ctools - Bug #2263

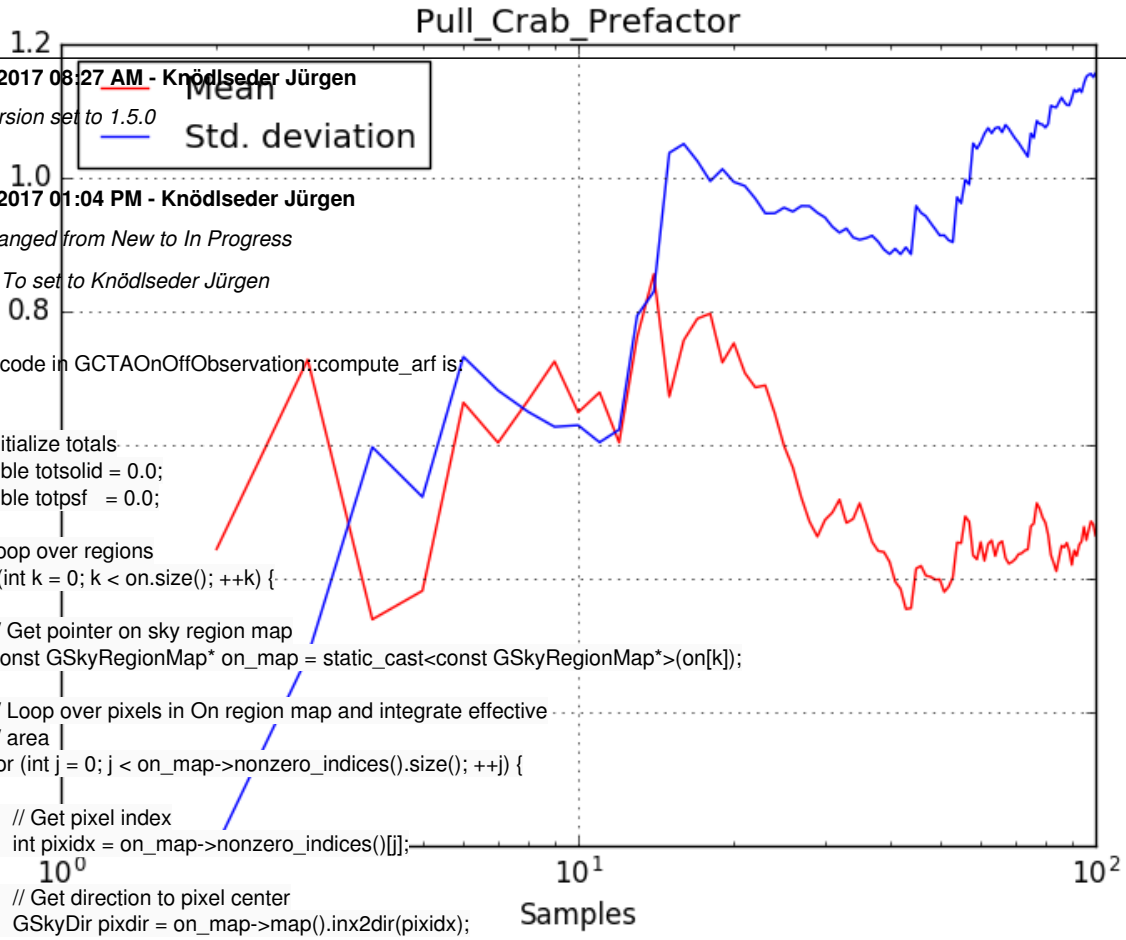
Remove small bias in pull distributions of On/Off analysis

10/27/2017 08:26 AM - Knödlseder Jürgen

Status:	Closed	**Start date:**	10/27/2017
Priority:	Normal	**Due date:**	
Assigned To:	Knödlseder Jürgen	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.5.0		

Description

There is a small bias in the pull distribution of the Prefactor towards too large values. See pull distribution evolution plot below.



History

#1 - 10/27/2017 08:27 AM - Knödlseder Jürgen

- Target version set to 1.5.0

#2 - 10/27/2017 01:04 PM - Knödlseder Jürgen

- Status changed from New to In Progress

- Assigned To set to Knödlseder Jürgen

The actual code in GCTAOnOffObservation::compute_arf is:

```
// Initialize totals
double totsolid = 0.0;
double totpsf   = 0.0;

// Loop over regions
for (int k = 0; k < on.size(); ++k) {

  // Get pointer on sky region map
  const GSkyRegionMap* on_map = static_cast<const GSkyRegionMap*>(on[k]);

  // Loop over pixels in On region map and integrate effective
  // area
  for (int j = 0; j < on_map->nonzero_indices().size(); ++j) {

    // Get pixel index
    int pixidx = on_map->nonzero_indices()[j];

    // Get direction to pixel center
    GSkyDir pixdir = on_map->map().inx2dir(pixidx);

    // Get solid angle subtended by this pixel
    double pixsolid = on_map->map().solidangle(pixidx);

    // Compute position of pixel centre in instrument coordinates
    double theta = obsdir.dist(pixdir);
    double phi   = obsdir.posang(pixdir);

    // Add up effective area
    m_arf[i] += response->aeff(theta,
                               phi,
                               zenith,
                               azimuth,
                               logEtrue) * pixsolid;

    // Add pixel solid angle to total for averaging later
    totsolid += pixsolid;

    // Compute offset angle to source
```

```
            double delta = srcdir.dist(pixdir);

            // Integrate PSF
            totpsf += response->psf(delta,
                                    theta,
                                    phi,
                                    zenith,
                                    azimuth,
                                    logEtrue) * pixsolid;

        } // endfor: looped over all pixels in region map

    } // endfor: looped over all regions

    // Average effective area over solid angle
    if (totsolid > 0.0) {
        m_arf[i] /= totsolid;
    }

    // Correct effective area by containment fraction
    if (totpsf >= 0.0 && totpsf <= 1.0) {
        m_arf[i] *= totpsf;
    }
```

which means that the effective area is averaged over the On region.

This is not accurate as the effective area should in fact be weighted by the PSF, meaning that the regions where the PSF is large count more. Here the respective code (which is in fact simpler):

```
    // Loop over regions
    for (int k = 0; k < on.size(); ++k) {

        // Get pointer on sky region map
        const GSkyRegionMap* on_map = static_cast<const GSkyRegionMap*>(on[k]);

        // Loop over pixels in On region map and integrate effective
        // area
        for (int j = 0; j < on_map->nonzero_indices().size(); ++j) {

            // Get pixel index
            int pixidx = on_map->nonzero_indices()[j];

            // Get direction to pixel center
            GSkyDir pixdir = on_map->map().inx2dir(pixidx);

            // Get solid angle subtended by this pixel
            double pixsolid = on_map->map().solidangle(pixidx);

            // Compute position of pixel centre in instrument coordinates
            double theta = obsdir.dist(pixdir);
            double phi   = obsdir.posang(pixdir);

            // Compute offset angle to source
            double delta = srcdir.dist(pixdir);

            // Get PSF value times the solid angle
            double psf = response->psf(delta,
                                       theta,
                                       phi,
                                       zenith,
                                       azimuth,
                                       logEtrue) * pixsolid;

            // Add up effective area
            m_arf[i] += response->aeff(theta,
                                       phi,
                                       zenith,
                                       azimuth,
                                       logEtrue) * psf;

        } // endfor: looped over all pixels in region map

    } // endfor: looped over all regions
```

#3 - 10/27/2017 01:37 PM - Knödlseder Jürgen

- File Prefactor_deadc1.png added

- File Index_deadc1.png added

- File Prefactor_weight_by_psf_deadc1.png added

- File Index_weight_by_psf_deadc1.png added

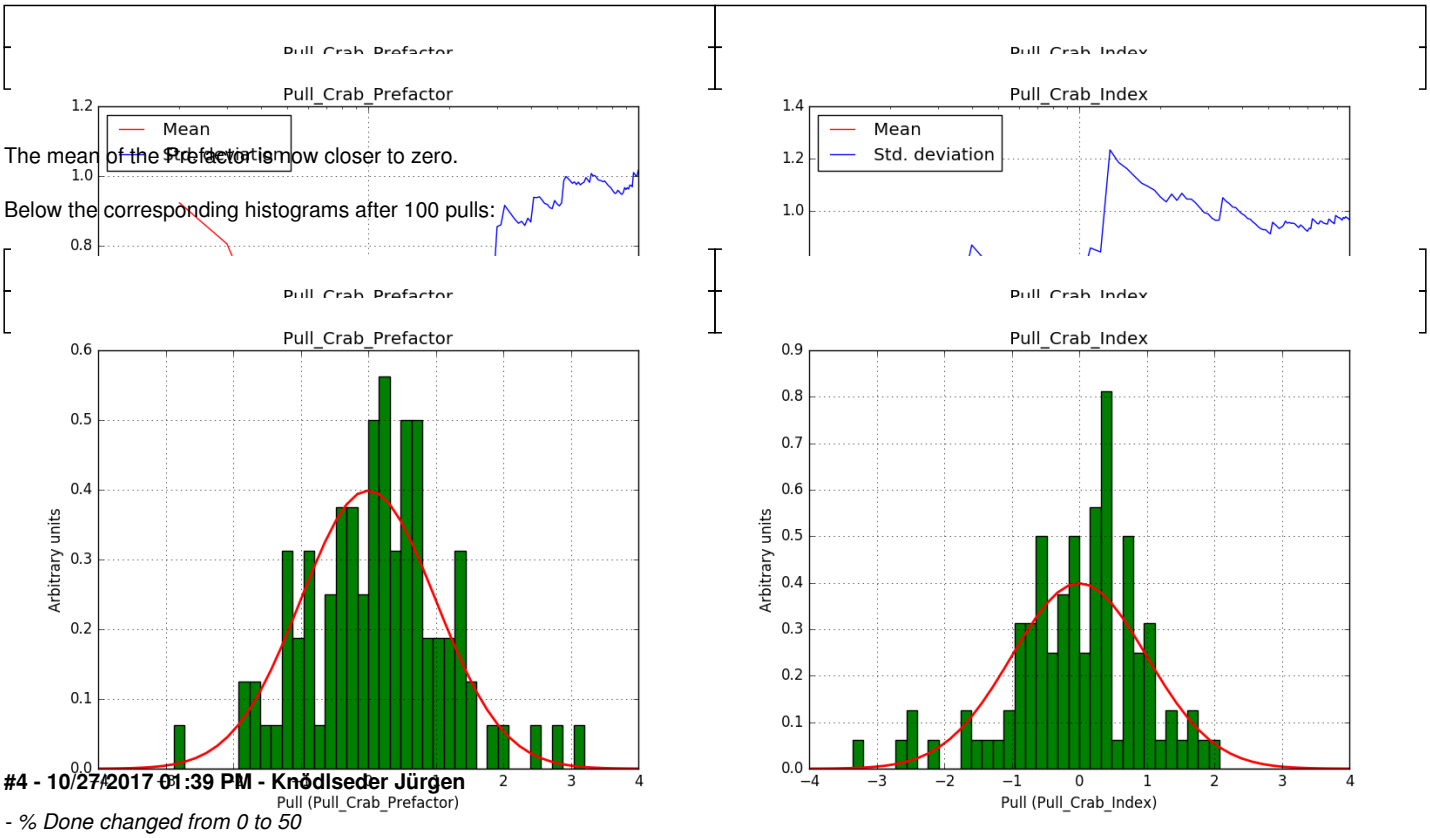
- File PrefactorHist_weight_by_psf_deadc1.png added

- File IndexHist_weight_by_psf_deadc1.png added

- File PrefactorHist_deadc1.png added

- File IndexHist_deadc1.png added

Below the comparison between the old computation (top) and the new computation (bottom). The pull distributions were generated for 10 reconstructed energy bins between 0.1 and 100 TeV, a on region radius of 0.2 deg and a deadtime correction factor of 1.0.



The mean of the Prefactor is now closer to zero.

Below the corresponding histograms after 100 pulls:

#4 - 10/27/2017 01:39 PM - Knödlseder Jürgen

- % Done changed from 0 to 50

#5 - 10/27/2017 04:51 PM - Knödlseder Jürgen

- File PrefactorHist_newarf_deadc05.png added

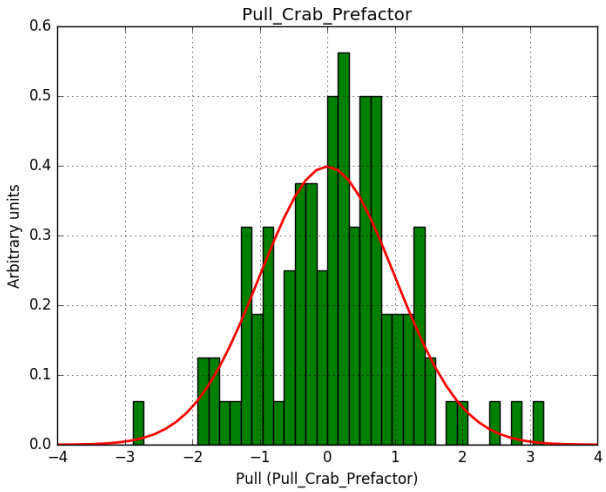
- File IndexHist_newarf_deadc05.png added

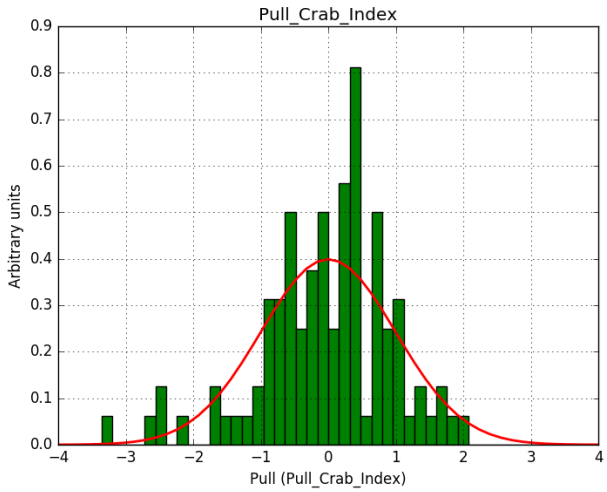
- File Prefactor_newarf_deadc05.png added

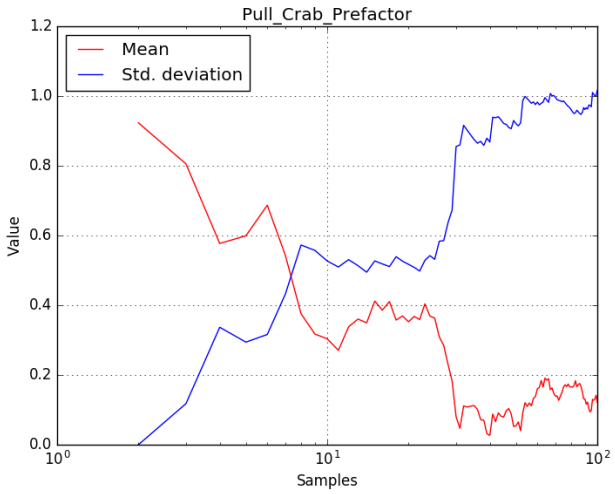
- File Index_newarf_deadc05.png added

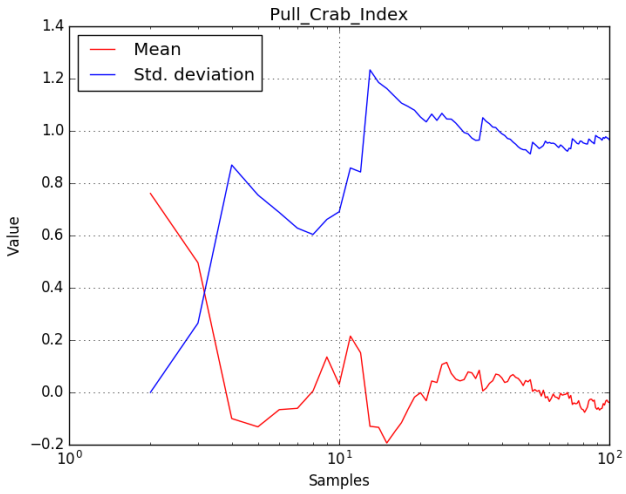
- % Done changed from 50 to 60

I checked that the value of deadc has no impact on the pull distribution by running 100 pulls for deadc=0.5. The pull histograms and pull evolutions are shown below. Everything looks ok.


Pull_Crab_Prefactor
Arbitrary units
Pull (Pull_Crab_Prefactor)


Pull_Crab_Index
Arbitrary units
Pull (Pull_Crab_Index)


Pull_Crab_Prefactor
Mean
Std. deviation
Value
Samples
1.2
1.0
0.8
0.6
0.4
0.2
0.0
10^0
10^1
10^2


Pull_Crab_Index
Mean
Std. deviation
Value
Samples
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
10^0
10^1
10^2

#6 - 10/27/2017 05:44 PM - Knödlseder Jürgen

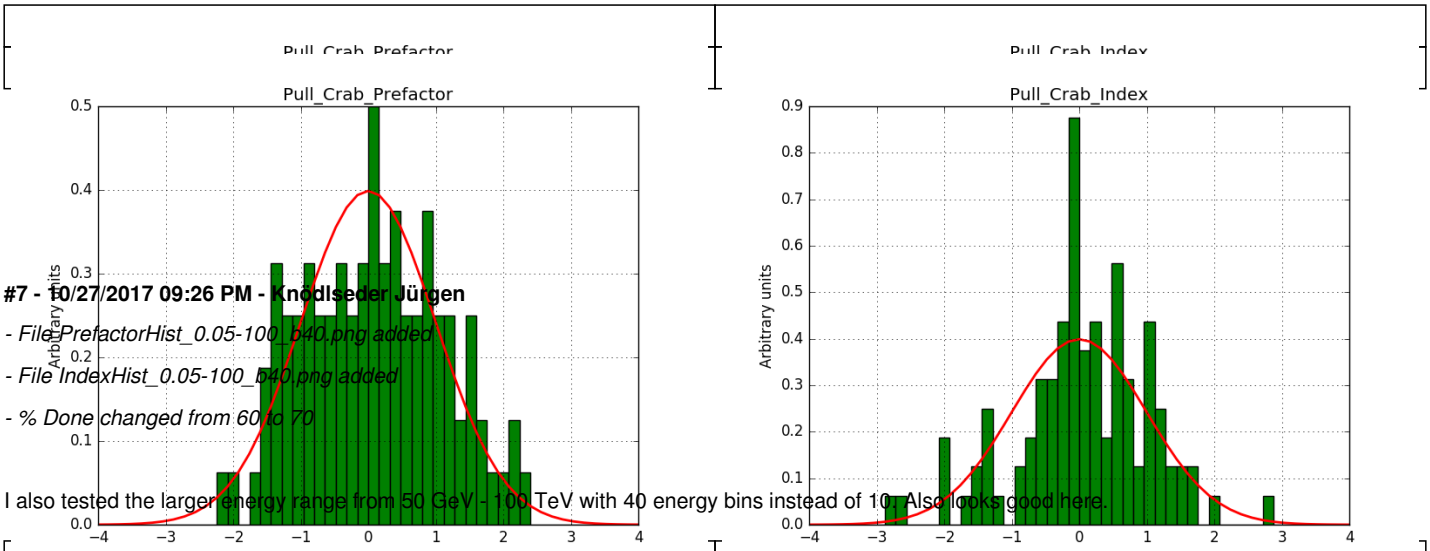
- File IndexHist_newarf_r01.png added

- File PrefactorHist_newarf_r03.png added

- File PrefactorHist_newarf_r01.png added

- File IndexHist_newarf_r03.png added

I also checked the impact of the On region radius. Before I used 0.2 deg, below results for 0.1 deg (top) and 0.3 deg (bottom). Also here everything is okay.



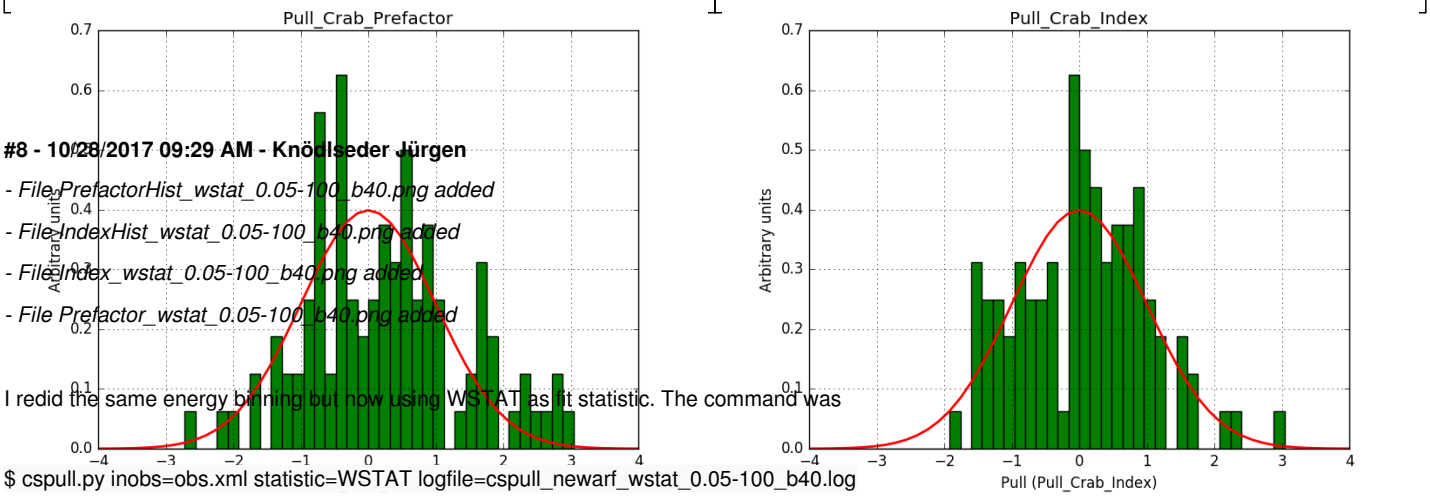
#7 - 10/27/2017 09:26 PM - Knödlseder Jürgen

- File PrefactorHist_0.05-100_b40.png added

- File IndexHist_0.05-100_b40.png added

- % Done changed from 60 to 70

I also tested the larger energy range from 50 GeV - 100 TeV with 40 energy bins instead of 10. Also looks good here.



#8 - 10/28/2017 09:29 AM - Knödlseder Jürgen

- File PrefactorHist_wstat_0.05-100_b40.png added

- File IndexHist_wstat_0.05-100_b40.png added

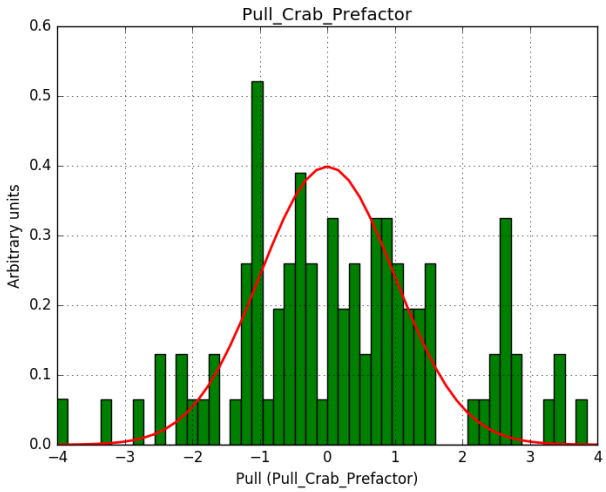
- File Index_wstat_0.05-100_b40.png added

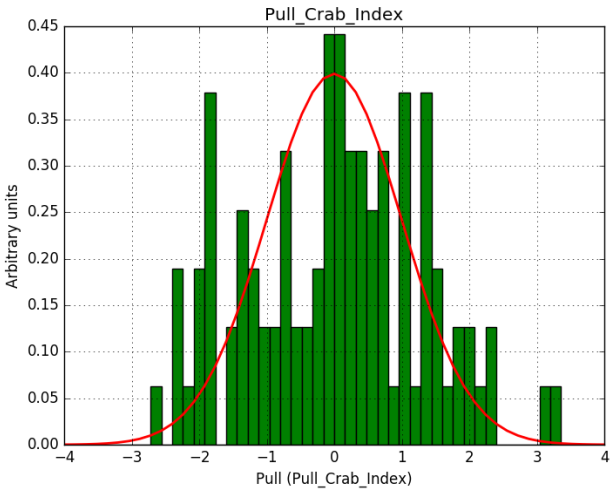
- File Prefactor_wstat_0.05-100_b40.png added

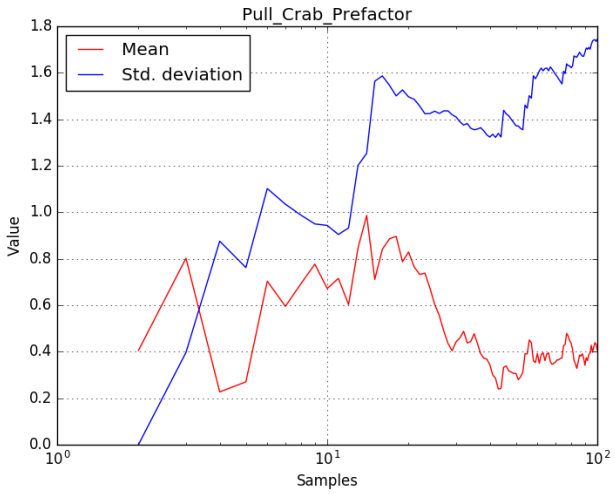
I redid the same energy binning but now using WSTAT as fit statistic. The command was

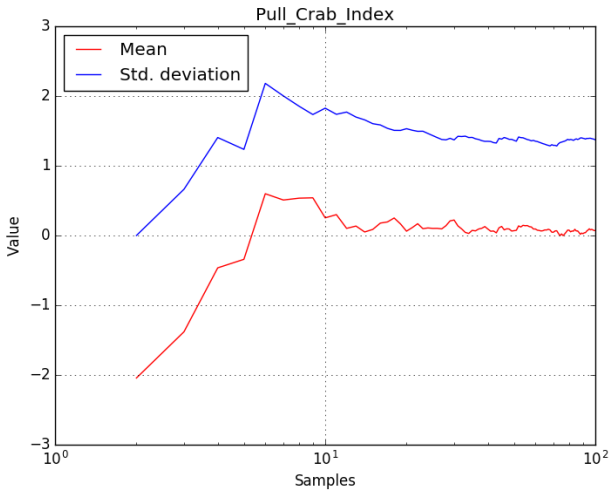
$ cspull.py inobs=obs.xml statistic=WSTAT logfile=cspull_newarf_wstat_0.05-100_b40.log

The result is shown below. Obviously, the spread of the pull distributions is too large. I'm wondering whether the issue arises tue to many bins with Non and/or Noff = 0.


Pull_Crab_Prefactor
Arbitrary units
Pull (Pull_Crab_Prefactor)


Pull_Crab_Index
Arbitrary units
Pull (Pull_Crab_Index)


Pull_Crab_Prefactor
Mean
Std. deviation
Value
Samples
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
10^0
10^1
10^2


Pull_Crab_Index
Mean
Std. deviation
Value
Samples
3
2
1
0
−1
−2
−3
10^0
10^1
10^2

#9 - 10/28/2017 10:02 AM - Knödlseder Jürgen

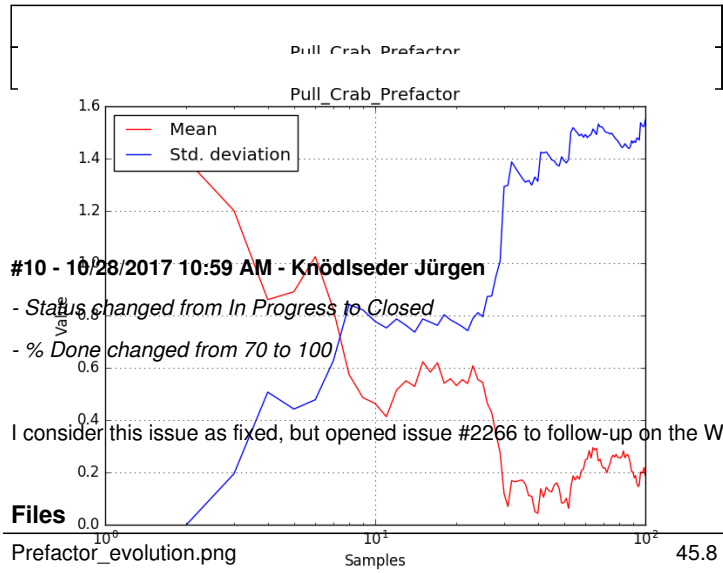
- File PrefactorHist_wstat.png added

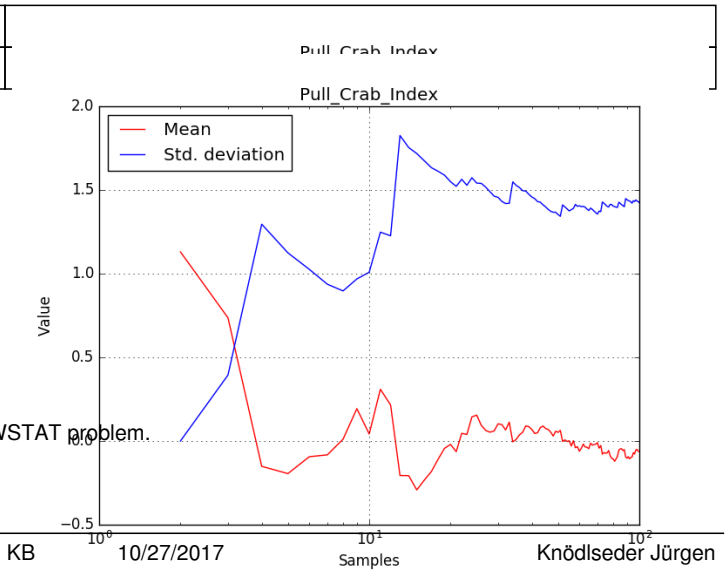
- File Index_wstat.png added

- File IndexHist_wstat.png added

- File Prefactor_wstat.png added

Here the WSTAT result with 10 energy bins between 100 GeV and 100 TeV. Results are still not as expected. Bias is small, but width is too large.





#10 - 10/28/2017 10:59 AM - Knödlseder Jürgen

- Status changed from In Progress to Closed

- % Done changed from 70 to 100

I consider this issue as fixed, but opened issue #2266 to follow-up on the WSTAT problem.

Files

Prefactor_evolution.png	45.8 KB	10/27/2017	Knödlseder Jürgen
Prefactor_deadc1.png	42.5 KB	10/27/2017	Knödlseder Jürgen
Index_deadc1.png	43.7 KB	10/27/2017	Knödlseder Jürgen
Prefactor_weight_by_psf_deadc1.png	43.3 KB	10/27/2017	Knödlseder Jürgen
Index_weight_by_psf_deadc1.png	43.9 KB	10/27/2017	Knödlseder Jürgen
PrefactorHist_weight_by_psf_deadc1.png	40.4 KB	10/27/2017	Knödlseder Jürgen
IndexHist_weight_by_psf_deadc1.png	39.4 KB	10/27/2017	Knödlseder Jürgen
PrefactorHist_deadc1.png	40.3 KB	10/27/2017	Knödlseder Jürgen
IndexHist_deadc1.png	39.4 KB	10/27/2017	Knödlseder Jürgen
PrefactorHist_newarf_deadc05.png	40.4 KB	10/27/2017	Knödlseder Jürgen
IndexHist_newarf_deadc05.png	39.4 KB	10/27/2017	Knödlseder Jürgen
Prefactor_newarf_deadc05.png	43.3 KB	10/27/2017	Knödlseder Jürgen
Index_newarf_deadc05.png	43.9 KB	10/27/2017	Knödlseder Jürgen
IndexHist_newarf_r01.png	38.7 KB	10/27/2017	Knödlseder Jürgen
PrefactorHist_newarf_r03.png	39.9 KB	10/27/2017	Knödlseder Jürgen
PrefactorHist_newarf_r01.png	40.5 KB	10/27/2017	Knödlseder Jürgen
IndexHist_newarf_r03.png	39.3 KB	10/27/2017	Knödlseder Jürgen
PrefactorHist_0.05-100_b40.png	40.6 KB	10/27/2017	Knödlseder Jürgen
IndexHist_0.05-100_b40.png	38.7 KB	10/27/2017	Knödlseder Jürgen
PrefactorHist_wstat_0.05-100_b40.png	40.9 KB	10/28/2017	Knödlseder Jürgen
IndexHist_wstat_0.05-100_b40.png	46.5 KB	10/28/2017	Knödlseder Jürgen
Index_wstat_0.05-100_b40.png	33.3 KB	10/28/2017	Knödlseder Jürgen
Prefactor_wstat_0.05-100_b40.png	48.3 KB	10/28/2017	Knödlseder Jürgen
PrefactorHist_wstat.png	40.7 KB	10/28/2017	Knödlseder Jürgen
Index_wstat.png	40.7 KB	10/28/2017	Knödlseder Jürgen
IndexHist_wstat.png	39.3 KB	10/28/2017	Knödlseder Jürgen
Prefactor_wstat.png	46.2 KB	10/28/2017	Knödlseder Jürgen