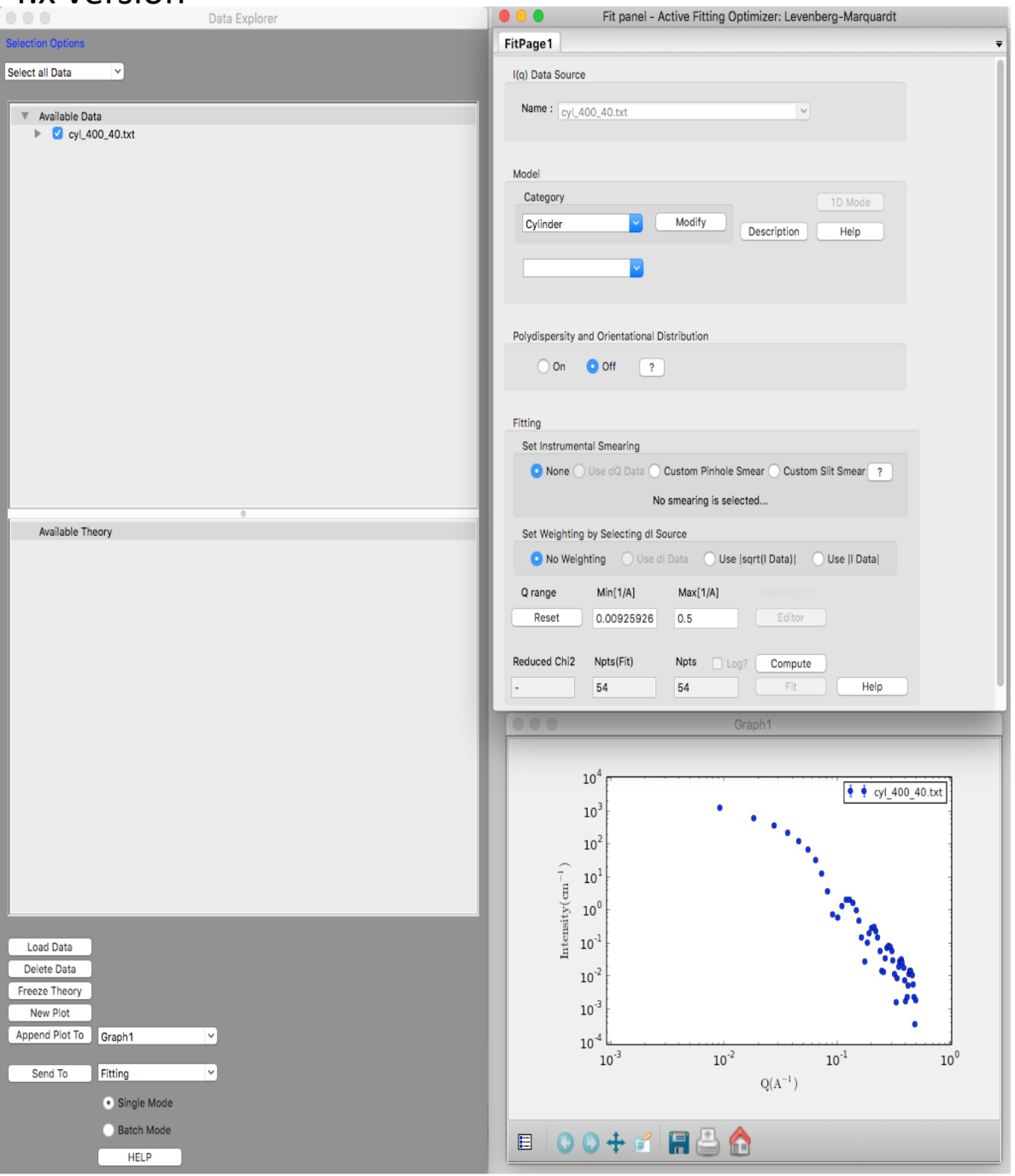
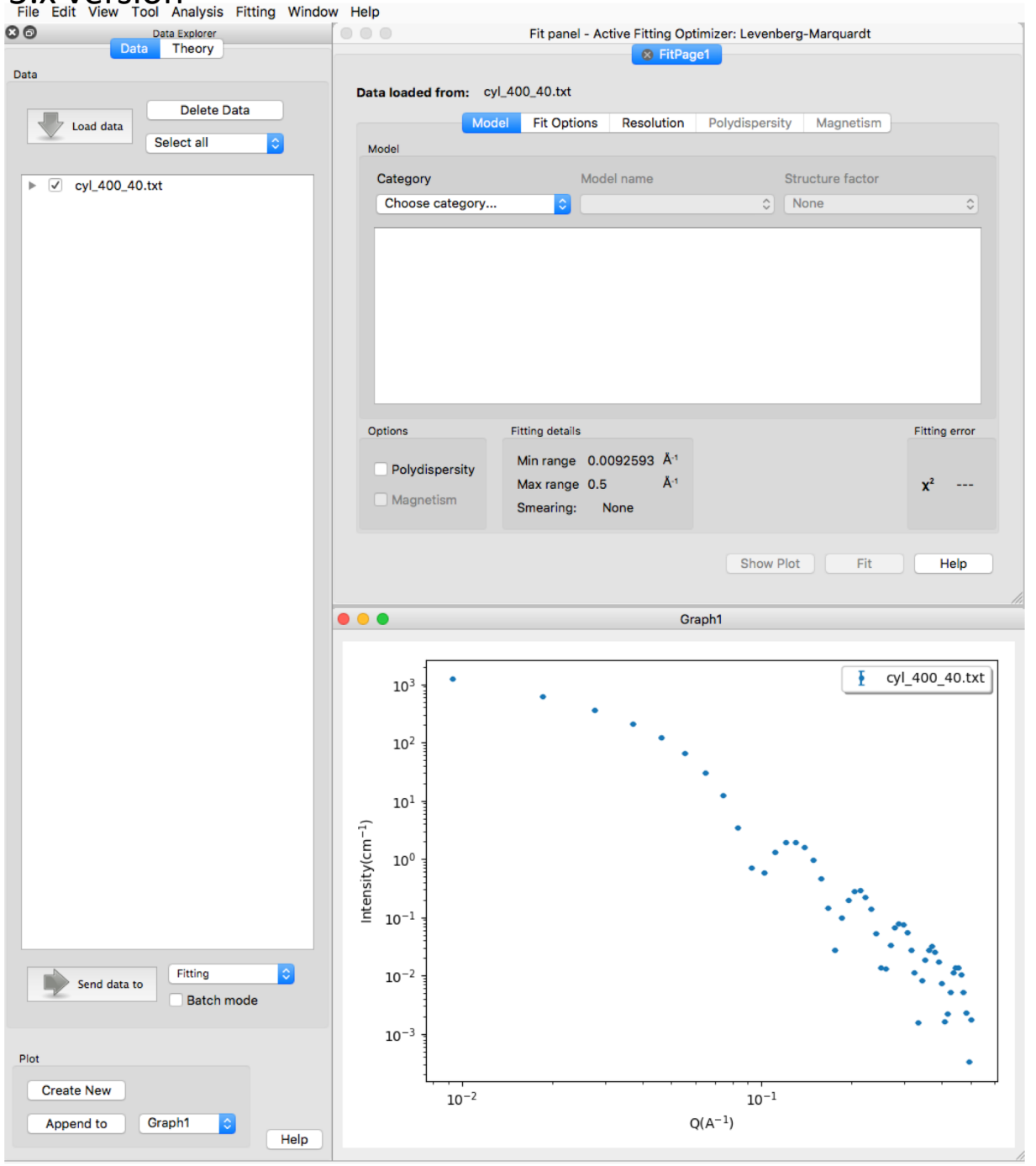
SasView
new developers intro

4.x version



5.x version



Setting up developers enviroment

Conda based installation

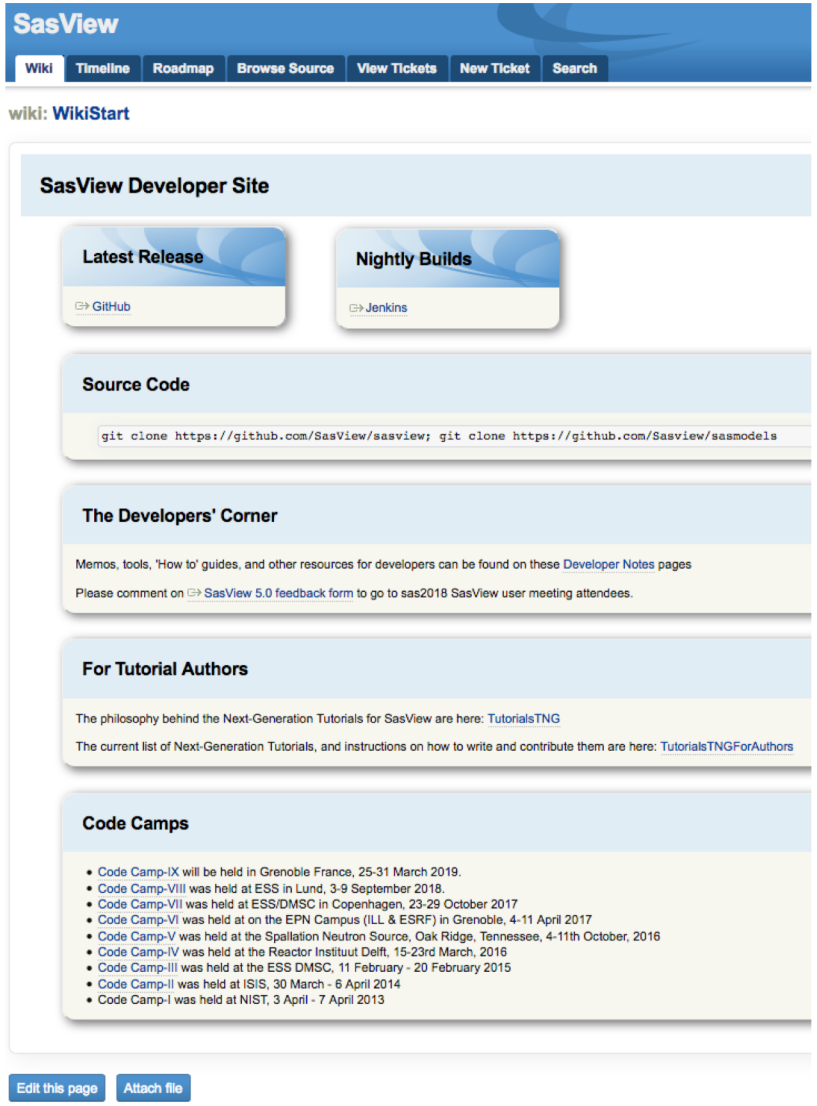
- http://trac.sasview.org/wiki/DevNotes/DevGuide/CondaDevSetup5.0
- http://trac.sasview.org/wiki/DevNotes/DevGuide/CondaDevSetup4.x

Github account

Trac account

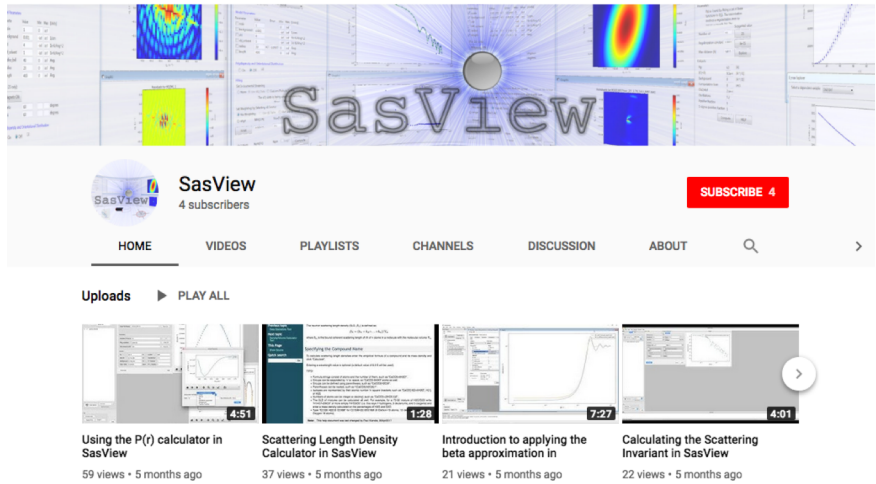
JIRA account

Trac and website



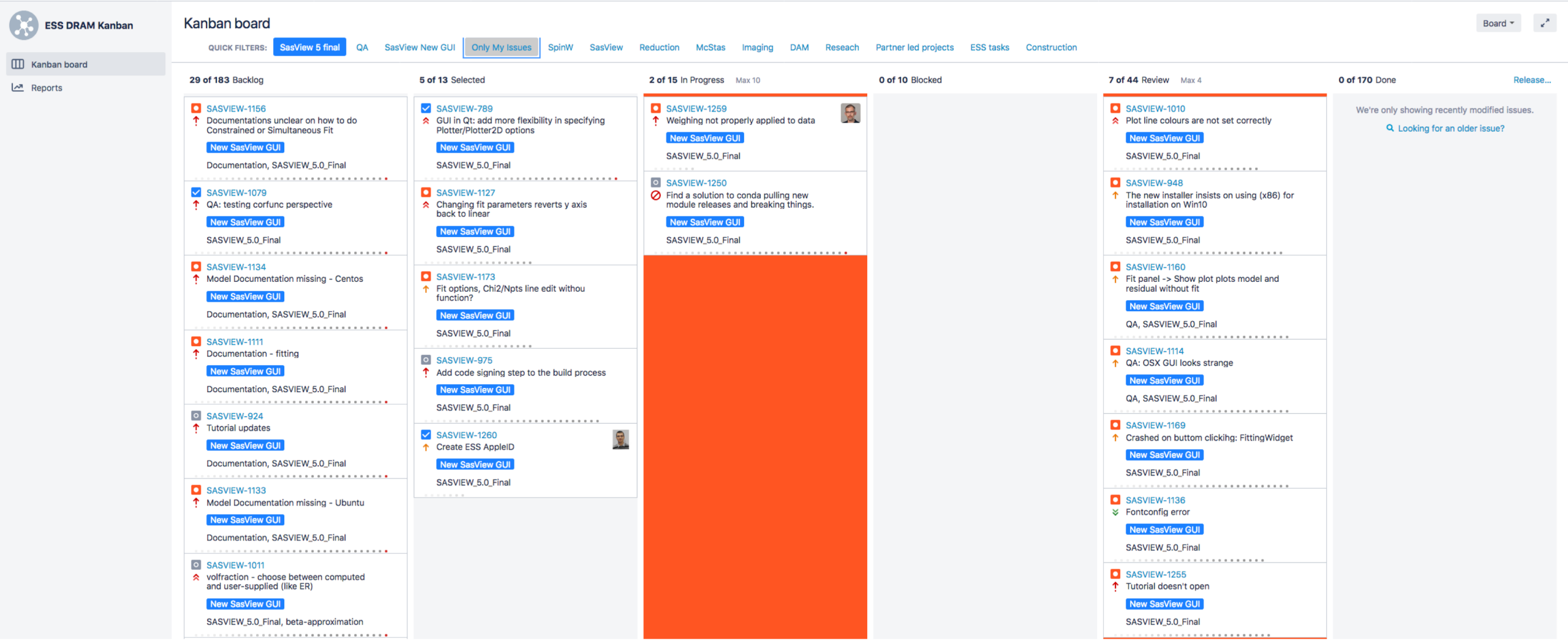
http://trac.sasview.org/

http://www.sasview.org/

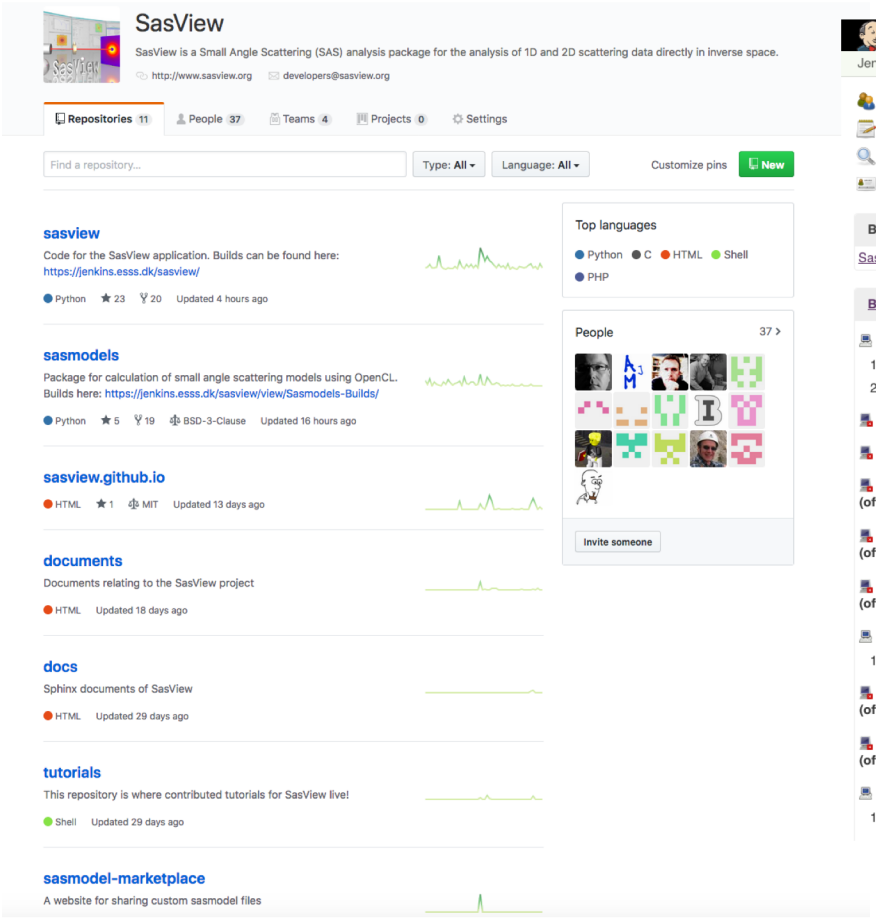


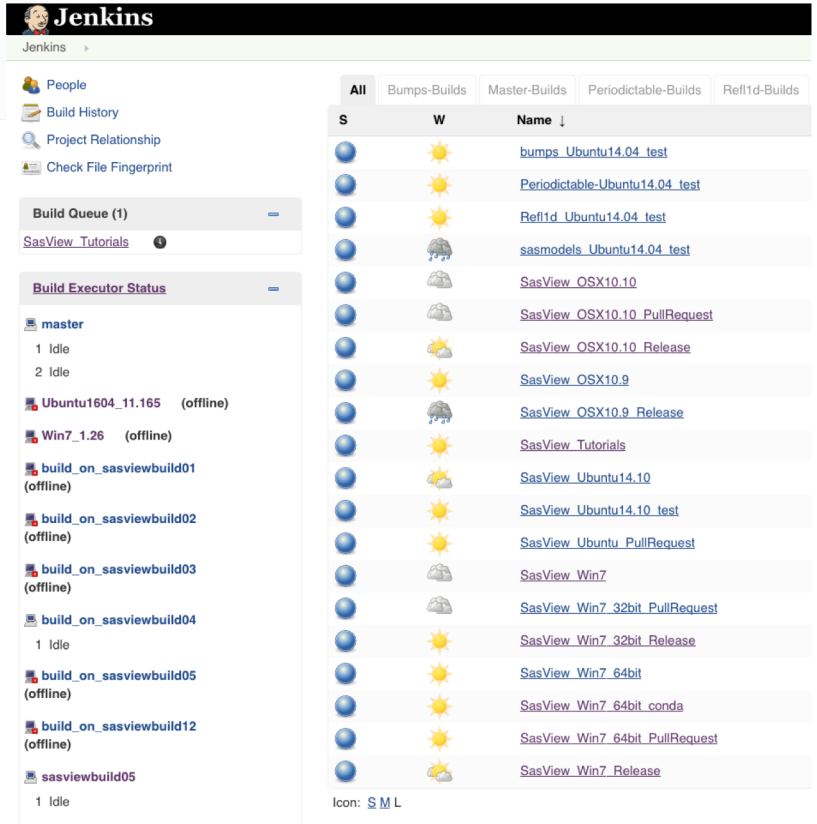
YouTube channel

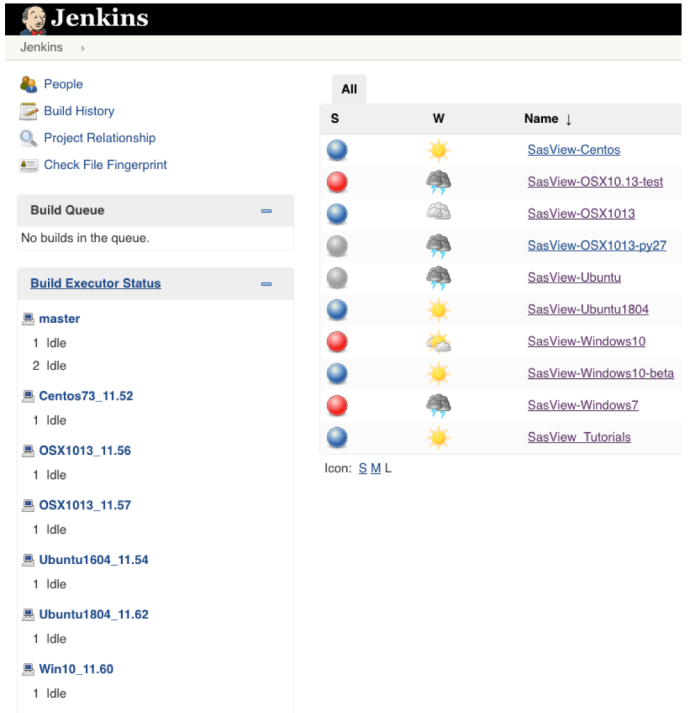
JIRA (SasView 5.x only)



Github and build servers





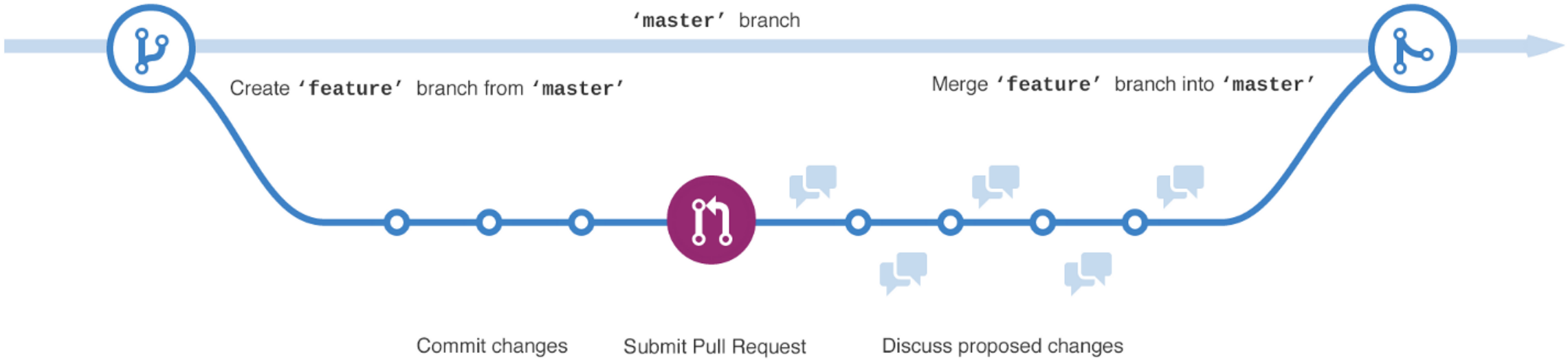


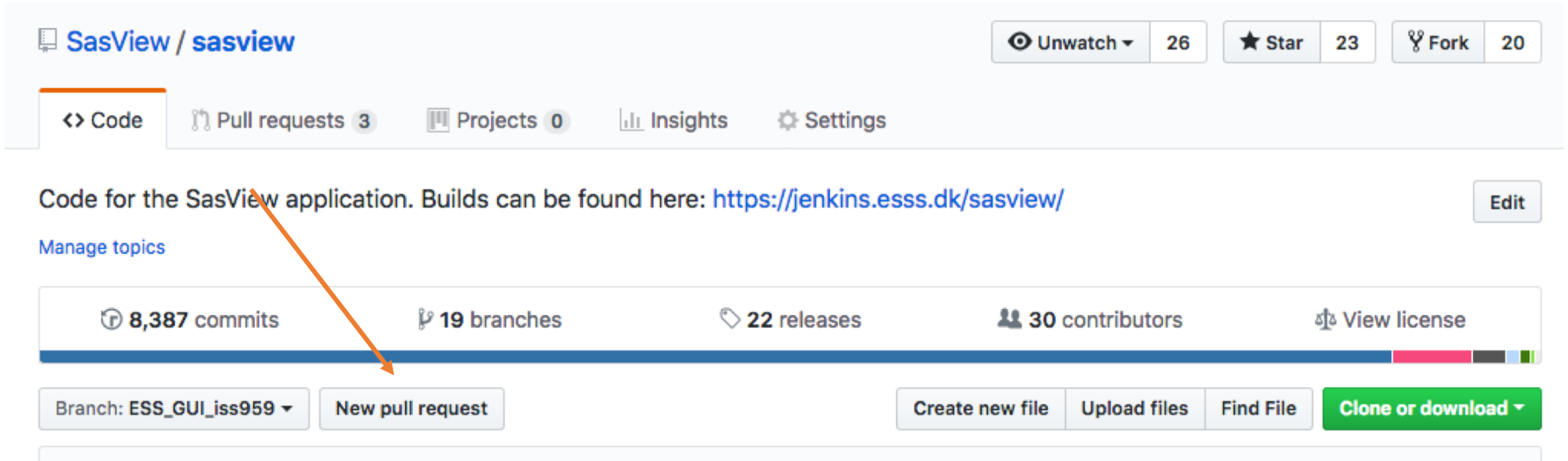
https://github.com/SasView

https://jenkins.esss.dk/sasview/

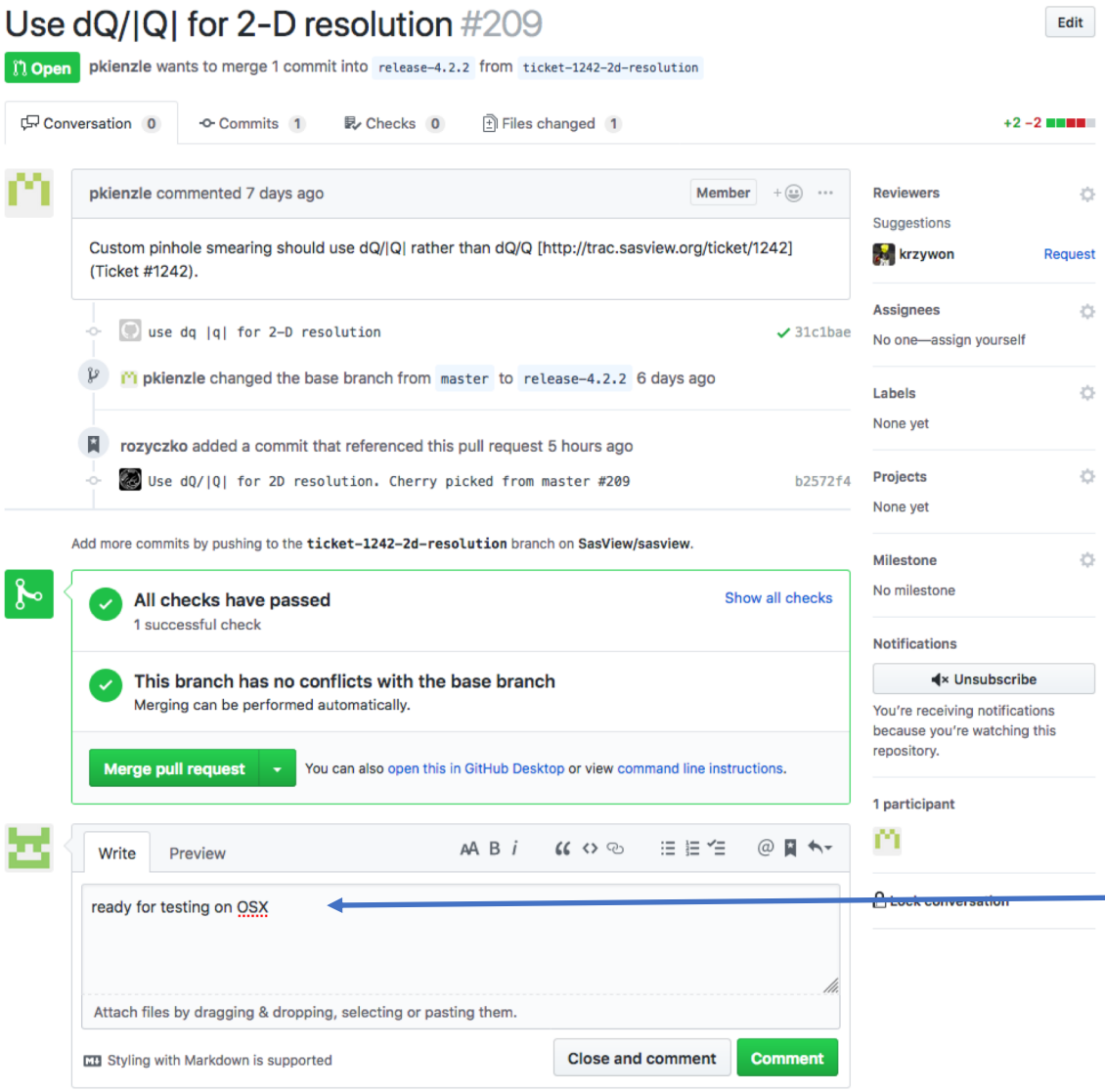
https://jenkins.esss.dk/sasview-beta/

Pull Requests



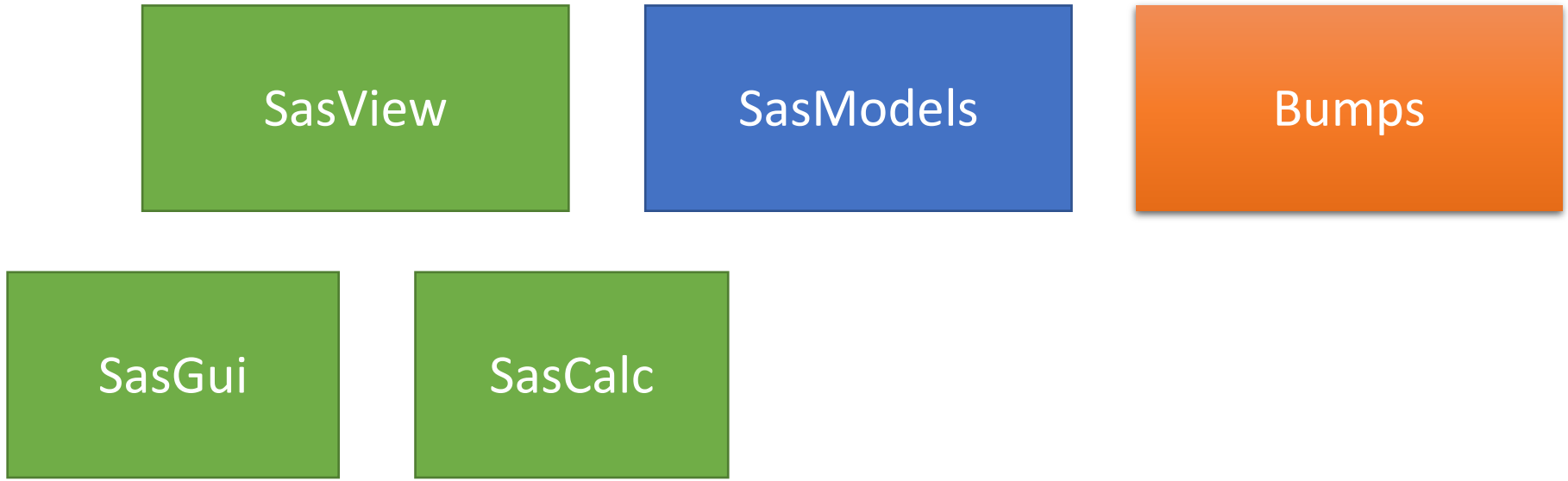


Pull request testing



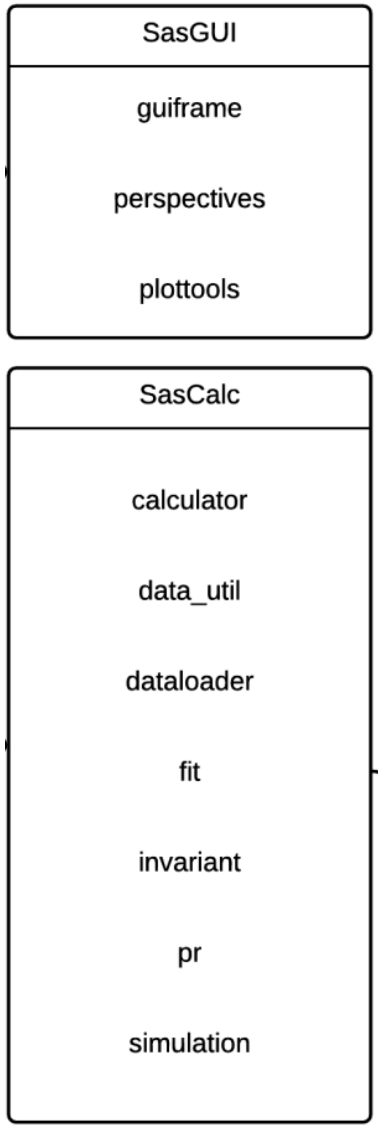
Magic phrase:
ready for testing on OSX
ready for testing on Linux
ready for testing on Win64
ready for testing on Win32

SasView components



SasGUI

- 4.x developed with wx python
- 5.0 developed with Qt: src/sas/sasgui/qtgui in (ESS_GUI branch)
- SasGui consists of:
  - Perspectives: src/sas/sasgui/perspectives (fitting, invariant, calculators, corfunc)
  - GuiFrame: src/sas/sasgui/guiframe (non-perspective gui elements)
  - PlotTools: src/sas/sasgui/plottols (ploting tools)

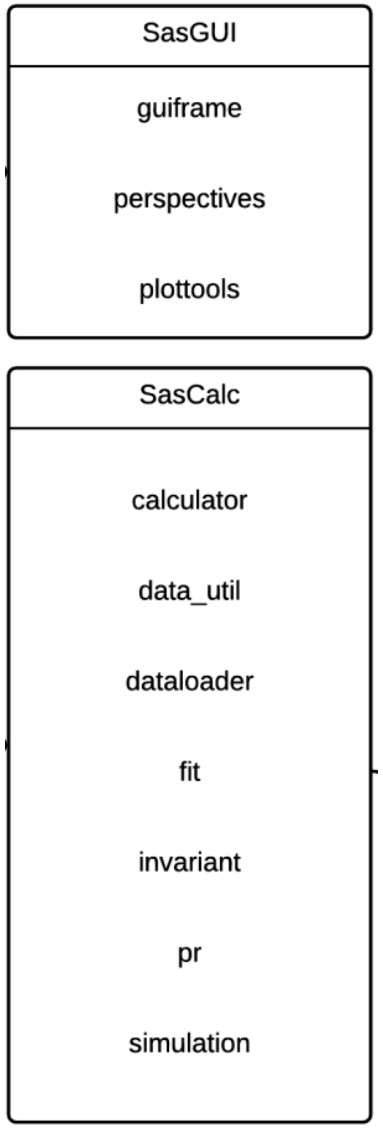


SasGUI – less obvious cases

- Model editor:
  src/sas/sasgui/perspectives/calculator/model_editor.py
- Category manager:
  src/sas/sasgui/guiframe/CategoryManager.py
- Settings for OpenCL:
  src/sas/sasgui/perspectives/fitting/gpu_options.py
- Fitting options (interface to bumps):
  bumps/gui/fit_dialog

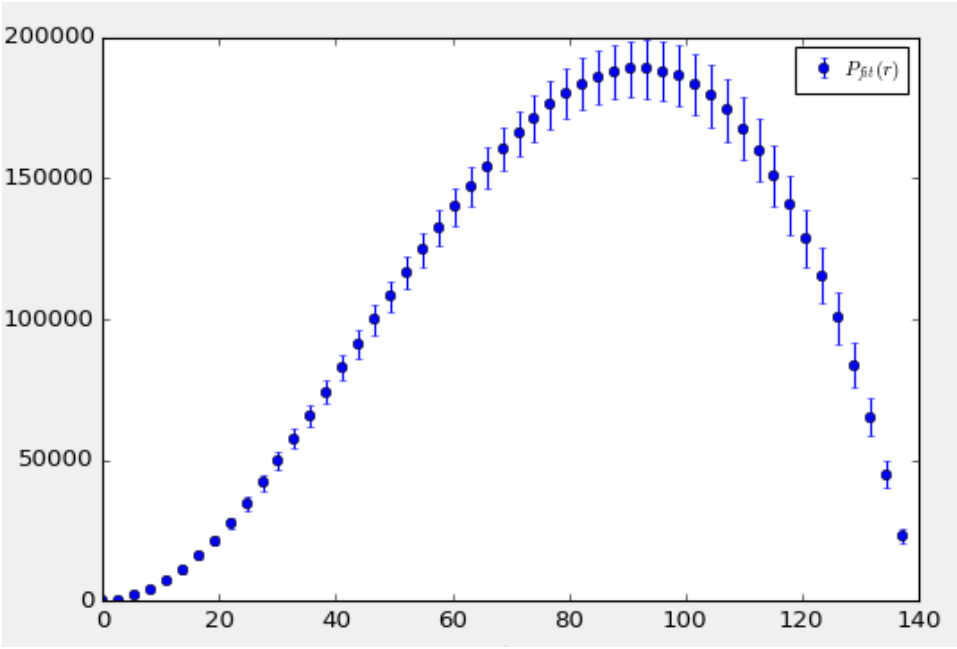
SasCalc

- Back-end calculations for sasgui perspective
- Data fitting:
  - src/sas/sascalc/fit/
- Pr inversion
  - src/sas/sascalc/pr/
- Data loader: src/sas/sascalc/dataloader/loader.py



SasCalc

```
from sas.sascalc.dataloader.loader import Loader
from sas.sascalc.pr.invertor import Invertor

loader = Loader()
test_data = loader.load("sphere_80.txt")

pr = Invertor()

# Set data
pr.x   = test_data.x
pr.y   = test_data.y

# Perform inversion and show graph
x, y = pr.invert()

import matplotlib.pyplot as plt
plt.plot(x, y)
plt.show()
```



And more at: https://github.com/SasView/documents/blob/master/Meetings/User_Meeting_2018/SasView%2BCLI%2Boverview.html

SasModels

- Models located in: sasmodels/models
- Three different types of models:
  - Python only: broad_peak.py
  - Python with embedded C: stickyhardsphere.py
  - Python and separate C: barbell.py and barbell.c

SasModels essential components

- Model Description: Docstring
- Name, title, short description
- Parameters table
- C files to be included during compilation
- Unit tests

SasModels functions

- Iq - 1D scattering intensity functions for the case where the scatterer is randomly oriented
- Iqxy - the 2D scattering intensity functions provide I(Q,ϕ) for an oriented system as a function of a Q
- Form_volume - calculates the volume of particle V, which is used to normalize form factor.
- ER – calculates effective radius
- VR - calculate particle volume/total volume for shape models
- INVALID(v) returns False if v.parameter is invalid
  for some parameter or other (e.g., v.bell_radius < v.radius).

SasModels core modules

- Model computation:
  - sasmodels/sasview_model.py – interface to sasview
  - sasmodels/kernel.py -interface to all kernel models
  - sasmodels/kerneldll.py –dll driver for c kernels
  - sasmodels/kernelpy.py – python driver
  - sasmodels/kernelcl.py – opencl driver
  - sasmodels/core.py - prepares the model for the execution
  - sasmodels/generate.py – generates model file based on the template

Sasmodels compare

- Computes model and compares between different platforms (opencl, dll)
- Allows for comparison with old sasview version
- ./compare.sh modelname -1d
- ./compare.sh modelname -2d
- Multiple precision, q ranges options

Unit tests

- SasView tests are located: sasview/test
  - calculatorview
  - corfunc
  - fileconverter
  - pr_inversion
  - sascalculator
  - sasdataloader
  - sasguiframe
  - sasinvariant
  - sasrealspace
- Sasmodels unit tests defined inside models

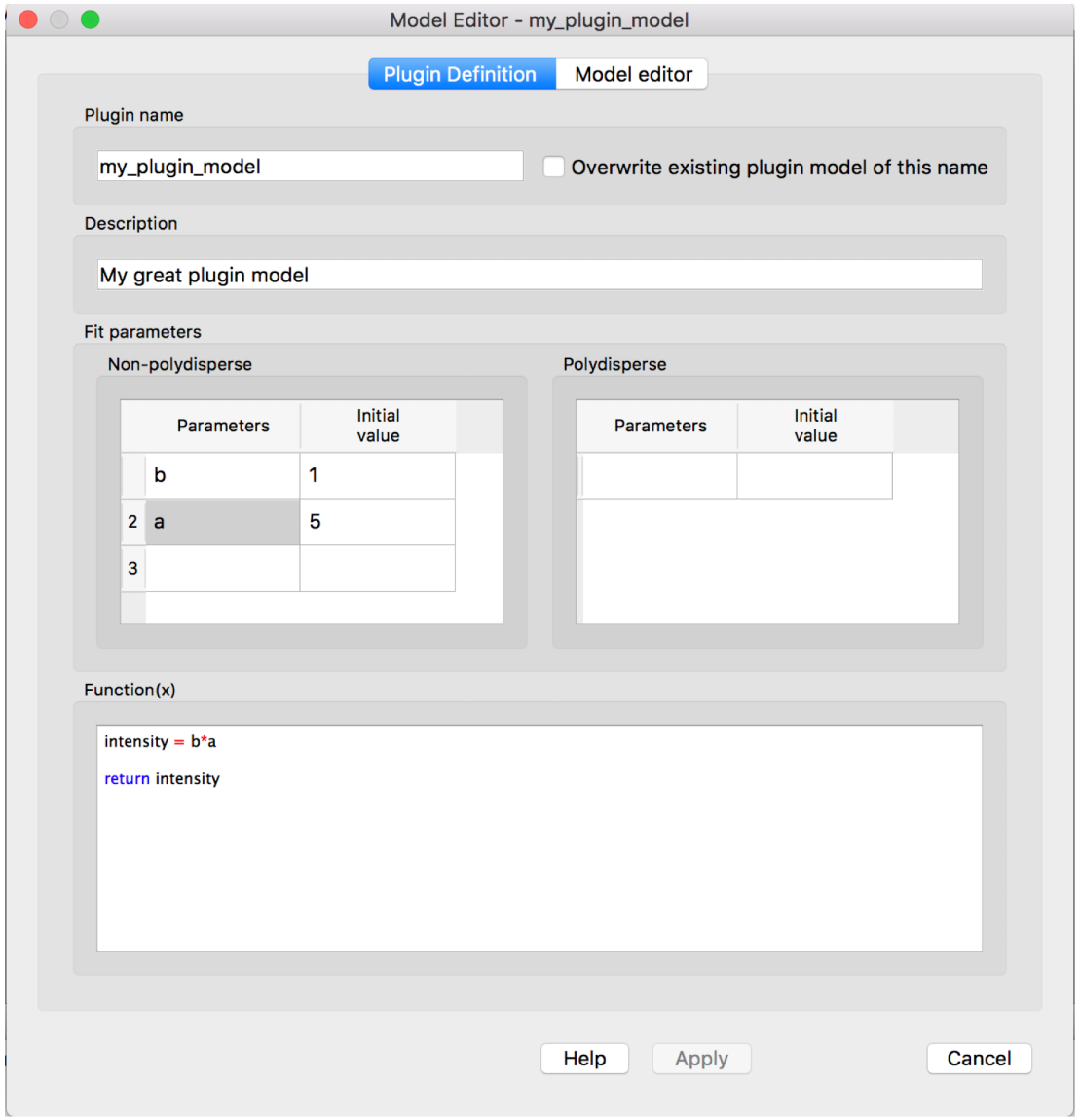
Running bumps with sasmodels

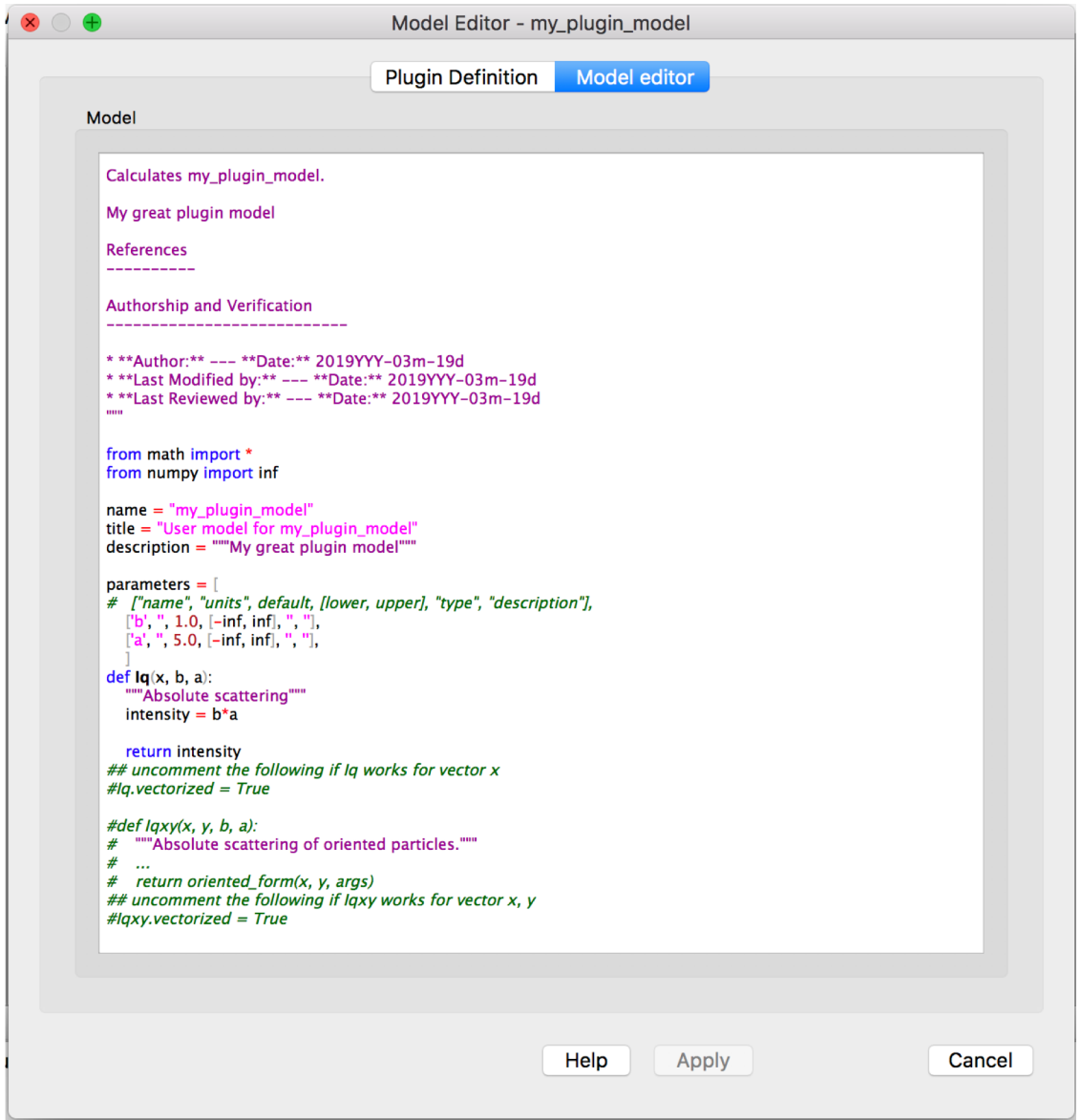
- Sasmodels example contains data set for oriented rod-like shape
- sasmodels/examples/fit.py (sasmodels interface to bumps)
- Running
  - bumps fit.py cylinder –preview
  - Fit.py accepts two arguments: type of model and view (radial and tnagential)

Custom settings and models

- Local configuration
  - ~/.sasview/config/custom_config.py
- Plugin models:
  - ~/.sasview/plugin_models/
- Model categories:
  - ~/.sasview/categories.json

Plugin model editor (Fitting->Add Custom Model)





Python and C model files

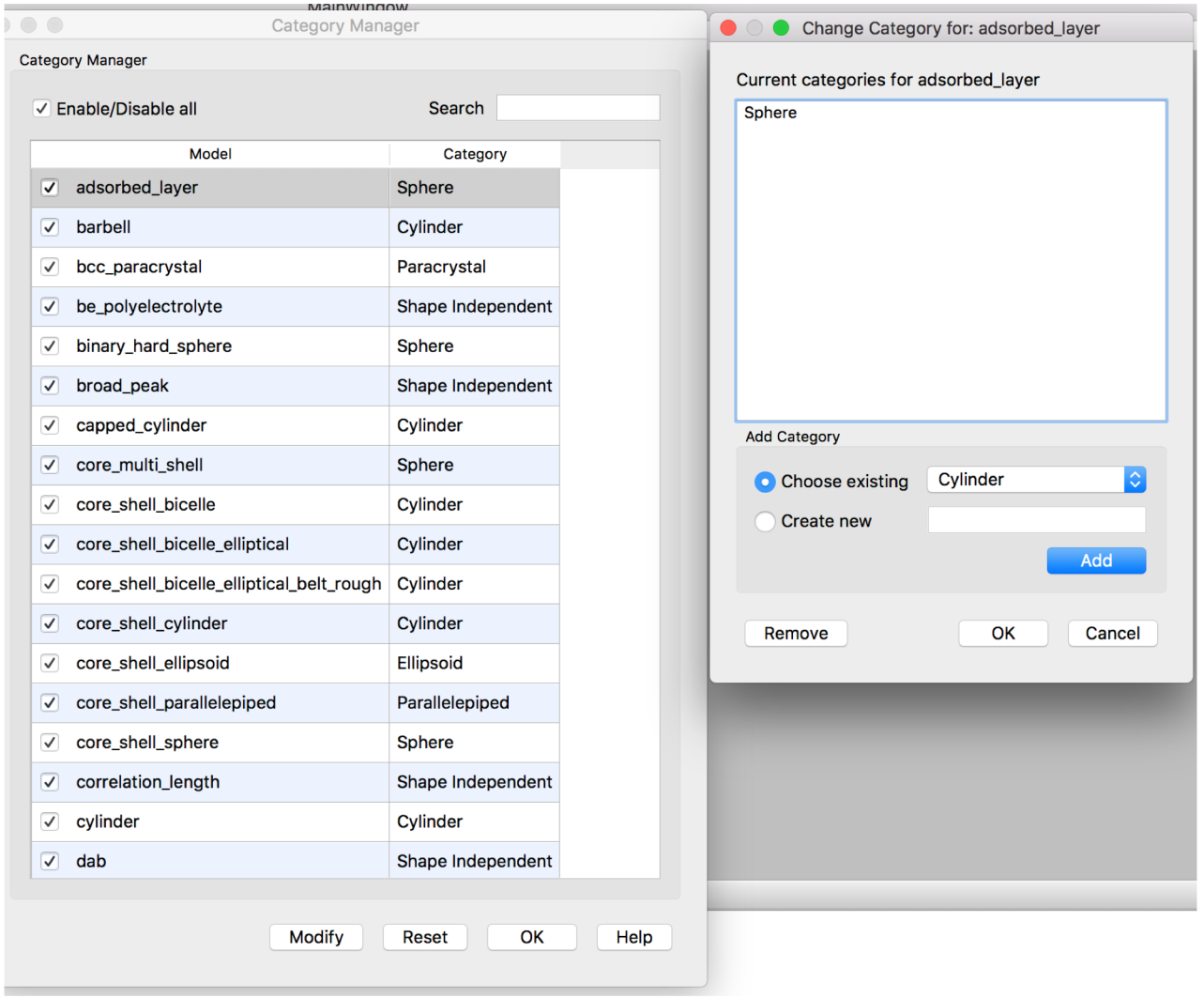
cylinder.py

```
import ...

name = "cylinder"
title = "Right circular cylinder with uniform scattering length density."
description = """
     f(q,alpha) = 2*(sld - sld_solvent)*V*sin(qLcos(alpha)/2))
                /[qLcos(alpha)/2]*J1(qRsin(alpha))/[qRsin(alpha)]

           P(q,alpha)= scale/V*f(q,alpha)^(2)+background
           V: Volume of the cylinder
           R: Radius of the cylinder
           L: Length of the cylinder
           J1: The bessel function
           alpha: angle between the axis of the
           cylinder and the q-vector for 1D
           :the ouput is P(q)=scale/V*integral
           from pi/2 to zero of...
           f(q,alpha)^(2)*sin(alpha)*dalpha + background
"""
category = "shape:cylinder"

#            [ "name", "units", default, [lower, upper], "type", "description"],
parameters = [["sld", "1e-6/Ang^2", 4, [-inf, inf], "sld",
               "Cylinder scattering length density"],
              ["sld_solvent", "1e-6/Ang^2", 1, [-inf, inf], "sld",
               "Solvent scattering length density"],
              ["radius", "Ang", 20, [0, inf], "volume",
               "Cylinder radius"],
              ["length", "Ang", 400, [0, inf], "volume",
               "Cylinder length"],
              ["theta", "degrees", 60, [-360, 360], "orientation",
               "cylinder axis to beam angle"],
              ["phi", "degrees", 60, [-360, 360], "orientation",
               "rotation about beam"],
             ]

source = ["lib/polevl.c", "lib/sas_J1.c", "lib/gauss76.c", "cylinder.c"]

def ER(radius, length):
    """
        Return equivalent radius (ER)
    """
    ddd = 0.75 * radius * (2 * radius * length + (length + radius) * (length + pi * radius))
```

cylinder.c

```
#define INVALID(v) (v.radius<0 || v.length<0)

static double
form_volume(double radius, double length)
{
    return M_PI*radius*radius*length;
}

static double
fq(double qab, double qc, double radius, double length)
{
    return sas_2J1x_x(qab*radius) * sas_sinx_x(qc*0.5*length);
}

static double
orient_avg_1D(double q, double radius, double length)
{
    // translate a point in [-1,1] to a point in [0, pi/2]
    const double zm = M_PI_4;
    const double zb = M_PI_4;

    double total = 0.0;
    for (int i=0; i<GAUSS_N ;i++) {
        const double theta = GAUSS_Z[i]*zm + zb;
        double sin_theta, cos_theta; // slots to hold sincos function output
        // theta (theta,phi) the projection of the cylinder on the detector plane
        SINCOS(theta , sin_theta, cos_theta);
        const double form = fq(q*sin_theta, q*cos_theta, radius, length);
        total += GAUSS_W[i] * form * form * sin_theta;
    }
    // translate dx in [-1,1] to dx in [lower,upper]
    return total*zm;
}

static double
Iq(double q,
    double sld,
    double solvent_sld,
    double radius,
    double length)
{
    const double s = (sld - solvent_sld) * form_volume(radius, length);
    return 1.0e-4 * s * s * orient_avg_1D(q, radius, length);
}
```

Category manager (View->Category Manager)



SasView marketplace

Welcome to the SasView Marketplace!

The Marketplace allows members of the SAS Community to contribute plug-in fitting models for the popular **SasView** data analysis program for all to use. (Please note: these plug-in models require SasView version 4.0 or later).

Contributed models should be written in Python (compatibility with python 2.7.x is currently required) or, if computational speed is an issue, in a combination of Python and C. You only need to upload the .py/.c source code files to the Marketplace!

Instructions on how to code a new plug-in model are available here. It may also be helpful to examine the library models in the \sasmodels-data\models sub-folder of your SasView installation directory.

You also have the option to upload a SasView text file of the scattering function data computed by your model. If you do this a graph of the scattering function will appear under the Marketplace entry for your model.

Disclaimer: Models are expected to be contributed to the Marketplace in good faith and with the best intentions, and come without any warranty as to their correctness and suitability for purpose. Neither the author of the model or the SasView Development Team will be held liable for any loss or damage arising from conclusions drawn from the use of a model contributed to the Marketplace. This includes models that are marked as 'verified' by the SasView Development Team.

Categories:

- Cylinder
- Ellipsoid
- Lamellae
- Other
- Paracrystal
- Parallelepiped
- Shape-Independent
- Sphere
- Structure Factor
- All Models