SILENCE - there is no warning or error anywhere that tells you that bad stuff happened
    - The bug leads to **inconsistent data**, even though the code - arguably - is correct
    - This is very bad if you care about the correctness of your data
    - Also, this is a **known problem** (at least in numpy, going back to 2012 - see e.g. https://github.com/numpy/numpy/issues/593)
    - The reason why this hasn't been addressed yet is performance (see https://github.com/numpy/numpy/issues/8987#issuecomment-327378779)
        - Floating point overflows are detected at the hardware level

So whats the deal with this?

- The major problem is
    - SILENCE - there is no warning or error anywhere that tells you that bad stuff happened
    - The bug leads to **inconsistent data**, even though the code - arguably - is correct
    - This is very bad if you care about the correctness of your data
    - Also, this is a **known problem** (at least in numpy, going back to 2012 - see e.g. https://github.com/numpy/numpy/issues/593)
    - The reason why this hasn't been addressed yet is performance (see https://github.com/numpy/numpy/issues/8987#issuecomment-327378779)
        - Floating point overflows are detected at the hardware level
        - Integer overflows aren't - they would need to be checked by `numpy`/`scipy`, which is too costly for arrays

So whats the deal with this?

- The major problem is
    - SILENCE - there is no warning or error anywhere that tells you that bad stuff happened
    - The bug leads to **inconsistent data**, even though the code - arguably - is correct
    - This is very bad if you care about the correctness of your data
    - Also, this is a **known problem** (at least in numpy, going back to 2012 - see e.g. https://github.com/numpy/numpy/issues/593)
    - The reason why this hasn't been addressed yet is performance (see https://github.com/numpy/numpy/issues/8987#issuecomment-327378779)
        - Floating point overflows are detected at the hardware level
        - Integer overflows aren't - they would need to be checked by `numpy`/`scipy`, which is too costly for arrays
    - Trouble is, even a `uint64` with a max value of 18446744073709551615 can theoretically overflow