Université de Montréal

This document contains the method for creating a temperature map for given chandra data.

Contents

1 Temperature Maps **2**

1.1 Intermediate Step: Binning the Data . . . 2

1.2 Generating Temperature Maps . . . 2

1.3 zpow . . . 3

1.3.1 zphabs . . . 3

1.3.2 apec . . . 3

1.3.3 χ^2 statistic . . . 4

2 The Algorithm **4**

3 Additional Python Fun **5**

3.1 Lookback Time . . . 5

1 Temperature Maps

Before we can get to the temperature map (I promise I'm getting there...), we first have to actually use the data we got from the Weighted Voronoi Tessellation Algorithm; we need the pixels binning information.

1.1 Intermediate Step: Binning the Data

As this subsection's title suggests (look up), we must first go through the process of binning the data (our fits file) based of the recommendation of our WVT algorithm.

Instead of describing this in great detail, I will simply include an algorithm outlining the process and point the reader toward the python file (*Bin_data.py*) for more in depth analysis.

Data: WVT Bins and Fits File
Result: Binned PHA files
initialization $--> $ read in WVT bin information;
for *bin* **do**
  for *pixel in bin* **do**
    Create fits file for pixel;
    Generate PI/PHA file;
  end
  Combine Pixel's PI/PHA files;
end

Algorithm 1: Binning Algorithm

Clearly the implementation is a little more complicated and uses several *CIAO* tools such as *specextract* and *dmextract*. For more information on these wonderful tools, check out the *CIAO* website:

http://cxc.harvard.edu/ciao/

1.2 Generating Temperature Maps

We can now (FINALLY) use our combined pixels to generate a temperature map assuming you have already created a binning of some sort – I would suggest my Weighted Voronoi Tesselations algorithm[1]. We will be employing *XSPEC*[2] for our modeling needs.

There are two main types of models:

- Addative: Source of emission i.e. producer of photons
- Multiplicative: Modifies spectrum i.e. acts on photons

Let's examine the major componenets of our model...

[1] https://github.com/crhea93/WVT
[2] https://heasarc.gsfc.nasa.gov/xanadu/xspec/

1.3 zpow

`zpow` is a simple photon power law taking into account redshift. This is just classic blackbody goodness...

$$A(E) = K[E(1+z)]^{-\alpha} \tag{1}$$

where α is the dimensionless photon index of the power law, z is our objects redshift, and K is a normalization factor at $1keV$ in units of $photons/keV/cm^2$. Photon indices of AGN are typically between $1.5 - 2.5$; this value needs to be determined from the literature.

Its worth noting that `zpow` is an addative model.

1.3.1 zphabs

`zphabs` is a model for hydrogen column density. Hydrogen column density is the "number of units of matter observed along a line of sight that has an area of observation"[3]. Basically, it takes into account all the stuff in between us and the object we are looking at (though it could be missing things such as a warm absorber like a galactic cluster which would require another model component – wabs in this instance). zphabs has a very simple equation:

$$M(E) = e^{-n_H * \sigma(E(1+z))} \tag{2}$$

where n_H is the equivalent column density in 10^{22} $atoms$ cm^{-2}, $\sigma(E)$ is the electron cross section – not Thompson – where z is the redshift.

All we specify are n_H and z.

`zphabs` is a multiplicative model

1.3.2 apec

`apec` is a wonderful model to handle the emission spectrum of a collisionally-drive optically-thin plasma (think AGN). Hence we use it to constrain the temperature of the object. For real details please check out the following website:

$$http://www.atomdb.org/physics.php$$

The most important thing to know is that it does in fact model thermal bremmstrahlung.

We need only specify T (KeV), as a guess, and z which is our redshift.

`Apec` is a multiplicative model

[3] https://en.wikiversity.org/wiki/Column_densities

1.3.3 χ^2 statistic

We will be using the χ^2 statistic for now (yes yes there are other and potentially better statistics to use...). The χ^2 is defined as the following:

$$\chi^2 = \sum_{i=1}^{N_{el}} \left(\frac{x_i - \mu_i}{\sigma_i} \right)^2 \tag{3}$$

Where x_i is the data value at point i, σ_i is the corresponding error, and μ_i is the model value.

By calculating the χ^2 we can get a decent guess on how well our fit is. Optimally, our χ^2 will equal the number of **degrees of freedom**. We call this ratio the **Reduced Chi-Squared Statistic**. Hence we can very easily calculate the reduced chi-square if we know the number of degrees of freedom for a model:

$$\chi^2_{red} = \frac{\chi^2}{N_{dof}} \tag{4}$$

So an optimal fit using a reduced χ^2-statistic means $\chi^2_{red} = 1$. Hence for our algorithm, we will try to hit a $\chi^2_{red} = 1$ ... or as close as we can get ...

2 The Algorithm

Data: Binning Map
Result: Temperature Map and Graphic
Step 1: Bin data using *Bin_data.py* ;
Step 2: Run XSPEC Model on binned data by "simply"[a] running *XSPEC.py* ;
Step 3: Create Visual Map in ParaView using *Cones_Paraview.py* ;
Algorithm 2: Temperature Map Pipeline

[a] Like fitting a model is simple..... But seriously this is where a mistake could be made resulting in incorrect Temperature Profiles so play around with the model on a *single* observation to ensure it is a descent fit...

Oh yes, you need to have paraview installed. If you are a linux user, its a simple command `sudo apt-get install paraview`. ParaView is an amazing tool for data rendering. Take a look at the associated website: https://www.paraview.org/.

3 Additional Python Fun

3.1 Lookback Time

```
from astropy import units as u
from astropy import cosmology
from astropy.coordinates import *
from astropy.cosmology import Planck13, default_cosmology
default_cosmology.set(Planck13)
z_RXJ = 0.658
d_RXJ = Distance(z=z_RXJ, unit=u.kpc)
```

Bibliographies

References