Extended

November 18, 2019

1 How to analyze observations of an extended source

In this tutorial you will learn how to analyze observations of an extended source. In this case you need to not only model the spectrum, but also the morphology of the source, along with the background. We will consider the case of the pulsar wind nebula MSH 15-52.

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_msh.xml')
        print(obs)
```

We have 20 observations, for a total of 228k candidate photons detected by H.E.S.S.

The first 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(obs)
        select['usethres'] = 'DEFAULT'
        select['emin']     = 'INDEF' # no manual energy selection
        select['rad']      = 2.
        select['tmin']     = 'INDEF' # no temporal selection
        select.run()
```

1.2 Generating a skymap

First of all let's generate a skymap and inspect it. Since we don't have a good background model we will use the ring-background estimation method. Note that the ring-background method will subtract emission on characteristic scales comparable to the size of the background ring (inradius, outradius). To apply it to an extended source you need to choose the values appropriately. If you find significant gamma-ray emission on scales comparable to those you chose for the background ring you need to adapt the parameters iteratively.

```
In [ ]: skymap = ctools.ctskymap(select.obs())

        ax = plt.subplot()
```

1.3 Generating a background model

The public H.E.S.S. datasets/IRFs do not include 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.

```
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']       = 0.38
        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 380 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)
```

1.4 Performing a 3D stacked analysis

1.4.1 Preparing the stacked observations

When you are deadling with multiple observations with many potential free free parameters if you use a background model per observation (and, in case, with many photons) it may be convenient to bin the events according to reconstructed arrival direction and energy, and stack all the observations together.

We bin events using the ctbin tool.

```
In [ ]: emin = 0.4
        emax = 40.
        ebins = 40
        xref = 228.4817
        yref = -59.1358
        binsz = 0.02


        ctbin = ctools.ctbin(select.obs())
        ctbin['stack'] = True
        ctbin['ebinalg'] = 'LOG'
        ctbin['emin'] = emin
        ctbin['emax'] = emax
        ctbin['enumbins'] = ebins
        ctbin['coordsys'] = 'CEL'
        ctbin['proj'] = 'CAR'
        ctbin['xref'] = xref
        ctbin['yref'] = yref
        ctbin['nxpix'] = 200
        ctbin['nypix'] = 200
        ctbin['binsz'] = binsz
        ctbin.run()
```

In this example the selected events are binned into a counts cube centred on the approximate direction of the source. A cartesian projection aligned in celestial coordinates is used and the counts cube has 200 x 200 spatial pixels of 0.02 x 0.02 degrees in size, covering an area of 4 deg x 4 deg, and 40 logarithmically spaced energy bins, covering an energy range from 0.4 TeV to 40 TeV. Note that you need to choose a region large enough to separate background and source, and a binning in arrival direction and energy fine enough to properly sample the model for the likelihood evaluation.

Let's check the content of the event cube.

```
In [ ]: print(ctbin.cube())
```

The next step consists in pre-computing the binned response in order to speed up the fitting process.

The exposure cube is computed using the ctexpcube tool and provides the effective area multiplied by the livetime of the observation. Tip: you may want to calculate the exposure on an area and energy interval somewhat larger than the count cube ones to ensure proper convolution of the model.

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

Next, we use the ctpsfcube tool to compute the point spread function (PSF) cube for the counts cube. Tip: in this case you should reduce the spatial binning to save memory since anyway the PSF varies smoothly over the field of view.

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

We will ignore energy dispersion in the interest of time. The last step is producing a cube background model by using ctbkgcube. While for the gamma-ray response we can use a grid arbitrarily defined with respect to the count cube, the background cube needs to be computed for the very same grid. We will therefore pass the count cube in input.

```
In [ ]: ctbkgcube = ctools.ctbkgcube(select.obs())
        ctbkgcube.cntcube(ctbin.cube())
        ctbkgcube.run()
```

Finally we can assemble all the pieces into a stacked observation.

```
In [ ]: # stacked observation
        stacked_obs = ctbin.obs().copy()

        # response
        stacked_obs[0].response(ctexpcube.expcube().copy(),
                                ctpsfcube.psfcube().copy(),
                                ctbkgcube.bkgcube().copy())

        # name and instrument
        stacked_obs[0].name('MSH 15-52')
        stacked_obs[0].instrument('HESS')

        # stacked background model
        stacked_obs.models(ctbkgcube.models().copy())
```

1.4.2 Likelihood fit

Now we want to fit to the data a model for the source and determine its properties. From the skymap we can guess that the source may be extended, therefore we will use a Gaussian as spatial model (with parameter values we can guess from the skymap), plus a power law for the spectrum.

```
In [ ]: # spectral model
        spectral = gammalib.GModelSpectralPlaw(1.0e-18,
                                               -3.0,
                                                gammalib.GEnergy(1.0,'TeV'))
        # spatial model
        pos = gammalib.GSkyDir()
        pos.radec_deg(228.55, -59.17)
        spatial = gammalib.GModelSpatialRadialDisk(pos, 0.1)
        spatial['Radius'].min(0.0001)
```

```
spatial['Radius'].free()
spatial['RA'].free()
spatial['DEC'].free()

# source creation
source = gammalib.GModelSky(spatial, spectral)
source.name('MSH 15-52')

# append source to stacked observation models

stacked_obs.models().append(source)
```

Run the likelihood fit and inspect the results. Tip: it is normal that this takes a little while, spatial parameters are optimized through the computation of numerical gradients.

Check the fit residuals.

1.5 Revisiting the skymap

Now that we have a good model of the region, including the background, we can revisit the skymap. In fact, we can isolate the emission associated to the source as the residual that remains after subtracting all other models from the data. One advantage of this procedure is that you don't apply any smoothing, contrarily to the ring background estimation.

Let's make a copy of the observation container with the best fit models and drop the source model.

```
In [ ]: resmap_obs = like.obs().copy()
        resmap_obs.models().remove('MSH 15-52')
```

And now calculate a residual map and inspect it.

```
In [ ]: resmap = cscripts.csresmap(resmap_obs)
```

You should be able to see that the background is modeled satisfactorily over the field of view, and that emission associated to the source seems elongated along one characteristic direction.

1.6 On/Off spectral analysis

You can determine the spectrum of an extended source using the classical On/Off technique with reflected background. Two important remarks: * you need to define the size of the On region appropriately based on the skymap inspection, so that you include most of the emission from the source; you can use the reflected-region background as long as the On region is sufficiently small w.r.t. the field of view that you have enough suitable Off regions; * in the computation of the On/Off instrument response the PSF is folded with the effective area; you need to provide a moprhological model if you want to do this correctly for an extended source.

We will base our On/Off spectral analysis on the best-fit morphology model from the stacked analysis.

```
In [ ]: phagen  = cscripts.csphagen(like.obs())
```

Now perform the On/Off fitting, compare the results for the spectral parameters with those from the stacked analysis and inspect the residuals.

1.7 For further exercise

- From the latest iteration of the skymap it seems that the source may have an elongated shape. Try to fit it with an elliptical disk (tip: see the notebook on advanced model manipulation and fitting).
- From the spectral residuals it seems that the spectrum of the source at high energies may deviate from a simple power law. Try to fit it with a curved model (tip: see the notebook on advanced model manipulation and fitting).
- Simulate CTA observations of MSH 15-52 based on the best-fit H.E.S.S. model. Characterize the significance at which CTA can detect this source, its extension and spectral curvature.