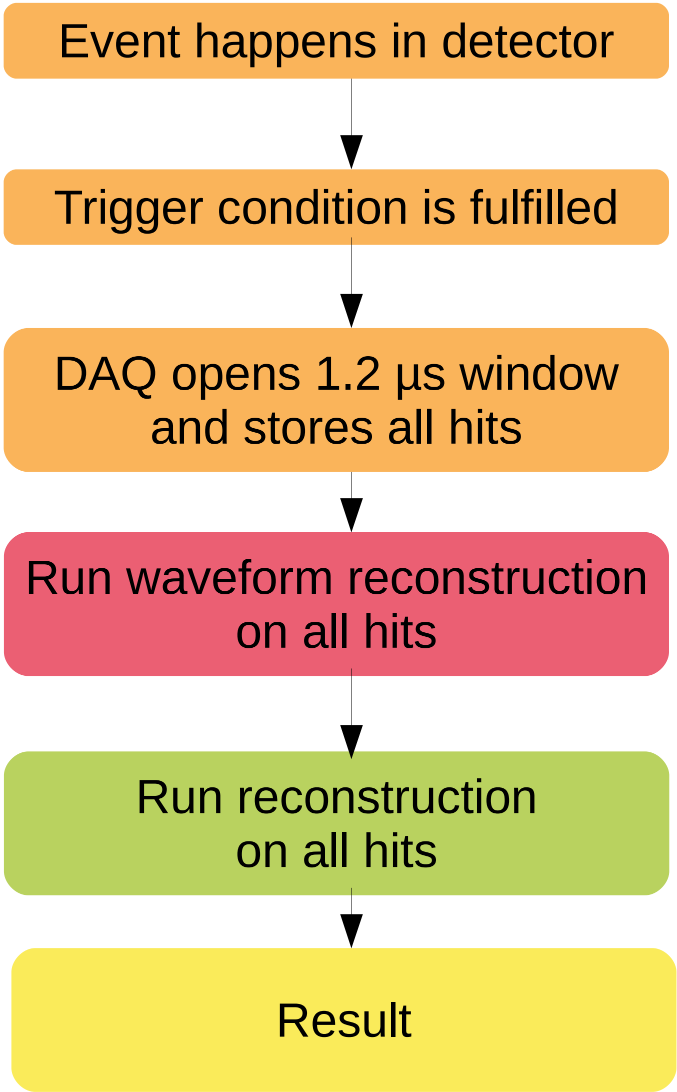
CLUSTER STRUCTURE IN OFFLINE

Software Review@JUNO Collaboration Meeting in Nanjing

23.01.2018 I PHILIPP KAMPMANN

JÜLICH
Forschungszentrum

Event processing workflow



Event happens in detector

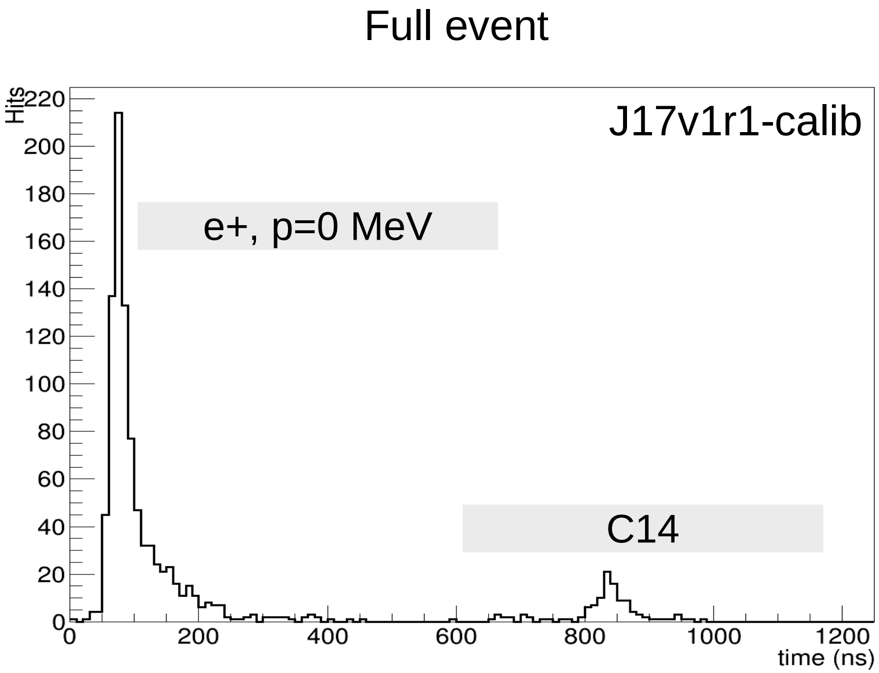
Trigger condition is fulfilled

DAQ opens 1.2 µs window and stores all hits

Run waveform reconstruction on all hits

Run reconstruction on all hits

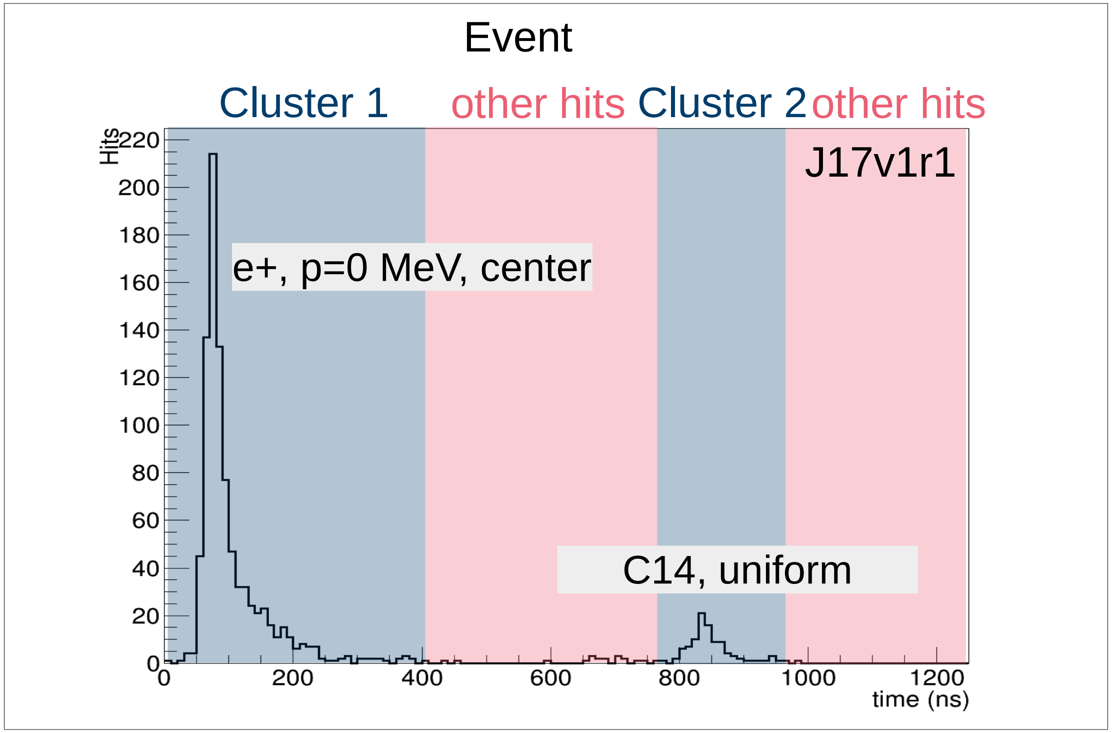
Result



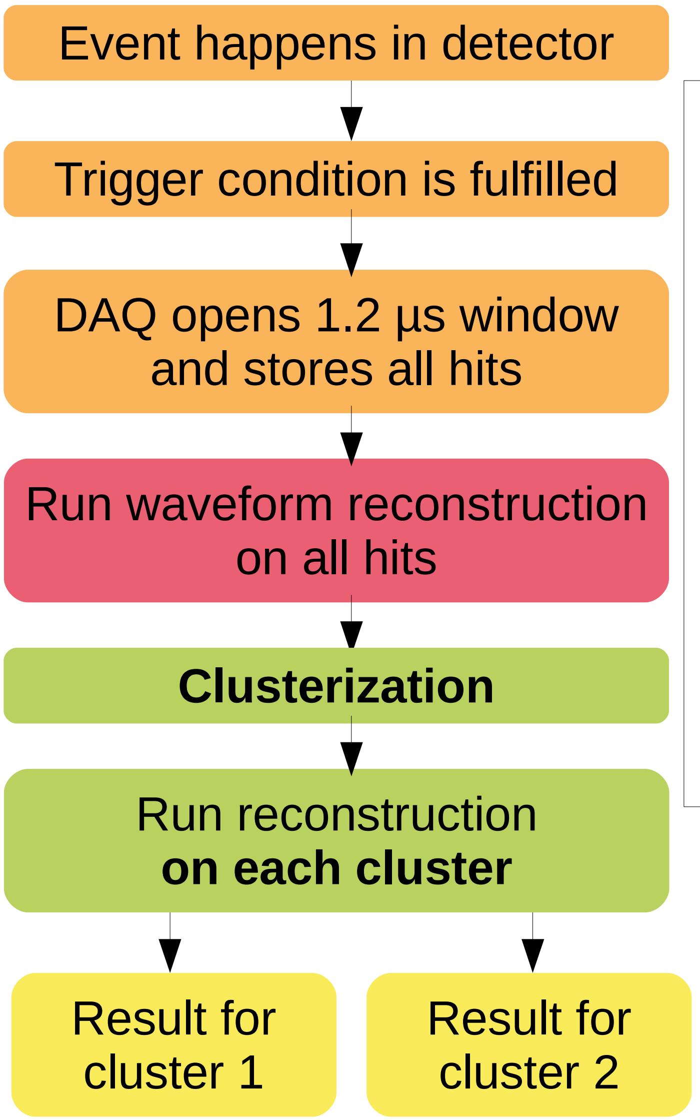
**In this case:
Two physical events in one DAQ window!
Reconstruction will give wrong results for the positron**!

Idea of having clusters

- Sub-structure of an event
- Divide DAQ window into its physical events
- Run reconstruction algorithms on clusters, not on full DAQ windows
- Release of constraints on DAQ window size from analysis point of view
- Removes Dark Noise



Event processing workflow (with clusterization)



Event happens in detector

Trigger condition is fulfilled

DAQ opens 1.2 µs window and stores all hits

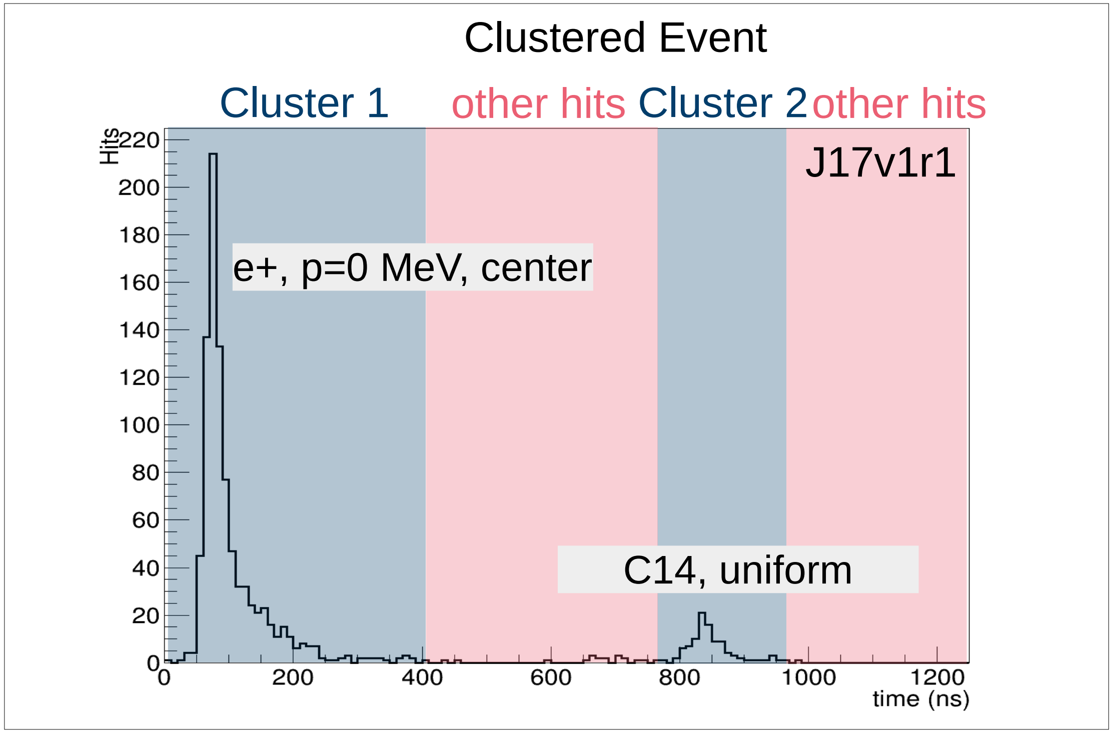
Run waveform reconstruction on all hits

Clusterization

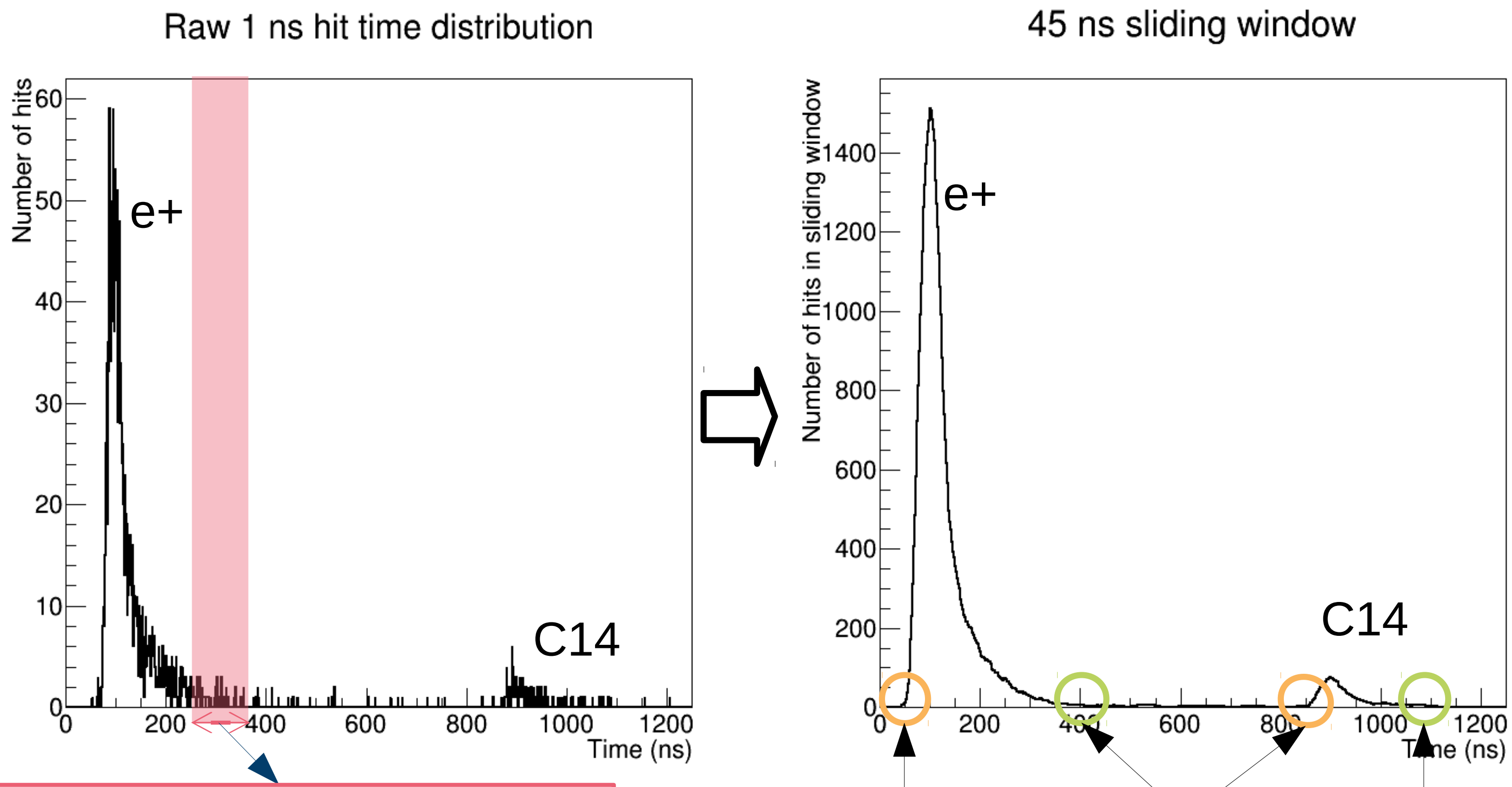
Run reconstruction **on each cluster**

Result for cluster 1

Result for cluster 2



The clusterization algorithm



Sliding window:

- Sums up consecutive bins
- Moves in 1 ns steps
- Smoothens curve

Cluster start:

- Signal increases over start threshold

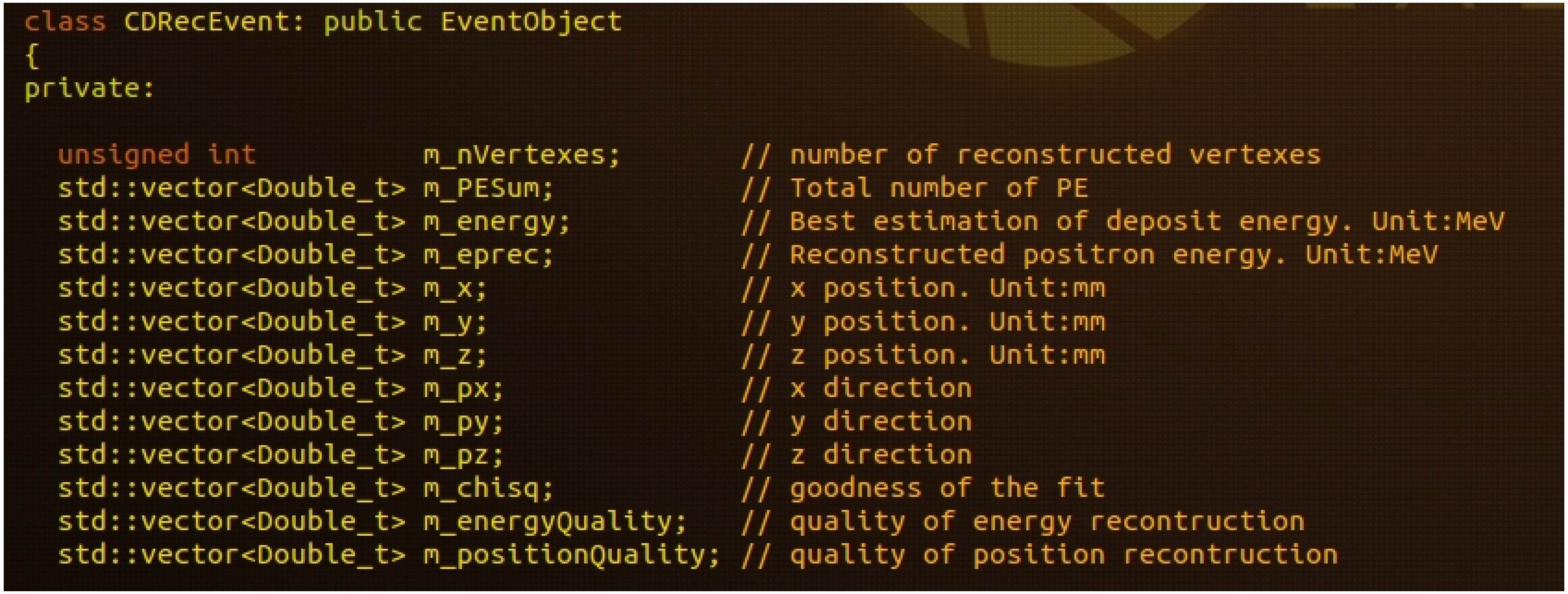
Cluster end:

- Signal falls under end threshold

Similar algorithms are used already by Qin Liu and Ding Xuefeng (and others?) for DN removal in reconstruction

How to implement it

- After waveform reconstruction
- As first step in reconstruction (at “tut_calib2rec.py - stage”)
  - Running as pre-filter for pos-reco, energy-reco
- CDRecEvent already allows the storage of multiple reconstructions
- Missing: Informations about reconstructed clusters (length, number of hits, start time, end time, etc?)

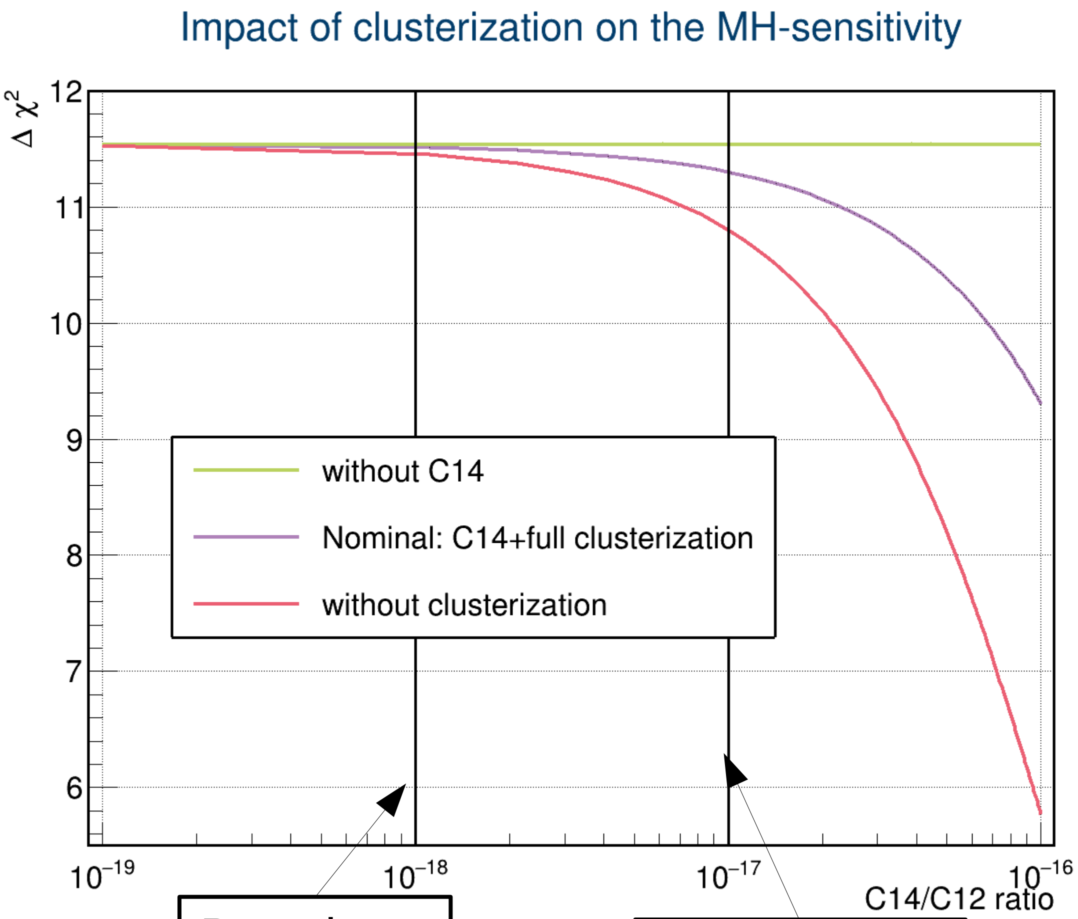


```
class CDRecEvent: public EventObject
{
private:

  unsigned int          m_nVertexes;       // number of reconstructed vertexes
  std::vector<Double_t> m_PESum;           // Total number of PE
  std::vector<Double_t> m_energy;          // Best estimation of deposit energy. Unit:MeV
  std::vector<Double_t> m_eprec;           // Reconstructed positron energy. Unit:MeV
  std::vector<Double_t> m_x;               // x position. Unit:mm
  std::vector<Double_t> m_y;               // y position. Unit:mm
  std::vector<Double_t> m_z;               // z position. Unit:mm
  std::vector<Double_t> m_px;              // x direction
  std::vector<Double_t> m_py;              // y direction
  std::vector<Double_t> m_pz;              // z direction
  std::vector<Double_t> m_chisq;           // goodness of the fit
  std::vector<Double_t> m_energyQuality;   // quality of energy recontruction
  std::vector<Double_t> m_positionQuality; // quality of position recontruction
```

BACKUP

C14 ratio scan



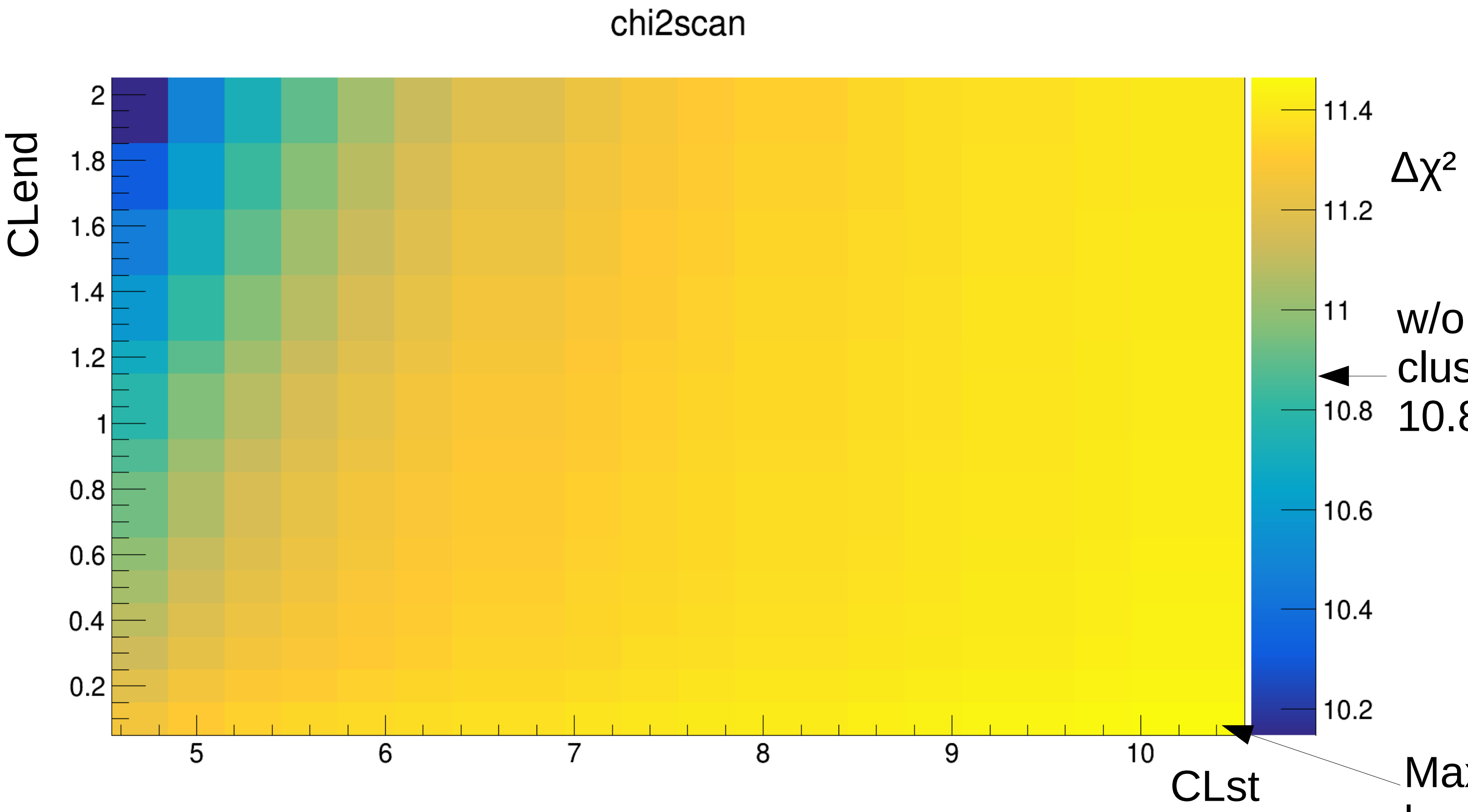
Impact of clusterization on the MH-sensitivity

Borexino level

Exp. JUNO level

- Scan in C14-concentration
- Clusterization performs well for C14-suppression

Sensitivity scan



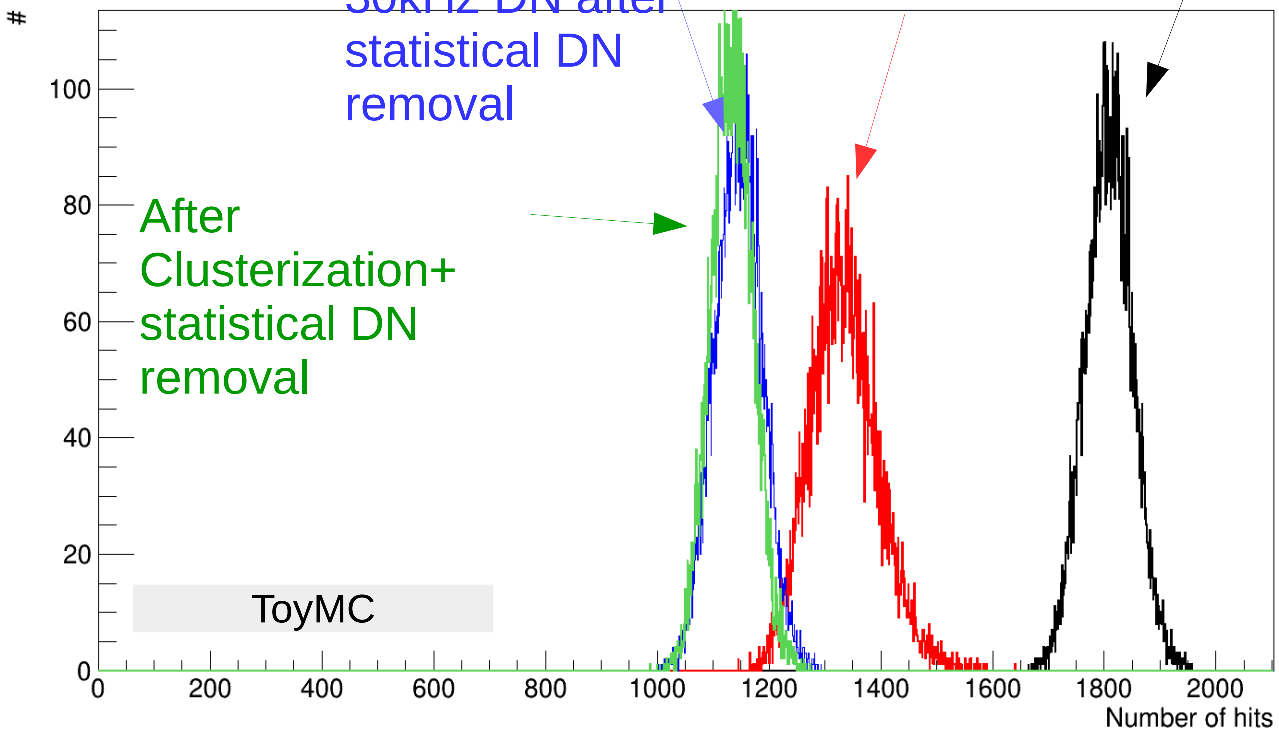
w/o clusterization: 10.86

Maximum is here!

- Implementation ready for a Delta_chi2 scan to determine the best clusterization paramters
- Scan needs to be reevaluated in a wider range

DN removal test

Example plot:



Hits of 1 MeV events with 30kHz DN after statistical DN removal

Hits of 1 MeV events with 30kHz DN after clusterization

Total hits of 1 MeV events with 30kHz DN

After Clusterization+ statistical DN removal

Total hits:	Mean = 1809,	StdDev = 42.21
After Clusterization:	Mean = 1335,	StdDev = 60.65
Statistical DN removal:	Mean = 1145,	StdDev = 42.21
Clusterization+DN removal:	Mean = 1135,	StdDev = 37.43