LCFIPlus status

Ryo Yonamine
Tohoku University

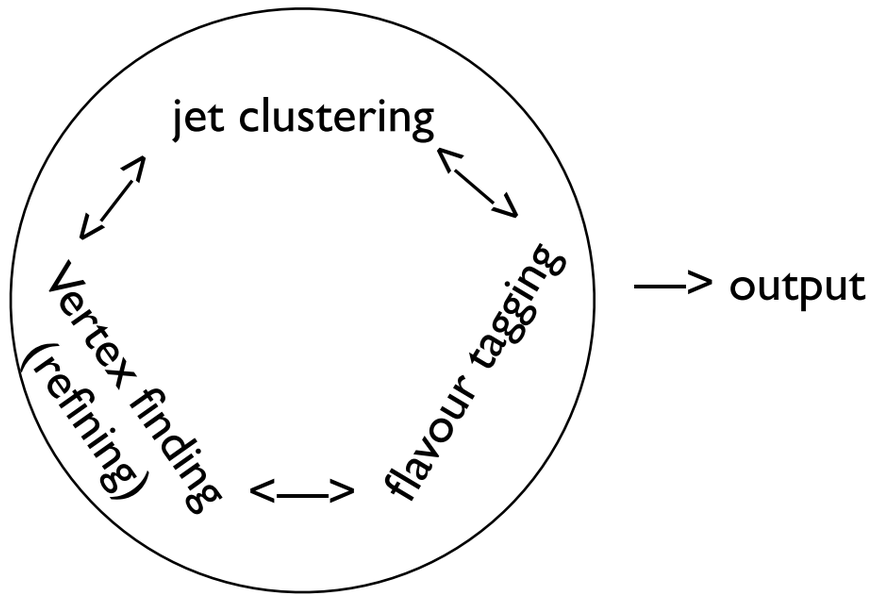
The 58th general meeting of the ILC physics working group @ KEK, 1st Sep. 2018

What's LCFIPlus?

- **A framework for jet flavour identification.**
  - does vertex finding, jet clustering, and flavour tagging
  - each process is implemented as a modular algorithm.
    - gives flexibility to iterate or reverse the processes.

w/o LCFIPlus : jet clustering —> vertex finding —> flavour tagging —> output

w/ LCFIPlus :

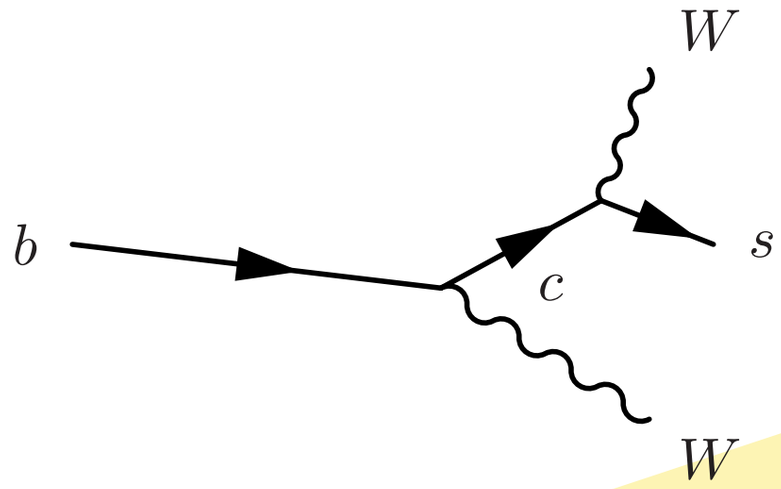


- It is flexible but typical usage is very simple;
  "vertex finding —> jet clustering —> vertex refining —>flavour tagging"

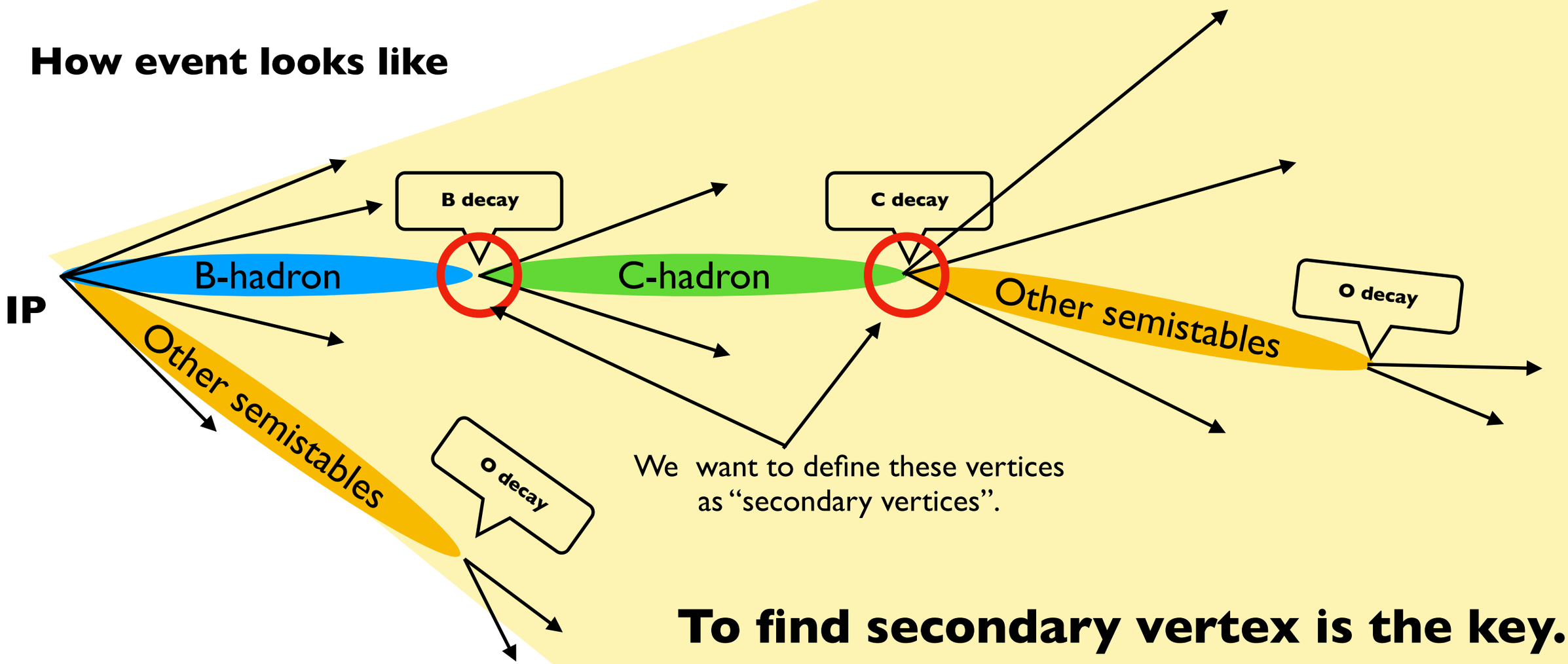
- originated from LCFIVertex (e.g. arXiv:0908.3019)

Principle of b-tag and c-tag

Feynman diagram



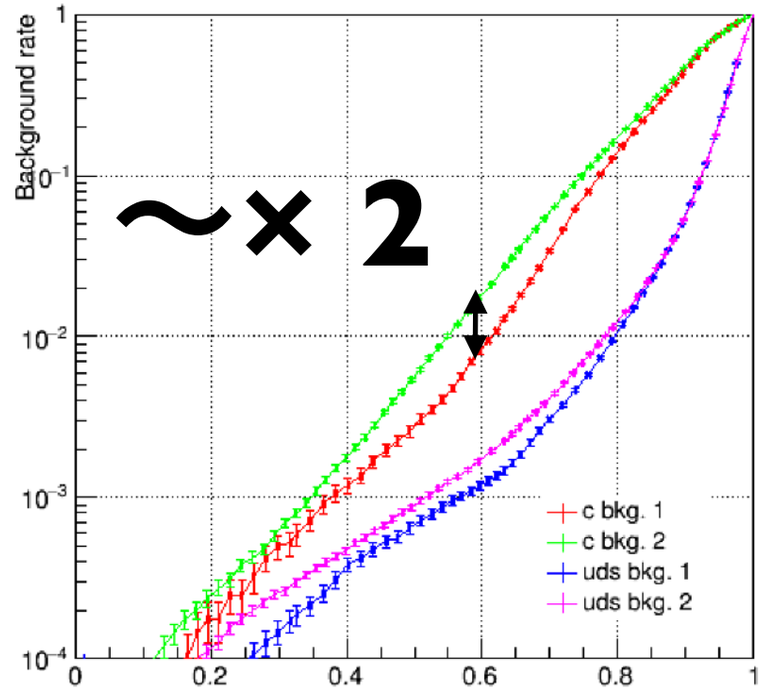
How event looks like



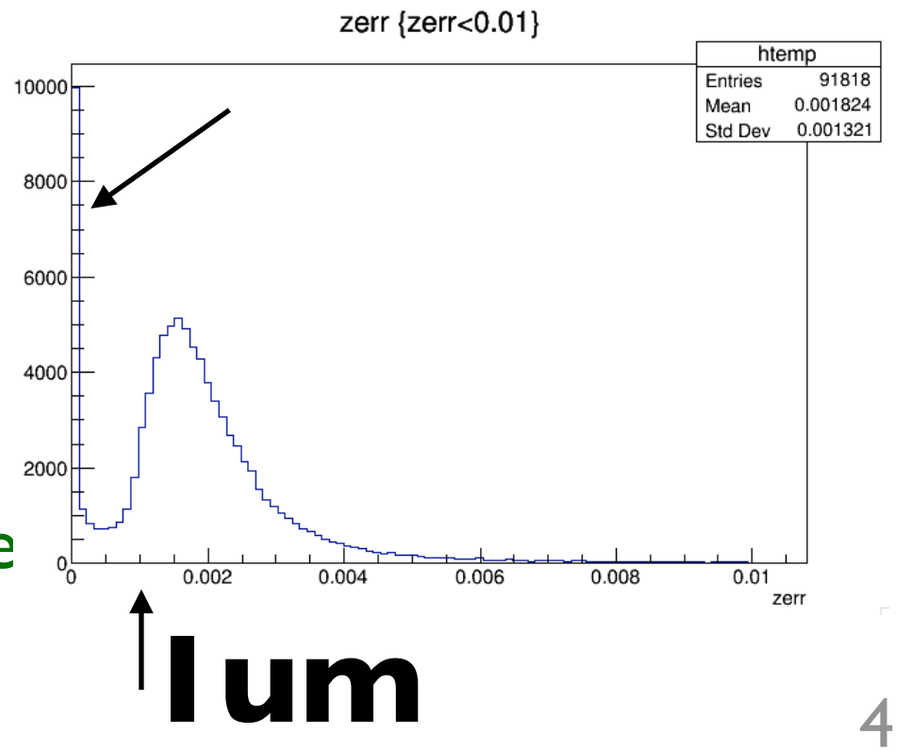
To find secondary vertex is the key.

Current concerns

- **I reported a phenomenon that larger statistics of training samples give worse performance in flavour tagging.**
  - I call it "statistical dependency problem".
  - We found that it can only be seen with samples in which the beam spot smearing exists.
  - "vtxmasspc" seems to be the main source.



- **Another issue : Too-small error on vertex position**
  - This has been pointed out by Junping.
  - Fit failure is suspected.
  - This issue seems to have been there since the DBD era.
  - One of variables used in the MVA (multivariate analysis) in flavour tagging may be affected by this error (vtxmasspc, aka pt corrected mass).



New findings

- **Unexpected behaviour found**
  - This caused a wrong error estimation.
  - This fixed our error estimation problem on 3d minimization. (1d minimization had another problem. will be discussed later.)

***Two different minimizations**

- 1) finding a position on a helix trajectory which gives the minimum distance to a target point. This is one-dimensional minimization.
- 2) finding a position that gives minimum chi2 defined as a sum of variances to helices. To compute variance for each helix, we use 1). This is 3 dimensional minimization.

Remarks for those who know my recent report :

- The change I made in the previous report : ROOT::Minuit2::MnStrategy(1) —> ROOT::Minuit2::MnStrategy(0) is turned out to be wrong direction. "0" means fa fast, "2" means accurate, "1" is in-between.
- Tolerance 10^-3 to 10^3 was needed to remove fitting error warning due to the problem above.

Code snippet (I)

Full description can be found at
https://github.com/ryonamin/LCFIPlus/blob/master/include/VertexFitterSimple.h#L20-L29

```
20        GeometryHandler* gh = GeometryHandler::Instance();
21        if (pointConstraint) {
22          Point* ip = new Point(pointConstraint);
23          vector<PointBase*> tracks;
24        if (!pointInitialOnly)
25          tracks.push_back(ip);
26        int ntracks = 0;
27        for (Iterator it = tracksBegin; it != tracksEnd; it++,ntracks++) {
28          tracks.push_back(new Helix(*it));
29        }
```

Note that “tracks” is composed of “Point” and “Helix”s.

Code snippet (2)

```
226     class PointFitFunctor {
227      public:
228       PointFitFunctor(const vector<PointBase*>& points) : _points(points) {}
229       double operator() (const double* xx) {
230         TVector3 p(xx[0], xx[1], xx[2]);
231
232         double ll = 0.;
233         for (unsigned int i=0; i<_points.size(); i++) {
234           ll += _points[i]->LogLikelihood(p);
235         }
236
237         return -ll;
238       }
239      private:
240       const vector<PointBase*> _points;
241     };
```

Note that Elements of “tracks” calls Loglikelihood(…) in fit procedure.

PointBase::Loglikelihood method is overridden in its inherited class : Point, Helix, etc. .

The problem and the (temporal?) measure

- **Problem**
  - The code looks ok to me, but I found that even for the constraint point (Point object) it calls Helix::Loglikelihood, and give apparently too-large, random numbers ! (It should return a fixed value we set in our steering file.)
- **First aid**
  - Modified so that a proper function (overridden in its inherited class) is explicitly called after clarifying the object class using "dynamic_cast".
  - Dump results look ok.
- **Still fitting failures seen…**
  - But it turned out that it comes from 1d fitting.
  - Found that it can be solved by optimizing parameters for the fit.

https://github.com/ryonamin/LCFIPlus/blob/master/src/geometry.cc#L318-L323

```
318     // TODO: primary vertex finder sometimes fails
319     double tllimit = -1. / r;
320     double tulimit = 1000. / r;
321
322     // todo: t<0 treatment
323     min.SetLimitedVariable(0,"t",t, 1e-6,tllimit, tulimit);
```

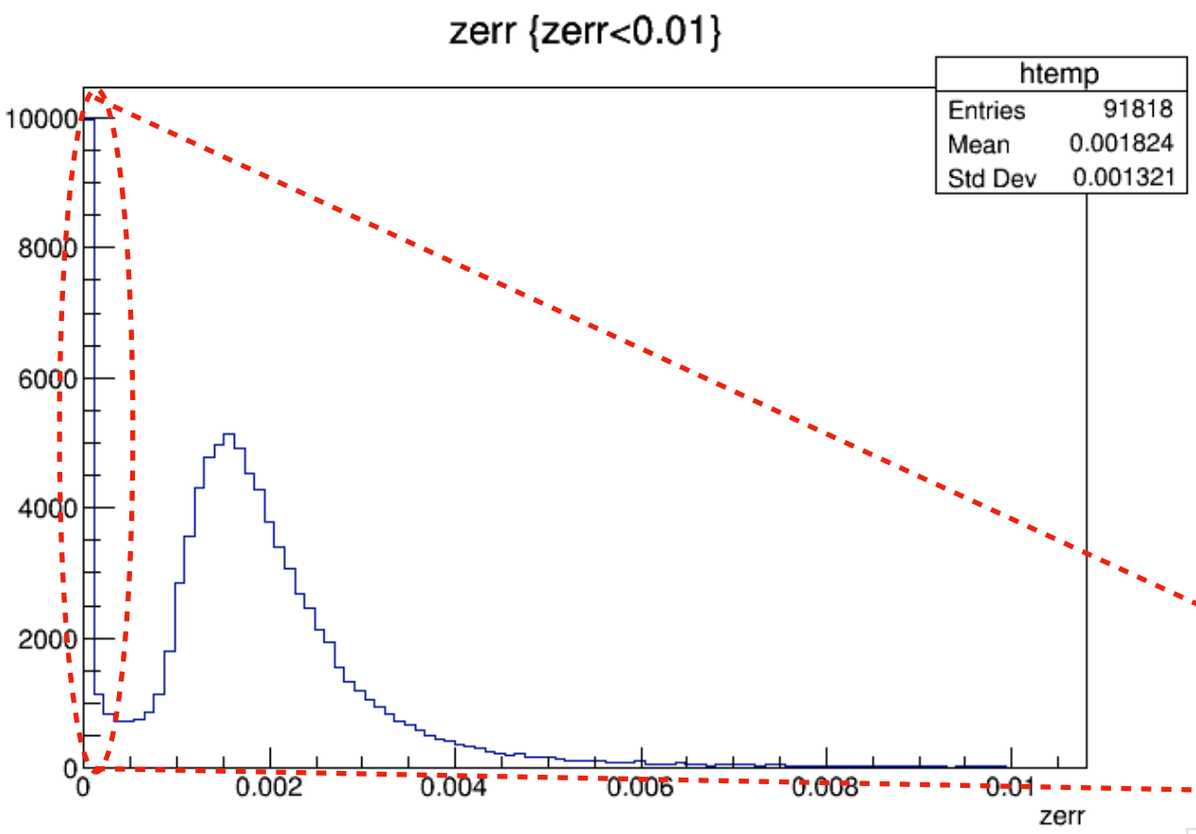
t - π

t + π

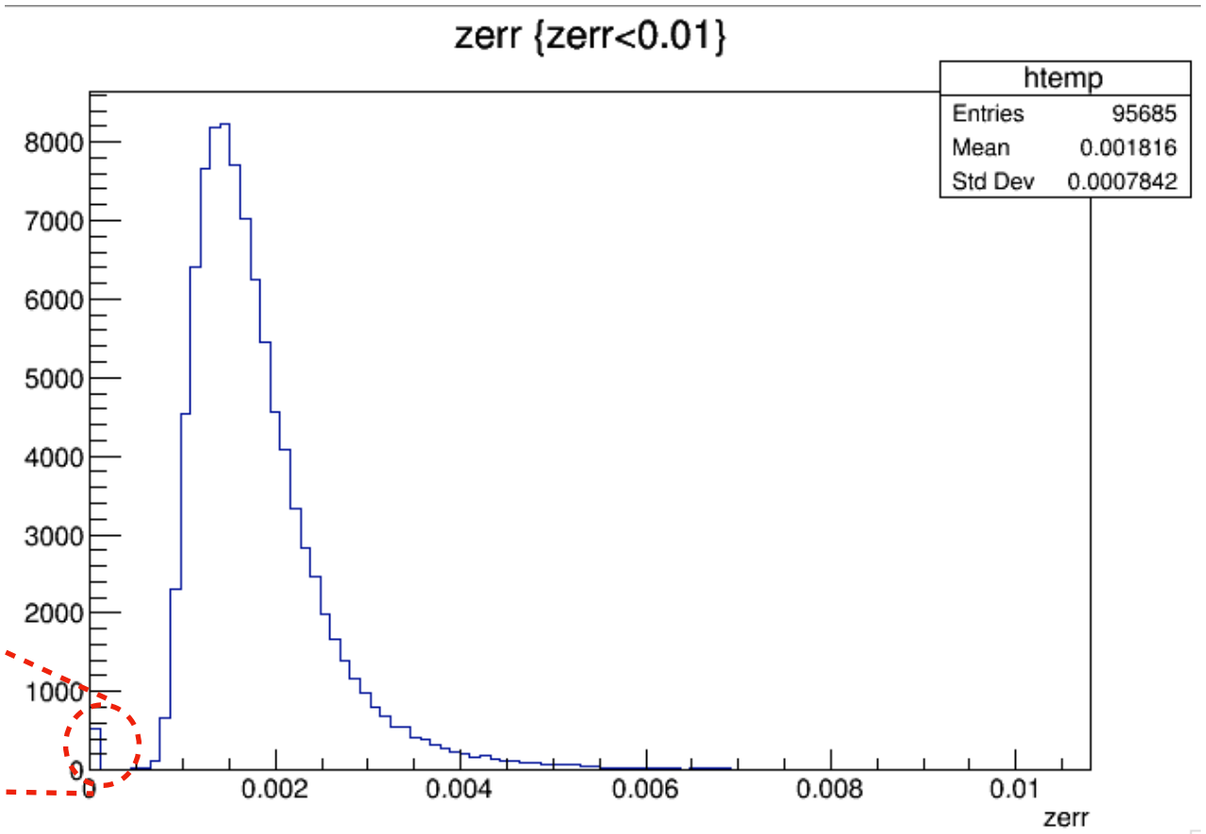
1e-4*(tulimit-tllimit)

Some preliminary plots

Error distribution on primary vertex z-position

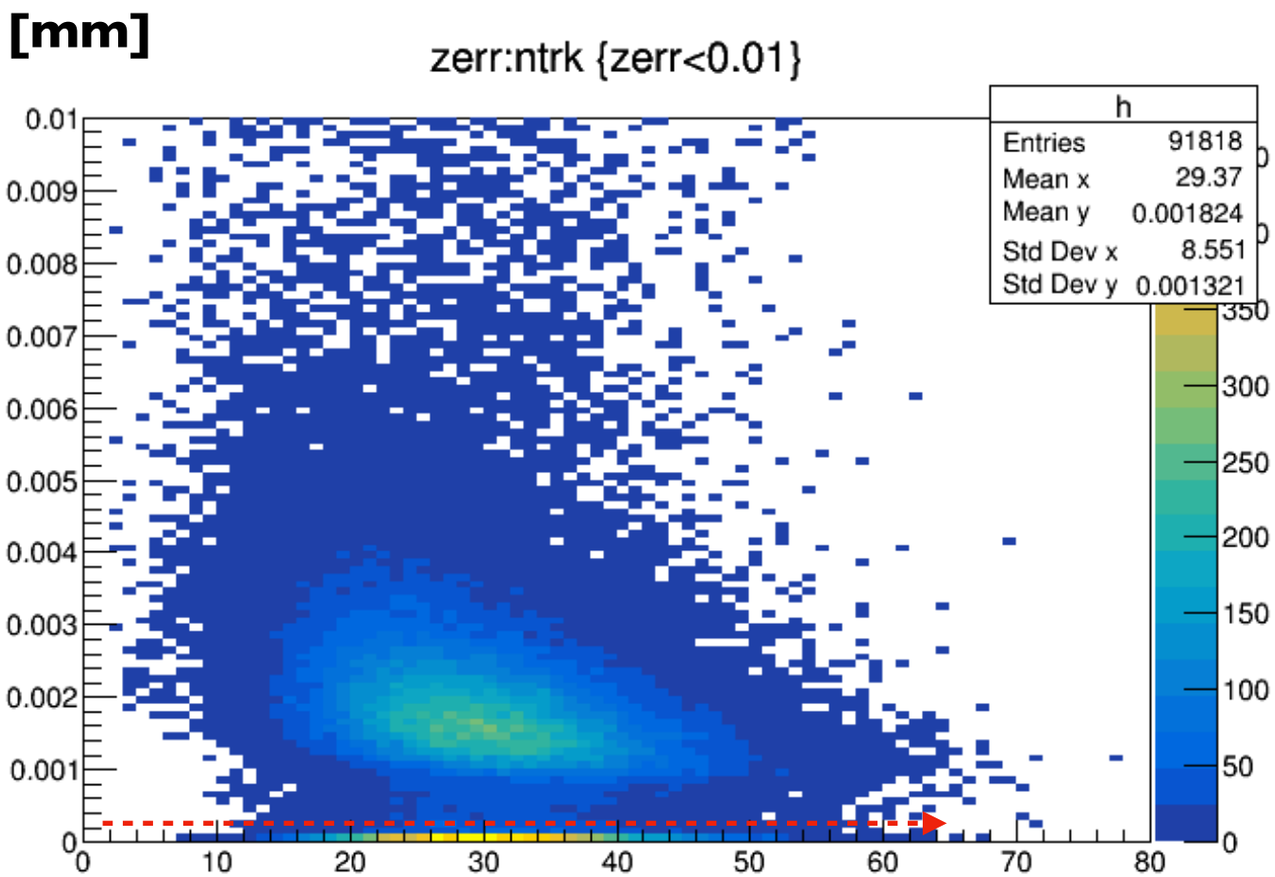


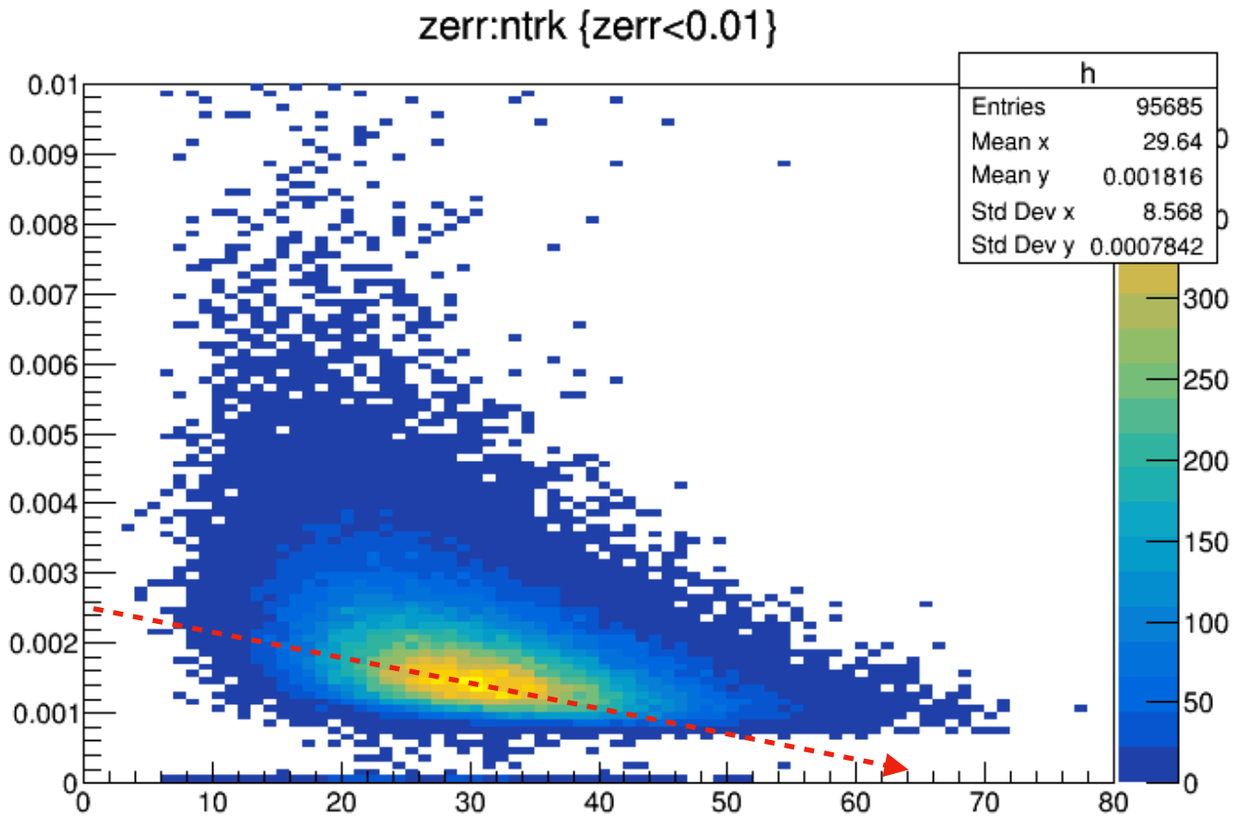
default



new

Vtx z-pos error vs # of vtx tracks





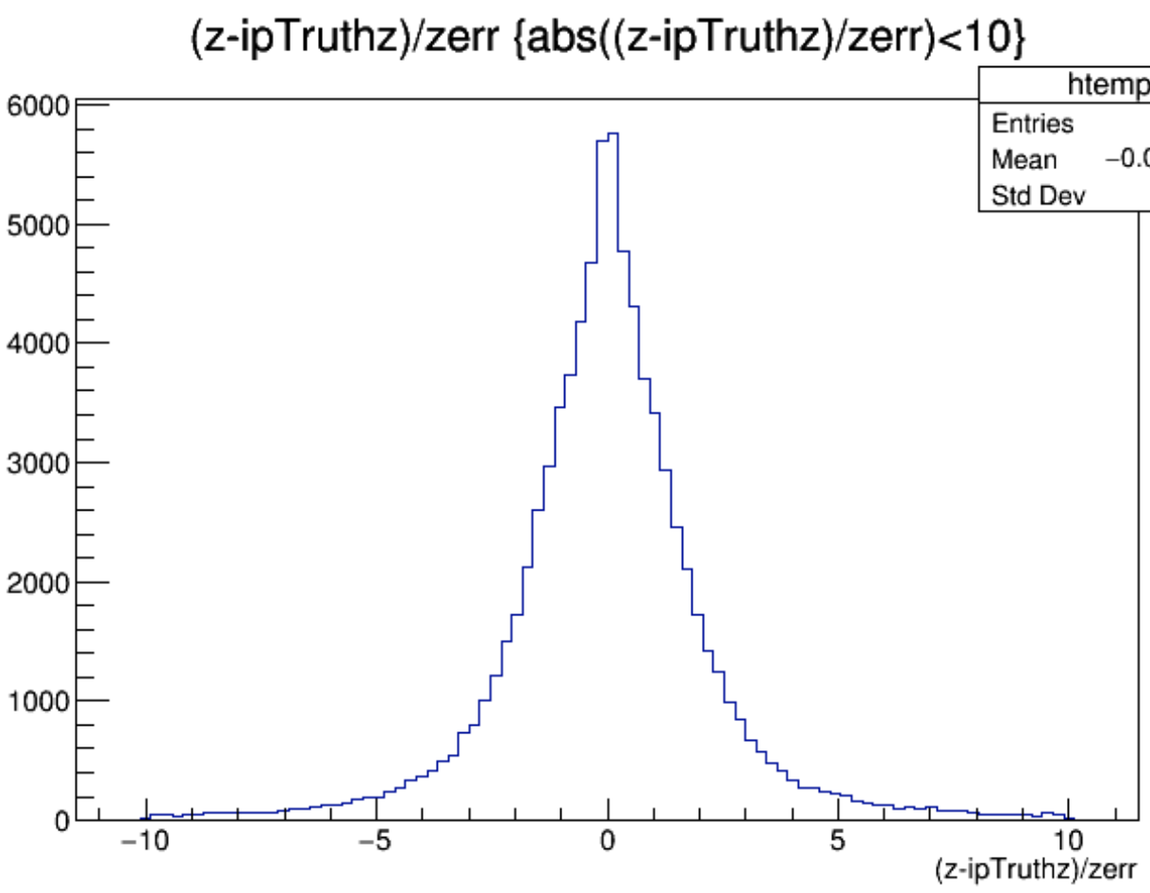
default

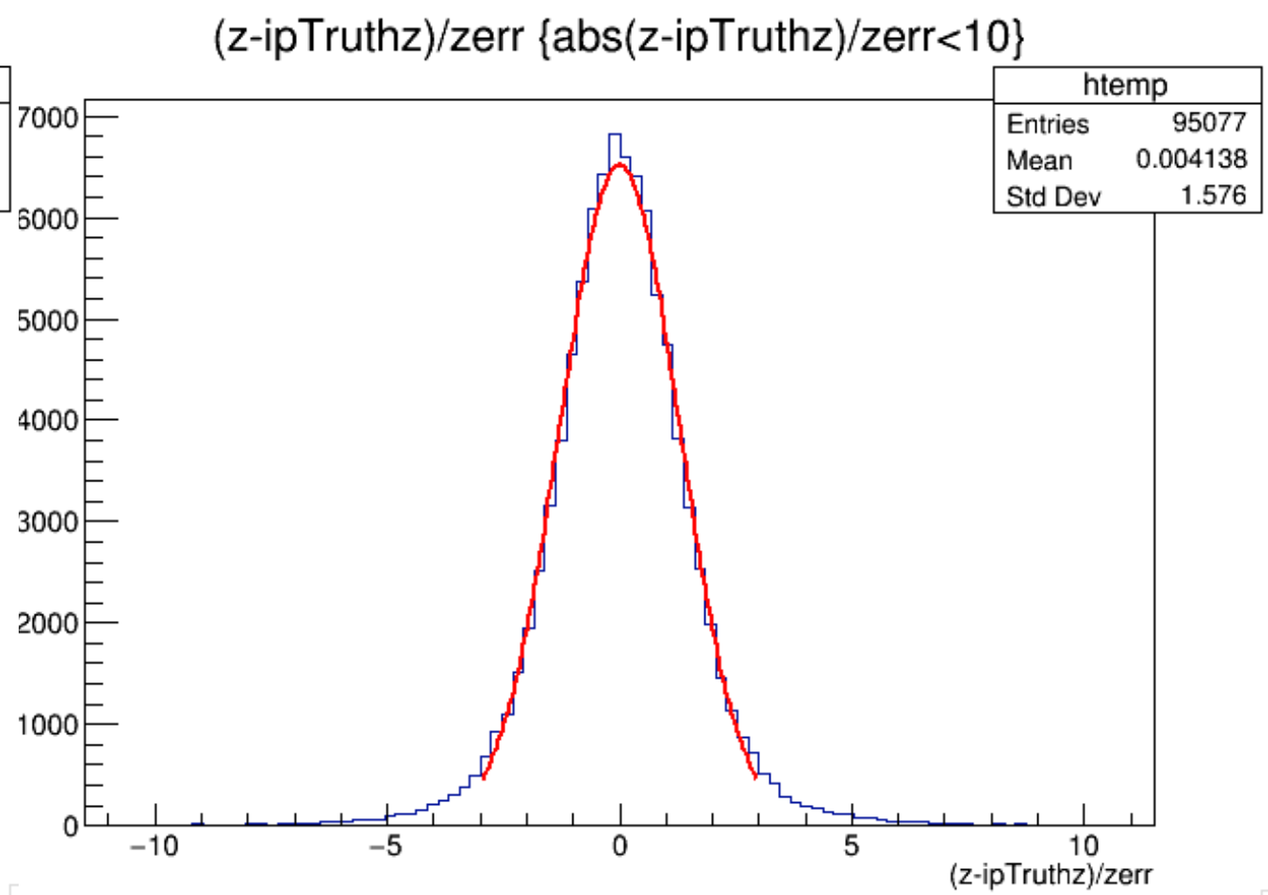
(No #-of-track dependency)

new

(sqrt{#-of-track} dependency?)

pull distribution of primary vertex z-position





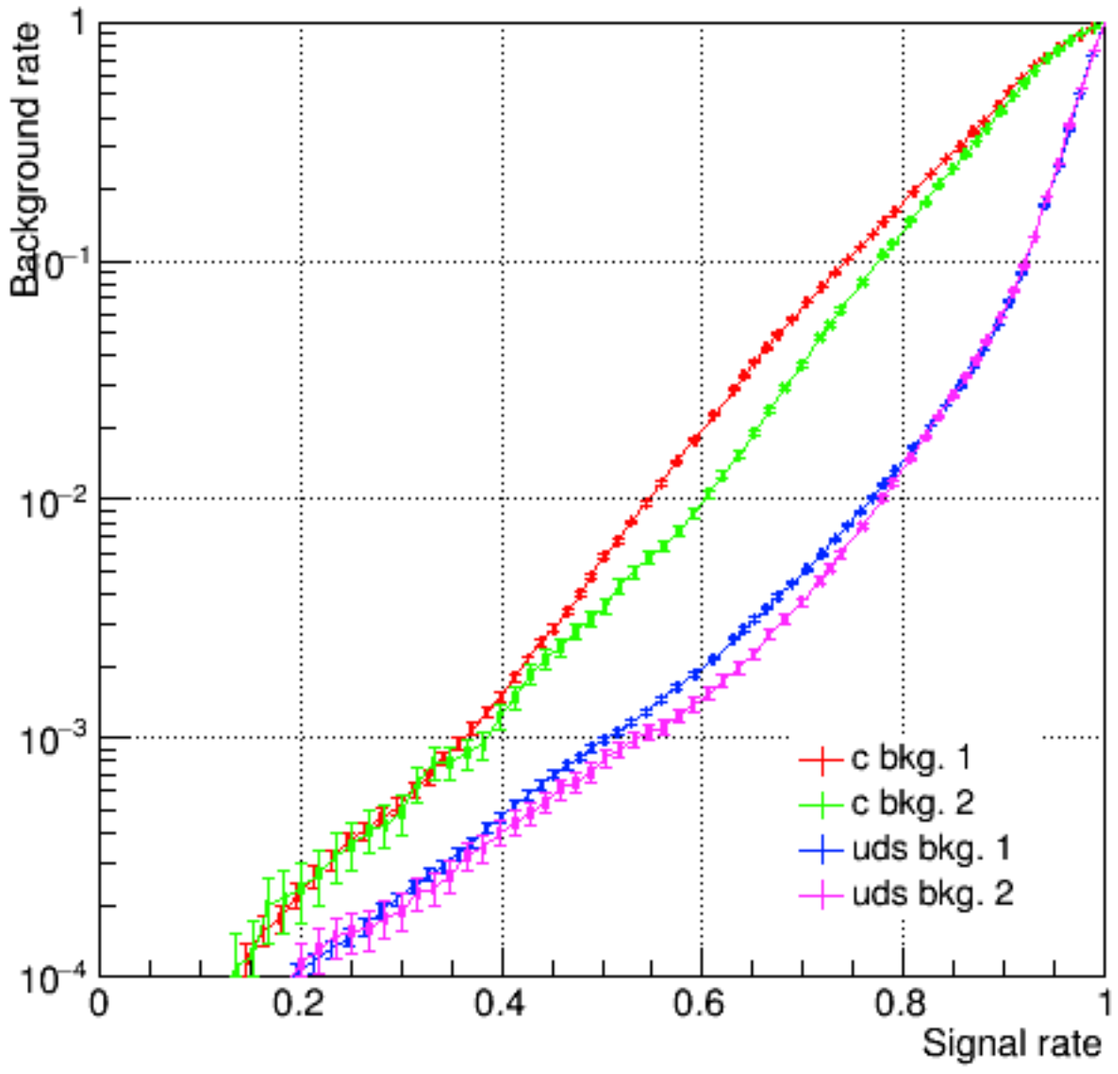
default

new

Fit : sigma = 1.29+-0.01

Still a bit broad?? but looks better than the one by default

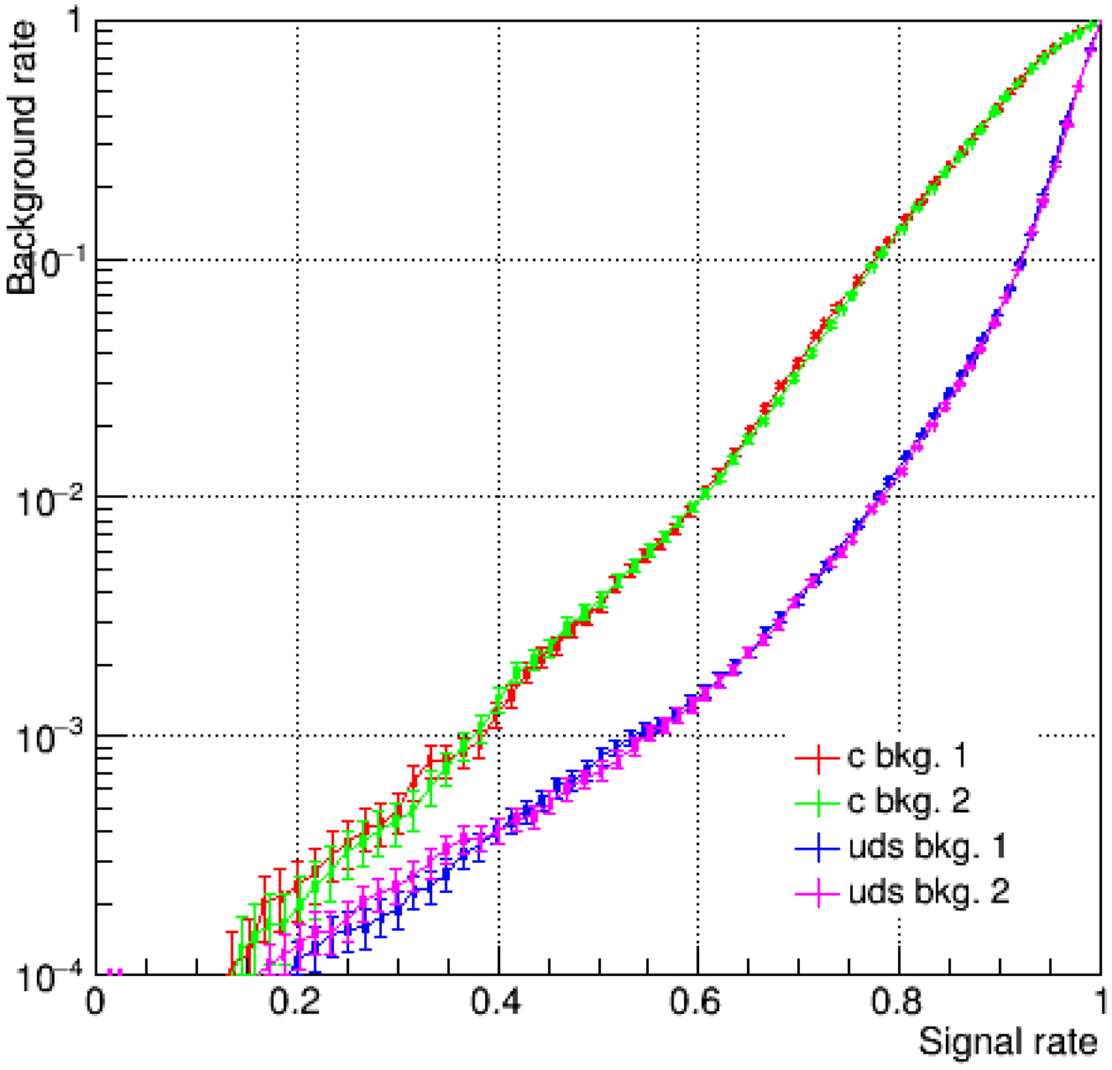
Flavour tagging performance new 20k, 100k



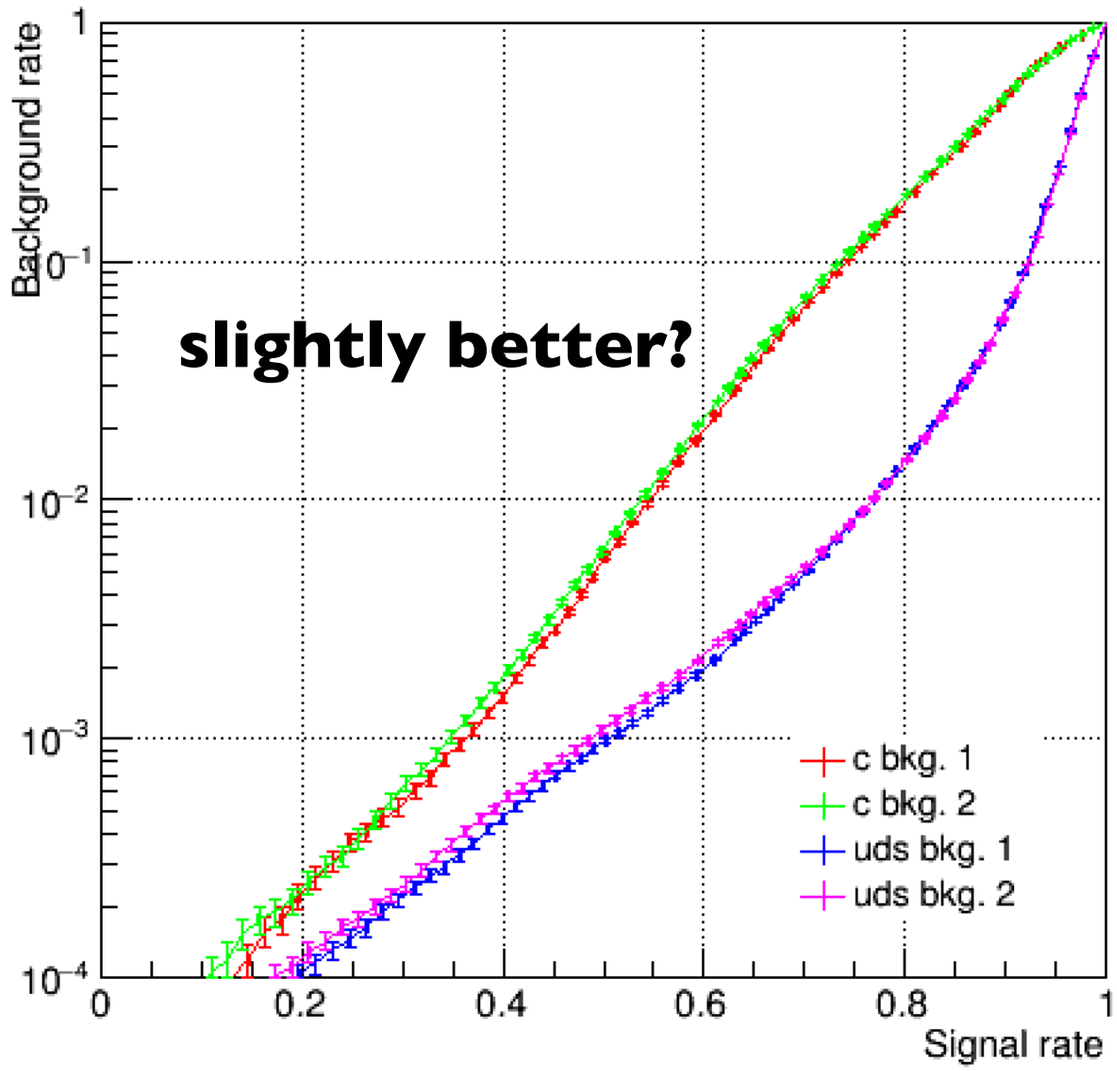
Still “statistical dependency” can be seen.

Flavour tagging performance

No significant difference in flavour tagging performance.



1. new 20k
2. default 20k

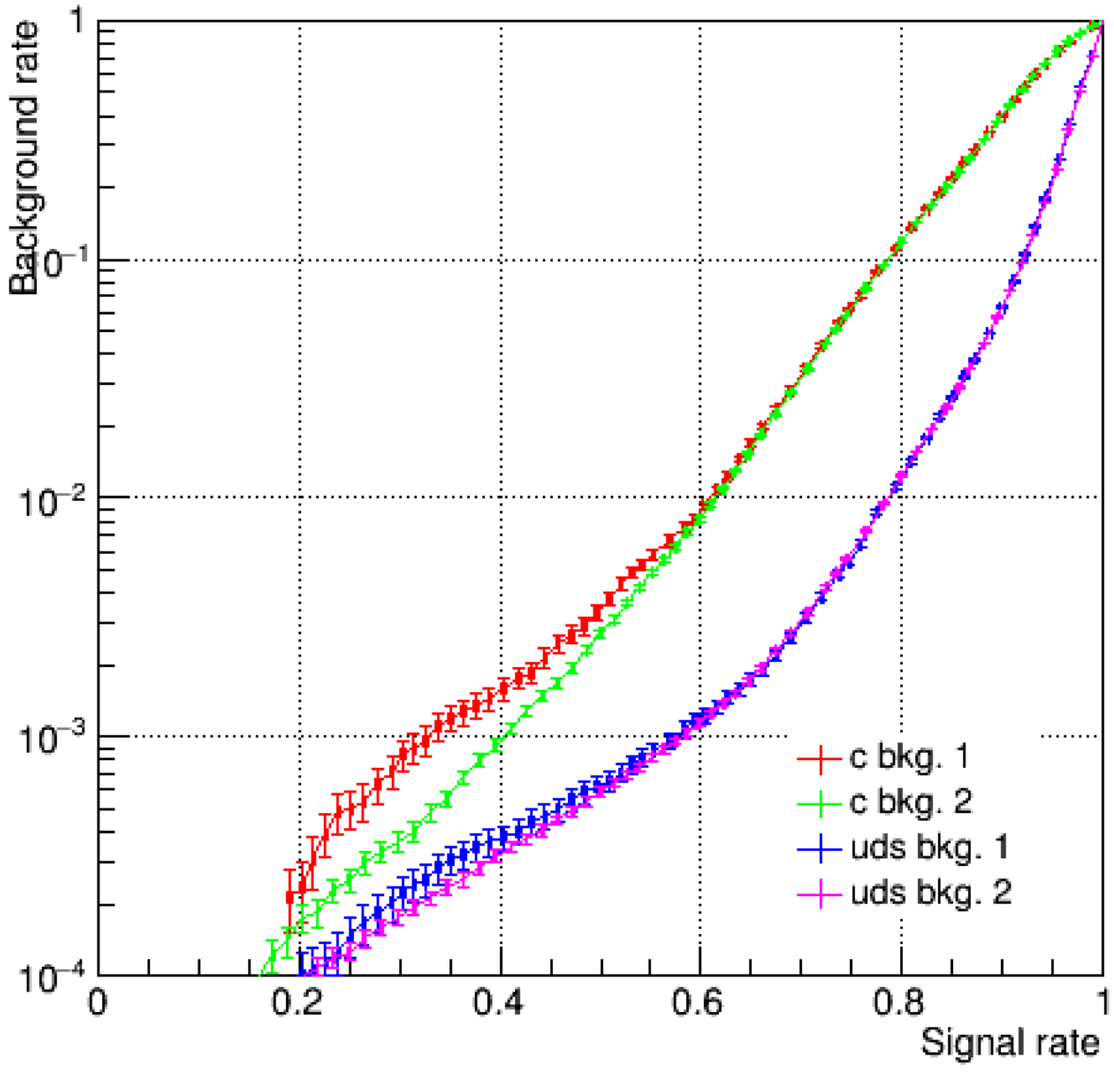


1. new 100k
2. default 100k

Flavour tagging performance

As Kurata-san reported, a BDT parameter (depth) improves the performance at 100k training.

https://agenda.linearcollider.org/event/8016/contributions/42046/attachments/33452/51231/LCFIPlus_check6.pdf

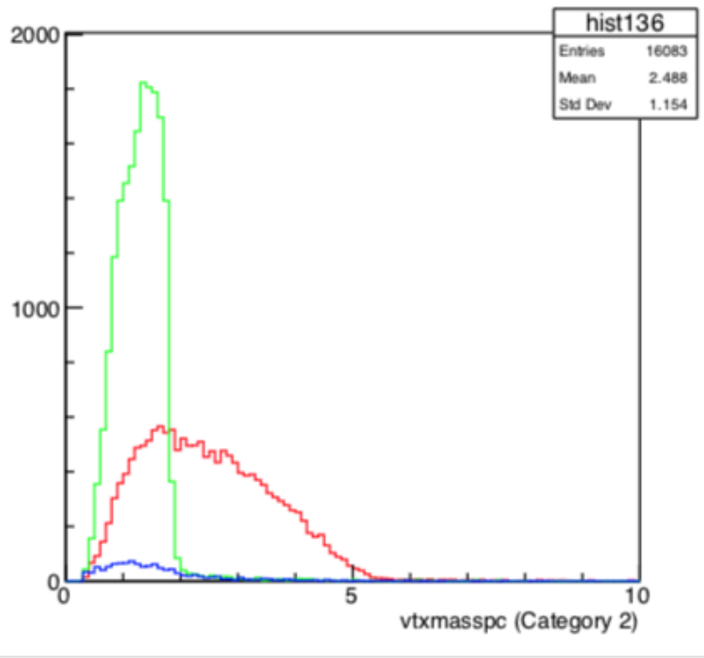


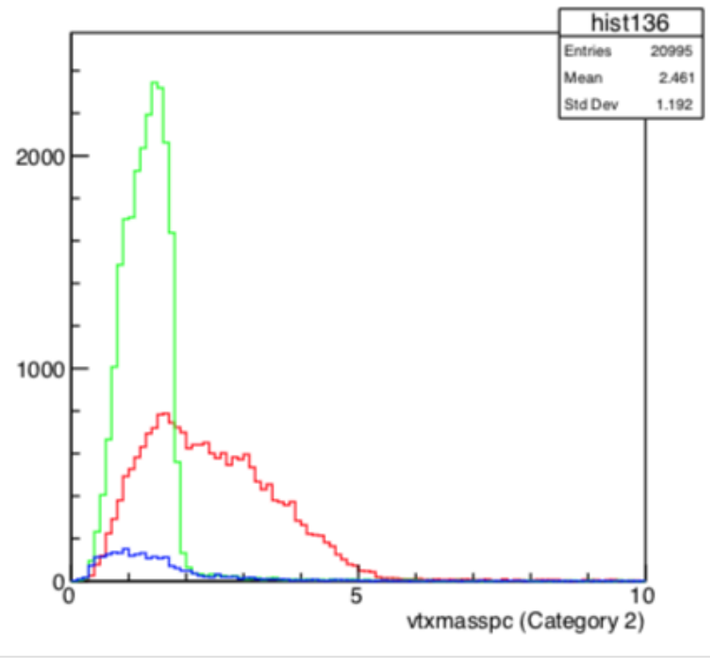
1. **new+depth6 20k**
2. **new+depth6 100k**

“Statistical dependency” problem remains

vtxmasspc distribution (“category2”; #svtx=1 in a jet)

~20k





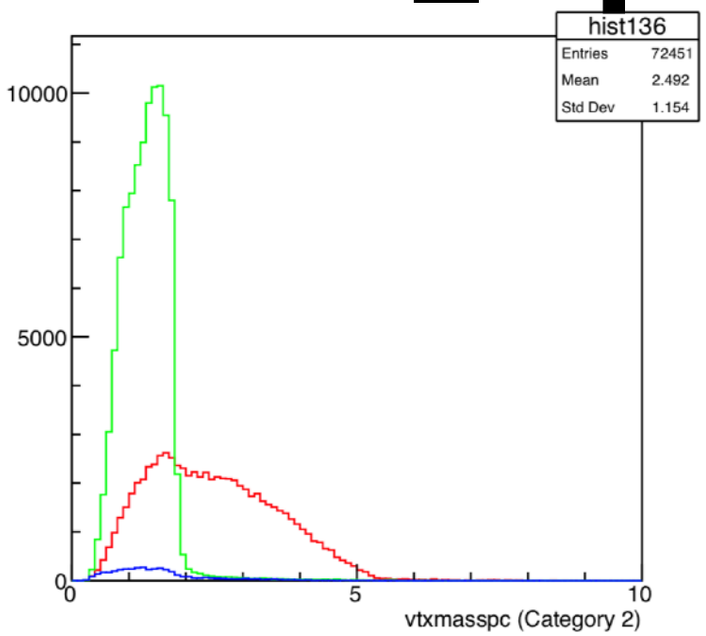
—b

—c

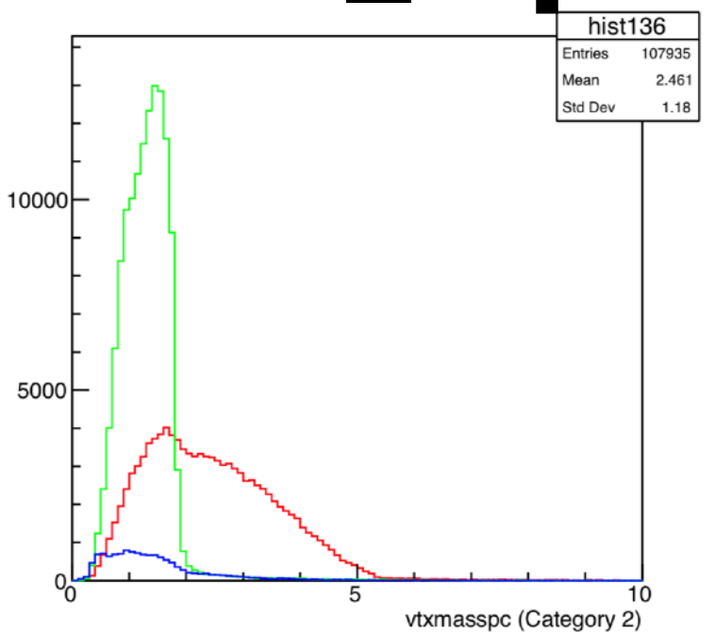
—q

~100k

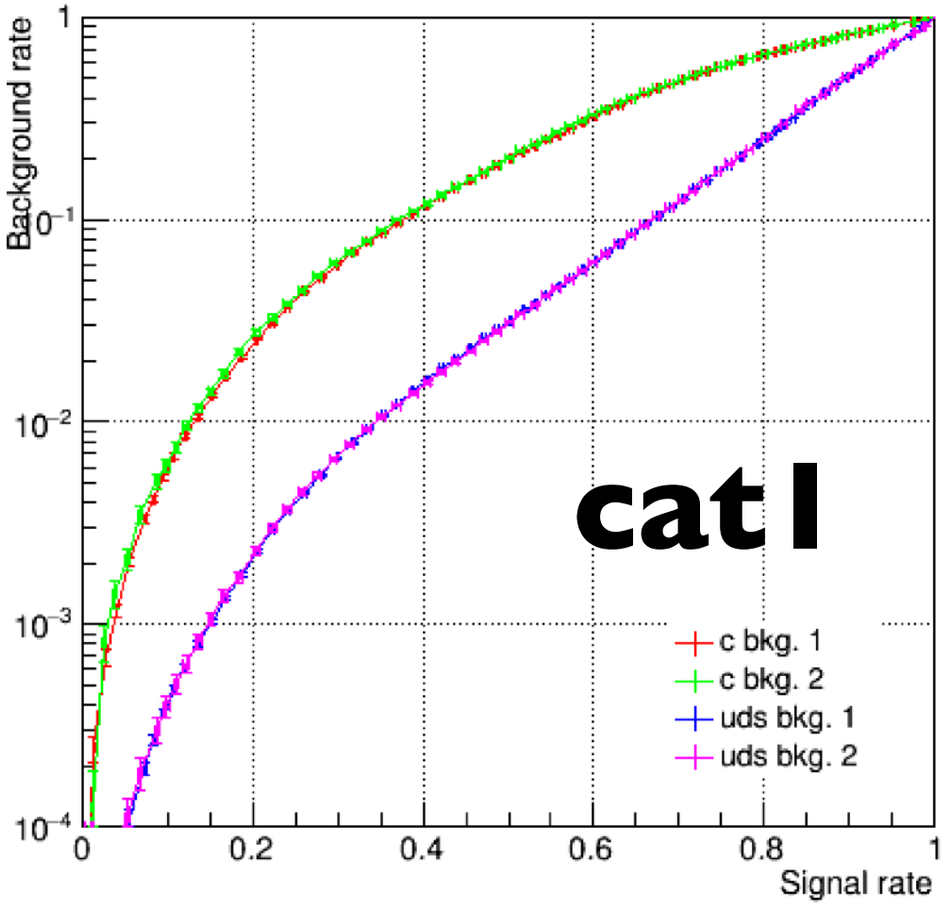
DBD_6q



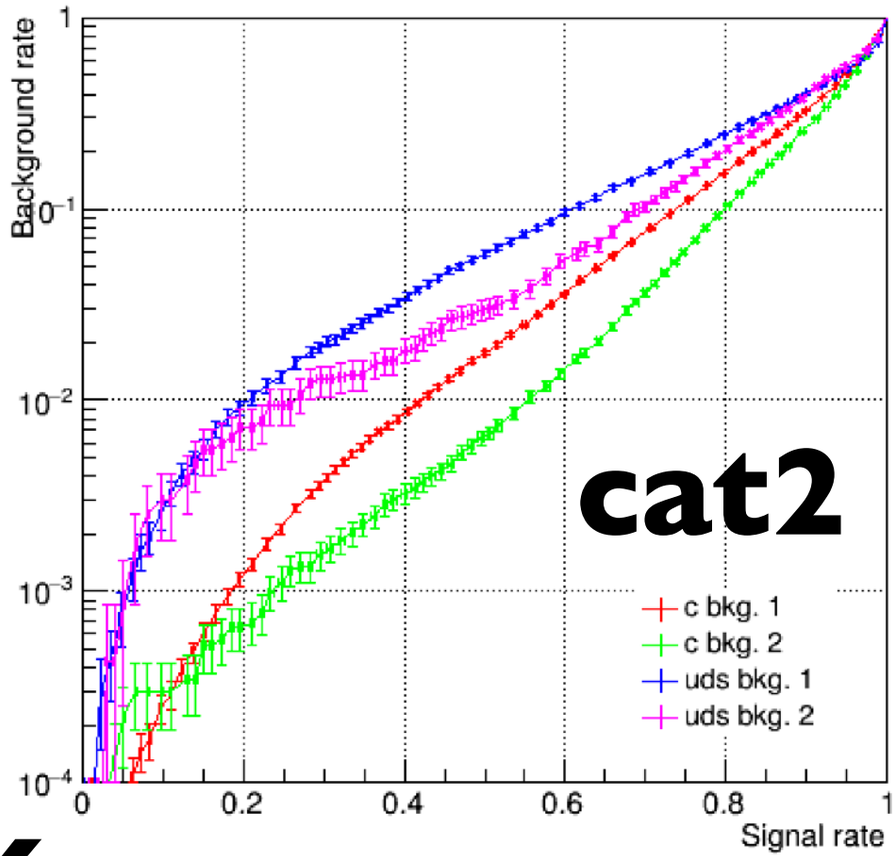
l5_6q



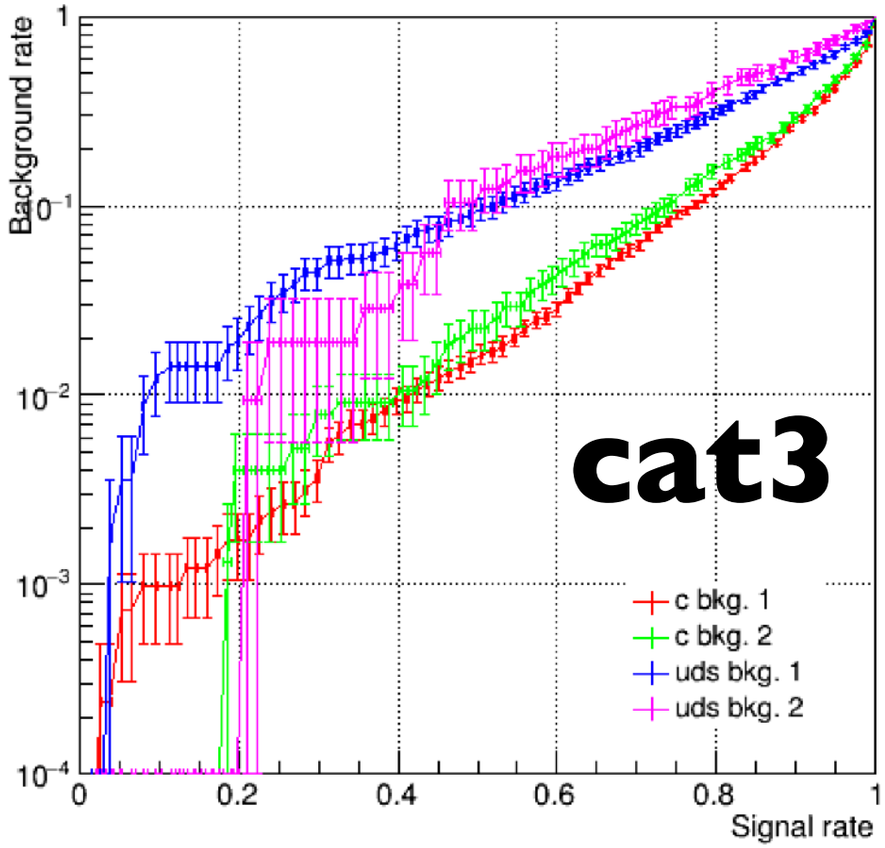
They look very similar…

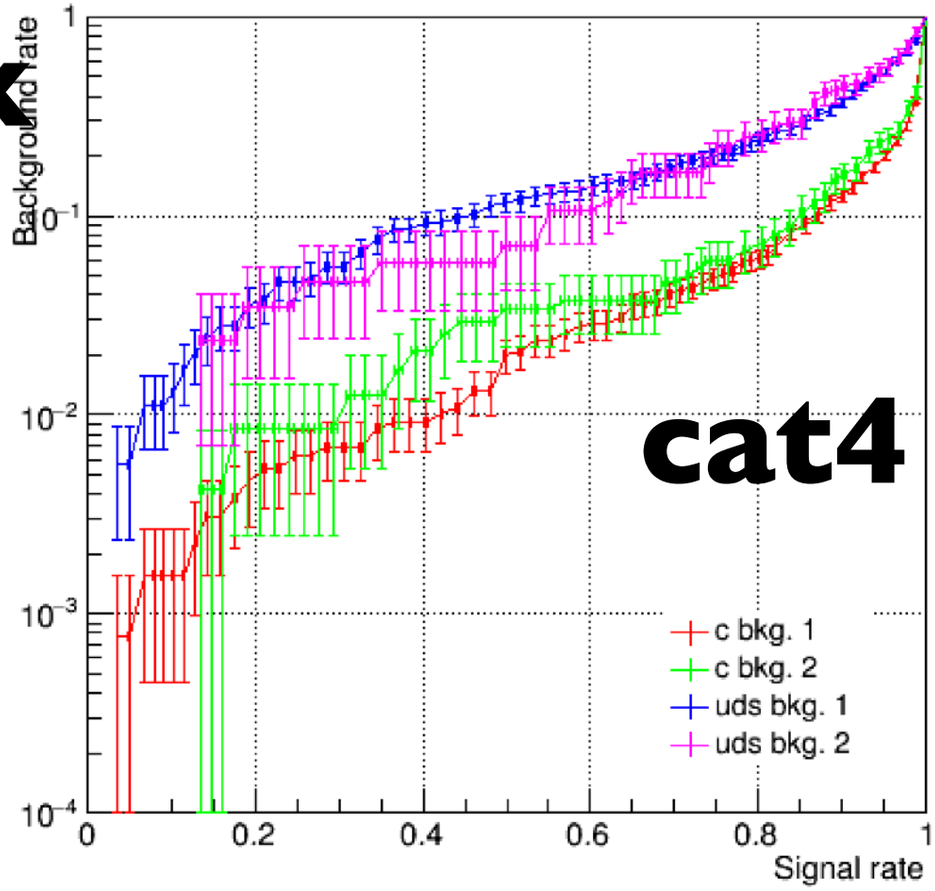

Background rate
Signal rate
c bkg. 1
c bkg. 2
uds bkg. 1
uds bkg. 2
cat1

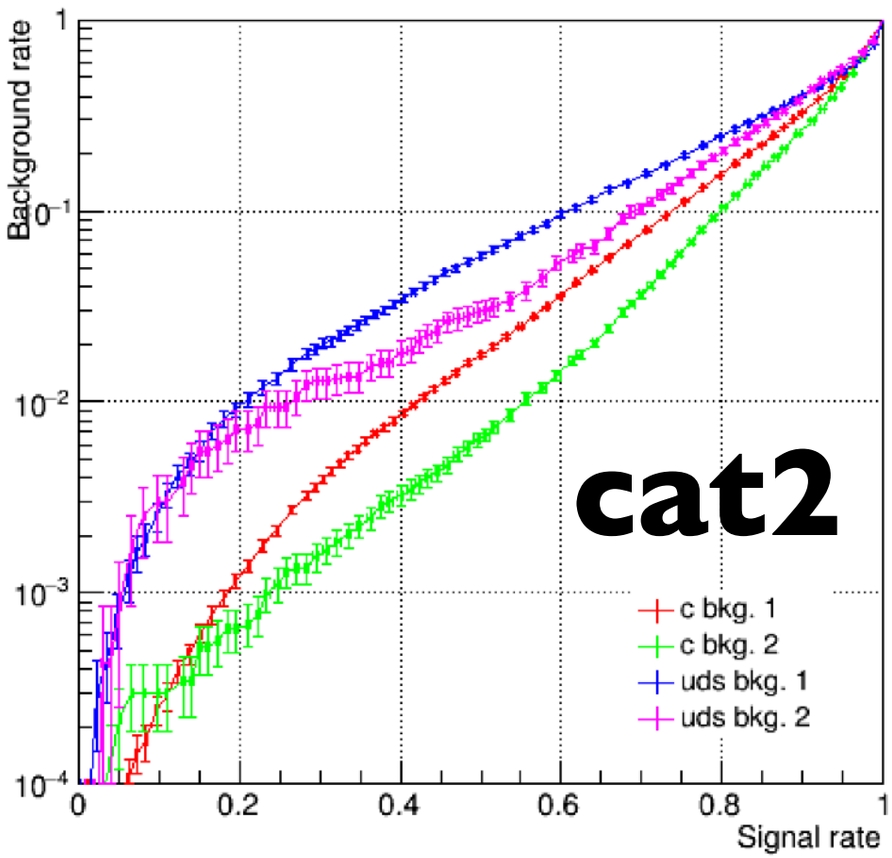
1: 100k


Background rate
Signal rate
c bkg. 1
c bkg. 2
uds bkg. 1
uds bkg. 2
cat2

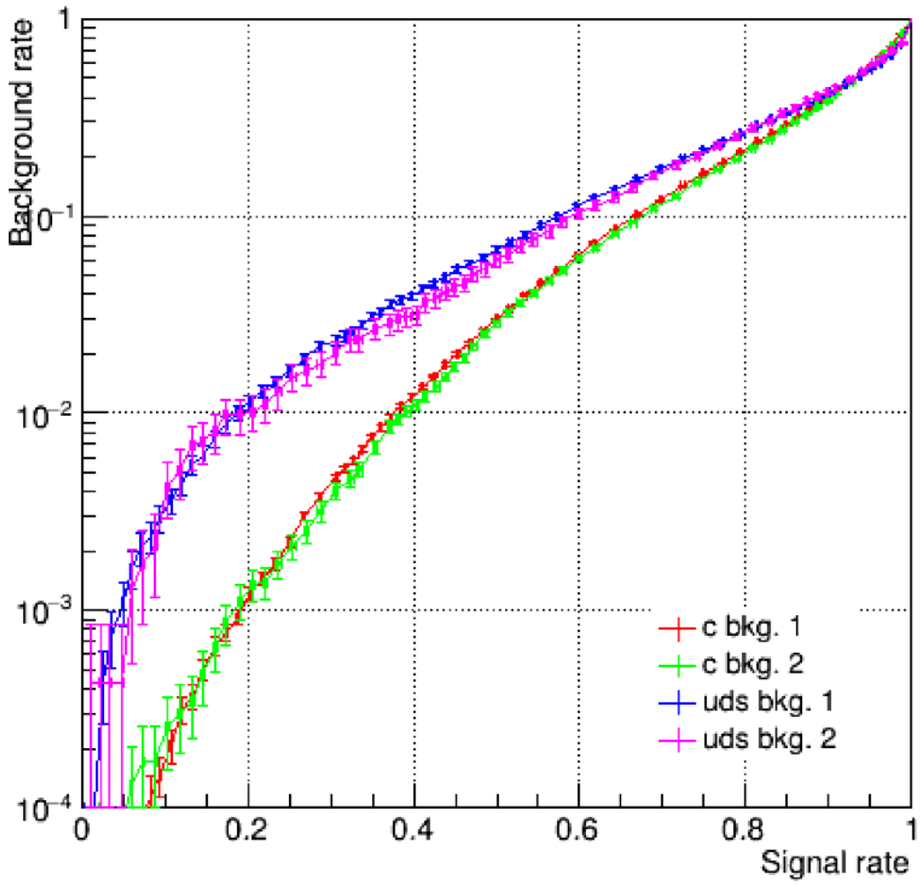
2: 20k


Background rate
Signal rate
c bkg. 1
c bkg. 2
uds bkg. 1
uds bkg. 2
cat3


Background rate
Signal rate
c bkg. 1
c bkg. 2
uds bkg. 1
uds bkg. 2
cat4



remove vtxmasspc



same as the one in the previous page

Conclusion & Plan

- **Too-small error problem solved (at least the cause identified).**
  - May need a bit more modification to make the code in a better manner. To be checked by experts.
  - Unfortunately this doesn't help for the "statistical dependency" problem.
  - The performance of flavour tagging doesn't change much.
  - I think there is no practical problem to use the samples produced already.
- **Remaining issue : "Statistical dependency"**
  - vtxmasspc (pt corrected mass) seems to have the key, especially in the case of single secondary vertex in a jet found (so called "category 2").
  - I think this is not so urgent problem now because it is more likely a problem at user-analysis level (we can redo later), not at common production level.
  - I don't mean I stop investigating now, but probably I should resume other businesses (ttbar hadronic, H->bb/cc/gg) to catch up the works.

Backup

Flavour tagging (statistical dependency issue)

- **vtxmasspc in flavour tagging**
  - I found this variable seems to be the main cause. (I checked by removing each input variable.)
  - Powerful to indentify b-jet (Mb~5GeV, Mc~2GeV)
  - Used in earlier experiments e.g. SLD ("Pt-corrected mass").
  - Taking neutrinos into account to correct secondary vertex mass.
  - Estimate minimum contributions from neutrinos by comparing the direction of the secondary vertex from the primary vertex and the momentum sum of the tracks from the secondary vertex.

$$M_{\mathrm{pc}} = \sqrt{M_{\mathrm{vis}}^2 + p_t^2} + |p_t|$$

if the primary- and secondary-vertex position errors are not precise, pt is set to be 0. —> vertex position error estimation could affect on this variable.

- **Checking helix trajectory, coordinate, etc**
  - No problem found

- **Checking minimization process.**
  - There are two different minimization :
    - 1) finding a position on a helix trajectory which gives the minimum distance to a target point. This is one-dimensional minimization.
    - 2) finding a position that gives minimum chi2 defined as a sum of variances to helices. To compute variance for each helix, we use 1). This is 3 dimensional minimization.
  - A method : PrintLevel(2) in minimization of 1) dumps more detailed info. though they are not so clear to me. But at least I found that some cases the minimization fails because of "nan" as error estimation.)
- **Findings**
  - Too-small error appears only when turning the beam constraint on.
  - ROOT::Minuit2::MnStrategy(1) —> ROOT::Minuit2::MnStrategy(0) in minimization (2) seems to be more stable.
  - Tolerance = 10^{-3} —> 10+3 in minimization (1) solve the "nan" problem above.
  - I tried these test parameters and see what happened.