Variable

November 18, 2019

1 How to study a variable source

In this tutorial you will learn how to study the temporal properties of emission from a variable source and reconstruct its lightcurve. We will consider the case of the blazar PKS 2155-304 observed vt H.E.S.S.

As usual start by importing the gammalib, ctools, and cscripts Python modules.

```
In [ ]: import gammalib
        import ctools
        import cscripts
```

You may want to use the matplotlib package for plotting.

```
In [ ]: %matplotlib inline
        import matplotlib.pyplot as plt
```

1.1 Dataset and selection

For this exercise we will use a public dataset from the H.E.S.S. experiment.

H.E.S.S. data must be already available on your computer. Let's check this is the case

```
In [ ]: obs = gammalib.GObservations('$HESSDATA/obs/obs_pks.xml')
        print(obs)

        for observation in obs:
            print(observation)
```

We have 21 observations, for a total of 179k candidate photons detected by H.E.S.S. You can also see that there are two separate time intervals covered bt the observations, from MJD 53945.85 to 53946.17, and from MJD 54705.8 to 54706.9. For the study of the source variability we will concentrate on the first time interval (the source is steady over the second one). The first thing we want to do is therefore to select only observations from the first time interval. We will also select observations pointed within 4 degrees from the source. We will achieve this via the csobselect tool.

```
In [ ]: obsselect = cscripts.csobselect(obs)
        obsselect['coordsys'] = 'CEL'
        obsselect['pntselect'] = 'CIRCLE'
```

```
obsselect['ra'] = 329.71694
obsselect['dec'] = -30.22559
obsselect['rad'] = 4.
obsselect['tmin'] = 'MJD 53945.85'
obsselect['tmax'] = 'MJD 53946.17'
obsselect.run()

print(obsselect.obs())
```

We have reduced the data sample to 15 observations and 14k candidate photons. The next thing we want to do is selecting the events. In particular there are two important parameters to select on.

- Energy threshold: for a real dataset this varies depending on atmospheric conditions, night-sky background ... The H.E.S.S. collaboration provides a recommended energy threshold as part of the IRF, that we can select by setting the (hidden) parameter usethres to DEFAULT. In this case we want to skip the manual energy selection by setting emin (or emax) to INDEF.
- Offset from the center of the camera: in practise modelling the instrument response far away from the camera center is difficult, so we want to keep only events within a certain offset, that for H.E.S.S. can be be set to 2 deg.

```
In [ ]: select  = ctools.ctselect(obsselect.obs())
        select['usethres'] = 'DEFAULT'
        select['emin']     = 'INDEF' # no manual energy selection
        select['rad']      = 2.
        select['tmin']     = 'INDEF' # no time selection
        select.run()
```

We can now retrieve the information about our final dataset.

```
In [ ]: obsinfo = cscripts.csobsinfo(select.obs())
        obsinfo.run()

        print(select.obs())
        print(obsinfo.ebounds())
        print(obsinfo.gti())
```

We are left with 60k candidate photons for our analysis, covering an energy range between 0.25 and 100 TeV.

1.2 Generating a skymap

It is useful to create a skymap and inspect it. Since we have no background model provided for H.E.S.S. data you can use the ring background.

2 Generating a background model

We can use the csbkgmodel tool to generate a background model from the data. This tool makes the hypothesis that, over the entire field of view, background dominates over gamma-ray emission. Later on the background model will be refined by fitting it to the data along with a model for the gamma-ray signal. Tip: set the energy range by looking at the properties of the observations left after the selection.

```
In [ ]: bkgmodel = cscripts.csbkgmodel(select.obs())
        bkgmodel['instrument'] = 'HESS'
        bkgmodel['spatial']    = 'LOOKUP'
        bkgmodel['slufile']    = '$CTOOLS/share/models/hess_bkg_lookup.fits'
        bkgmodel['gradient']   = True
        bkgmodel['spectral']   = 'NODES'
        bkgmodel['ebinalg']    = 'LOG'
        bkgmodel['emin']       = 0.25
        bkgmodel['emax']       = 40.0
        bkgmodel['enumbins']   = 8
        bkgmodel['runwise']    = True
        bkgmodel.run()
```

Some important remarks:

- the spatial shape of the background is modelled using a lookup table that provides the background rate as a function of offset from the center of the camera and measured energy, generated from the empty-field observations made available by the H.E.S.S. collaboration;
- we multiply the azimuthally symmetric lookup model by a bi-linear gradient in camera coordinates (2 free parameters);
- the average spectrum is modelled using a piece-wise broken power law with 8 energy nodes between 250 GeV and 40 TeV (8 free parameters). The free parameters are fitted to the data for each observation (runwise = True), to get a first reasonable estimate of their values.

We shall append the background model thus generated to the selected observations.

```
In [ ]: models = gammalib.GModels()
        for model in bkgmodel.models():
            models.append(model)
        select.obs().models(models)
```

2.1 Fit over the entire time interval

To study temporal properties of the source and generate a lightcurve we need a model for the source. We will start by adding a source at the known position of PKS 2155 with a power-law spectrum, and fit this model to the data over the entire time interval to determine the average emission properties of the source.

We start by adding a model for the source with reasonable guesses for the parameters.

```
In [ ]: spectral = gammalib.GModelSpectralPlaw(7.0e-17,
                                               -3.0,
                                                gammalib.GEnergy(1.0,'TeV'))
        pos = gammalib.GSkyDir()
        pos.radec_deg(329.71694, -30.22559)
        spatial = gammalib.GModelSpatialPointSource(pos)
        spatial['RA'].free()
        spatial['DEC'].free()

        source = gammalib.GModelSky(spatial, spectral)
        source.name('PKS 2155-304')

        select.obs().models().append(source)
```

Now we can fit the model to the data.

```
In [ ]: like = ctools.ctlike(select.obs())
```

Inspect the fit results and the fit residuals.

2.2 Generating the source lightcurve

We will generate a lightcurve by using the cslightcrv tool. For the computation of the lightcurve we will fix all the source parameters except the prefactor.

```
In [ ]: like.obs().models['PKS 2155-304']['RA'].fix()
        like.obs().models['PKS 2155-304']['DEC'].fix()
        like.obs().models['PKS 2155-304']['Index'].fix()
```

As a first step we will compute the lightcurve using the 3D unbinned analysis method.

```
In [ ]: lc = cscripts.cslightcrv(like.obs())
        lc['srcname'] = 'PKS 2155-304'
        lc['tbinalg'] = 'LIN'
        lc['tmin'] = 'MJD 53945.85'
        lc['tmax'] = 'MJD 53946.17'
        lc['tbins'] = 15
        lc['method'] = '3D'
        lc['emin'] = 0.25
        lc['emax'] = 40.
        lc['enumbins'] = 0 # unbinned analysis
        lc['outfile'] = 'lc_3D.fits'
        lc.execute()
```

We have required to compute the lightcurve using 15 linearly-spaced time bins over the time interval of interest. Note that scripts like cslightcurve are using parallel compunting: by default each time bin is analyzed independently over as many threads as there are available on your machine. You can control this feature by using the nthread parameter (set to 1 to run the fits sequentially, or to the maximum number of threads you want to be used).

We can now visualize the lightcurve using the example script show_lightcurve.

```
In [ ]: import sys
        import os
        sys.path.append(os.environ['CTOOLS']+'/share/examples/python/')

        from show_lightcurve import plot_lightcurve
        plot_lightcurve('lc_3D.fits','')
```

You can appreciate that the source was highly variable during the time interval considered.

Within cslightcrv you can choose the analysis method most suitable for your case. Compute the lightcurve with the classical On/Off method, and compare the results with those from the 3D unbinned analysis

2.3 For further excercise

- Assume that you do not know that there is a variable source at the position of the blazar. Use the ctfindvar tool to "discover" the variable source.
- What is the minimum timescale over which you can assess the source's variability?
- Simulate CTA observations of this source for CTA, and compare the uncertainties in the lighctuve points for the two instruments.