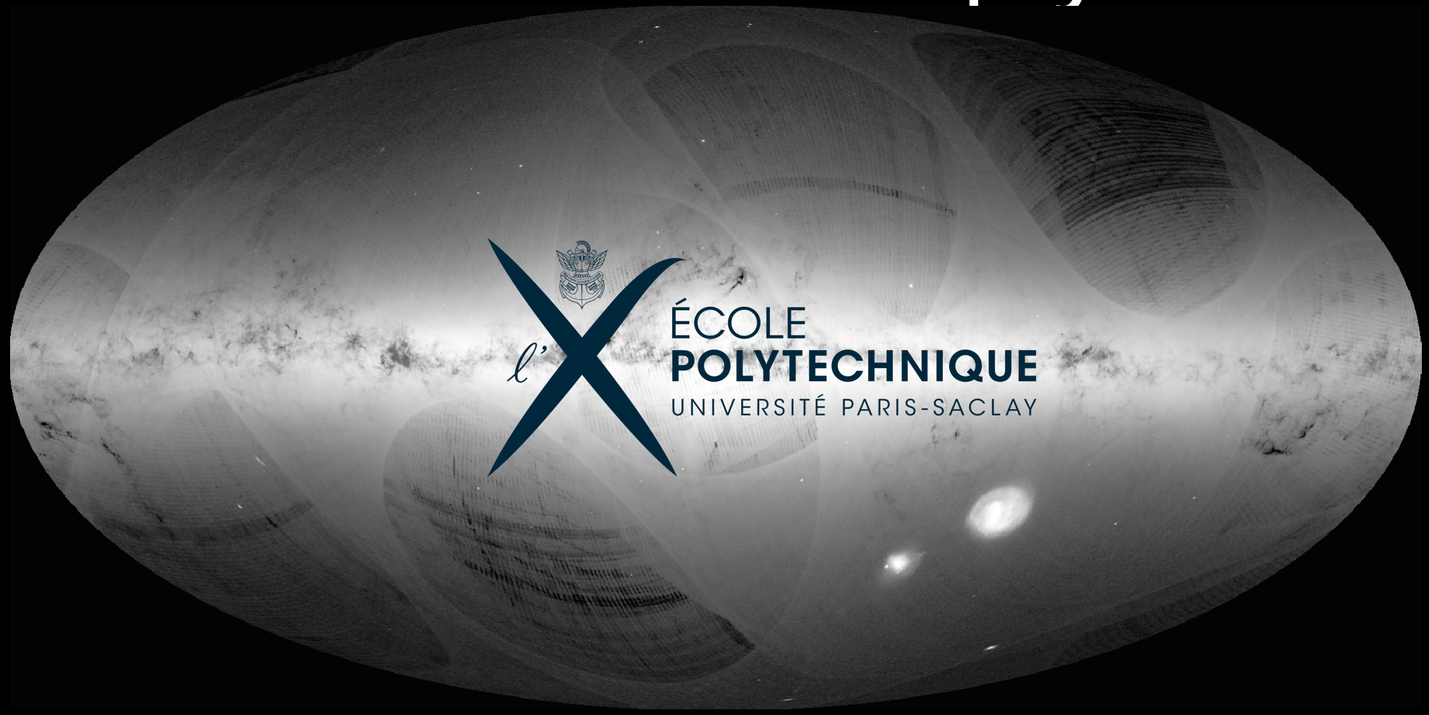
# A Billion Stars in the Jupyter Notebook




**MAARTEN BREDDELS**
data science / python / jupyter

Paris - Jupyter Day - 2018

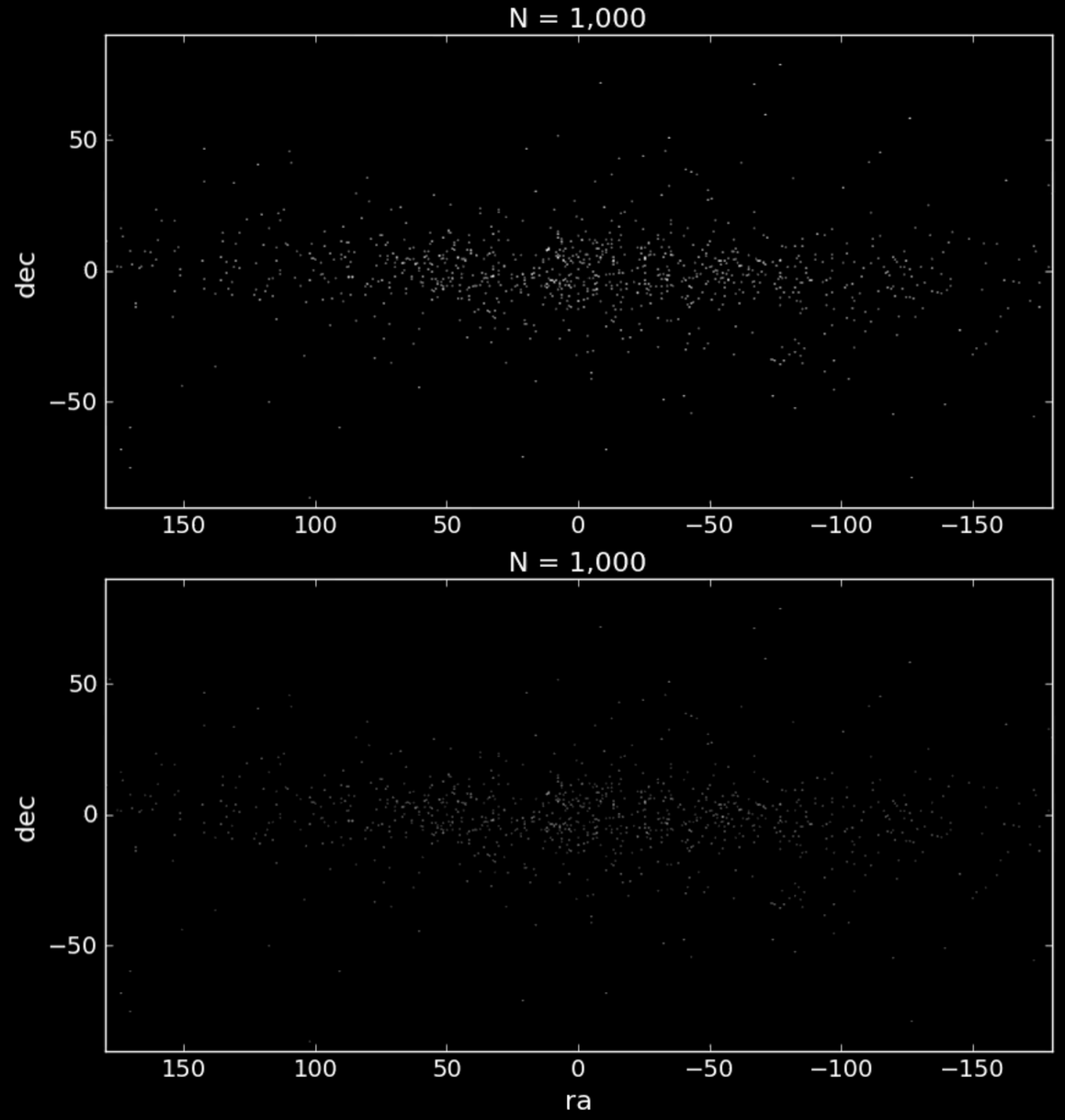
university of groningen / faculty of mathematics and natural sciences / kapteyn astronomical institute

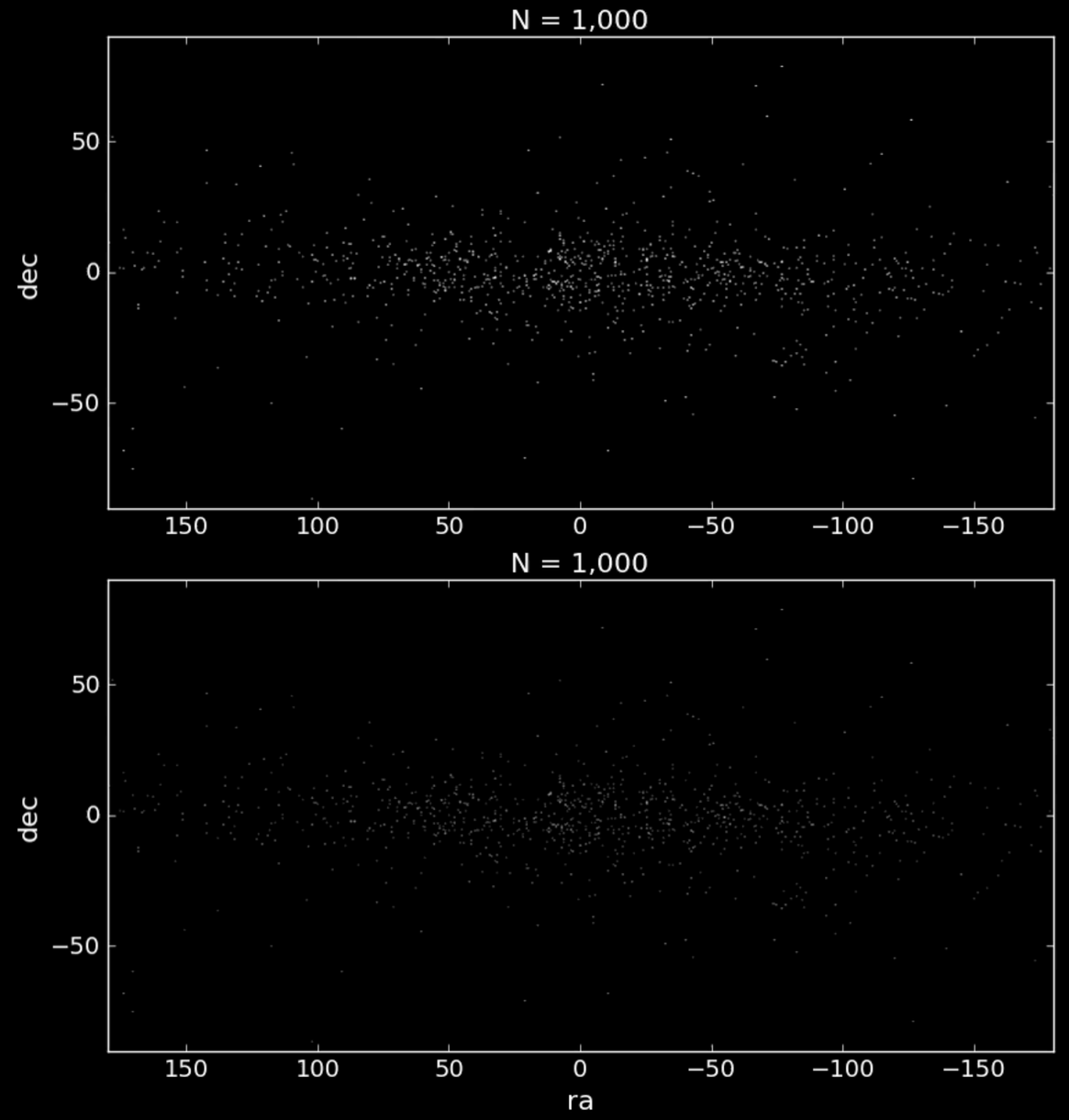
# Agenda

- Show how to deal with a billion objects/rows/stars?
- Why use/move to the notebook?
- How can we make the notebook interactive?

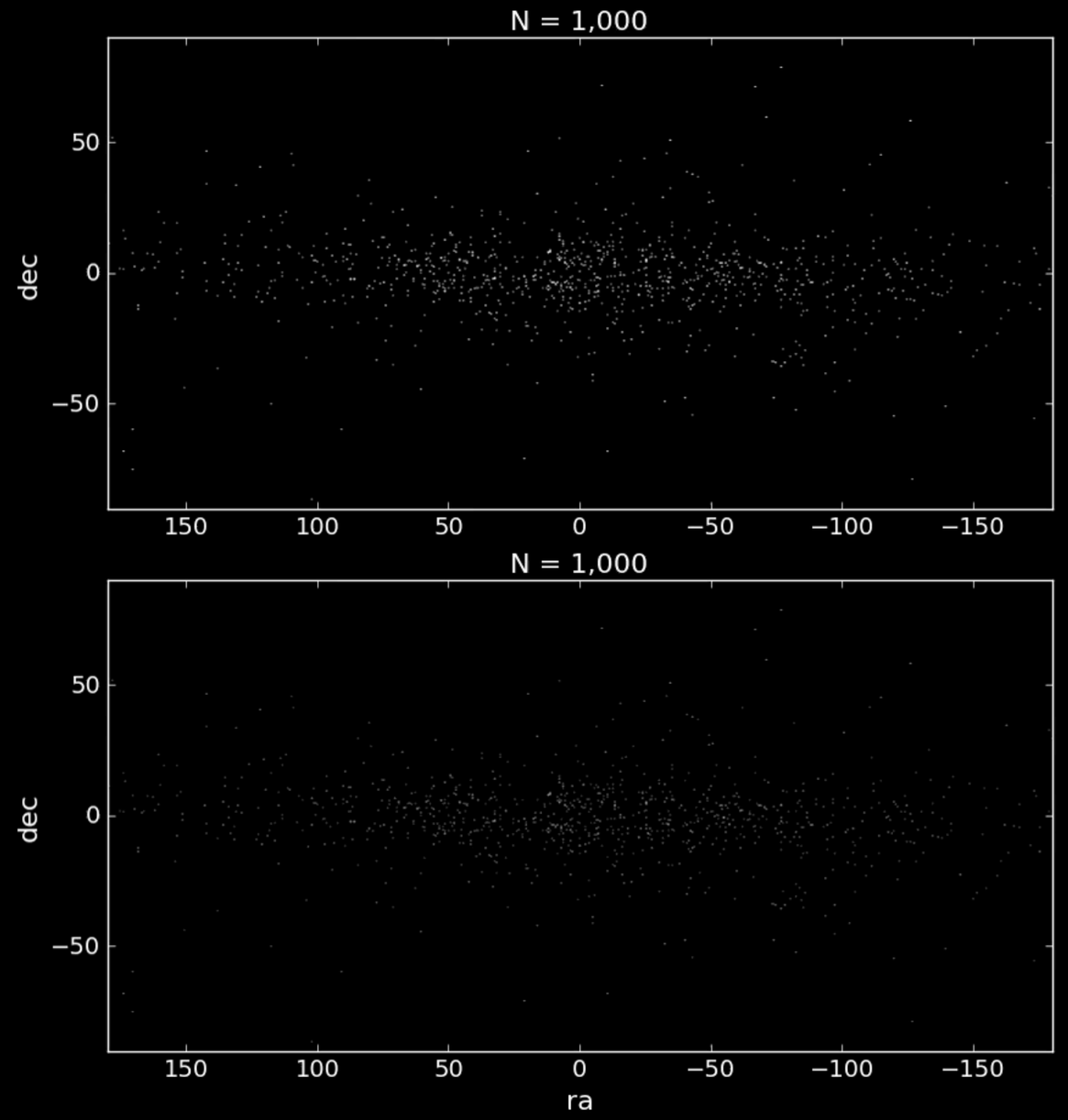
# Motivation: Gaia

>1,000,000,000 stars/objects


N = 1,000
dec
50
0
−50
150
100
50
0
−50
−100
−150
N = 1,000
dec
50
0
−50
150
100
50
0
−50
−100
−150
ra


N = 1,000
50
0
−50
dec
150
100
50
0
−50
−100
−150
N = 1,000
50
0
−50
dec
150
100
50
0
−50
−100
−150
ra

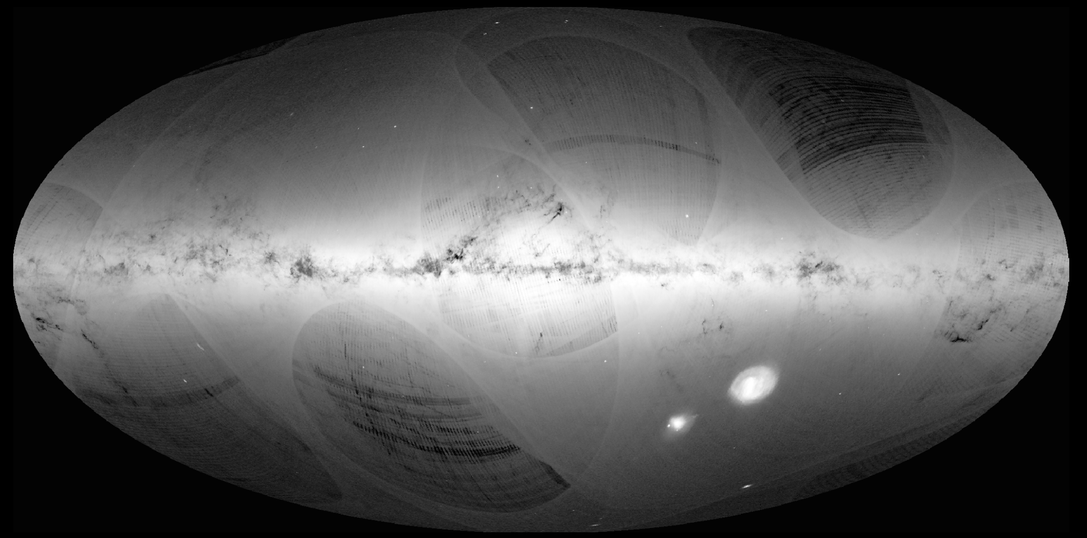
scatter



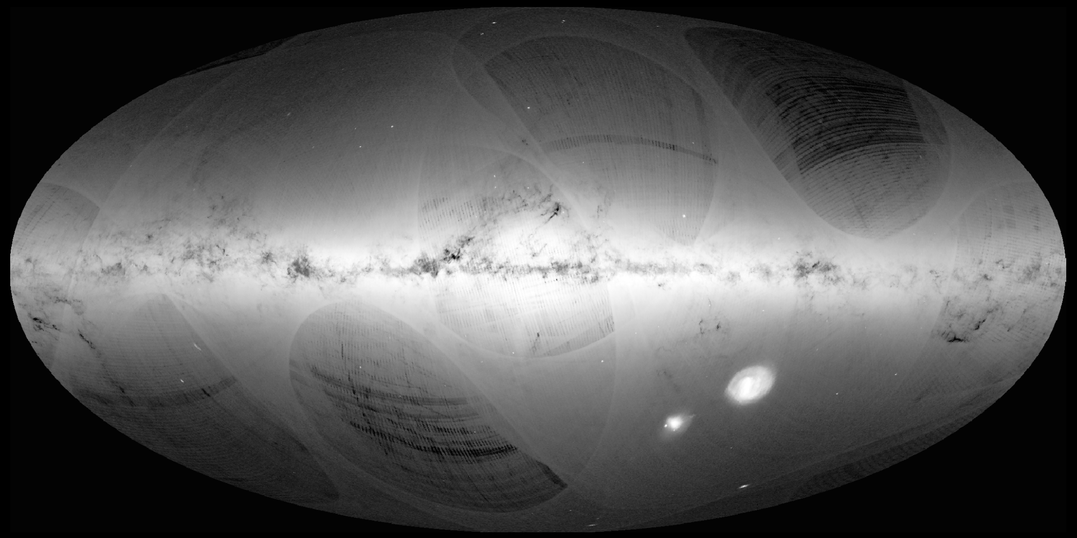

scatter

density

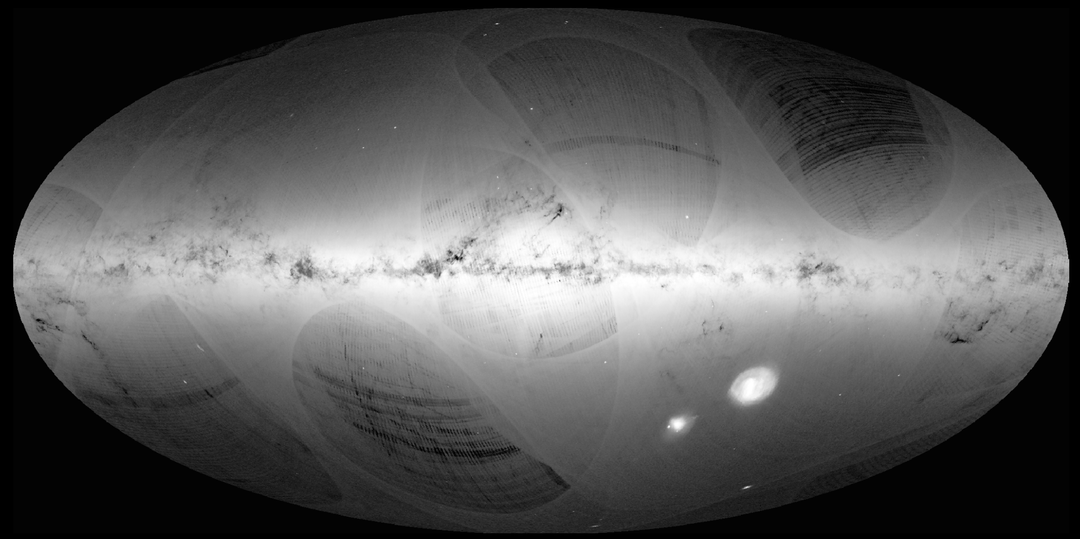
- How fast can it be done?



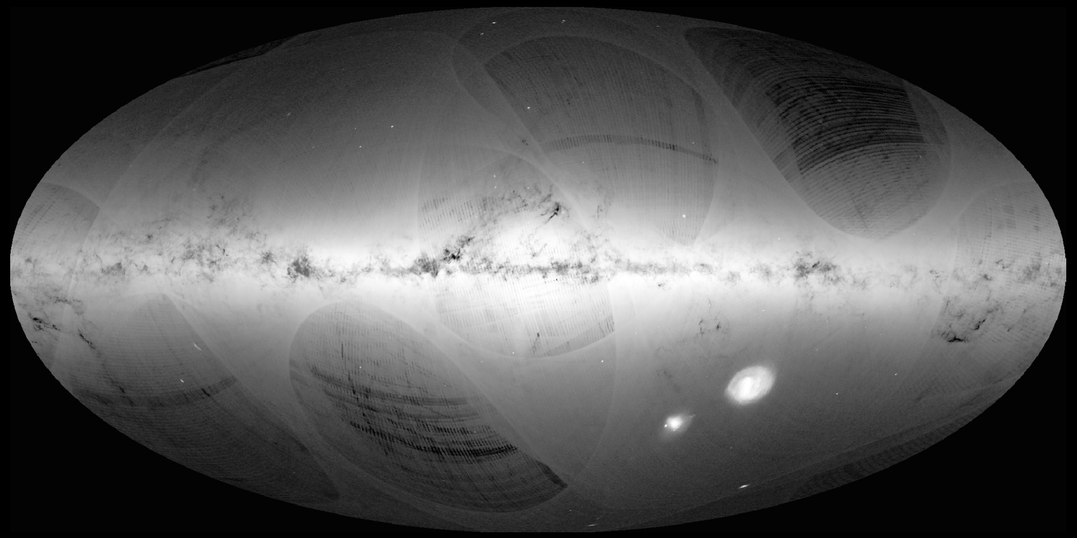
- How fast can it be done?
  - $10^9$ * 2 * 8 bytes = 15 GiB (double is 8 bytes)



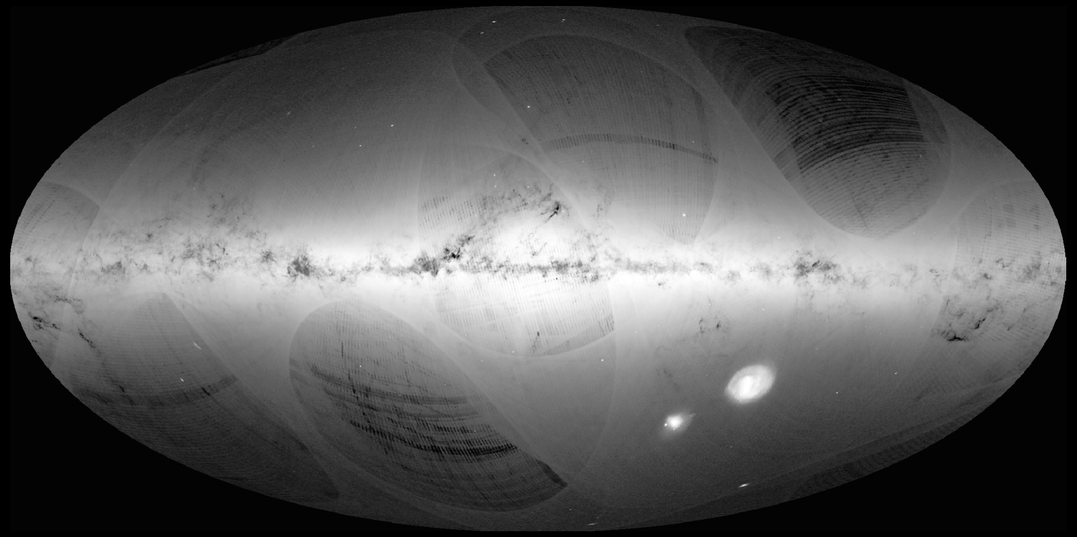
- How fast can it be done?
  - $10^9$ * 2 * 8 bytes = 15 GiB (double is 8 bytes)
  - Memory bandwidth: 10-30 GiB/s: ~1 second



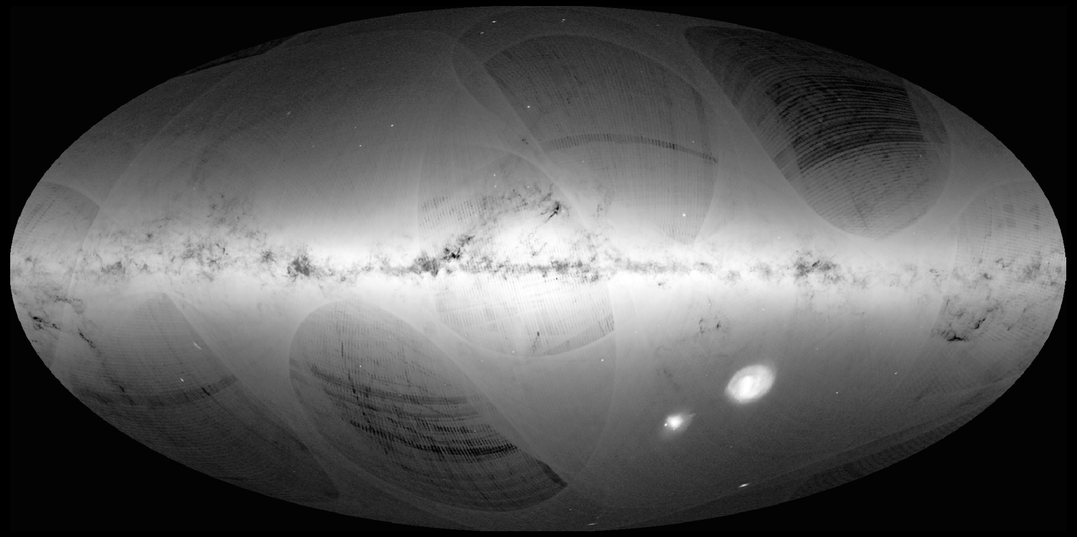
- How fast can it be done?
  - $10^9$ * 2 * 8 bytes = 15 GiB (double is 8 bytes)
  - Memory bandwidth: 10-30 GiB/s: ~1 second
  - CPU: 3 Ghz (but multicore, say 4-8): 12-24 cycles/second

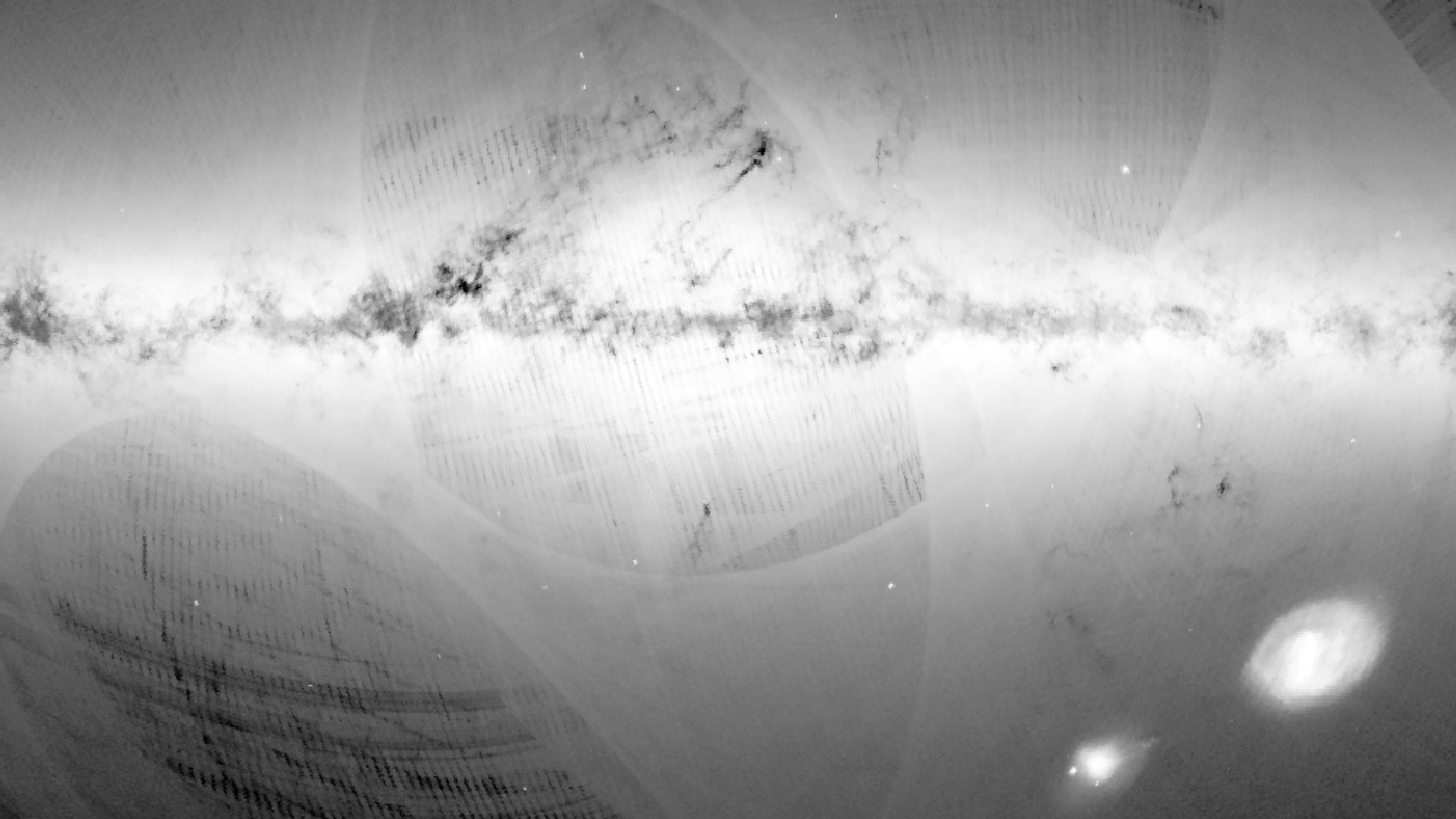


- How fast can it be done?
  - $10^9$ * 2 * 8 bytes = 15 GiB (double is 8 bytes)
  - Memory bandwidth: 10-30 GiB/s: ~1 second
  - CPU: 3 Ghz (but multicore, say 4-8): 12-24 cycles/second
  - Few cycles per row/object, simple algorithm



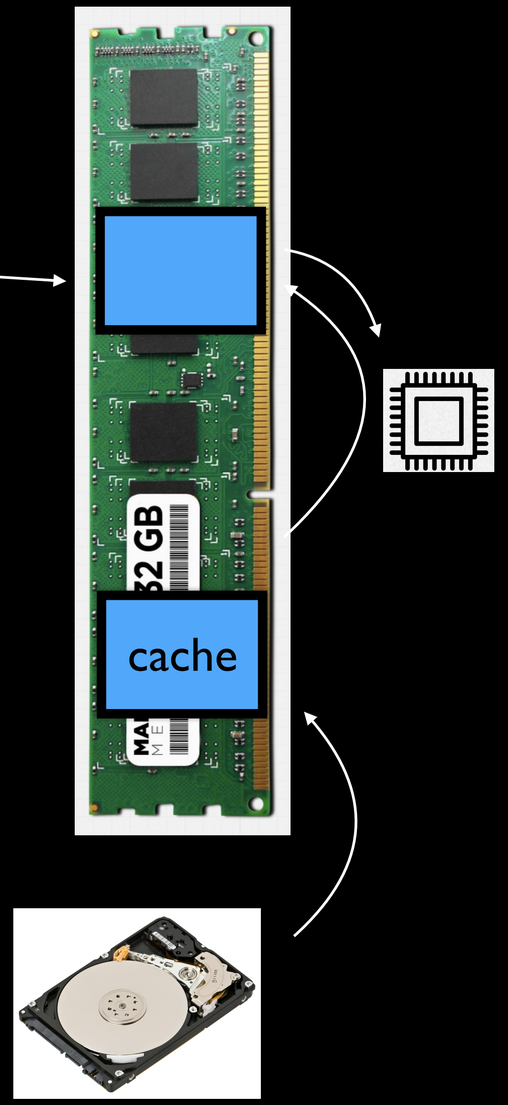
- How fast can it be done?
  - $10^9$ * 2 * 8 bytes = 15 GiB (double is 8 bytes)
  - Memory bandwidth: 10-30 GiB/s: ~1 second
  - CPU: 3 Ghz (but multicore, say 4-8): 12-24 cycles/second
  - Few cycles per row/object, simple algorithm
  - Histograms/Density/Statistics grids





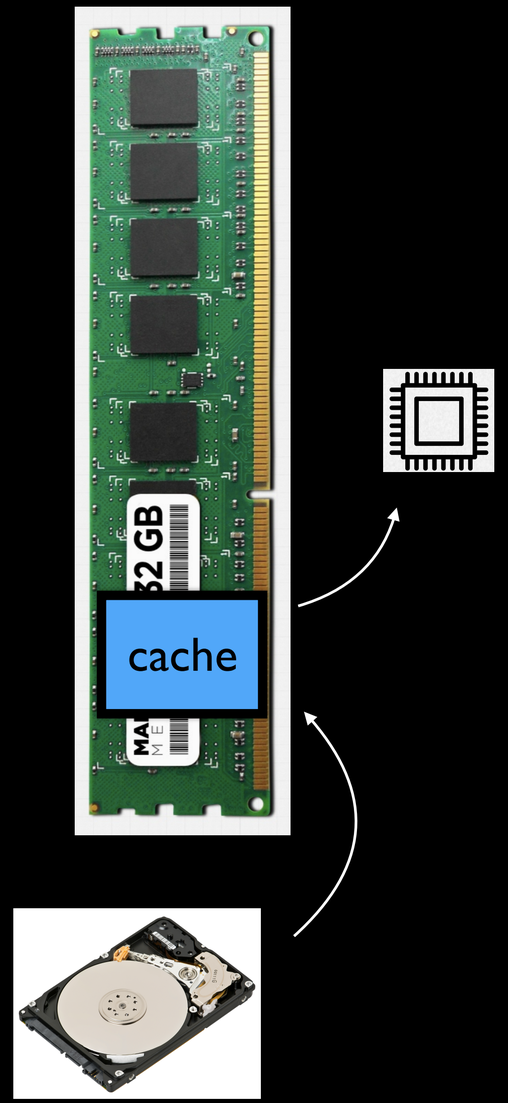
# How to store and read the data

- Storage: native, column based (hdf5)
- ‘Normal’ 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



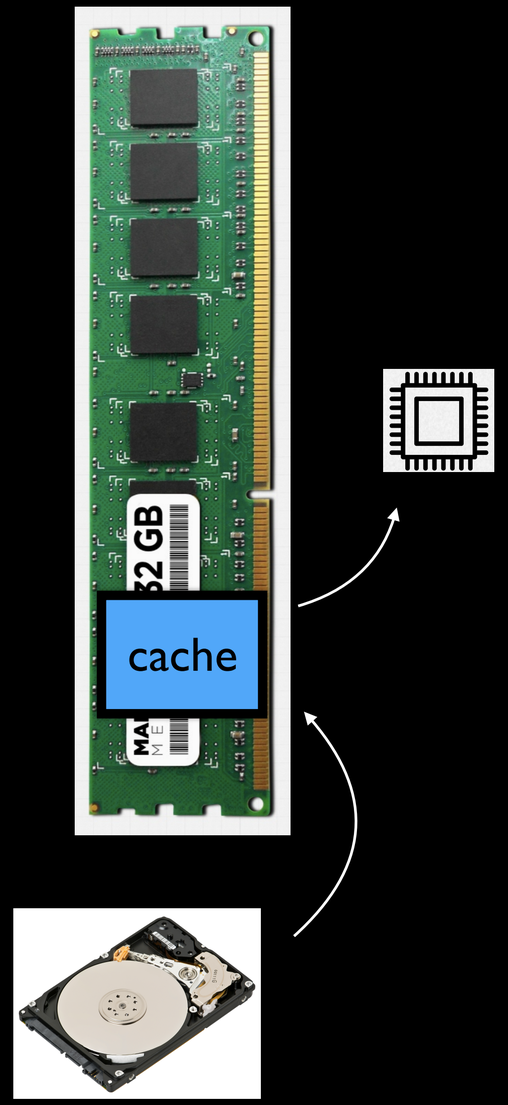
# How to store and read the data

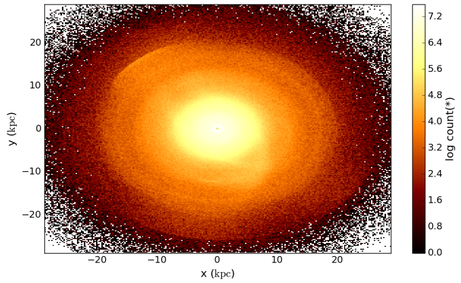
- Storage: native, column based (hdf5)
- ‘Normal’ method:
  - Allocate memory
  - read from disk to memory
  - Actually: from disk, to OS cache, to memory
  - Wastes memory/cache
  - 15 GB data , requires 30 GB is you want to use the file system cache



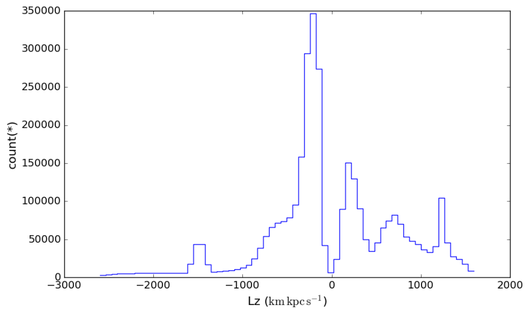
# How to store and read the data

- Storage: native, column based (hdf5)
- ‘Normal’ 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)

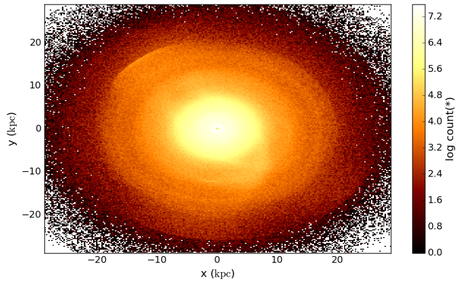



20
10
0
−10
−20
y (kpc)
−20
−10
0
10
20
x (kpc)
7.2
6.4
5.6
4.8
4.0
3.2
2.4
1.6
0.8
0.0
log count(+)

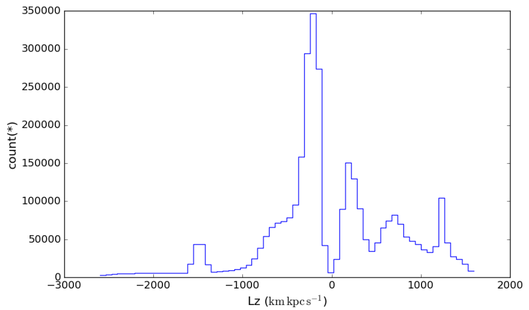
1d



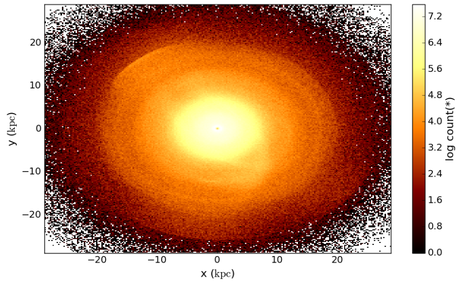

2d



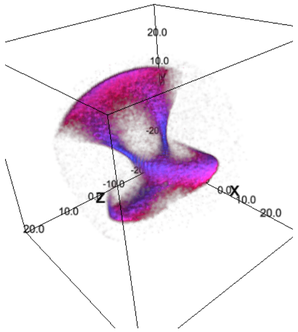
1d




2d

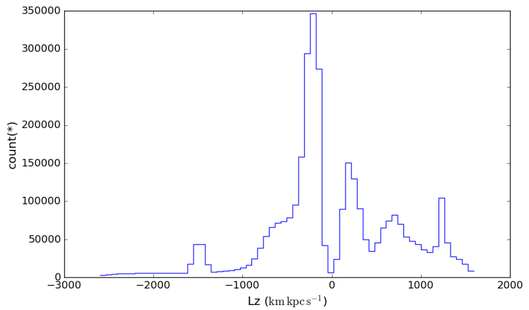



3d

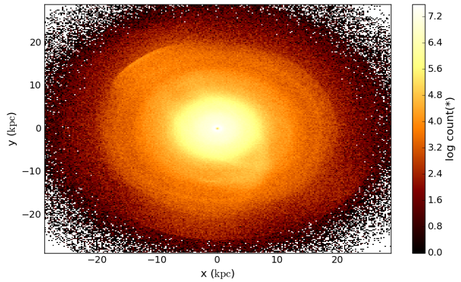


0d 330,000 rows

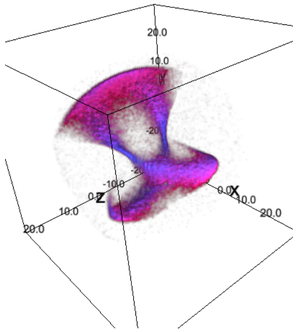
1d




2d

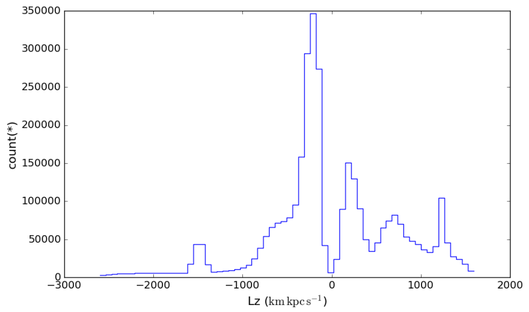



3d

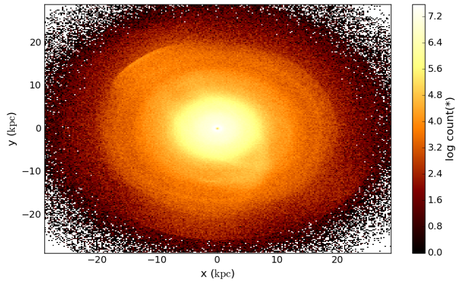


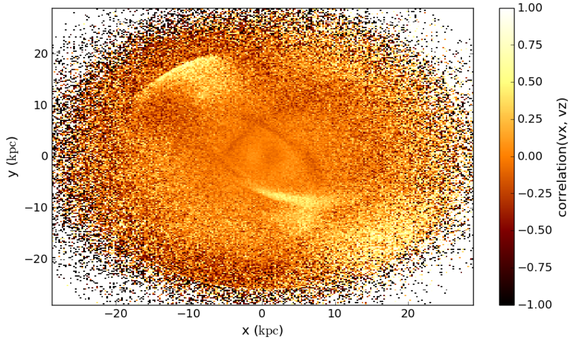
0d 330,000 rows

1d

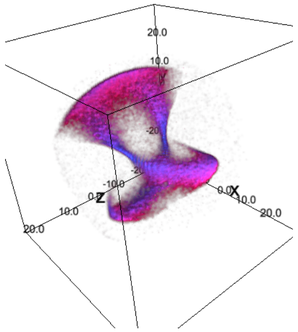



2d



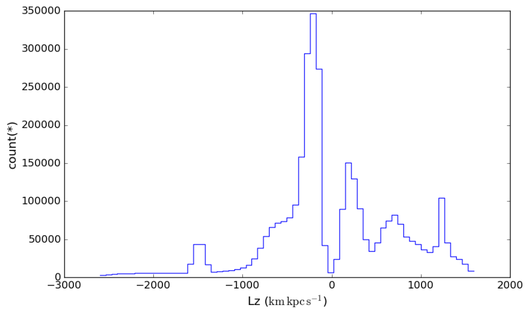



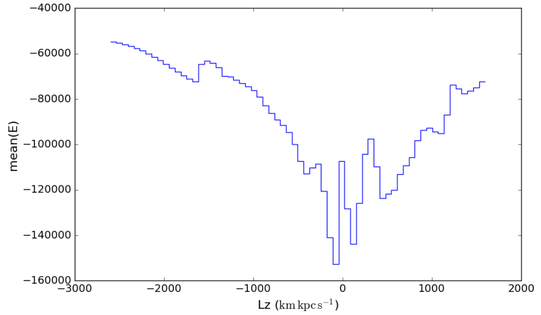

3d



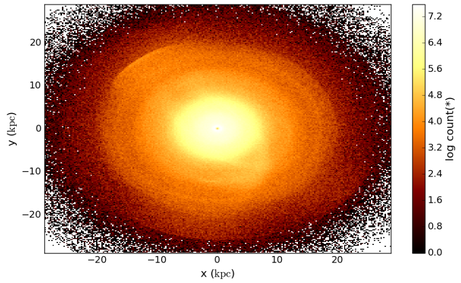
0d 330,000 rows

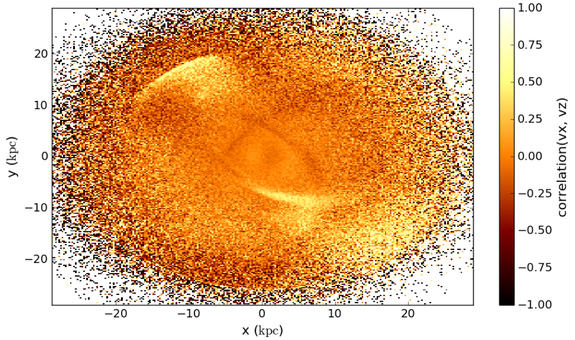
1d



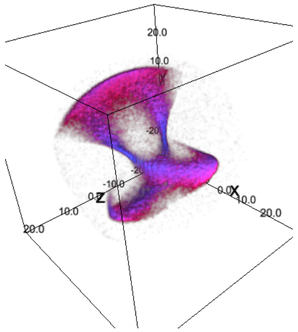




2d



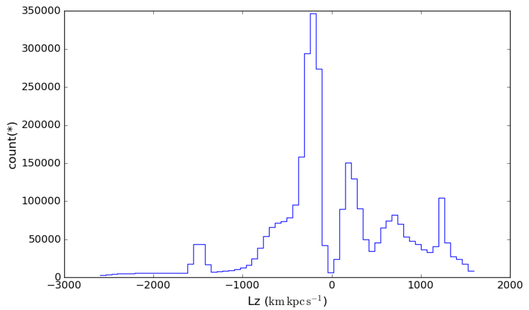



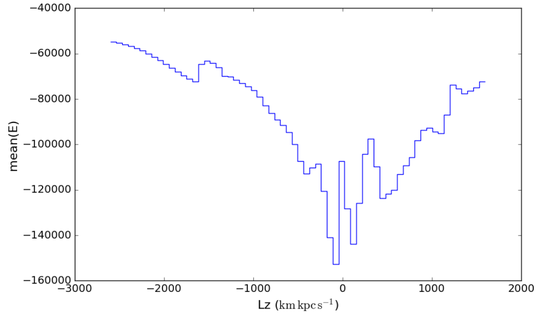

3d



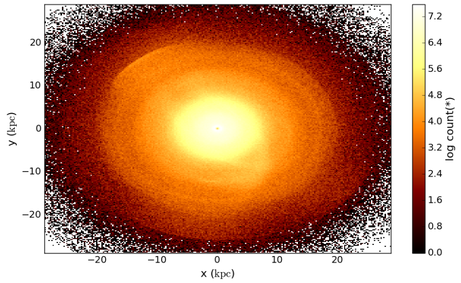
0d 330,000 rows mean: -0.083

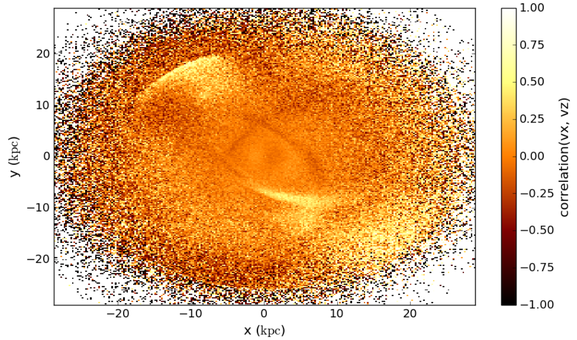
1d



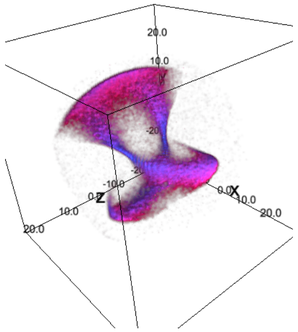




2d



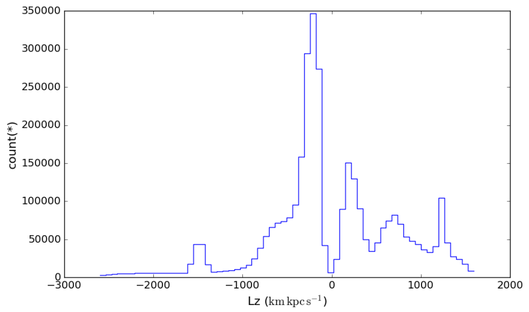



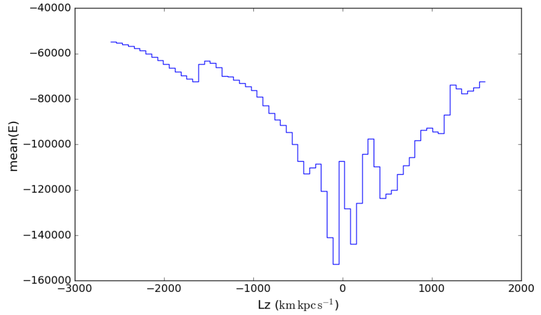

3d



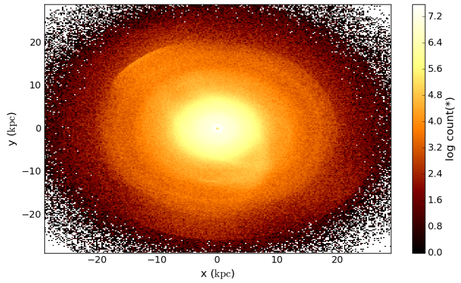
0d 330,000 rows mean: -0.083

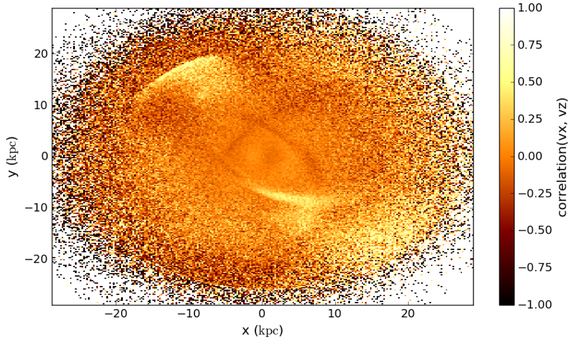
1d



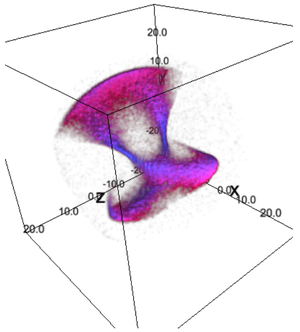



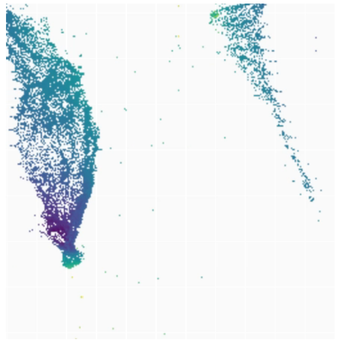

2d




3d





# vaex

# vaex

- Python library (conda/pip installable)

# vaex

- Python library (conda/pip installable)
- pandas-like but for large datasets
  - Out-of-core, lazy expression, works on chunks

# vaex

- Python library (conda/pip installable)
- pandas-like but for large datasets
  - Out-of-core, lazy expression, works on chunks
- Focusses mostly on statistics on N-d grids (count/mean/max/std/…)

# vaex

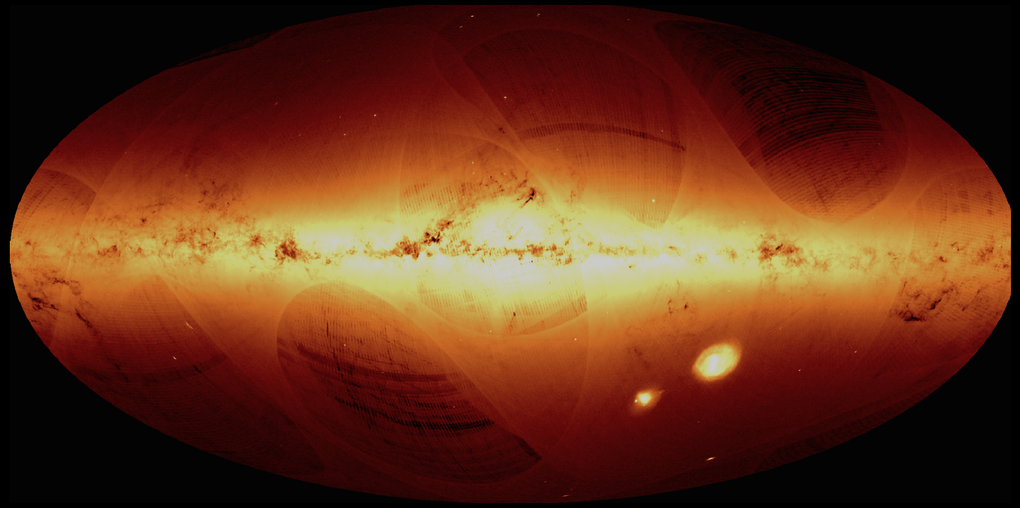
- Python library (conda/pip installable)
- pandas-like but for large datasets
  - Out-of-core, lazy expression, works on chunks
- Focusses mostly on statistics on N-d grids (count/mean/max/std/…)
- >1 billion rows / sec on a `decent` desktop (quad core 3Gz)
  - >50x faster than scipy.stats.binned_statistic_2d

# vaex

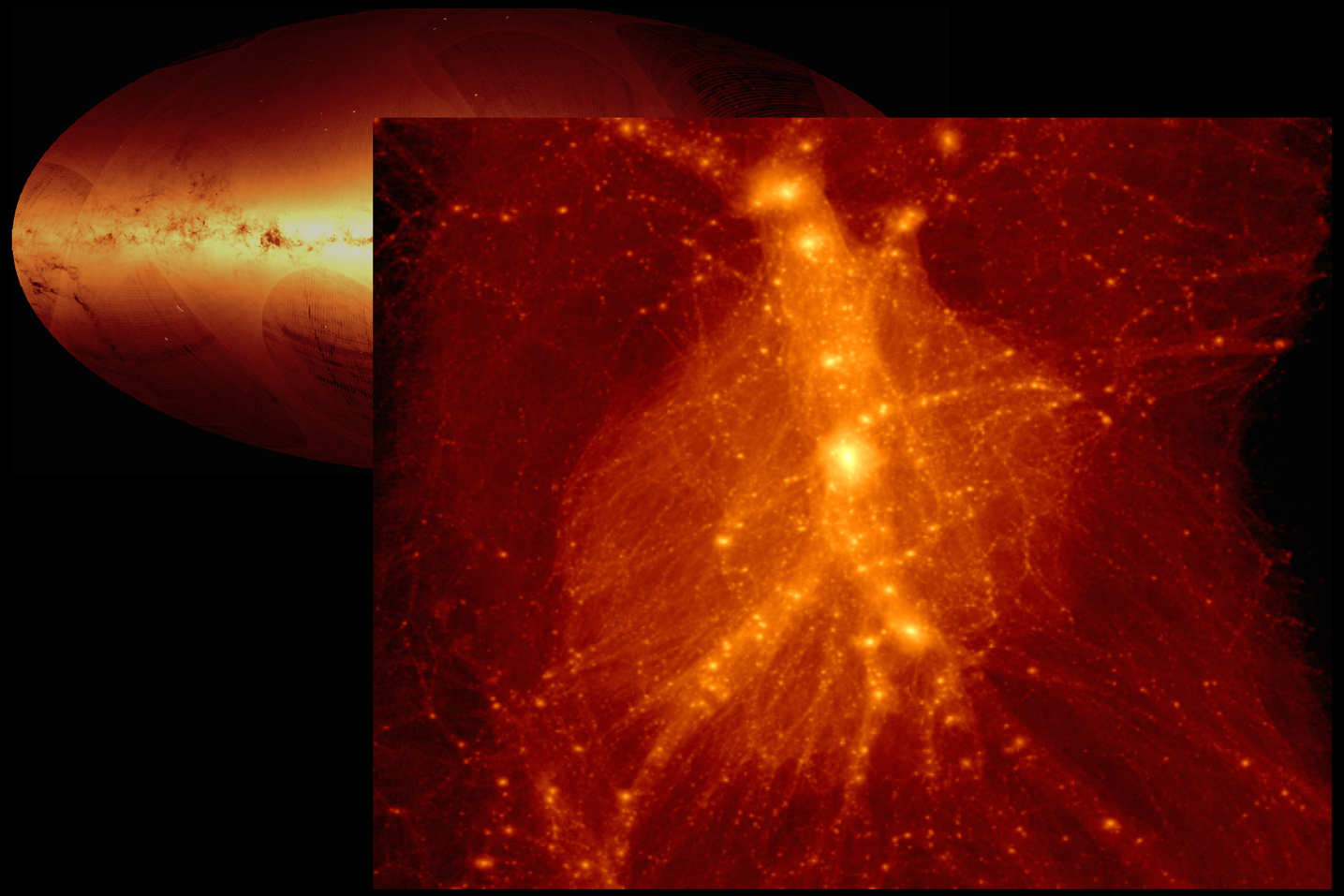
- Python library (conda/pip installable)
- pandas-like but for large datasets
  - Out-of-core, lazy expression, works on chunks
- Focusses mostly on statistics on N-d grids (count/mean/max/std/…)
- >1 billion rows / sec on a `decent` desktop (quad core 3Gz)
  - >50x faster than scipy.stats.binned_statistic_2d
- Does visualisation / matplotlib / bqplot / ipyvolume / ipyleaflet

# What kind of data?

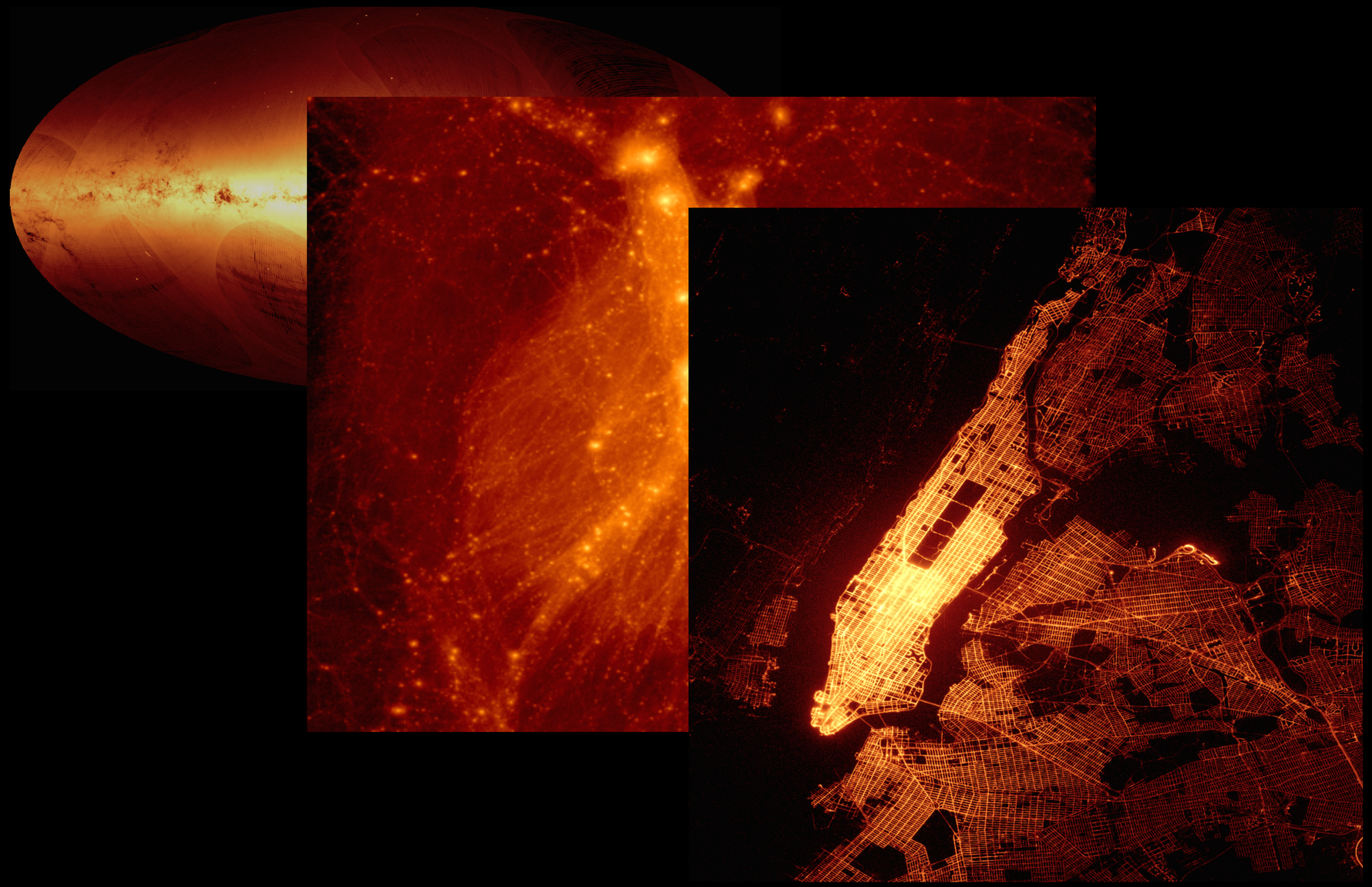
# What kind of data?



# What kind of data?



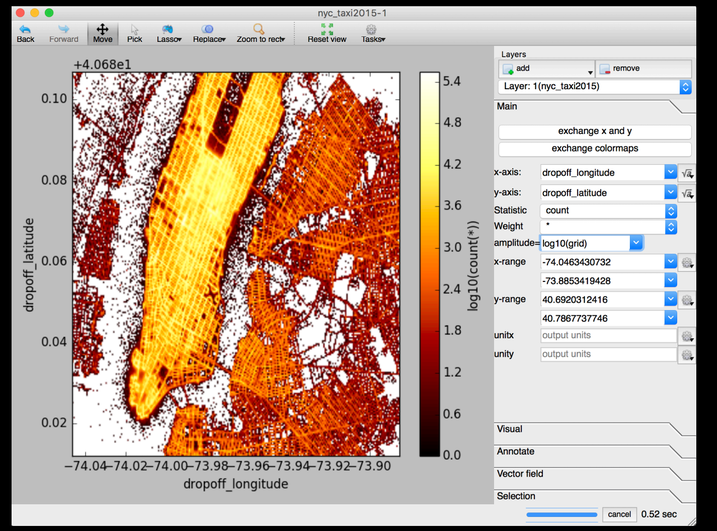
# What kind of data?



# Why the Jupyter notebook?

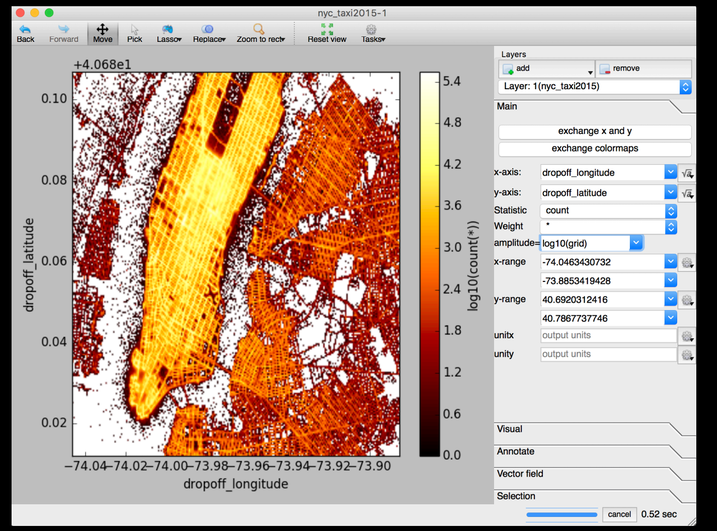
# Why the Jupyter notebook?

- GUI (vaex has/had this)
  - Limited in options, cannot replace a programming language



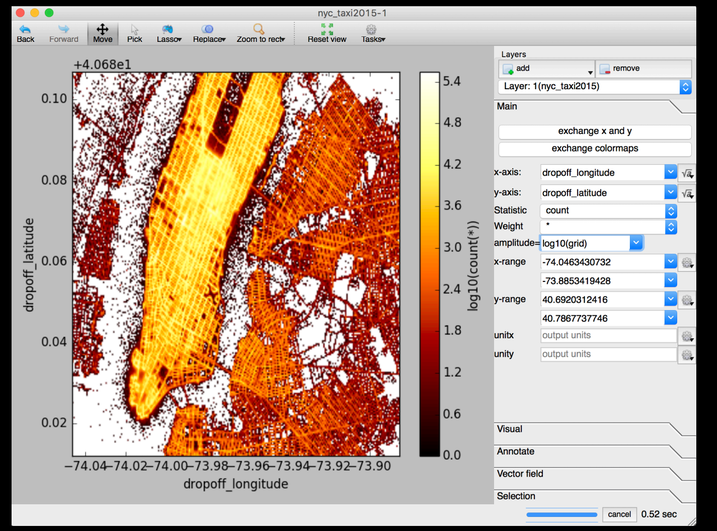
# Why the Jupyter notebook?

- GUI (vaex has/had this)
  - Limited in options, cannot replace a programming language
- Scripts
  - Anything possible
  - Slow in exploration phase



# Why the Jupyter notebook?

- GUI (vaex has/had this)
  - Limited in options, cannot replace a programming language
- Scripts
  - Anything possible
  - Slow in exploration phase
- Notebook
  - Quick exploration/iterating
  - What about interactive plots?



*“Never do a live demo”*

-Many people

# Answers

- How to deal with a billion objects/rows/stars?
  - statistics in N-d grids / vaex
- Why use/move to using the notebook?
  - Quick exploration + interactivity
- How can we make the notebook interactive?
  - ipywidgets+bqplot+ipyleaflet+ipyvolume

- maartenbreddels@gmail.com
  - www.maartenbreddels.com
  - Twitter @maartenbreddels
- vaex
  - https://vaex.io
  - 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
- Material:
  - https://github.com/maartenbreddels/jupyter-day-polytechnique-2018