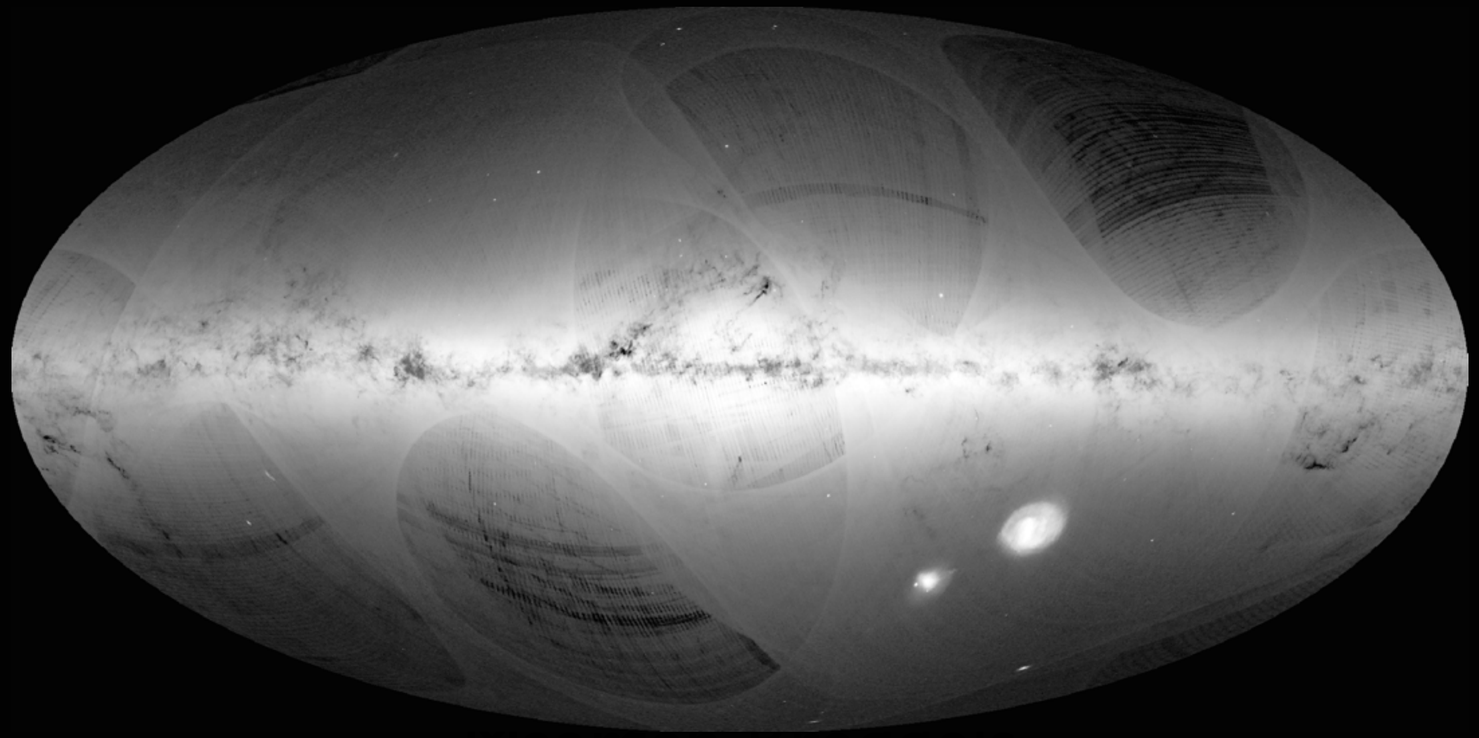
A Billion stars in the Jupyter notebook



Maarten Breddels

PyData Amsterdam - 2017

university of groningen / faculty of mathematics and natural sciences / kapteyn astronomical institute

Outline

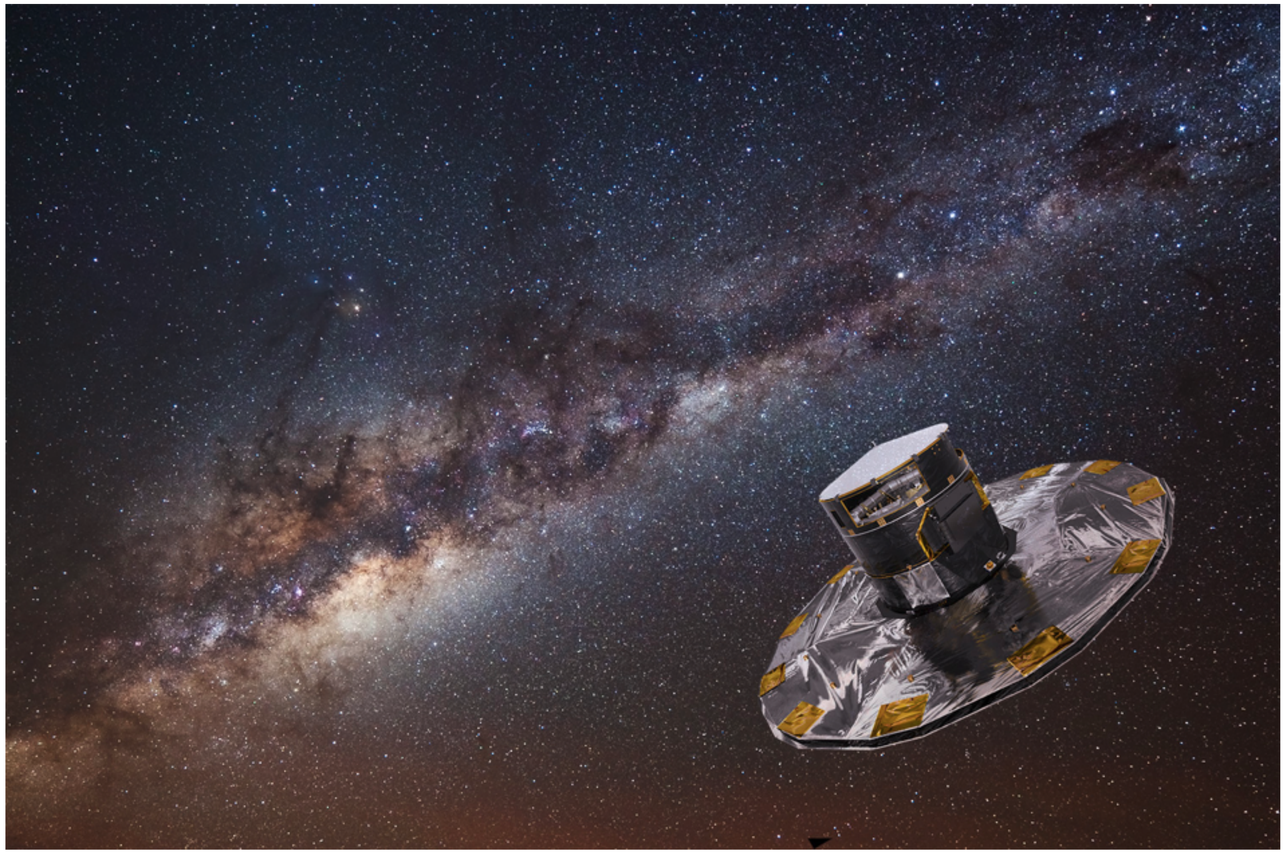
- Motivation/Problem
- Solution (vaex)
- Demo

 vaex

 ipwidgets / ipyvolume

Motivation

- Gaia satellite
  - launched by ESA in december 2013
  - determine for >10^9 stars/objects in our Milky Way
    - positions
    - velocities
    - astrophysical parameters
  - First catalogue (DR1) is out
    - sky positions, G magnitude



Motivation

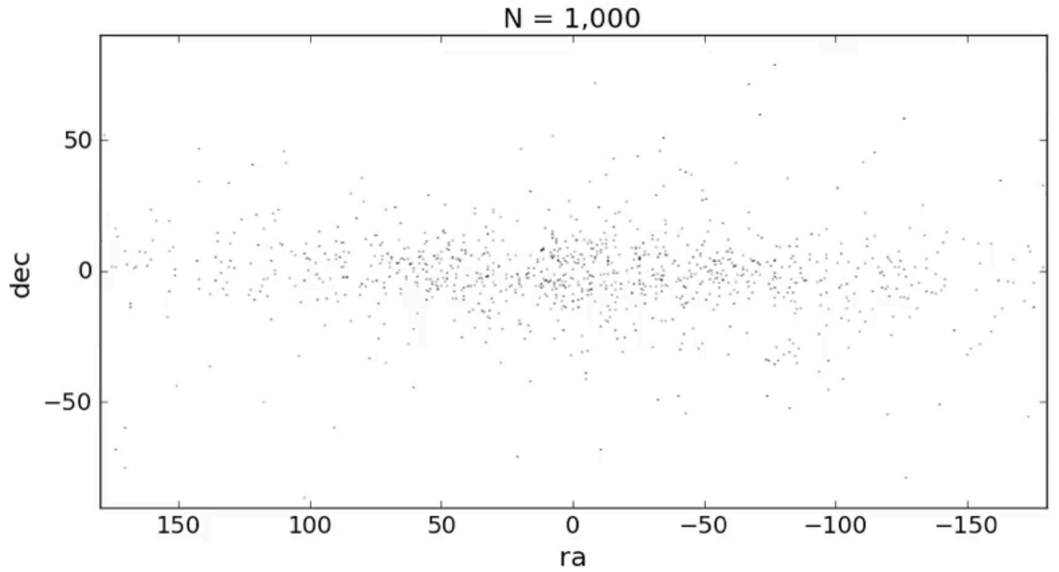
- We have Gaia DR1, soon DR2
  - > 10^9 objects/stars
- Can we visualise and explore this?
  - We want to ‘see’ the data
  - Data/Quality checks
- Science/Discovery
  - Trends, relations, clustering
  - You are the (biological) neutral network

Motivation

- We have Gaia DR1, soon DR2
  - > 10^9 objects/stars
- Can we visualise and explore this?
  - We want to ‘see’ the data
  - Data/Quality checks
- Science/Discovery
  - Trends, relations, clustering
  - You are the (biological) neutral network
- Problem
  - Scatter plots do not work well for 10^9 rows/objects
- Work with densities/statistics in 0,1,2 and 3d
  - Interactive?
  - Zoom, pan etc
  - Explore: selections/queries

Motivation

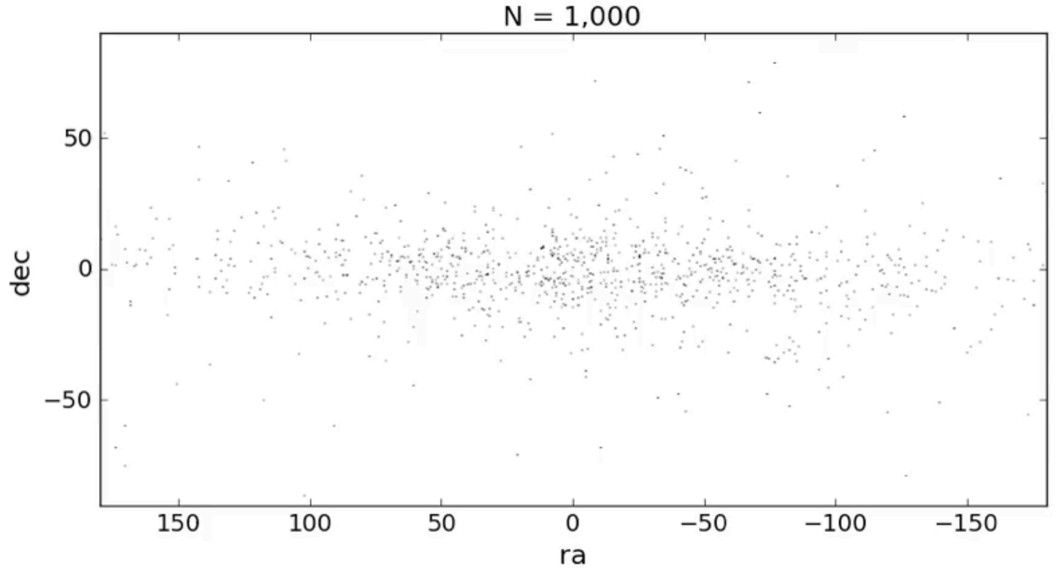
- We have Gaia DR1, soon DR2
  - > 10^9 objects/stars
- Can we visualise and explore this?
  - We want to ‘see’ the data
  - Data/Quality checks
- Science/Discovery
  - Trends, relations, clustering
  - You are the (biological) neutral network

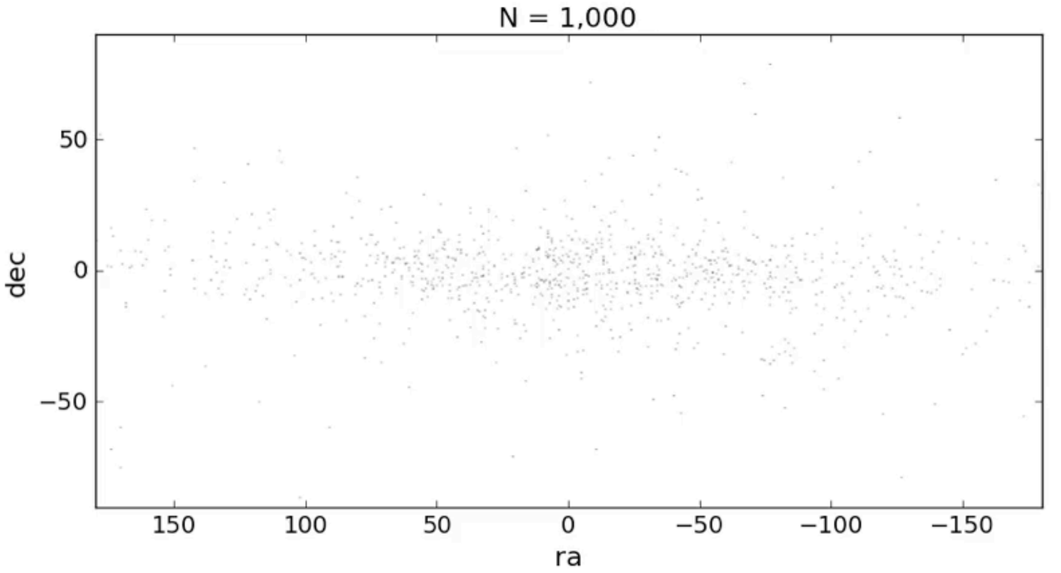


- Problem
  - Scatter plots do not work well for 10^9 rows/objects
- Work with densities/statistics in 0,1,2 and 3d
  - Interactive?
  - Zoom, pan etc
  - Explore: selections/queries

Motivation

- We have Gaia DR1, soon DR2
  - > 10^9 objects/stars
- Can we visualise and explore this?
  - We want to ‘see’ the data
  - Data/Quality checks
- Science/Discovery
  - Trends, relations, clustering
  - You are the (biological) neutral network
- Problem
  - Scatter plots do not work well for 10^9 rows/objects
- Work with densities/statistics in 0,1,2 and 3d
  - Interactive?
  - Zoom, pan etc
  - Explore: selections/queries



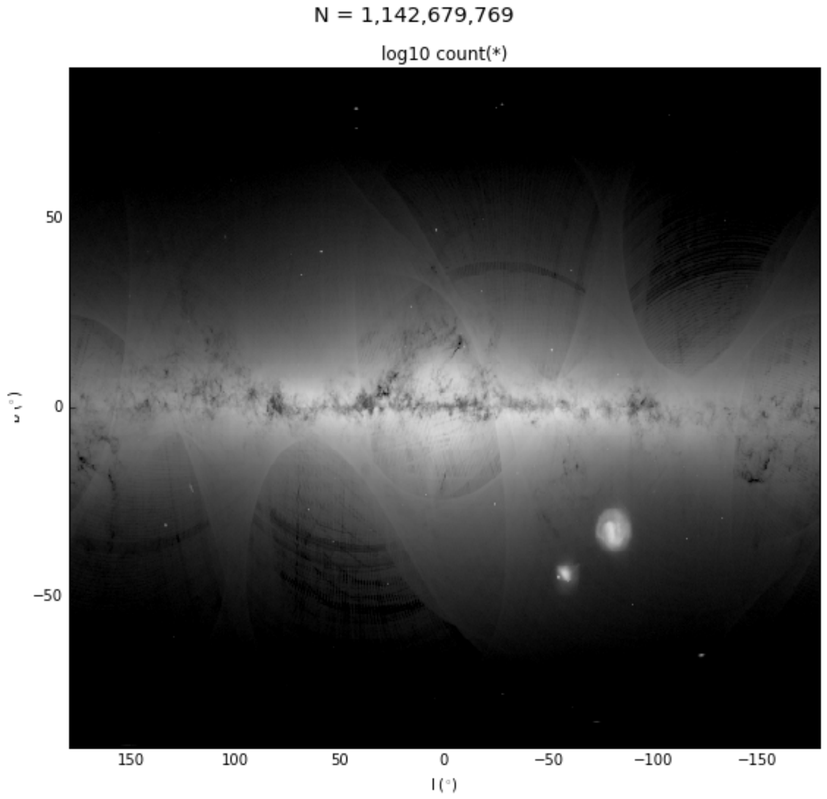


How fast can it be processed?

- What can be done?
  - 10^9 * 2 * 8 bytes = 15 GiB (double is 8 bytes)
  - Memory bandwidth: 10-20 GiB/s: ~1 second
  - CPU: 3 Ghz (but multicore, say 4-8): 12-24 cycles/second
  - Few cycles per row/object, simple algorithm
    - Histograms/Density grids

How fast can it be processed?

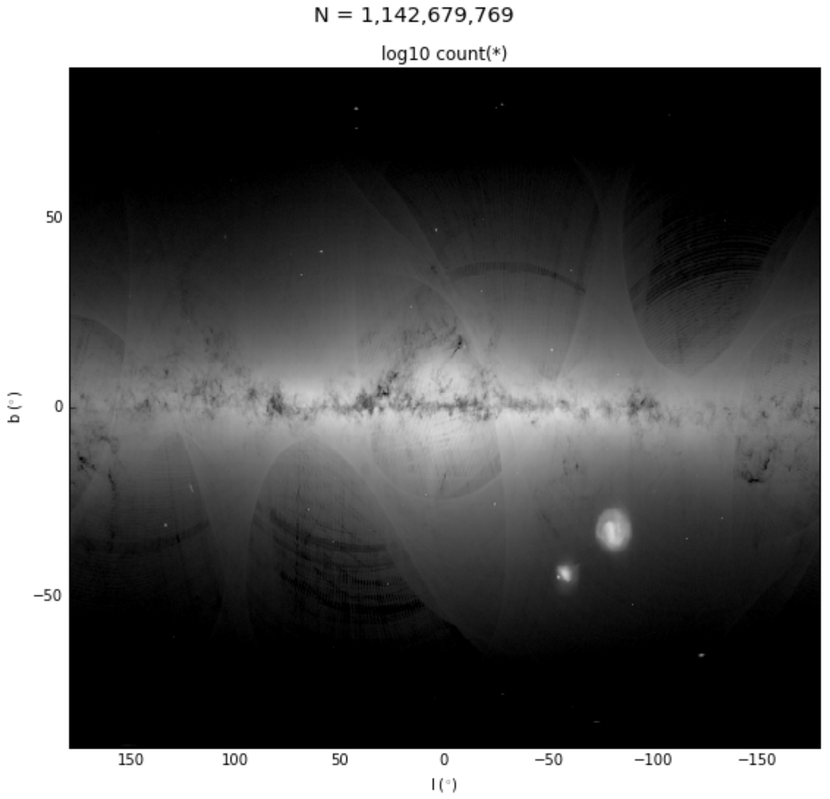
- What can be done?
  - 10^9 * 2 * 8 bytes = 15 GiB (double is 8 bytes)
  - Memory bandwidth: 10-20 GiB/s: ~1 second
  - CPU: 3 Ghz (but multicore, say 4-8): 12-24 cycles/second
  - Few cycles per row/object, simple algorithm
    - Histograms/Density grids
- ~1 second



How fast can it be processed?

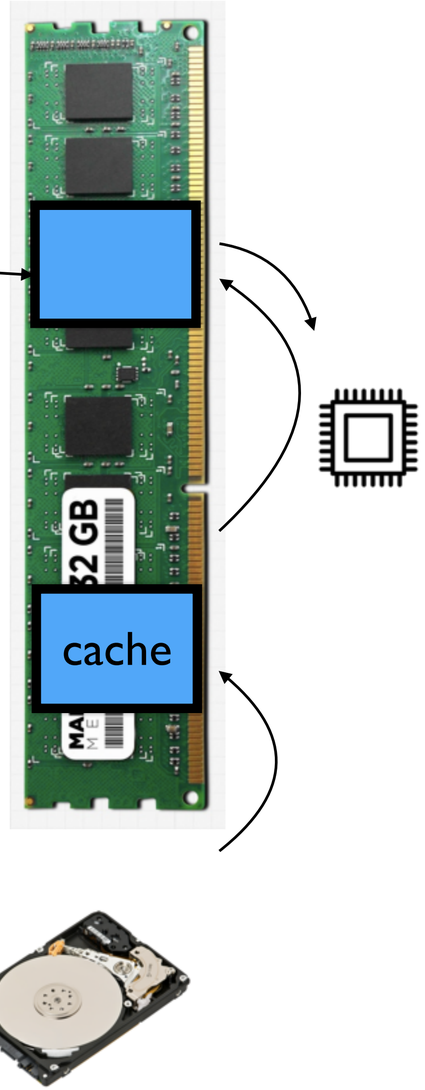
- What can be done?
  - 10^9 * 2 * 8 bytes = 15 GiB (double is 8 bytes)
  - Memory bandwidth: 10-20 GiB/s: ~1 second
  - CPU: 3 Ghz (but multicore, say 4-8): 12-24 cycles/second
  - Few cycles per row/object, simple algorithm
    - Histograms/Density grids
- Yes, but
  - If it fits/cached in memory, otherwise ssd/hdd speeds (10-100 seconds)
  - proper storage and reading of data (no .csv!)
  - simple and fast algorithm for binning

- ~1 second



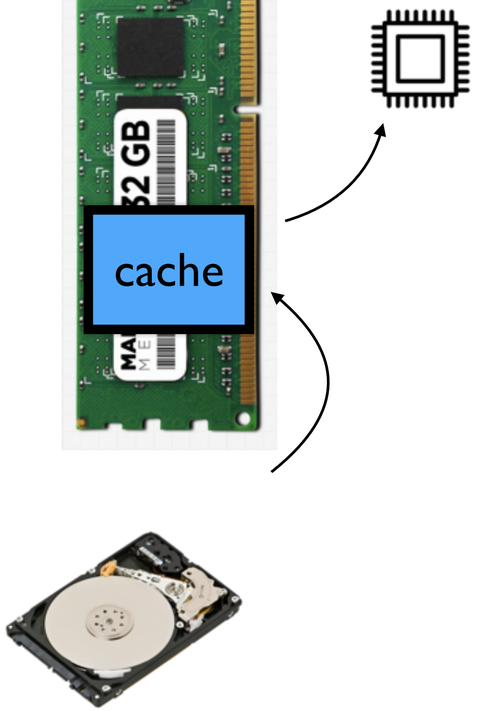
How to store and read the data

- Storage: native, column based (hdf5)
- Normal (POSIX read) method:
  - Allocate memory
  - read from disk to memory
  - Actually: from disk, to OS cache, to memory
  - Wastes memory/cache
  - 15 GB data , requires 30 GB is you want to use the file system cache



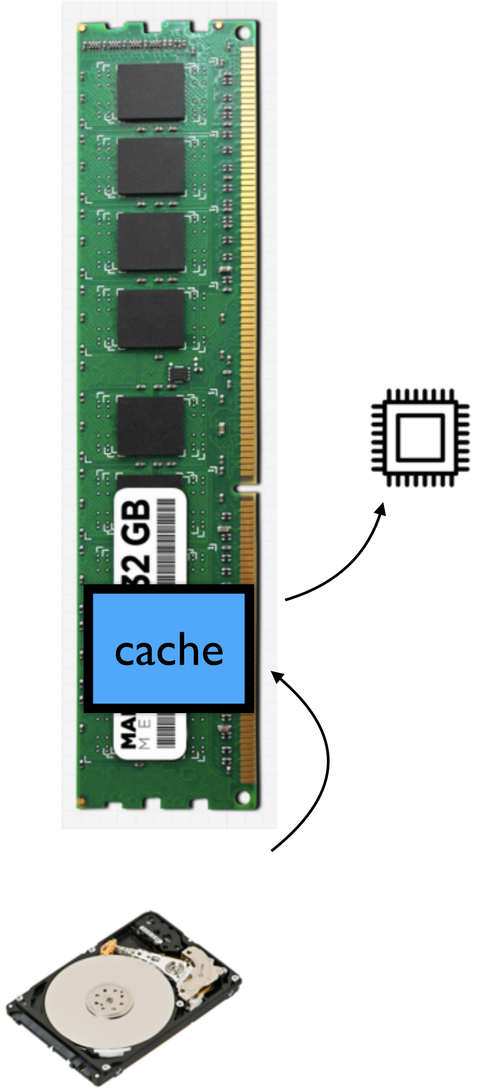
How to store and read the data

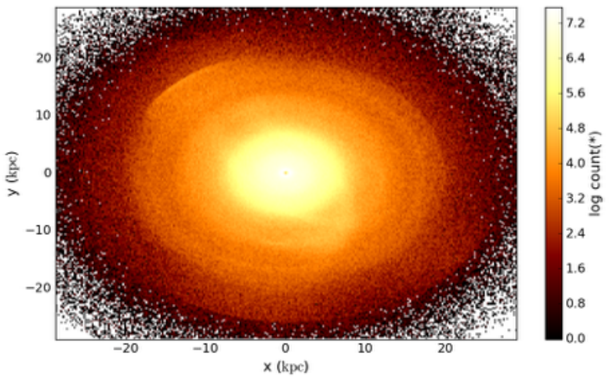
- Storage: native, column based (hdf5)
- Normal (POSIX read) method:
  - Allocate memory
  - read from disk to memory
  - Actually: from disk, to OS cache, to memory
  - Wastes memory/cache
  - 15 GB data , requires 30 GB is you want to use the file system cache



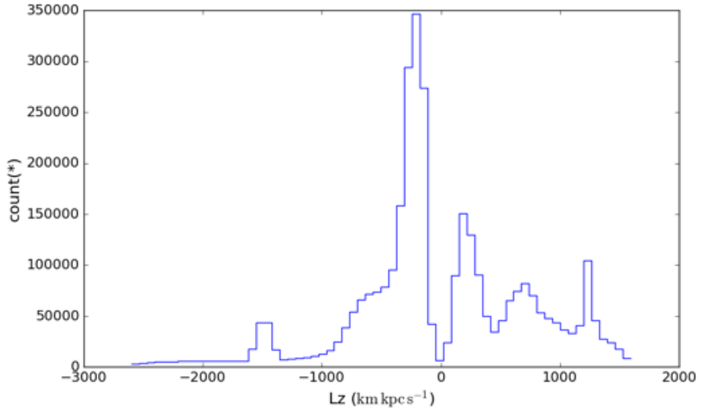
How to store and read the data

- Storage: native, column based (hdf5)
- Normal (POSIX read) method:
  - Allocate memory
  - read from disk to memory
  - Actually: from disk, to OS cache, to memory
  - Wastes memory/cache
  - 15 GB data , requires 30 GB is you want to use the file system cache
- Memory mapping:
  - get direct access to OS memory cache, no copy, no setup (apart from the kernel doing setting up the pages)
  - avoid memory copies, more cache available
- In previous example:
  - copying 15 GB will take about ~1.0 second, at 10-20 GB/s
  - Can be 2-3x slower (cpu cache helps a bit)

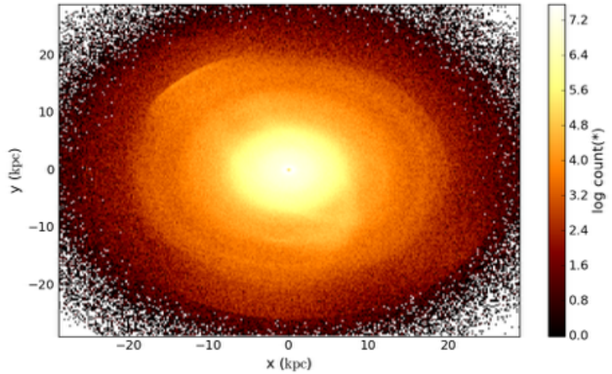



20
10
0
−10
−20
y (kpc)
−20
−10
0
10
20
x (kpc)
7.2
6.4
5.6
4.8
4.0
3.2
2.4
1.6
0.8
0.0
log count(*)

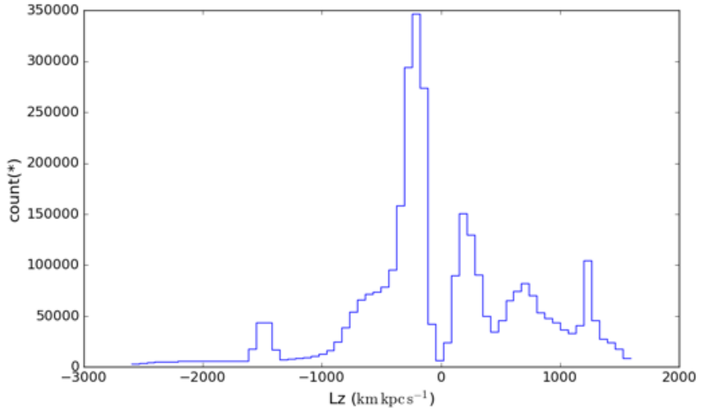
1d


350000
300000
250000
200000
150000
100000
50000
0
count(*)
−3000
−2000
−1000
0
1000
2000
Lz (km kpc s^{-1})

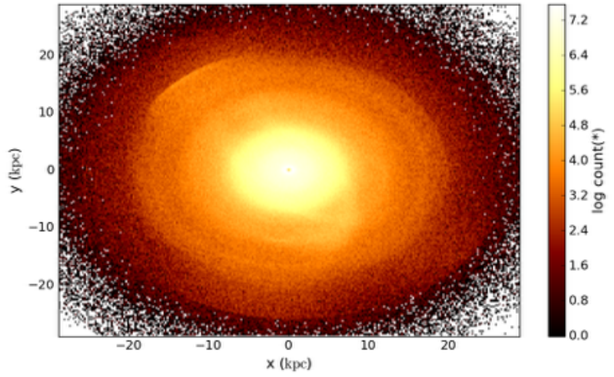
2d


20
10
0
−10
−20
y (kpc)
−20
−10
0
10
20
x (kpc)
7.2
6.4
5.6
4.8
4.0
3.2
2.4
1.6
0.8
0.0
log count(*)

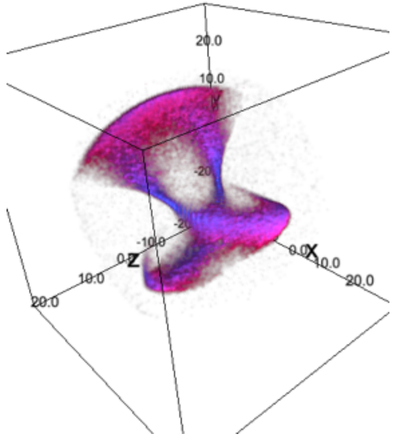
1d



2d

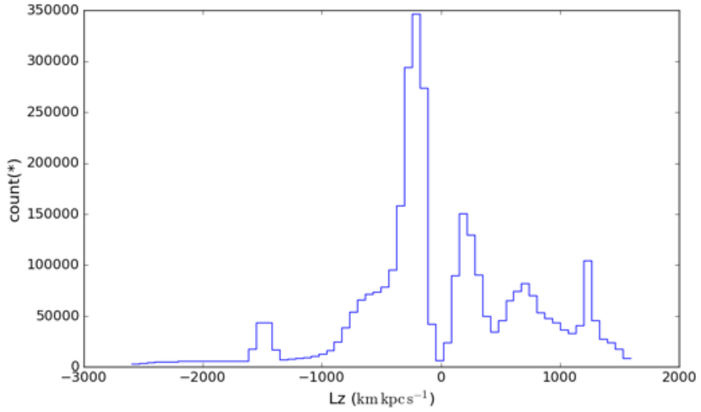


3d

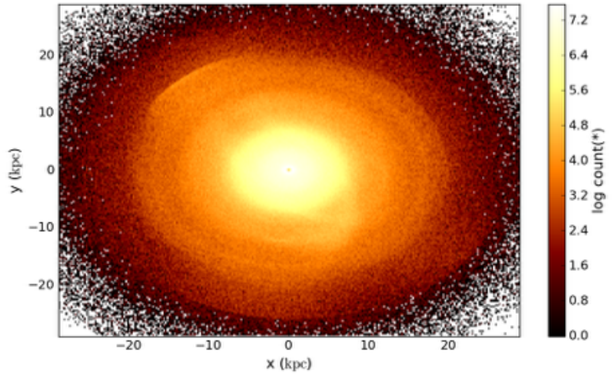


0d 330,000 rows

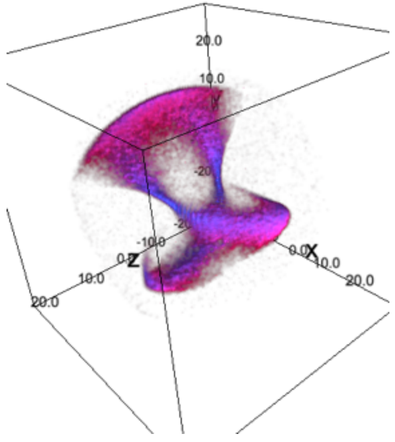
1d



2d

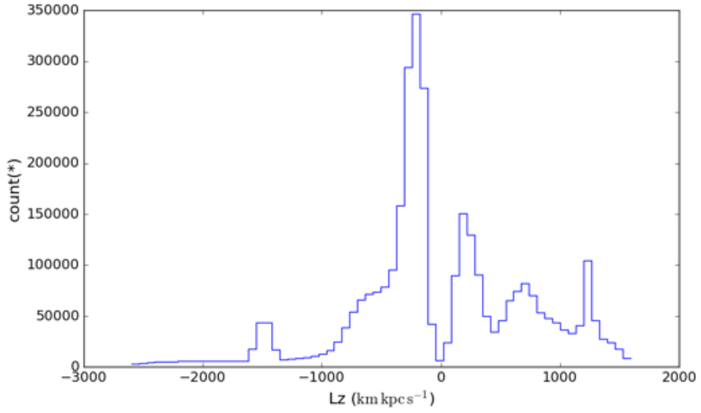


3d

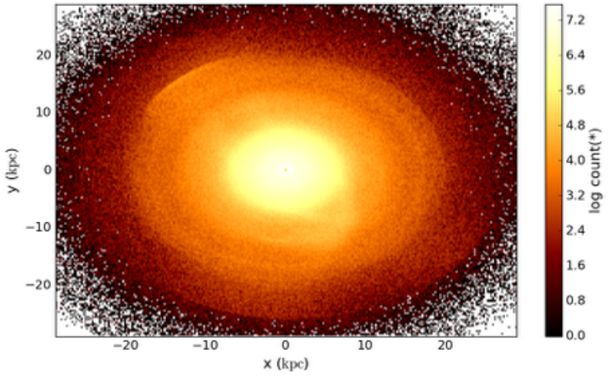


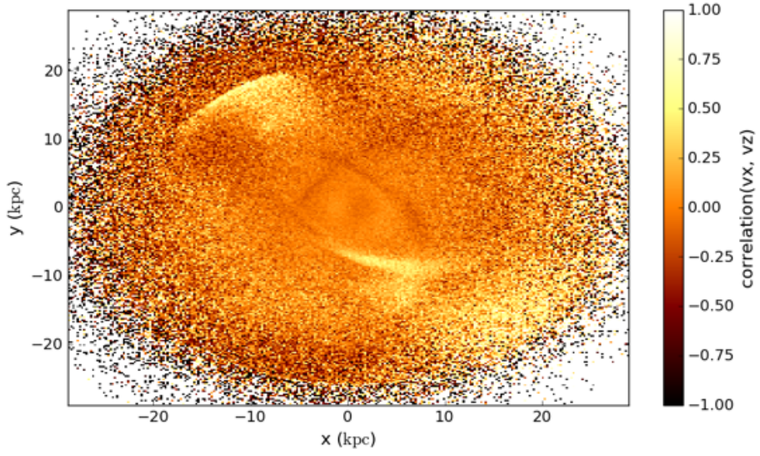
0d 330,000 rows

1d

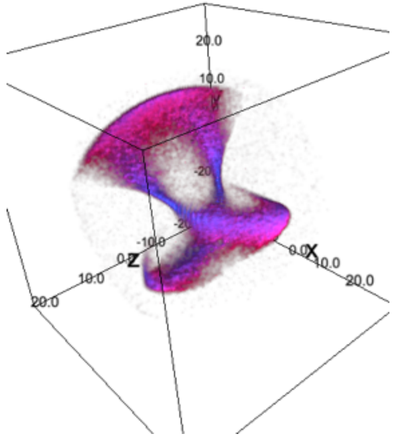


2d



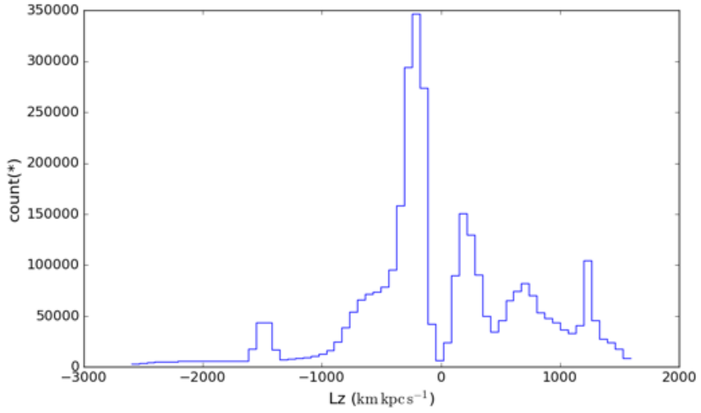


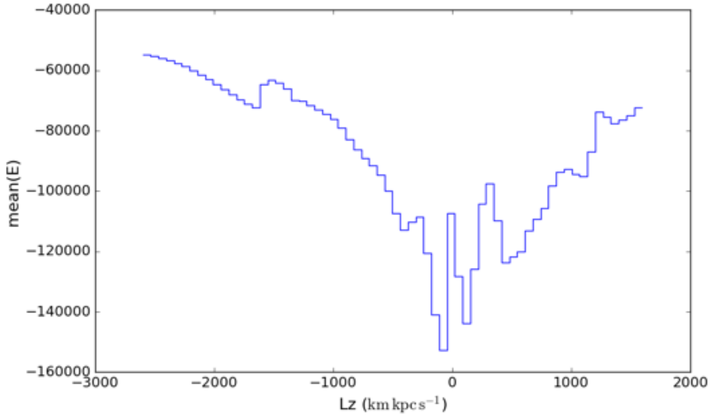
3d



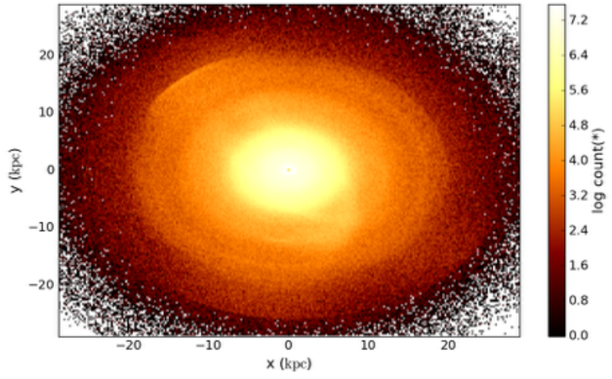
0d 330,000 rows

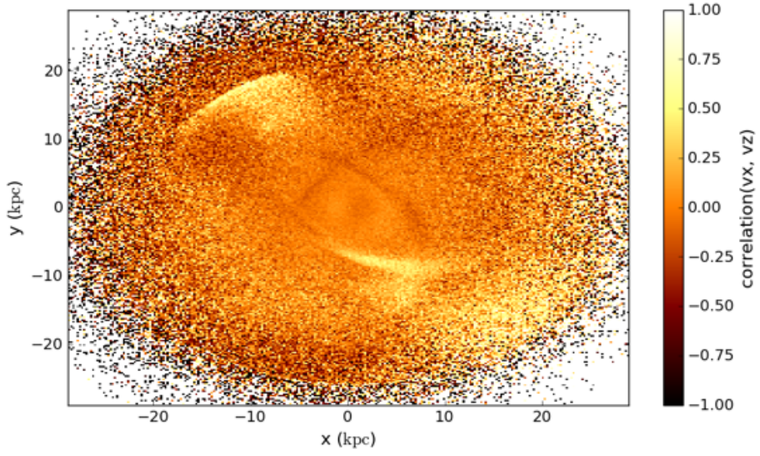
1d



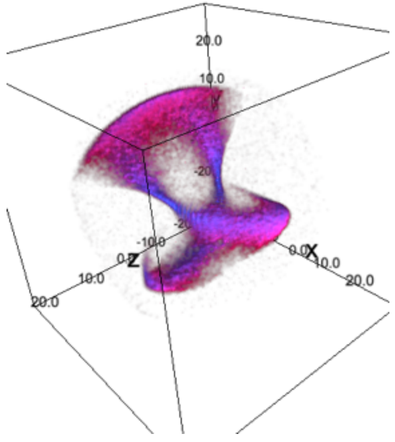


2d





3d

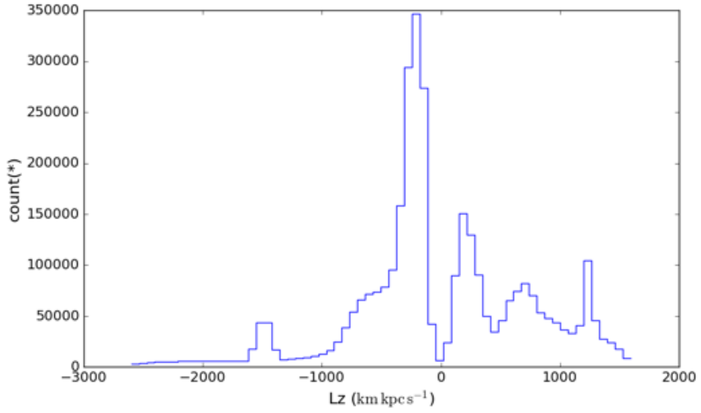


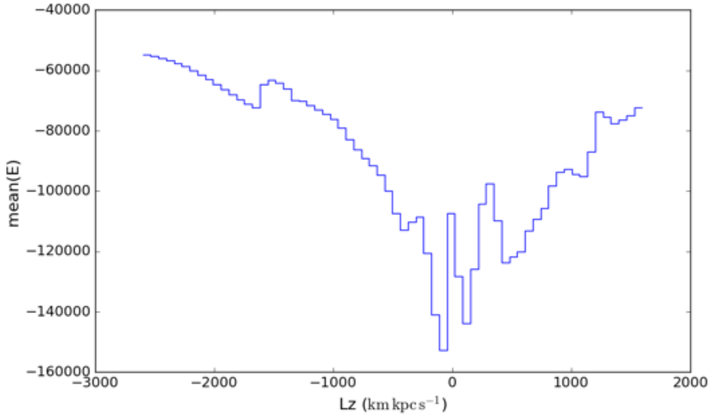
0d

330,000 rows

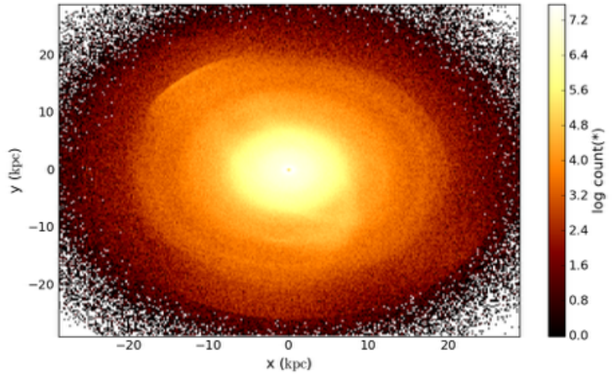
mean: -0.083

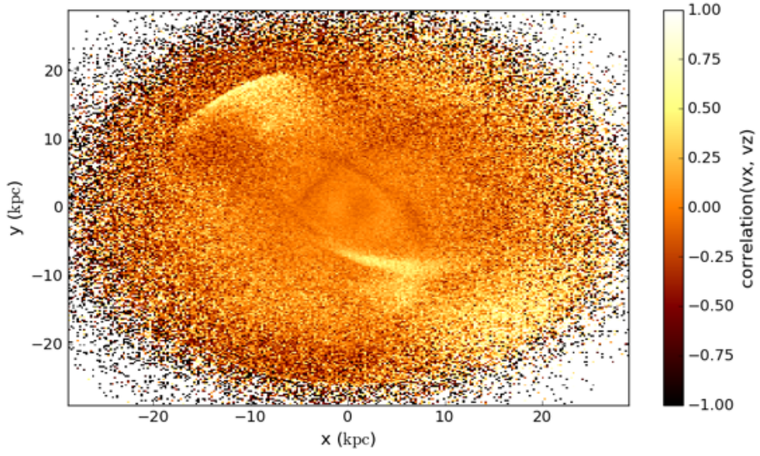
1d



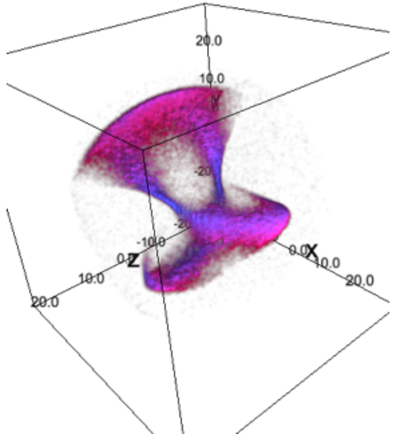


2d





3d

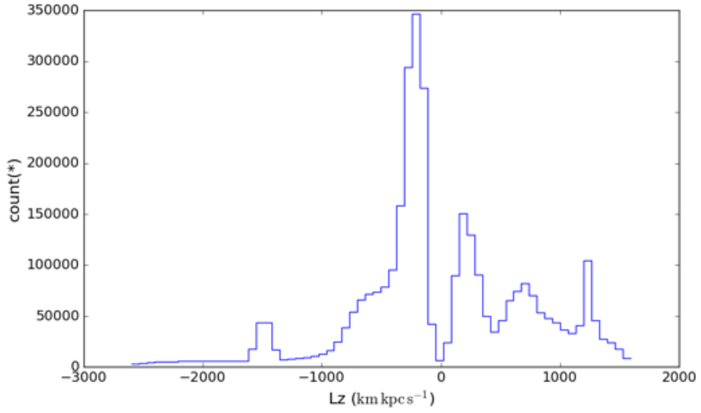


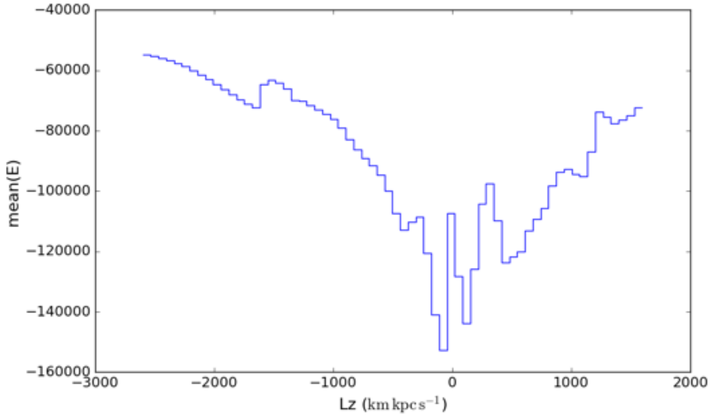
0d

330,000 rows

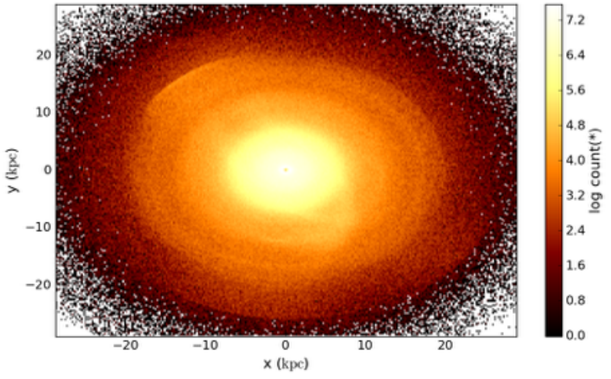
mean: -0.083

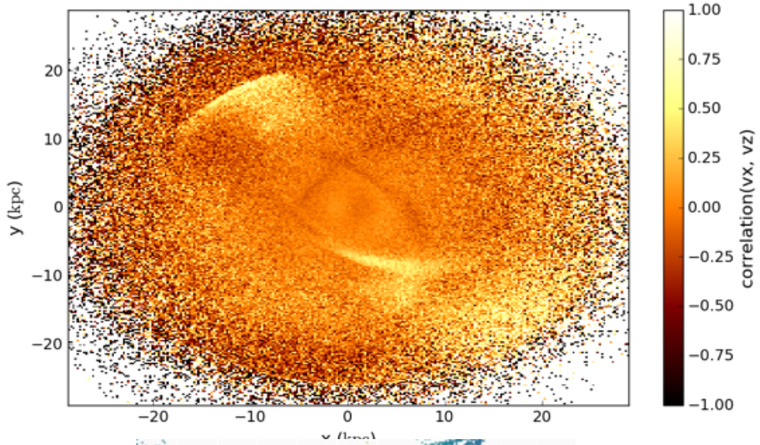
1d



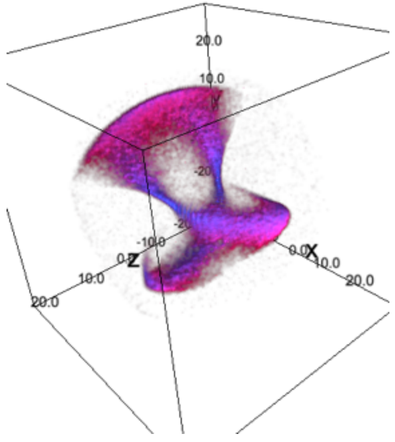


2d





3d



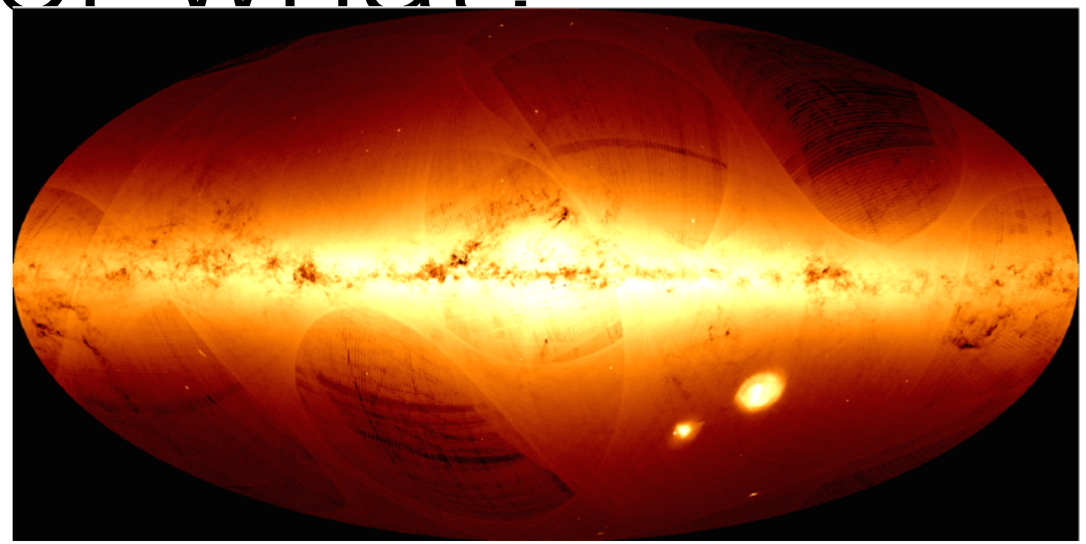
vaex

- Python library (conda/pip installable)
- Think pandas-like but for large datasets
- simple api: 1 class (Dataset), the rest are strings/dicts/tuples/lists/ndarray
- Focusses mostly on statistics on N-d grids (count/mean/max/std/…)
- >1 billion rows / sec on a `decent` desktop (quad core 3Gz)
  - >50x faster than scipy.stats.binned_statistic_2d
  - >2x faster then datashader
- 0.1-0.2 billion rows on this 2yr old Macbook Air
  - CPU bound
- Does visualisation / matplotlib / bqplot / ipyvolume / ipyleaflet

For what?

- Astronomical catalogues
  - Gaia > 1 billion
  - SDSS > 150 million
  - PANSTARRS > 1 billion
  - LSST > 10 billion
- N-body simulations (0.1-100 billion)
  - columns: x,y,z,vx,vy,vz
    - SPH: density, temperature, metallicity…
  - rows: particle
- Any tabular data
  - rows and columns

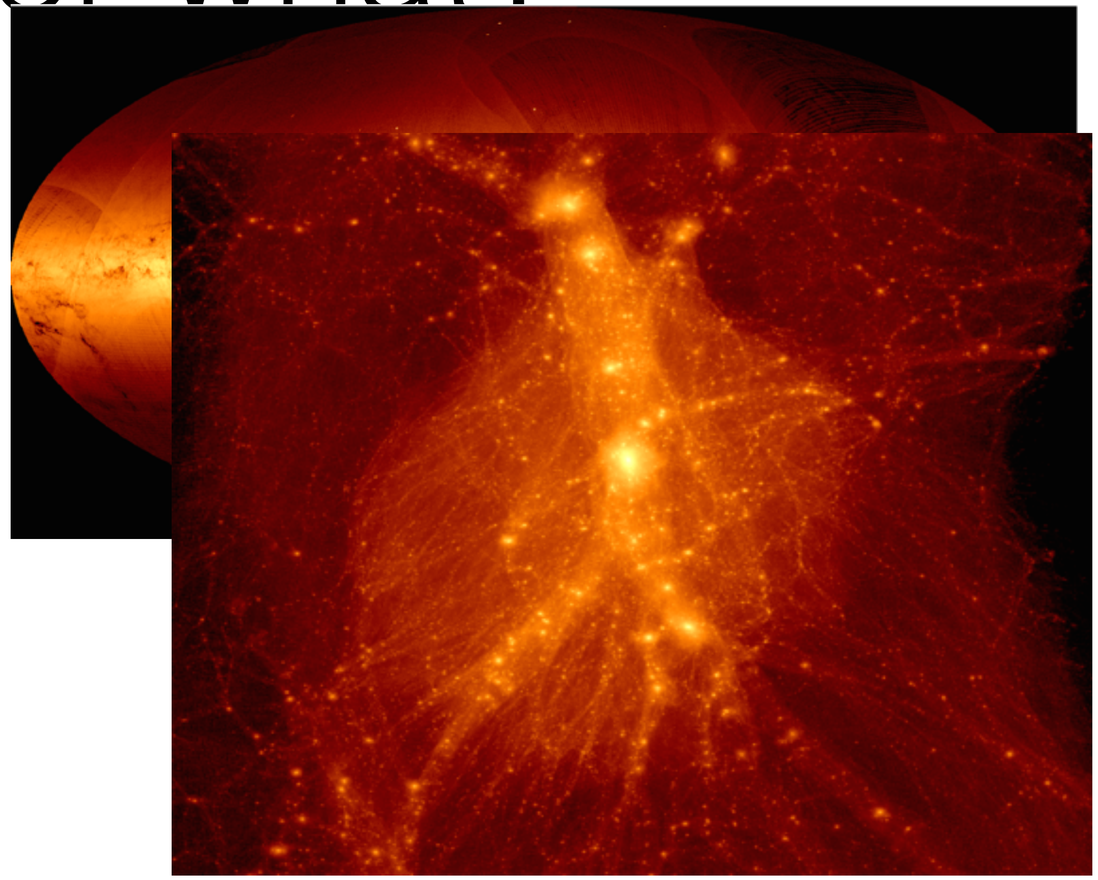
For what?



- Astronomical catalogues
  - Gaia > 1 billion
  - SDSS > 150 million
  - PANSTARRS > 1 billion
  - LSST > 10 billion
- N-body simulations (0.1-100 billion)
  - columns: x,y,z,vx,vy,vz
    - SPH: density, temperature, metallicity…
  - rows: particle
- Any tabular data
  - rows and columns

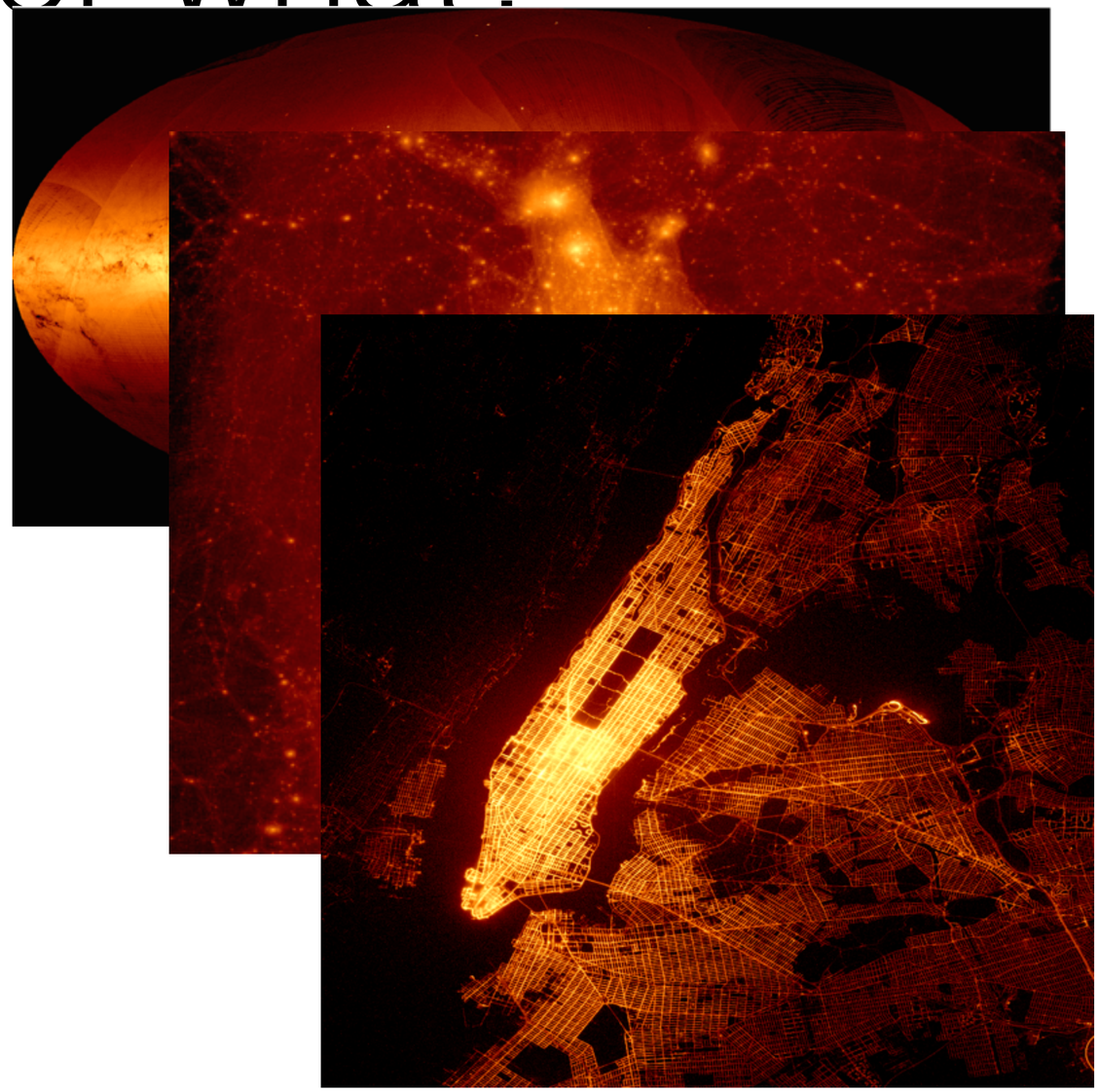
For what?

- Astronomical catalogues
  - Gaia > 1 billion
  - SDSS > 150 million
  - PANSTARRS > 1 billion
  - LSST > 10 billion
- N-body simulations (0.1-100 billion)
  - columns: x,y,z,vx,vy,vz
    - SPH: density, temperature, metallicity…
  - rows: particle
- Any tabular data
  - rows and columns



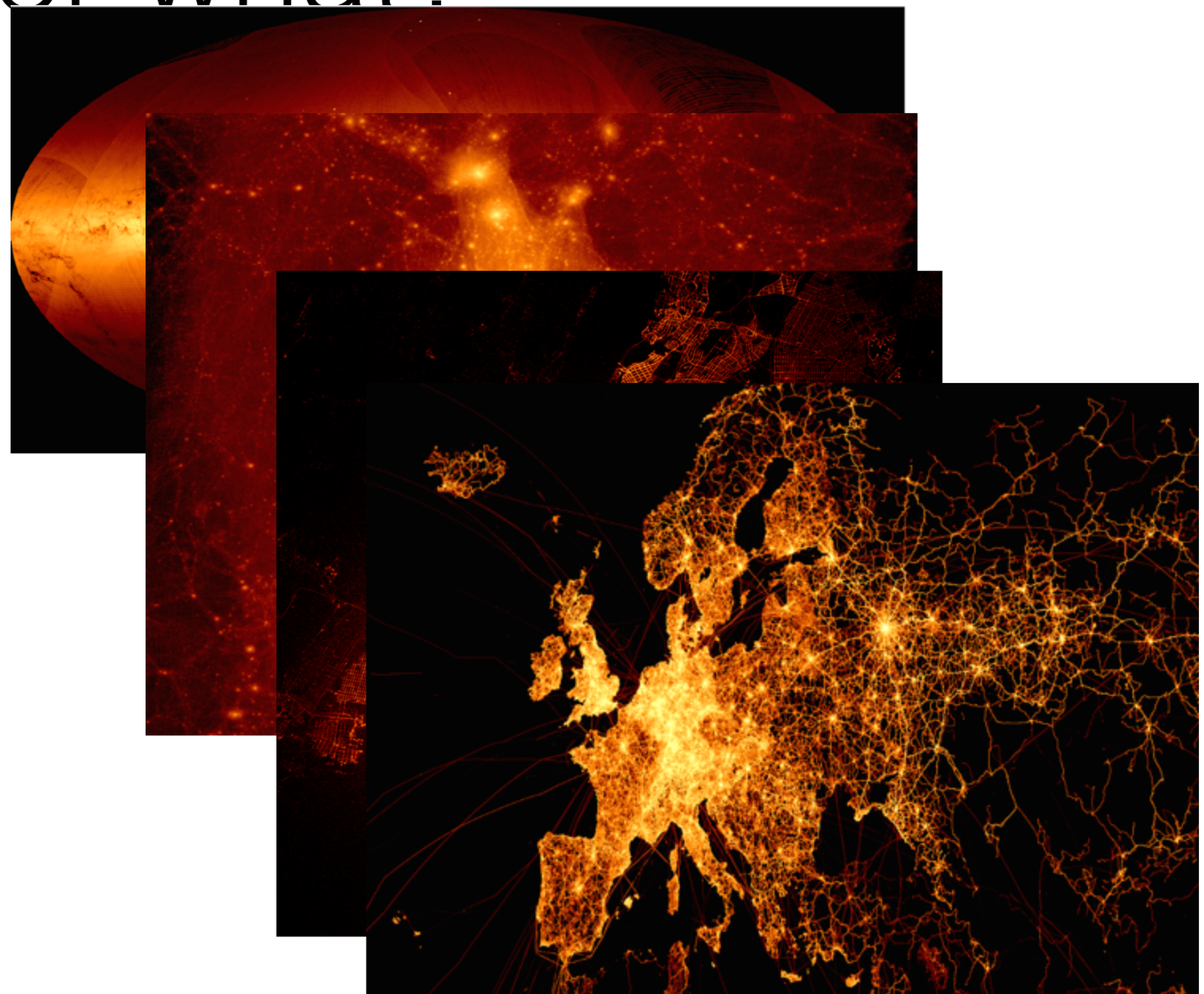
For what?

- Astronomical catalogues
  - Gaia > 1 billion
  - SDSS > 150 million
  - PANSTARRS > 1 billion
  - LSST > 10 billion
- N-body simulations (0.1-100 billion)
  - columns: x,y,z,vx,vy,vz
    - SPH: density, temperature, metallicity…
  - rows: particle
- Any tabular data
  - rows and columns



For what?

- Astronomical catalogues
  - Gaia > 1 billion
  - SDSS > 150 million
  - PANSTARRS > 1 billion
  - LSST > 10 billion
- N-body simulations (0.1-100 billion)
  - columns: x,y,z,vx,vy,vz
    - SPH: density, temperature, metallicity…
  - rows: particle
- Any tabular data
  - rows and columns



Transformation of the data

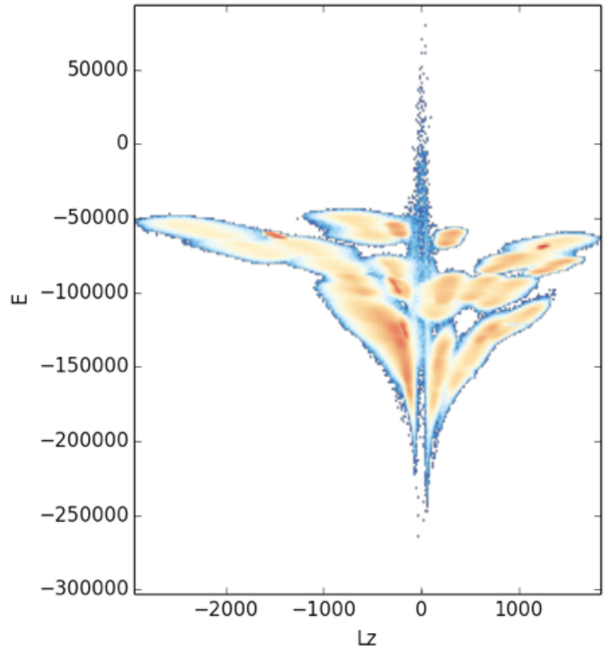
- Not ‘z = x + y’
  - for 1 billion rows: 8GB of RAM wasted
- Everything is an expression: `dataset.mean('x + y')`
- Computed in chunks not to waste RAM
  - Virtual columns:
    - `dataset.add_virtual_column('r', 'sqrt(x**2+y**2)')`
    - `dataset.mean('r')`

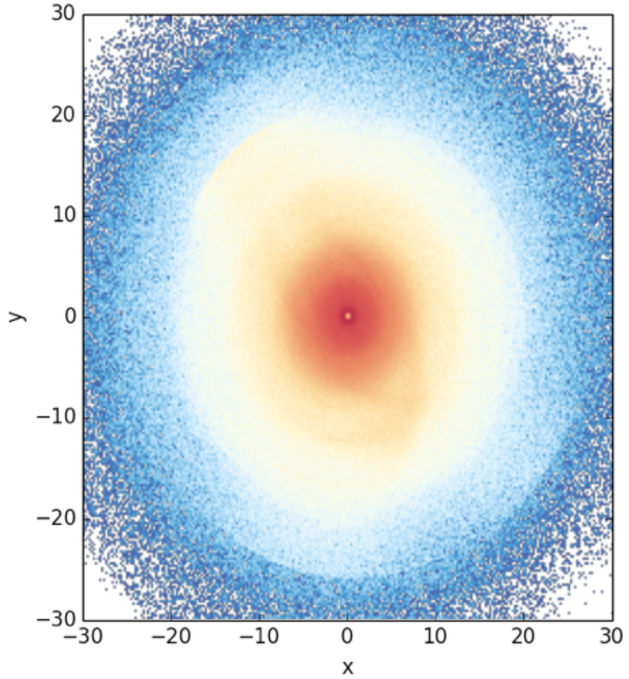
Selections/subsets

- You don’t use all data / filtering
- Efficient subsets (do NOT copy the data)
- vaex
  - Selections result in a boolean mask
    - `dataset.select(‘x > 0’, name=‘xpos’)`
  - Use in any statistic
    - `dataset.mean(‘x + y’, selection=‘xpos’)`

Selections/subsets

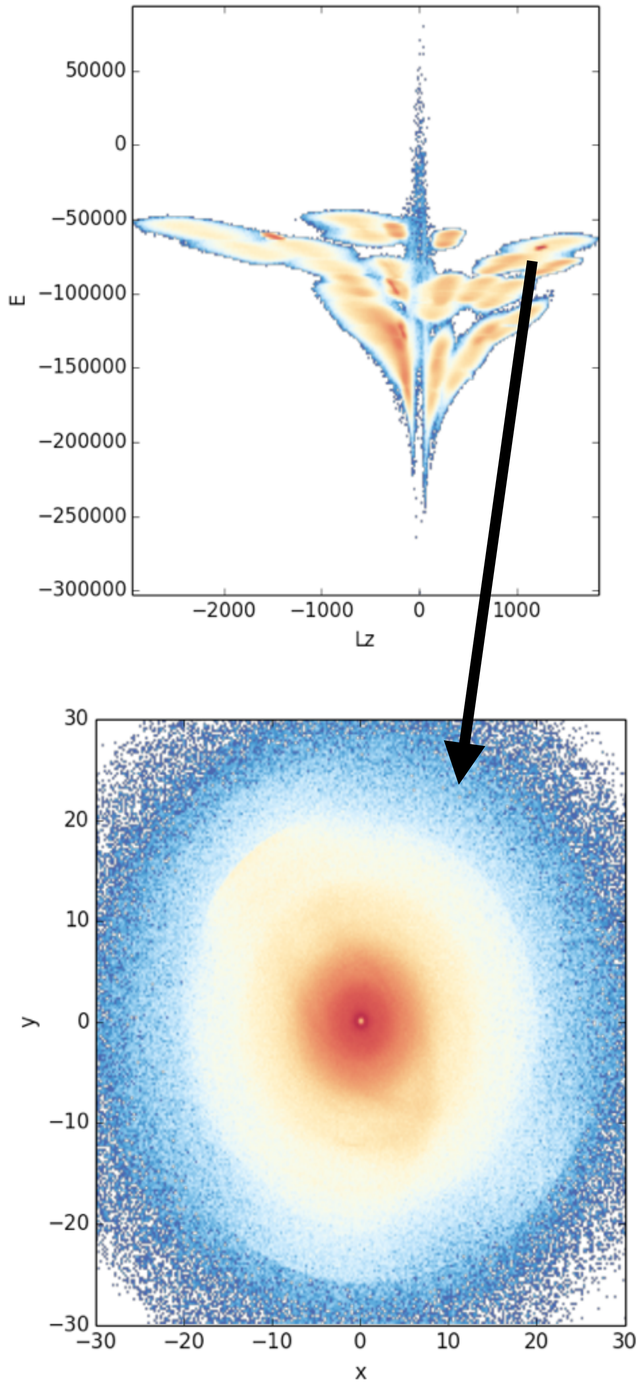
- You don’t use all data / filtering
- Efficient subsets (do NOT copy the data)
- vaex
  - Selections result in a boolean mask
    - `dataset.select('x > 0', name='xpos')`
  - Use in any statistic
    - `dataset.mean('x + y', selection='xpos')`





Selections/subsets

- You don’t use all data / filtering
- Efficient subsets (do NOT copy the data)
- vaex
  - Selections result in a boolean mask
    - `dataset.select('x > 0', name='xpos')`
  - Use in any statistic
    - `dataset.mean('x + y', selection='xpos')`



Demo library

Conclusions

- vaex
  - N-d statistics on regular grids
  - ~1 billion rows/sec
  - efficient transformations/selections
  - regular grids can be used for visualisation
  - Interactive in the Notebook due to ipywidgets/bqplot/ipyleaflet
- ipyvolume
  - adds interactive 3d plotting to the notebook, (as an ipywidget)
  - can be used as backend for vaex
  - works outside the notebook, in the browser
    - tablet (paperless office)
    - sharing, outreach, press release material

Questions

- maartenbreddels@gmail.com
- twitter @maartenbreddels
- vaex
  - http://vaex.astro.rug.nl
  - https://github.com/maartenbreddels/vaex
  - pip install —pre vaex / conda install -c conda-forge vaex
- ipyvolume
  - https://ipywidgets.readthedocs.io
  - https://github.com/maartenbreddels/ipyvolume
  - pip install ipyvolume / conda install -c conda-forge ipyvolume
  - if pip: jupyter nbextension enable --py --user ipyvolume (but really, use anaconda)
- Notebook:
  - https://github.com/maartenbreddels/pydata-amsterdam-2017