Data Analysis and Image Processing with Python

Sami-Matias Niemi
s.niemi@ucl.ac.uk
@AstroSami / #DAIPPython

MSSL - UCL, 4 - 5 November 2014

Me

- Euclid VIS Instrument Scientist:
  - Euclid is a ESA cosmology mission to study the dark Universe (dark matter and energy).
- Have written codes in:
  - Fortran, Java, IDL, Perl, C-shell.
- All programs in last 5yrs in Python.

Agenda

MAN, I SUCK AT THIS GAME. CAN YOU GIVE ME A FEW POINTERS?
0x3A28213A
0x6339392C,
0x7363682E.
I HATE YOU.

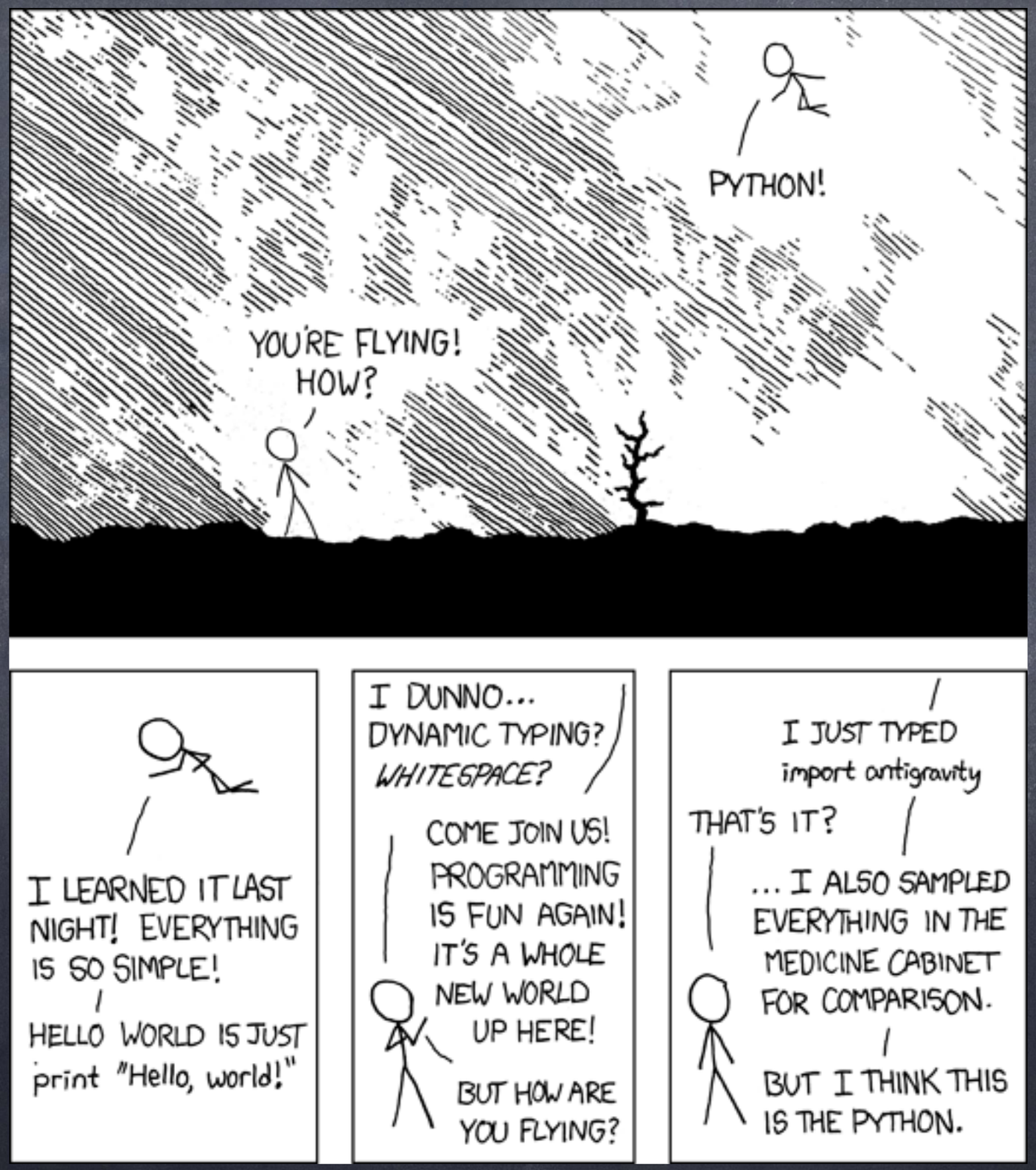
Day 1: November 4

Time	Topic
09:30 - 09:45	Logistics and Introduction
09:45 - 10:30	Basics of Python
15 mins	Coffee break
10:45 - 12:30	Basics and Numerical Arrays
60 mins	Lunch break
13:30 - 15:00	Scientific Computing
15 mins	Coffee break
15:15 - 17:30	Generating Figures

Day 2: November 5

Time	Topic
09:30 - 10:30	Model Fitting
15 mins	Coffee break
10:45 - 12:30	Model Fitting / Machine Learning
60 mins	Lunch break
13:30 - 15:00	Machine Learning
15 mins	Coffee break
15:15 - 17:30	Image Processing

Why Python?


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

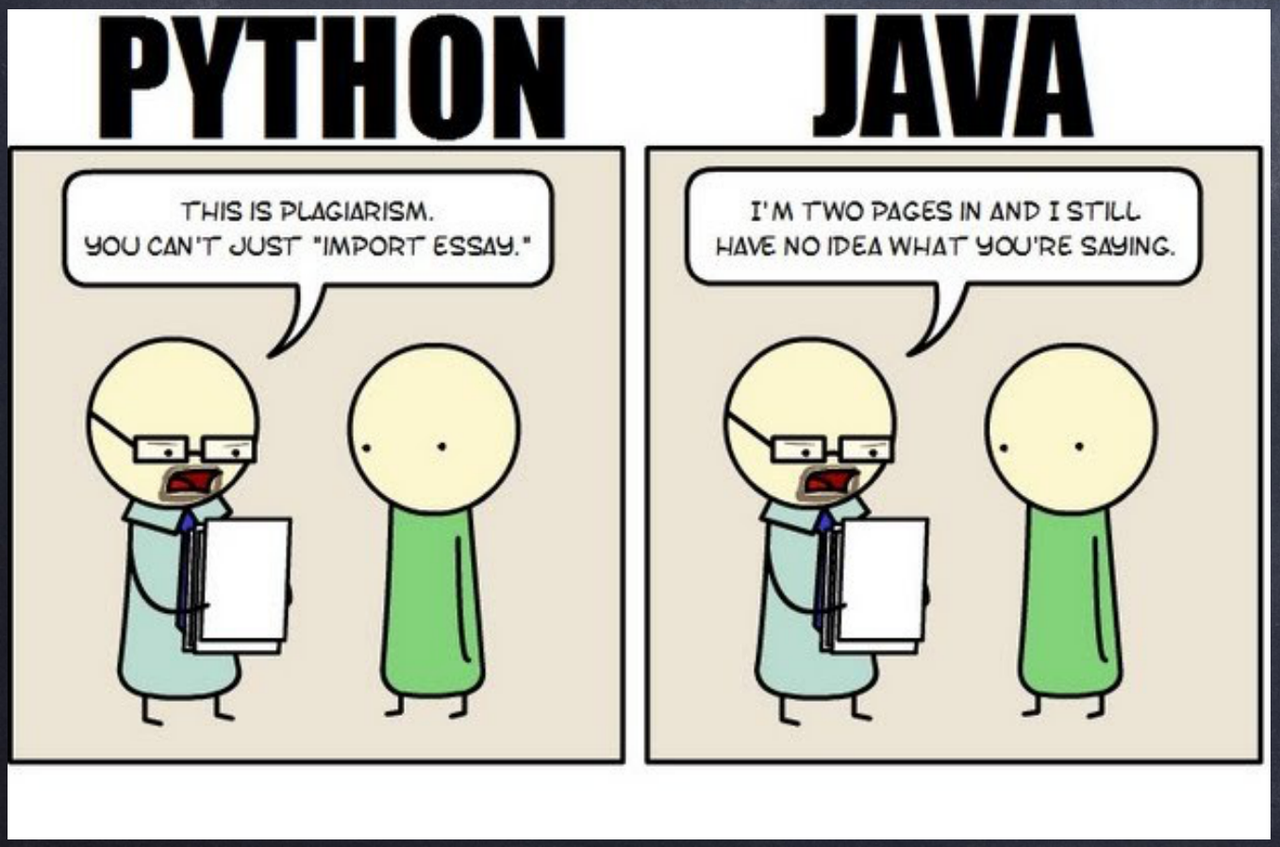
What is Python used for?

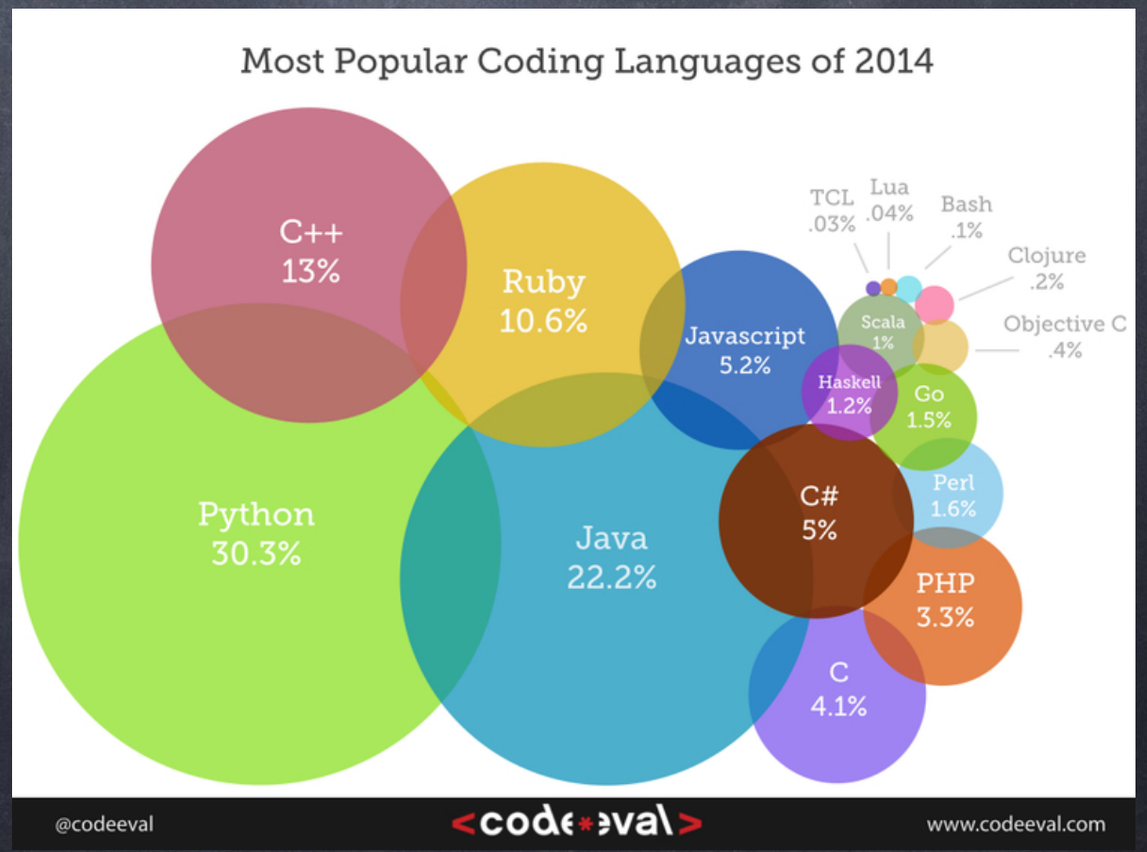
- Searching: Google search engine (uses also C and Java).
- Web: Youtube, Yahoo maps...
- Graphics: Industrial Light and Magic, Walt Disney Feature Animation...
- Science: National Weather Service, NASA, STScI (HST&JWST)...
- Software development: e.g. Red Hat installer
- Games: Eve Online, Battlefield 2, Civilization 4...

I have used Python for

- Euclid-VIS image simulator:
  - mock astronomy images with 600Mpix.
- Spectral and imaging data reduction pipeline.
- All analysis and figures in my papers, proposals, etc. since 2008.

So others use Python, why should I care?





Why Python?

- Free & Portable (Linux, OS X, Windows...)
- Easy to learn yet powerful
- Reduced development time:
  - No compilation phase
- Integration with C, Fortran, and Java is easy.
- Widely used in Science and Business.
- Many 3rd party packages (batteries included)...

Python Package Index

- http://pypi.python.org/pypi
- Software repository listing **50566** ~~**24343**~~ packages (as of 27th of October 2014 ~~30th of Sep 2012~~).

python™

» Package Index

search

PACKAGE INDEX
- Browse packages
- Package submission
- List trove classifiers
- List packages
- RSS (latest 40 updates)
- RSS (newest 40 packages)
- Python 3 Packages
- PyPI Tutorial
- PyPI Security
- PyPI Support
- PyPI Bug Reports
- PyPI Discussion
- PyPI Developer Info

ABOUT
NEWS
DOCUMENTATION
DOWNLOAD
COMMUNITY
FOUNDATION
CORE DEVELOPMENT

Index of Packages Matching 'science'

Not Logged In
- Login
- Register
- Lost Login?
- Use OpenID

Status
- Nothing to report

Package	Weight*	Description
colour-science 0.3.2	7	Colour Science for Python
Helmholtz 0.2.0	4	A framework for creating neuroscience databases
omniscience 0.1dev	4	Enables dynamic real-time monitoring/logging of code.
scicloud 3.0.4	4	Science VM client-side library
sciscicloud 3.0.2	4	Science VM client-side library
Sift 1.1.1.0	4	Python bindings for Sift Science's API
dst 0.1.5	3	Data Science Toolbox -- Start doing data science in minutes.
dstk 0.50	3	Data Science Toolkit Client
eq 0.0.8	3	Earthquake science library.
execo 2.3	3	Execo offers a Python API for asynchronous control of local or remote, standalone or parallel, unix processes. It is especially well suited for quickly and easily scripting workflows of parallel/distributed operations on local or remote hosts: automate a scientific workflow, conduct computer science experiments, perform automated tests, etc. The core python package is ``execo``. The ``execo_g5k`` package provides a set of tools and extensions for the `Grid5000 <https://www.grid5000.fr>`_ testbed. The ``execo_engine`` package provides tools to ease the development of computer sciences experiments.
kabuki 0.5.5	3	kabuki is a python toolbox that allows easy creation of hierarchical bayesian models for the cognitive sciences.
neuroinfo_toolkit 1.5.1	3	Neuroinfo Toolkit
nitime 0.5	3	Nitime: timeseries analysis for neuroscience data
psychopy_ext 0.5.5	3	A framework for a rapid reproducible design, analysis and plotting of experiments in neuroscience and psychology.
pyccm 0.2	3	Implementation of the Convergent Cross Mapping (Sugihara et al., 2012: Detecting Causality in Complex Ecosystems, Science 338: 496–500.)
pycse 1.50	3	python computations in science and engineering
pykit-sci 0.1.8	3	pykit-sci: collection of python modules for science.
pysift 0.0.2	3	SiftScience API.
python-bioformats 1.0.4	3	Read and write life sciences file formats
rosetta 0.2.5	3	Tools, wrappers, etc... for data science with a concentration on text processing

...a few worth mentioning

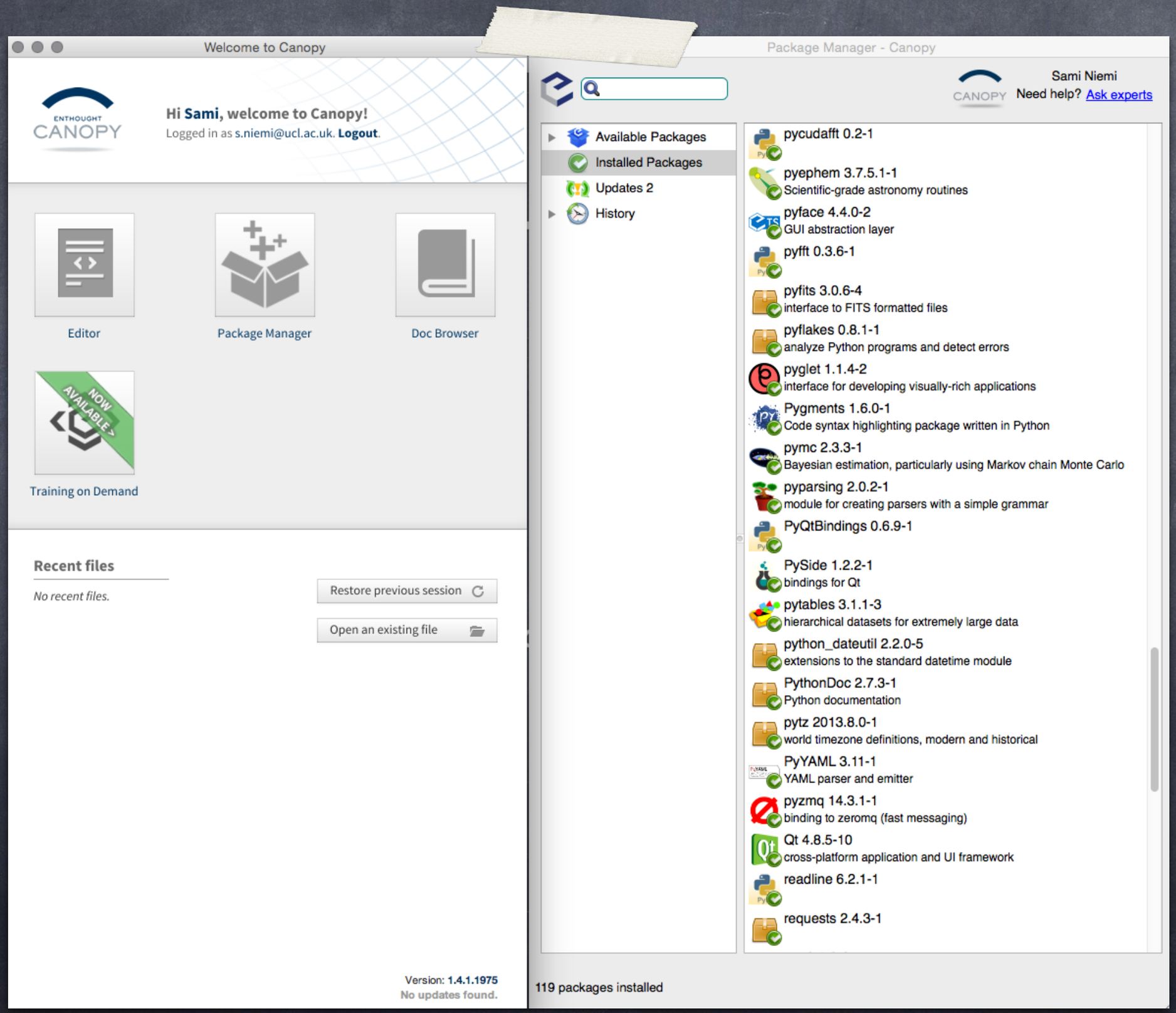
- NumPy, SciPy, matplotlib, ipython, PyFITS, astropy, ATpy, Kapteyn, PyWCS, SymPy, scikits, RPy2, PyMC, emcee, PyIDL, APLpy, astLib, statsmodels, MySQLdb, PIL, PyCUDA, numexpr, yt, mplstereonet, h5py, Patsy, SfePy, SQLAlchemy, TKinter, PyGtk, PyQt, Mutagen, Django, numexpr, PyPy, Sphinx...
  - but too many to fit on one slide.
- Examples will use some of the above.

I am sold,
can I play?

What Do You Need?

- Core Python is pre-installed in Linux (Debian, Ubuntu, etc.) and OS X.
- Many ways to install 3rd party packages (easy_install, pip, homebrew, etc.).
  - One of the easiest way is, however, to use Enthought Canopy:
    - https://www.enthought.com/products/canopy/
    - Free for Academics!
- Please make sure you have a working setup.

Canopy GUI


Welcome to Canopy
ENTHOUGHT CANOPY
Hi Sami, welcome to Canopy!
Logged in as s.niemi@ucl.ac.uk. Logout.
Editor
Package Manager
Doc Browser
NOW AVAILABLE
Training on Demand
Recent files
No recent files.
Restore previous session
Open an existing file
Version: 1.4.1.1975
No updates found.
Package Manager - Canopy
Sami Niemi
Need help? Ask experts
Available Packages
Installed Packages
Updates 2
History
pycudafft 0.2-1
pyephem 3.7.5.1-1
Scientific-grade astronomy routines
pyface 4.4.0-2
GUI abstraction layer
pyfft 0.3.6-1
pyfits 3.0.6-4
interface to FITS formatted files
pyflakes 0.8.1-1
analyze Python programs and detect errors
pyglet 1.1.4-2
interface for developing visually-rich applications
Pygments 1.6.0-1
Code syntax highlighting package written in Python
pymc 2.3.3-1
Bayesian estimation, particularly using Markov chain Monte Carlo
pyparsing 2.0.2-1
module for creating parsers with a simple grammar
PyQtBindings 0.6.9-1
PySide 1.2.2-1
bindings for Qt
pytables 3.1.1-3
hierarchical datasets for extremely large data
python_dateutil 2.2.0-5
extensions to the standard datetime module
PythonDoc 2.7.3-1
Python documentation
pytz 2013.8.0-1
world timezone definitions, modern and historical
PyYAML 3.11-1
YAML parser and emitter
pyzmq 14.3.1-1
binding to zeromq (fast messaging)
Qt 4.8.5-10
cross-platform application and UI framework
readline 6.2.1-1
requests 2.4.3-1
119 packages installed

Writing Python Code

- Any text editor will do...

but an interactive development environment might come handy :-)

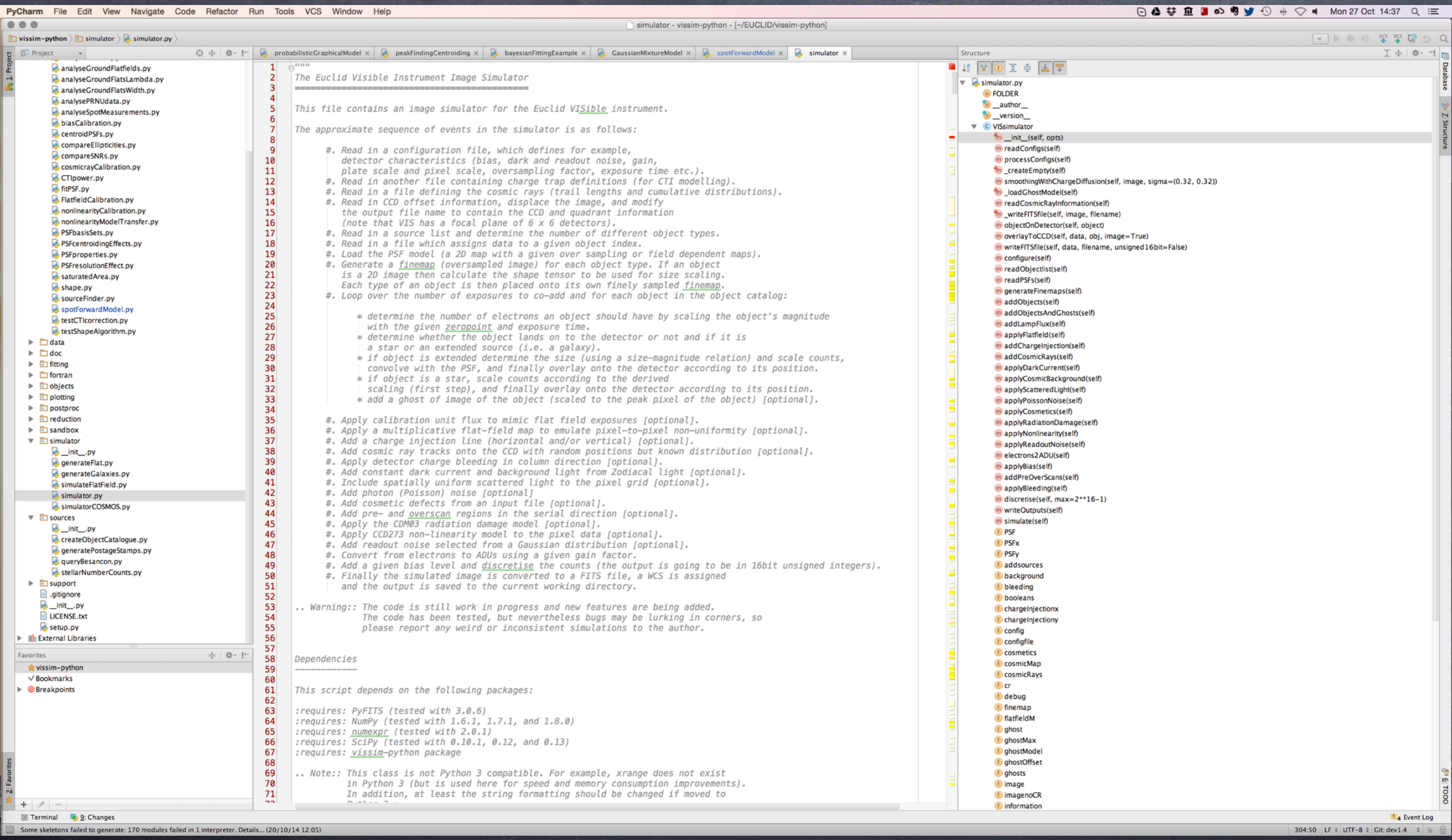
Terminal Shell Edit View Window Help

simulator — vim — bash — 127×71 — ⌘3

Python | bash | bash | Python | vim

```
"""
The Euclid Visible Instrument Image Simulator
=============================================

This file contains an image simulator for the Euclid VISible instrument.

The approximate sequence of events in the simulator is as follows:

     #. Read in a configuration file, which defines for example,
        detector characteristics (bias, dark and readout noise, gain,
        plate scale and pixel scale, oversampling factor, exposure time etc.).
     #. Read in another file containing charge trap definitions (for CTI modelling).
     #. Read in a file defining the cosmic rays (trail lengths and cumulative distributions).
     #. Read in CCD offset information, displace the image, and modify
        the output file name to contain the CCD and quadrant information
        (note that VIS has a focal plane of 6 x 6 detectors).
     #. Read in a source list and determine the number of different object types.
     #. Read in a file which assigns data to a given object index.
     #. Load the PSF model (a 2D map with a given over sampling or field dependent maps).
     #. Generate a finemap (oversampled image) for each object type. If an object
        is a 2D image then calculate the shape tensor to be used for size scaling.
        Each type of an object is then placed onto its own finely sampled finemap.
     #. Loop over the number of exposures to co-add and for each object in the object catalog:

           * determine the number of electrons an object should have by scaling the object's magnitude
             with the given zeropoint and exposure time.
           * determine whether the object lands on to the detector or not and if it is
             a star or an extended source (i.e. a galaxy).
           * if object is extended determine the size (using a size-magnitude relation) and scale counts,
             convolve with the PSF, and finally overlay onto the detector according to its position.
           * if object is a star, scale counts according to the derived
             scaling (first step), and finally overlay onto the detector according to its position.
           * add a ghost of image of the object (scaled to the peak pixel of the object) [optional].

     #. Apply calibration unit flux to mimic flat field exposures [optional].
     #. Apply a multiplicative flat-field map to emulate pixel-to-pixel non-uniformity [optional].
     #. Add a charge injection line (horizontal and/or vertical) [optional].
     #. Add cosmic ray tracks onto the CCD with random positions but known distribution [optional].
     #. Apply detector charge bleeding in column direction [optional].
     #. Add constant dark current and background light from Zodiacal light [optional].
     #. Include spatially uniform scattered light to the pixel grid [optional].
     #. Add photon (Poisson) noise [optional]
     #. Add cosmetic defects from an input file [optional].
     #. Add pre- and overscan regions in the serial direction [optional].
     #. Apply the CDM03 radiation damage model [optional].
     #. Apply CCD273 non-linearity model to the pixel data [optional].
     #. Add readout noise selected from a Gaussian distribution [optional].
     #. Convert from electrons to ADUs using a given gain factor.
     #. Add a given bias level and discretise the counts (the output is going to be in 16bit unsigned integers).
     #. Finally the simulated image is converted to a FITS file, a WCS is assigned
        and the output is saved to the current working directory.

.. Warning:: The code is still work in progress and new features are being added.
             The code has been tested, but nevertheless bugs may be lurking in corners, so
             please report any weird or inconsistent simulations to the author.


Dependencies
------------

This script depends on the following packages:

:requires: PyFITS (tested with 3.0.6)
:requires: NumPy (tested with 1.6.1, 1.7.1, and 1.8.0)
:requires: numexpr (tested with 2.0.1)
:requires: SciPy (tested with 0.10.1, 0.12, and 0.13)
:requires: vissim-python package

.. Note:: This class is not Python 3 compatible. For example, xrange does not exist
          in Python 3 (but is used here for speed and memory consumption improvements).
"simulator.py" [noeol] 2394L, 110053C
```

Writing Python Code (IDE)



https://www.jetbrains.com/pycharm/

Making Changes

- Code development requires iterations. Many ways to track changes...
  - Option 1: mycode_new.py, mycode_newer.py, mycode_works.py...
  - Option 2: Revision Control
- Many options for version control systems:
  - SVN, CVS, Mercurial, GIT...

Example: SourceTree and GIT Hub

SourceTree File Edit View Repository Actions Window Help — Mon 27 Oct 15:08

vissim-python (Git)

View | Commit | Checkout | Reset | Stash | Add | Remove | Add/Remove | Fetch | Pull | Push | Branch | Merge | Tag | Show in Finder | Git Flow | Terminal | Settings

FILE STATUS
- Working Copy 8

BRANCHES
- dev1.1
- dev1.2
- dev1.3
- dev1.4
- master

TAGS

REMOTES
- origin
  - dev1.1
  - dev1.2
  - dev1.3
  - dev1.4
  - HEAD
  - master

STASHES

SUBMODULES

SUBTREES

All Branches | Show Remote Branches | Ancestor Order | Jump to:

Graph	Description	Commit	Author	Date
	Added a new modules to include cosmic ray events to images and to analyse ground flat field measurements and the noise in these images.	598cb9b	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	25 Feb 2013 13:06
	Simulator version 1.2 is getting ready. Included a script that is the backbone for the VIS data reduction pipeline.	8db198a	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	23 Jan 2013 09:50
	Added a spatially uniform scattered light model to the simulator and updated the documentation to reflect this change. The version 1.2 is getting ready.	1a93709	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	11 Jan 2013 16:29
	Added a Bayesian MCMC fitting function to fitPSF.	e2d2331	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	17 Dec 2012 13:10
	Added a sandbox for examples and tests. Included a simple example demonstrating Bayesian fitting with pymc...	7fccc58	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	12 Dec 2012 13:59
	Included a script to fit basis sets to a PSF. Both least squares and Bayesian MCMC fitting is available, but the latter has not been tested properly...	efb8a76	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	12 Dec 2012 10:23
	Fixed the PSF centroiding script. This could be further improved by using a sinc interpolation for the shifts. However, the current (3rd order spline) is probably good enough.	297fa2d	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	6 Dec 2012 12:12
	Included a simple function to calculate a floating-window first moment centroid from a given image. Changed the PSF basis set derivation script to include centroiding. However, this will now centroid each object separately, not taking into account any other wavelengths or PSFs.	97bf8d6	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	26 Nov 2012 13:14
	Included a script to test CTI and how large of an ellipticity bias an imperfect correction might induce.	f683f41	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	23 Nov 2012 15:03
	Included a script that can be used to derive PCA basis set for PSFs.	46047d7	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	18 Nov 2012 15:15
	Added a script to plot the cosmic ray track distributions.	10fb692	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	2 Nov 2012 19:09
	Included a function to the CCDlabData.py that can be used to combine the different image areas to a single CCD FITS file. Updated documentation.	fa2b9a9	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	2 Nov 2012 13:10
	Added a part that allows to test different phases and multipliers in case of non-linearity calibration.	1b1e287	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	26 Oct 2012 16:58
	Patched the sinusoidal non-linearity model. The scaling was incorrect in the previous version.	85135a7	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	24 Oct 2012 12:26
	Included a script to convert CCD lab data (lab view binary files) to FITS files. Modifications to the calibration analysis scripts.	4334b9e	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	24 Oct 2012 11:32
	Modified to simulator to use np.float64 in the the PSF convolutions.	668305c	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	15 Oct 2012 12:18
	Modified the reduceVISdata script to allow quadrants to be reduced. A bug in the CTI correction for quadrant data still remains.	38a2390	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	12 Oct 2012 08:35
	simulator1.1 origin/dev1.1 dev1.1 Simulator version 1.1 is ready!	6ed2f5e	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	3 Oct 2012 13:47
	simulator1.08dev Updated the simulator and the ETC. Tried to speed up some of the operations in the simulator code with numexpr. Included an optional feature to the ETC to either calculate the readout noise when read noise is poorly sampled.	f49b1c5	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	20 Sep 2012 16:31
	simulator1.07dev Started working on non-linearity simulations and model. Included a non-linearity to the simulator.	7076c94	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	18 Sep 2012 17:41
	simulator1.06dev Improved the source finder: included a magnitude calculation to the aperture photometry method.	518fc57	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	17 Sep 2012 17:02
	Added an aperture photometry possibility to the source finder.	ffe3f1c	Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>	12 Sep 2012 15:10

Sorted by path | Search

- analysis/compareSNRs.py
- analysis/ETC.py
- analysis/fitPSF.py
- analysis/fitPSFmodel.py
- data/science.config
- doc/analysis.rst
- doc/_build/doctrees/analysis.doctree
- doc/_build/doctrees/environment.pickle
- doc/_build/doctrees/simulator.doctree
- doc/_build/html/analysis.html
- doc/_build/html/genindex.html
- doc/_build/html/index.html
- doc/_build/html/objects.inv
- doc/_build/html/py-modindex.html

Commit: e2d23317f0e9ab38087432722e3981c406af25dc [e2d2331]
Parents: 7fccc58dea
Author: Sami-Matias Niemi <smn2@mssl.ucl.ac.uk>
Date: 17 December 2012 13:10:33 GMT

Added a Bayesian MCMC fitting function to fitPSF.

Updated documentation.

analysis/compareSNRs.py

Hunk 1 : Lines 27-40 — Reverse hunk

```
27 27     return catalog
28 28
29 29
30     - def compareMagnitudes(catalog, min=22.9, max=26.1):
    30 + def compareMagnitudes(catalog, min=22.9, max=26.1, timestamp=False):
31 31     """
32 32     A simple plot to compare input and extracted magnitudes.
33 33     """
34 34     txt = '%s' % datetime.datetime.isoformat(datetime.datetime.now())
35 35
    36 +   msk = catalog.magerr_aper < 0.65
    37 +
36 38     fig = plt.figure(frameon=False)
37 39
38 40     left, width = 0.1, 0.8
```

Hunk 2 : Lines 45-51 — Reverse hunk

```
43 45     ax2 = fig.add_axes(rect2)  #left, bottom, width, height
44 46
45 47     ax1.plot([min, max], [min, max], 'k-')
46     -  ax1.errorbar(catalog.mag_input, catalog.mag_aper, yerr=catalog.magerr_aper,
    48 +  ax1.errorbar(catalog.mag_input[msk], catalog.mag_aper[msk], yerr=catalog.magerr_aper[msk],
47 49                 c='b', ls='None', marker='o', ms=3, label='r=0.65 Aperture')
48 50     ax1.errorbar(catalog.mag_input, catalog.mag_auto, yerr=catalog.magerr_auto,
49 51                 c='r', ls='None', marker='s', ms=3, label='Auto')
```

Hunk 3 : Lines 53-59 — Reverse hunk

```
51 53                 c='g', ls='None', marker='D', ms=3, label='Best')
52 54
53 55     ax2.plot([min, max], [0, 0], 'k-')
54     -  ax2.errorbar(catalog.mag_input, catalog.mag_aper-catalog.mag_input, yerr=catalog.magerr_aper,
    56 +  ax2.errorbar(catalog.mag_input[msk], catalog.mag_aper[msk]-catalog.mag_input[msk], yerr=catalog.magerr_aper[msk],
55 57                 c='b', ls='None', marker='o', ms=3, label='r=0.65 Aperture')
56 58     ax2.errorbar(catalog.mag_input, catalog.mag_auto-catalog.mag_input, yerr=catalog.magerr_auto,
57 59                 c='r', ls='None', marker='s', ms=3, label='Auto')
```

Hunk 4 : Lines 66-80 — Reverse hunk

```
64 66
65 67     ax1.set_xticklabels([])
66 68     ax1.set_yticks(ax1.get_yticks()[1:])
67     -  ax2.set_yticks(ax2.get_yticks()[::2])
    69 +  #ax2.set_yticks(ax2.get_yticks()[::2])
68 70
69 71     ax1.set_xlim(min, max)
70 72     ax1.set_ylim(min, max)
71 73     ax2.set_xlim(min, max)
    74 +  ax2.set_ylim(-0.4, 1.1)
```

dev1.4 | 2 Modified | 6 Not Tracked — Atlassian

https://github.com/

Documenting Your Code

- Remember to document your code!
  - After a few months it is next to impossible to remember why you decided to divide everything with pi...
- One of the best methods is to write the documentation with your code and use e.g. Sphinx to generate html and PDFs straight from your code.
  - Can embed figures, equations (LaTeX)...

HTML Example:

Welcome to Euclid Visible InStrument (VIS) Python Package (VIS-PP) Documentation

Author: Sami-Matias Niemi
Contact: s.niemi@ucl.ac.uk
issue: 1.1
version: 1.3
date: January 20, 2014

This Python package VIS-PP provides subpackages and methods related to generating mock data and reducing it, the main consideration being the visible instrument (VIS) that is being developed for the Euclid telescope. The subpackages include methods to e.g. generate object catalogues, simulate VIS images, study radiation damage effects and fit new trap species, reduce and analyse data, and to include instrumental characteristics such as readout noise and CTI to "pristine" images generated with e.g. the GREAT10 photon shooting code. In addition, an algorithm to measure ellipticities of galaxies is also provided. Thus, this package tries to provide an end-to-end simulation chain for the VIS instrument.

The documentation is also available in PDF format, please download the VIS Python Package Manual.

Installation

The VIS Python Package (VIS-PP) is held in a GIT repository. You can download or fork the repository here. The package contains a mixture of classes and scripts divided in subpackages based on the usage. Unfortunately, there is no official or preferred installation instructions yet. To get most scripts working you should place the path to the root directory of the package to your PYTHONPATH environment variable. In addition, it is useful to compile the Fortran code available in the fortran subdirectory with the following command:

```
f2py -c -m cdm03 cdm03.f90
```

and then copy the .so file to the CTI directory. Please note that f2py is available in the NumPy package, but you still need for example gFortran compiler to actually compile the Fortran source.

Dependencies

The VIS Python package depends heavily on other Python packages such as NumPy, SciPy, PyFITS, and matplotlib. Thus it is recommended that one installs a Python distribution like Enthought Python, which installs all dependencies at once.

Instrument Model

The *support* subpackage contains functions that define the VIS instrument model. This model contains information about noise, dark, cosmic rays, radiation damage parameters, sky background, and pixel scale. For the Python documentation, please see:

- VIS Instrument Model
- MTF and PSF

Instrument Characteristics

The *postproc* subpackage contains methods related to either generating a CCD mosaics from simulated data that is in quadrants like the VIS reference simulator produces or including instrument characteristics to simulated images that contain only Poisson noise and background. For more detailed documentation of the Python classes, please see:

- Postprocessing Tools
  - Inserting instrument characteristics
  - Generating a CCD mosaic
  - Generating an FPA mosaic

Exposure Times

The package provides a simple exposure time calculator (ETC) that allows to estimate a signal-to-noise ratio of an average galaxy or star with a given number of VIS exposures. The ETC also allows to calculate limiting magnitude or an exposure time for a given magnitude.

For the Python documentation, please see:

- Exposure Time Calculator
  - Calculating Exposure Times and Limiting Magnitude

Table Of Contents

Welcome to Euclid Visible InStrument (VIS) Python Package (VIS-PP) Documentation
Installation
- Dependencies
Instrument Model
- Instrument Characteristics
Exposure Times
Creating Object Catalogs
Creating Simulated Mock Images
Data reduction
Data Analysis
Charge Transfer Inefficiency
- Fortran code for CTI
Supporting methods and files
- Objects
- Code
Photometric Accuracy
Supporting Methods
Indices and tables

Next topic

VIS Instrument Model

Quick search

Go

Enter search terms or a module, class or function name.

PDF (LaTeX) Example:

Warning: The logging module used does not work well with multiprocessing, but starts to write multiple entries after a while. This should be fixed.

requires PyFITS

requires NumPy

requires CDM03 (FORTRAN code, f2py -c -m cdm03 cdm03.f90)

author Sami-Matias Niemi

contact smn2@mssl.ucl.ac.uk

version 0.9

class `postproc.postprocessing.PostProcessing(`*values*, *work_queue*, *result_queue*, *seed*`)`

Euclid Visible Instrument postprocessing class. This class allows to add radiation damage (as defined by the CDM03 model) and add readout noise to a simulated image.

`applyLinearCorrection(`*image*`)`

Applies a linear correction after one forward readout through the CDM03 model.

Bristow & Alexov (2003) algorithm further developed for HST data processing by Massey, Rhodes et al.

Parameters **image** (*ndarray*) – radiation damaged image

Returns corrected image after single forward readout

Return type ndarray

`applyRadiationDamage(`*data*, *iquadrant=0*`)`

Apply radian damage based on FORTRAN CDM03 model. The method assumes that input data covers only a single quadrant defined by the iquadrant integer.

Parameters

- **data** (*ndarray*) – imaging data to which the CDM03 model will be applied to.
- **iquandrant** (*int*) – number of the quadrant to process

cdm03 - Function signature: sout = cdm03(sinp,iflip,jflip,dob,rdose,in_nt,in_sigma,in_tr,[xdim,ydim,zdim])

Required arguments: sinp : input rank-2 array('f') with bounds (xdim,ydim) iflip : input int jflip : input int dob : input float rdose : input float in_nt : input rank-1 array('d') with bounds (zdim) in_sigma : input rank-1 array('d') with bounds (zdim) in_tr : input rank-1 array('d') with bounds (zdim)

Optional arguments: xdim := shape(sinp,0) input int ydim := shape(sinp,1) input int zdim := len(in_nt) input int

Return objects: sout : rank-2 array('f') with bounds (xdim,ydim)

Basics of Python

Powerful Inbuilt Functions

- Date and time (datetime, calendar)
- File and directory access (glob, shutil)
- OS service (os, sys, io, time)
- OS service ext. (thread, mmap)
- Math and complex numbers (math, cmath, fractions, decimals)
- Data persistence (cPickle, whichdb, sqlite3)
- Data compression (zlib, gzip, tarfile)

Python Syntax

- Designed to be highly readable.
- Indentation is crucial!
  - No keywords like “begin”, “endif” or {}
- No need to declare variable types.
  - public static final double c = 299792458.0D; [JAVA]
- Some reserved words:
  - and, as, assert, break, class, for, if, import..

Python Basics

- You can write "scripts":
  - Files containing complete "programs" ready for execution.
- or use an interactive session:
  - Python interpreter
  - iPython (shell + Python):
    - provides shell commands, tab completion, history, etc.

Demo (i)python from a shell and then open the first notebook (Basics)...

Numerical Arrays

- Core Python is slow and not designed for numerical work.
- NumPy:
  - a powerful N-dimensional arrays, tools for integrating C/C++ and Fortran code, and useful linear algebra, Fourier transform, and random number capabilities.
  - However, not always the fastest (an example will follow).

Go to

Numerical Arrays Notebook

Scientific Analysis

SciPy: numerical algorithms galore

- **linalg** : Linear algebra routines (including BLAS/LAPACK)
- **sparse** : Sparse Matrices (including UMFPACK, ARPACK,...)
- **fftpack** : Discrete Fourier Transform algorithms
- **cluster** : Vector Quantization / Kmeans
- **odr** : Orthogonal Distance Regression
- **special** : Special Functions (Airy, Bessel, etc).
- **stats** : Statistical Functions
- **optimize** : Optimization Tools
- **maxentropy** : Routines for fitting maximum entropy models
- **integrate** : Numerical Integration routines
- **ndimage** : n-dimensional image package
- **interpolate** : Interpolation Tools
- **signal** : Signal Processing Tools
- **io** : Data input and output

Go to

Scientific Analysis Notebook

Data Access

- Reading and Writing Ascii files
  - Many possibilities; pure Python, SciPy, NumPy, asciitable, pandas, BeautifulSoup...
- Databases
  - More convenient than Ascii files and not necessarily more complicated.
  - Will demonstrate:
    - Creating a database and querying it;
    - Accessing SDSS DR8 database.

scipy.io

- Loading and saving files, supports:
  - reading IDL savefiles
  - loading and saving MATLAB files
  - etc.

Data Processing

- An example how to do simple image processing:
  - cosmic ray rejection using median filtering,
  - background subtraction using fitting,
  - and finally spectral extraction.
- Real data from the Hubble Space Telescope.

Go to

Data Processing

Notebook

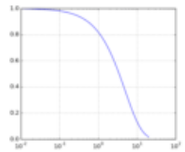
Visualising Data and Making Publication Quality Figures

Visualising Data

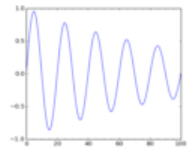
- Data visualisation is often crucial when trying to understand the data.
  - You should regularly plot everything against everything (pandas example) :-)
- Many Python packages like PyChar, igraph, graphviz, NetworkX, or you can call GnuPlot, Grace, etc. directly from Python.
- Will demonstrate one called matplotlib.

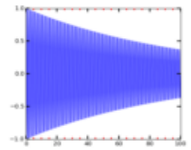
matplotlib

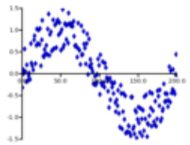
- Publication quality figures:
  - Supports e.g. LaTeX rendering.
  - Fonts, axes, etc. easily controllable.
  - Multiple output formats (ps, eps, PDF, ...)
- Resembles MATLAB in syntax.
- Can also be embedded to a GUI.
- Multiple backends: GTKAgg, QT4Agg, PDF...
- Active development community.

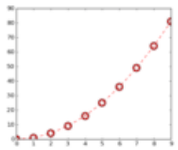

log_test

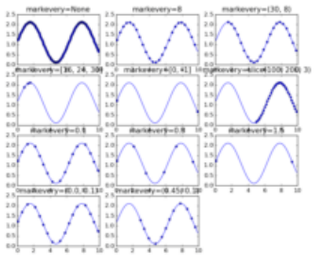
logo

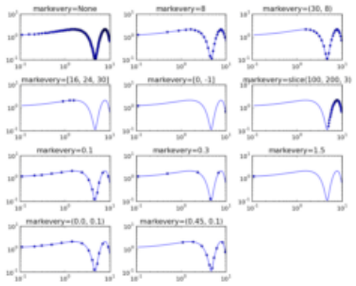

major_minor_demo1

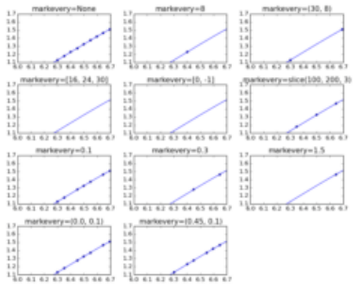

major_minor_demo2

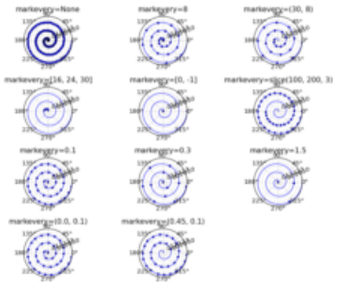

manual_axis

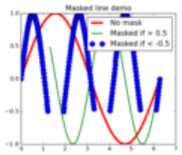

marker_path

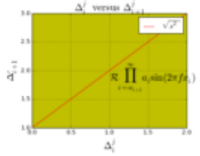

markevery_demo


markevery_demo

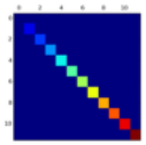

markevery_demo

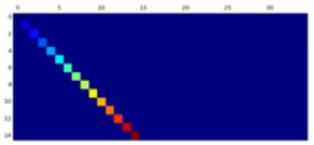

markevery_demo

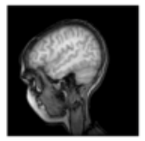

masked_demo

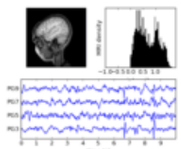

mathtext_demo

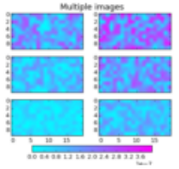
mathtext_examples

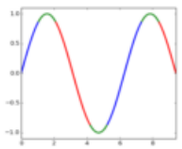

matshow

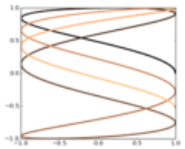

matshow

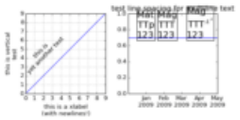

mri_demo

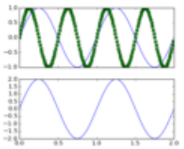

mri_with_eeg

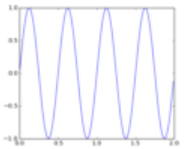

multi_image

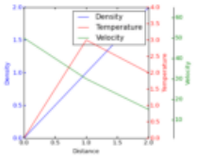

multicolored_line

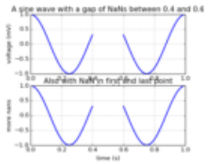

multicolored_line

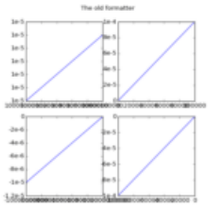

multiline

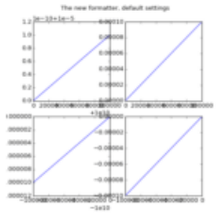

multiple_figs_demo

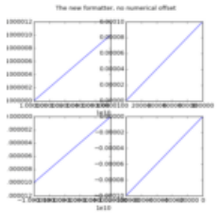

multiple_figs_demo

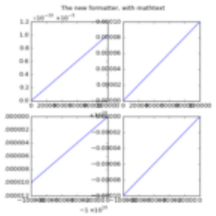

multiple_yaxis_with_spines

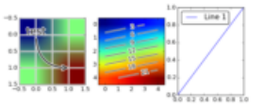

nan_test

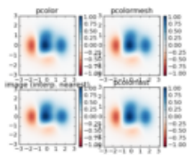

newscalarformatter_demo

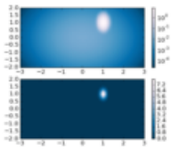

newscalarformatter_demo

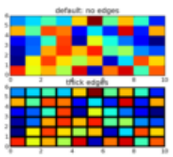

newscalarformatter_demo

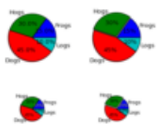

newscalarformatter_demo

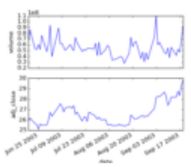

patheffect_demo

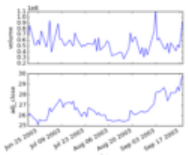

pcolor_demo

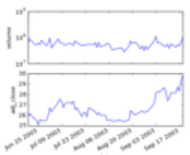

pcolor_log

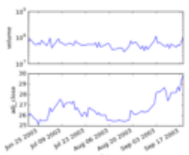

pcolor_small

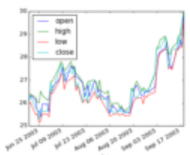

pie_demo2

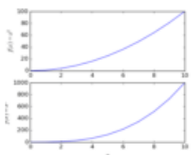

plotfile_demo

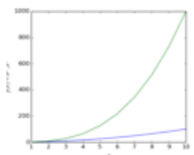

plotfile_demo


plotfile_demo

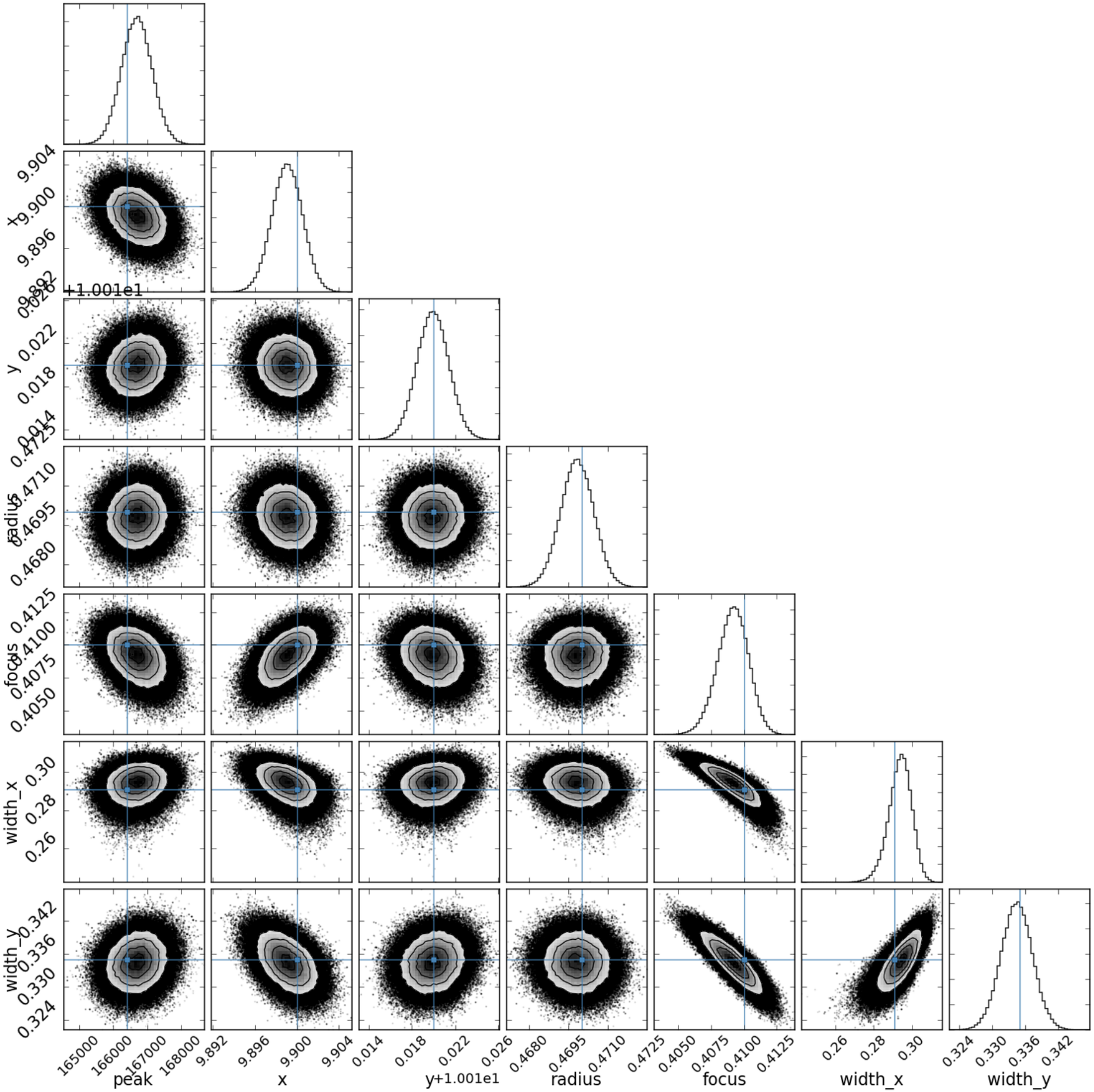

plotfile_demo

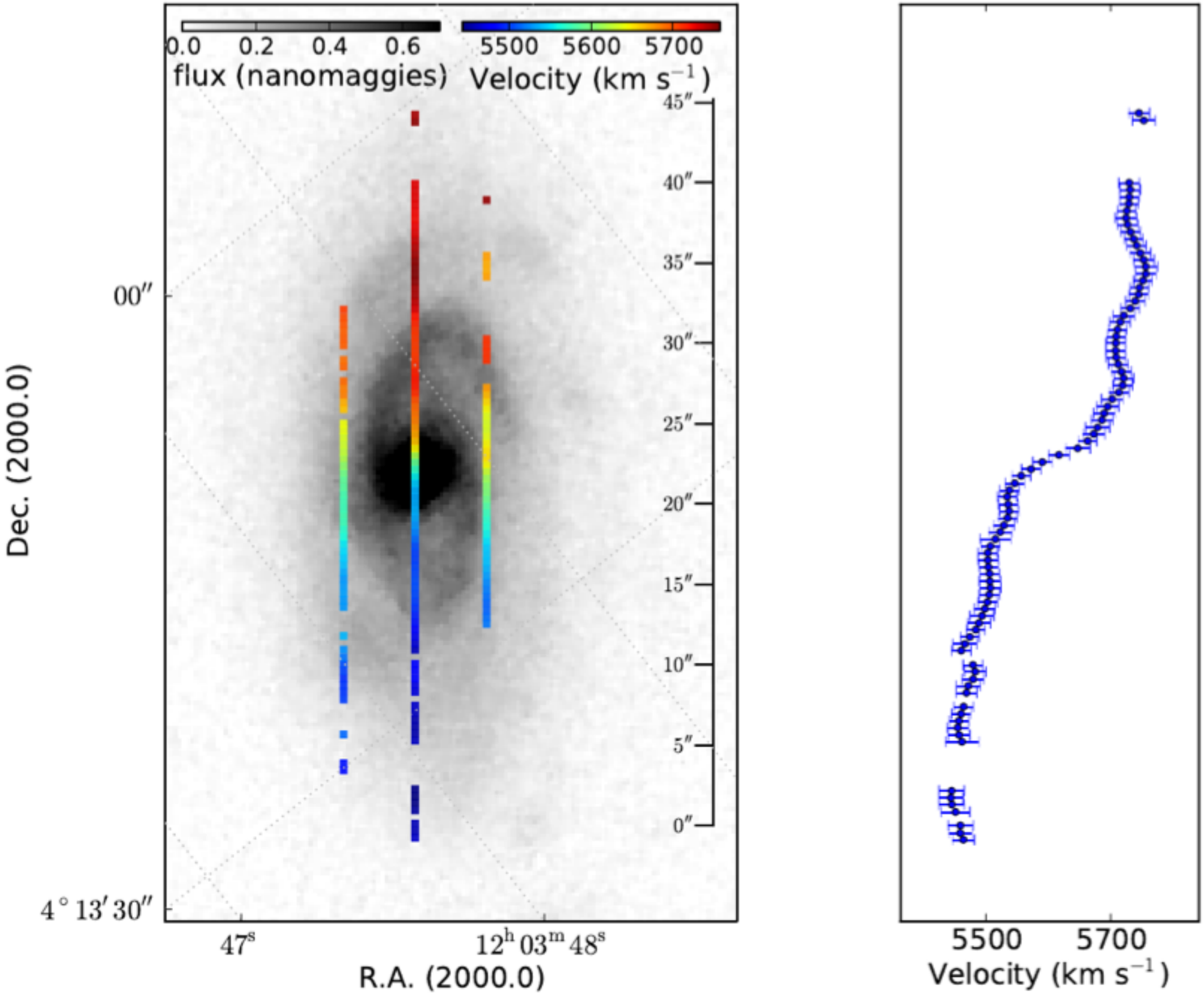

plotfile_demo

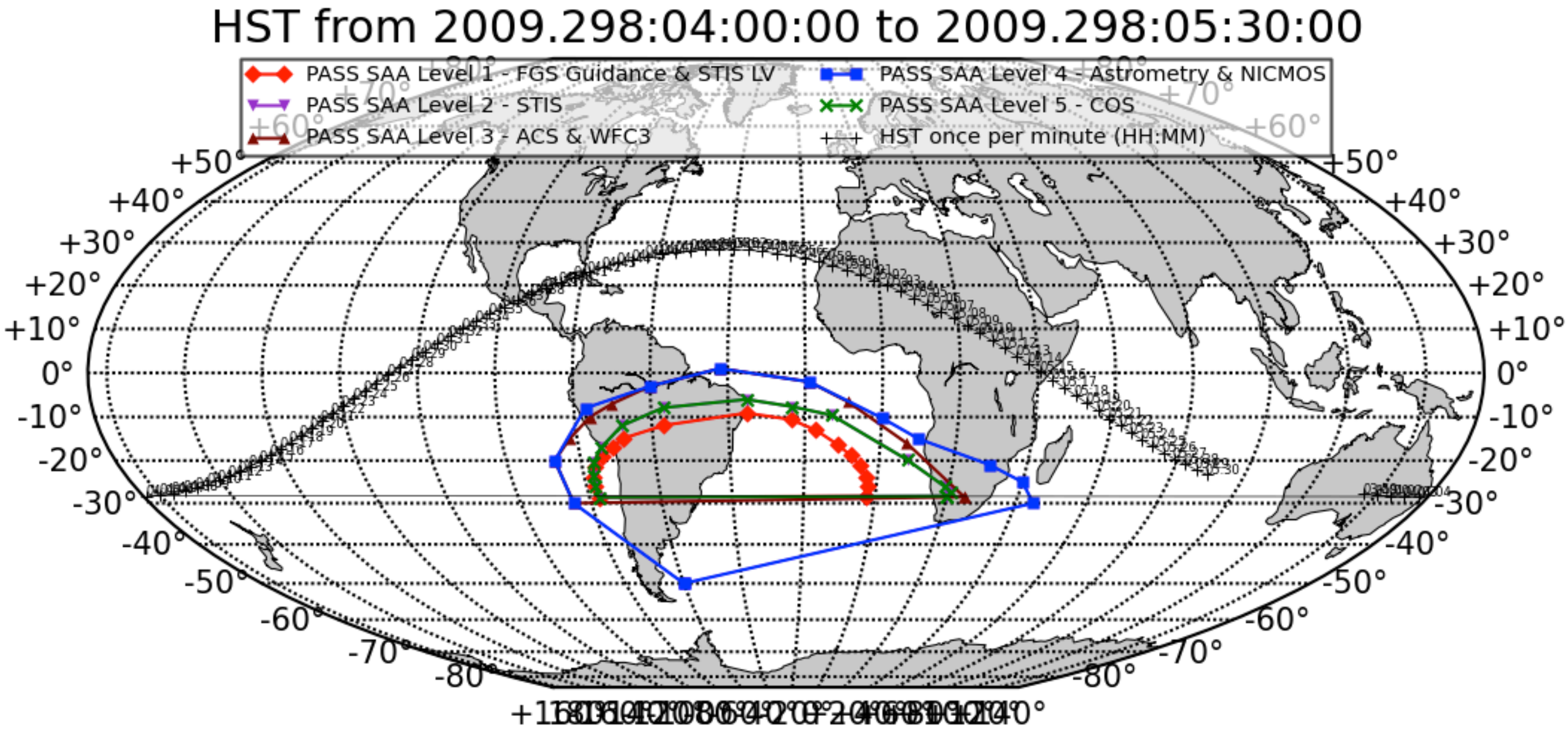

plotfile_demo

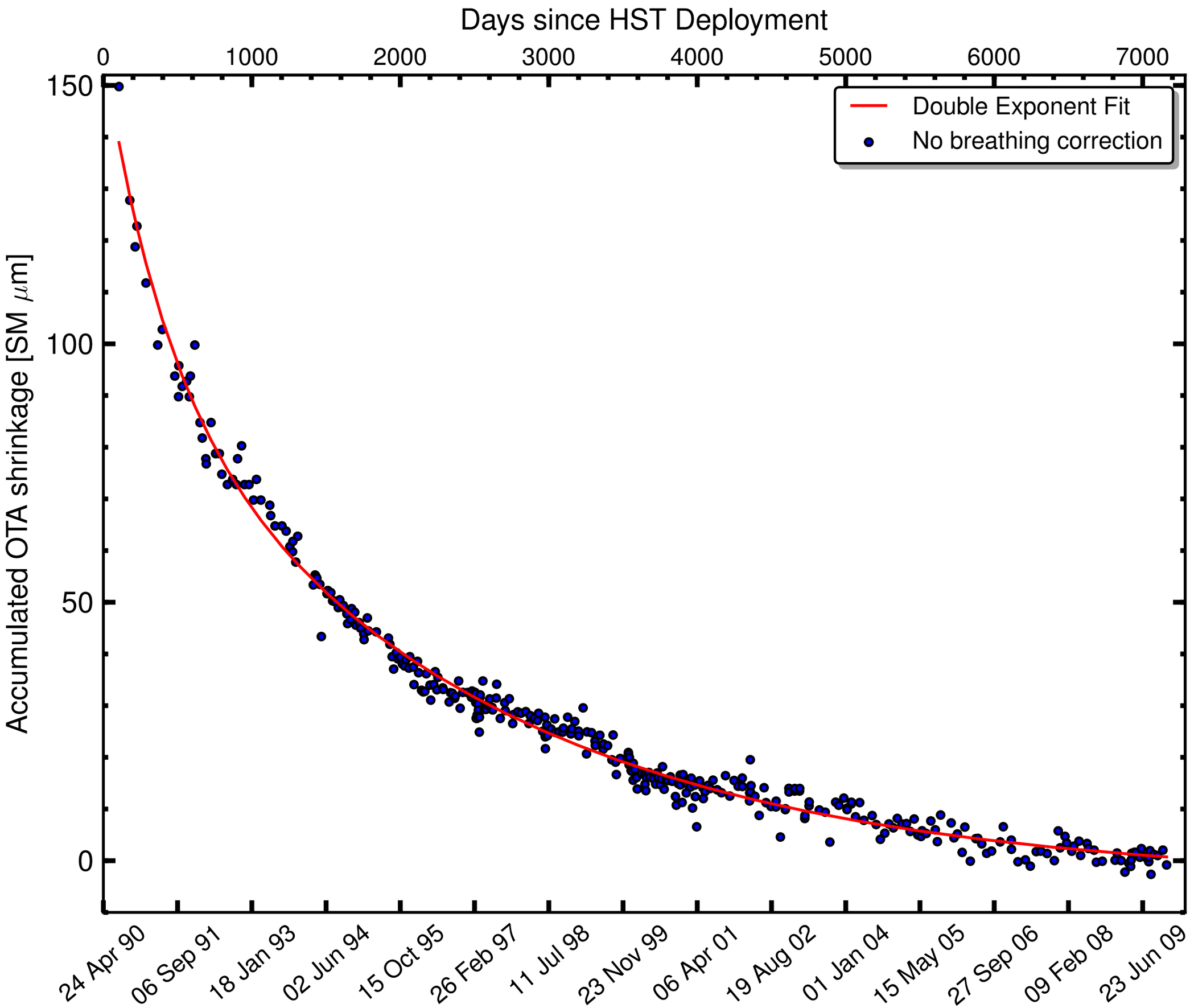

plotfile_demo

plotfile_demo

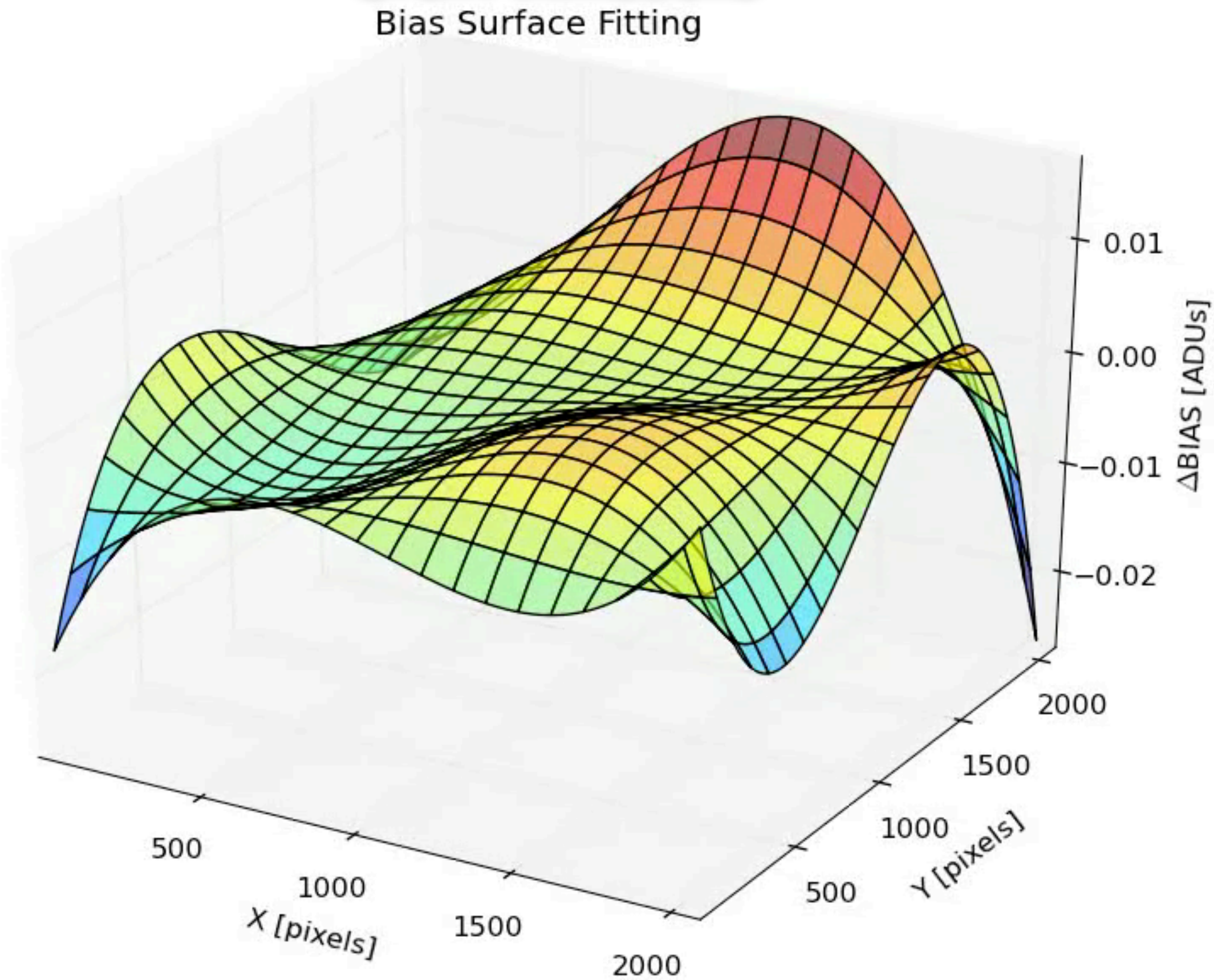

peak
x
y+1.001e1
radius
focus
width_x
width_y
+1.001e1


0.0
0.2
0.4
0.6
flux (nanomaggies)
5500
5600
5700
Velocity (km s^{-1})
45″
40″
35″
30″
25″
20″
15″
10″
5″
0″
00″
4° 13′ 30″
Dec. (2000.0)
47^{s}
12^{h} 03^{m} 48^{s}
R.A. (2000.0)
5500
5700
Velocity (km s^{-1})

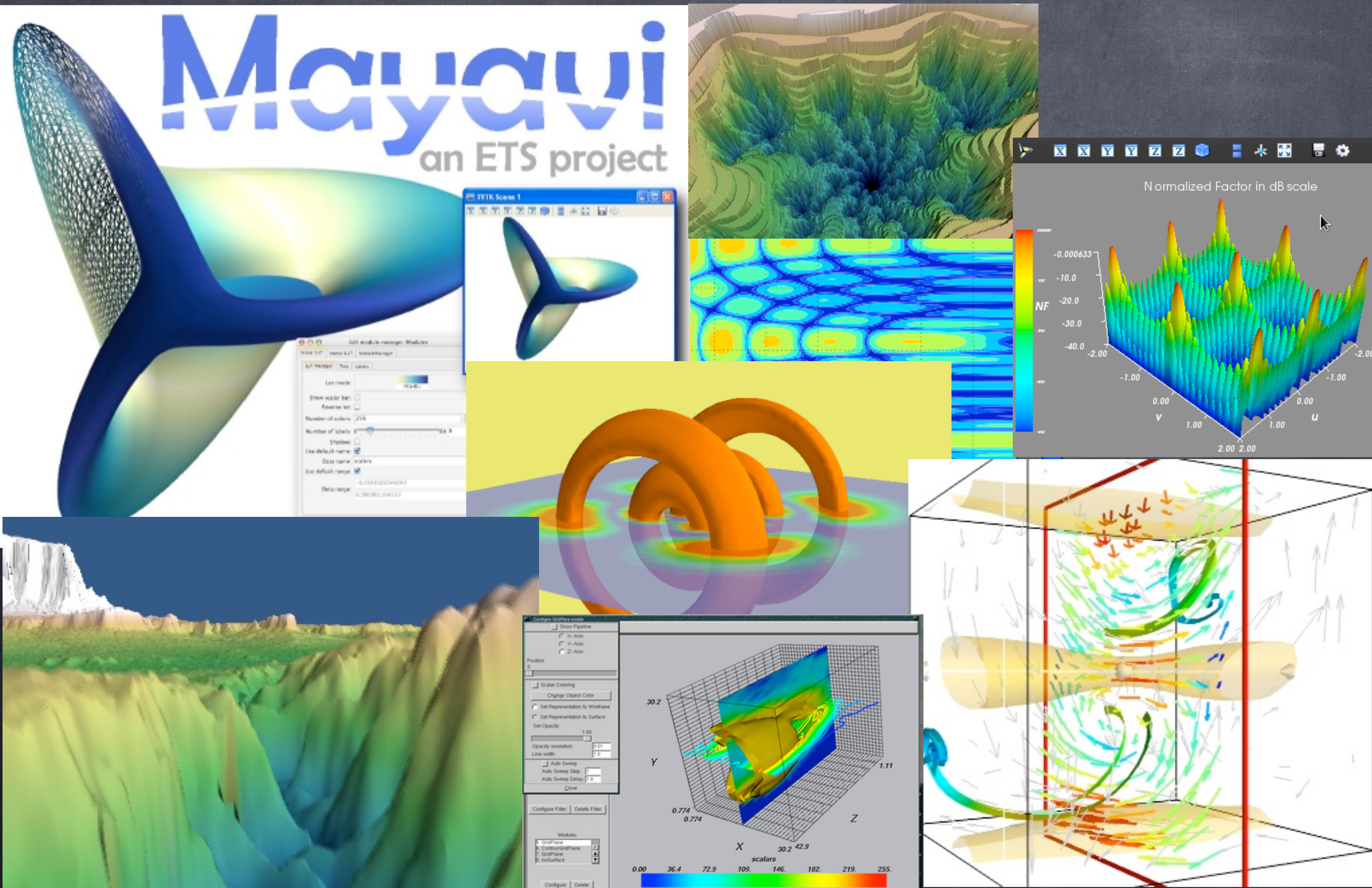

HST from 2009.298:04:00:00 to 2009.298:05:30:00
PASS SAA Level 1 - FGS Guidance & STIS LV
PASS SAA Level 2 - STIS
PASS SAA Level 3 - ACS & WFC3
PASS SAA Level 4 - Astrometry & NICMOS
PASS SAA Level 5 - COS
HST once per minute (HH:MM)
+50°
+40°
+30°
+20°
+10°
0°
-10°
-20°
-30°
-40°
-50°
-60°
-70°
-80°


Days since HST Deployment
0
1000
2000
3000
4000
5000
6000
7000
150
100
50
0
Accumulated OTA shrinkage [SM μm]
Double Exponent Fit
No breathing correction
24 Apr 90
06 Sep 91
18 Jan 93
02 Jun 94
15 Oct 95
26 Feb 97
11 Jul 98
23 Nov 99
06 Apr 01
19 Aug 02
01 Jan 04
15 May 05
27 Sep 06
09 Feb 08
23 Jun 09

Even Animations



3D Visualization: mayavi



Go to

Plotting Notebook

Model Fitting

Model Fitting

- We will fit a straight line (don't laugh) to data with outliers:
  - by estimating the maximum likelihood given Normally distributed uncertainties, and
  - by estimating the posterior probability of a model that allows outliers.
- We will fit more parameters than data points and I will (try to) convince you that it is the right thing to do, shocking I know...

What is probability?

- If I claim "the probability that this coin will land heads is 50%", what claim am I actually making?
  - classical statistician: in the limit of infinite number of coin flips, 50% of the results will be heads.
  - Bayesian: given what we know about the coin, we are 50% certain that the flip will land heads.
- Why does this matter?

Prior Information Matters

- Imagine 1000 boxes and one having prize.
  - If you choose randomly, the P(prize)=1/1000.
  - If someone who knows were the prize is opens 998 empty boxes from the 999 remaining, what happens?
    - The remaining box should have the prize (P(prize)=999/1000), why?
- If a third person walks in and chooses, without knowing that there were 1000 boxes initially, the P(prize) = 1/2.

Go to

Model Fitting Notebook

Machine Learning

Examples We Will Look

- Dimensional Reduction:
  - Principle Components Analysis and MDS.
- Classification:
  - Gaussian Naive Bayes and Random Forest.
- Clustering:
  - K Means.
- Regression:
  - Gaussian Process.

In Addition (if time)

- Outlier Detection:
  - Support Vector Machines and Convariance Estimator.

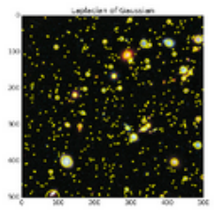
Image Processing

Image Processing

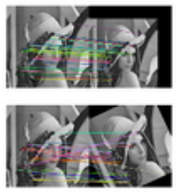
- Images can be considered as 2D arrays.
- scipy.ndimage provides e.g.:
  - **filters**: convolution, Gaussian, Laplace, Sobel, maximum, minimum, percentile...
  - **interpolations**: rotation, zoom, map_coorinates, arbitrary transform.
  - **measurements**: centre-of-mass, extrema, etc.
  - **morphologies**: dilation, erosion, opening, closing, etc.

Image Processing

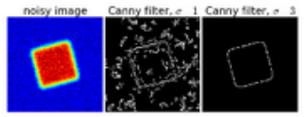
- scikits-image: extends scipy.ndimage



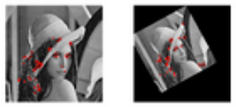
Blob Detection



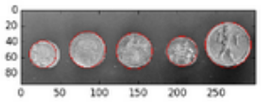
BRIEF binary descriptor



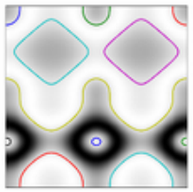
Canny edge detector



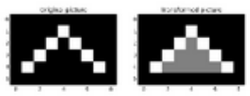
CENSURE feature detector



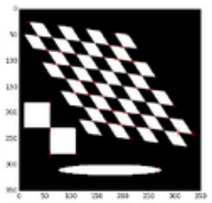
Circular and Elliptical Hough Transforms



Contour finding



Convex Hull

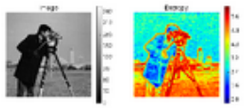


Corner detection

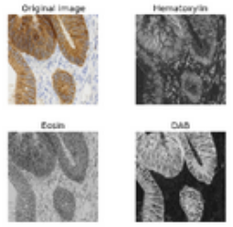
Dense DAISY feature description

Denoising the picture of Lena

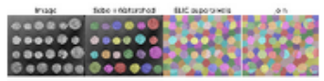
Edge operators



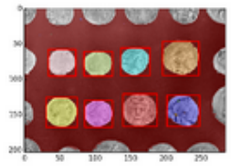
Entropy



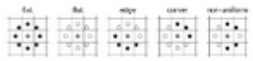
Immunohistochemical staining colors separation



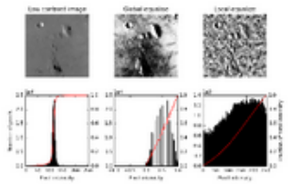
Find the intersection of two segmentations



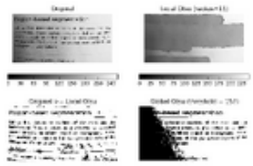
Label image regions



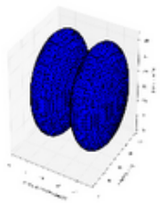
Local Binary Pattern for texture classification



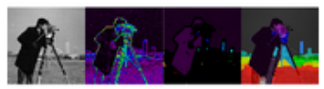
Local Histogram Equalization



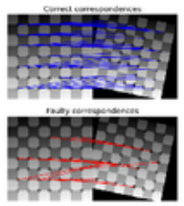
Local Otsu Threshold



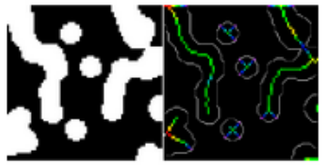
Marching Cubes



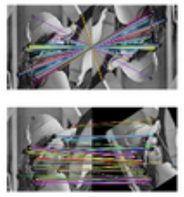
Markers for watershed transform



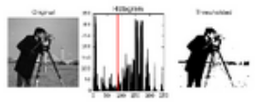
Robust matching using RANSAC



Medial axis skeletonization



ORB feature detector and binary descriptor



Thresholding

Examples We Will Look

- Drawing a chess board.
  - Simple array manipulation.
- Edge detection.
  - Sobel Filter.
- Template Matching.
- Object Detection.

Will show a few scikits examples, a detailed treatment of the topic would take weeks...

If time, we can look at e.g. symbolic calculus...

Further Reading

- http://scipy-lectures.github.com/_downloads/PythonScientific-simple.pdf
- http://stsdas.stsci.edu/perry/pydatatut.pdf
- http://python4mpia.github.com/intro/quick-tour.html
- http://minrk.github.com/scipy-tutorial-2011/
- http://python4astronomers.github.com/index.html
- https://github.com/profjsb/python-bootcamp