

# Status of LCFIPlus

Taikan Suehara, Tomohiko Tanabe  
(ICEPP, U. Tokyo)

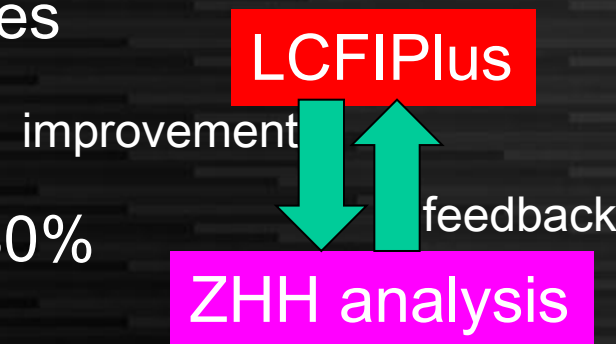
# Direction of LCFIPlus development

**LCFIVertex** The first realistic flavor tagging in ILC

- Incorporating modern flavor tagging techniques to obtain reasonable performance
- No other algorithms to be compared...
- Mainly tuned with Z-pole qqbar samples

**LCFIPlus** Our second version

- Clear target: Higgs self-coupling to  $\sim 30\%$   
➔ high demand for performance
- Focused on  $\geq 4$  jet environments
- Including jet clustering (performance driver for 6-jets)
- Trying many ideas for performance improvement



**LCFIPlus is more performance-driven,  
mainly concentrated on many-jet processes**

# Data/process flow

All in "lcfiplus" namespace

## EventStore

singleton for data pool

vector<Track \*>            vector<Vertex \*>  
vector<Neutral \*>        vector<Jet \*>  
vector<MCParticle \*>    any other types

- Automatic type identification  
(Allow one name with multiple types)
- Automatic creation/deletion  
(using ROOT class dictionary)

## Algorithm

PrimaryVertex   JetVertexRefiner  
BuildUpVertex   FlavorTag    TrainMVA  
JetClustering    MakeNtuple    ReadMVA etc.

- Parameters class used  
for type-safe configuration

## LCIOStorer

- Automatic conversion from  
LCIO to lcfiplus classes  
(using hook in EventStore)
- Conversion to LCIO  
is manually invoked by  
LcfiplusProcessor

## LcfiplusProcessor

- Marlin processor
- Process Marlin parameters  
to be passed to Algorithm
- LCIO I/O configuration

LCIO

configuration

Internal algorithms

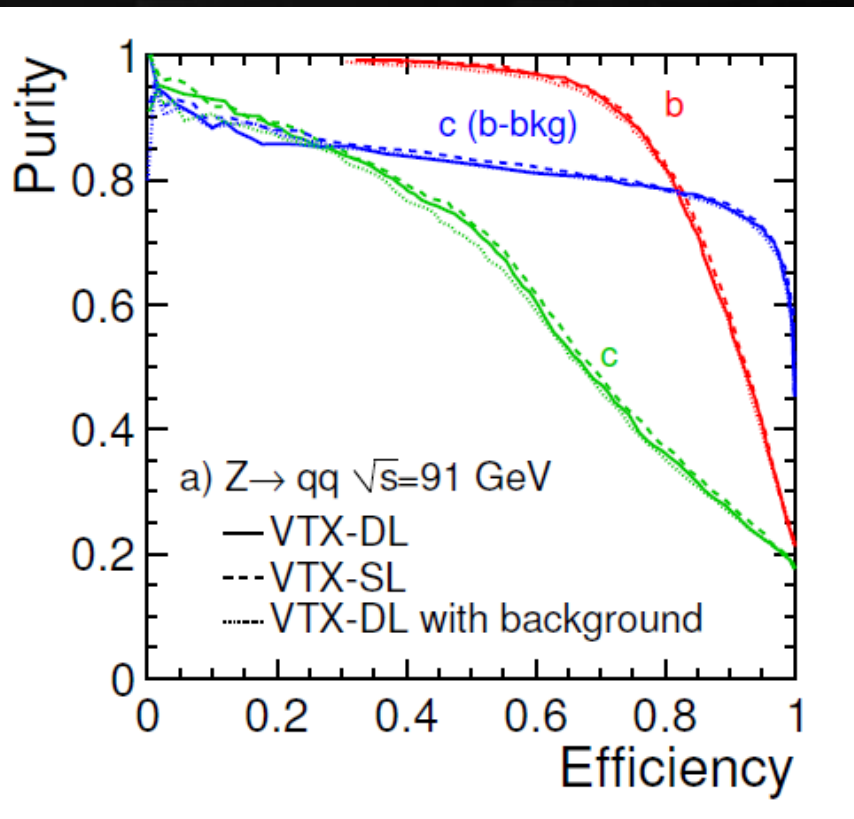
Taikan Suehara

Independent

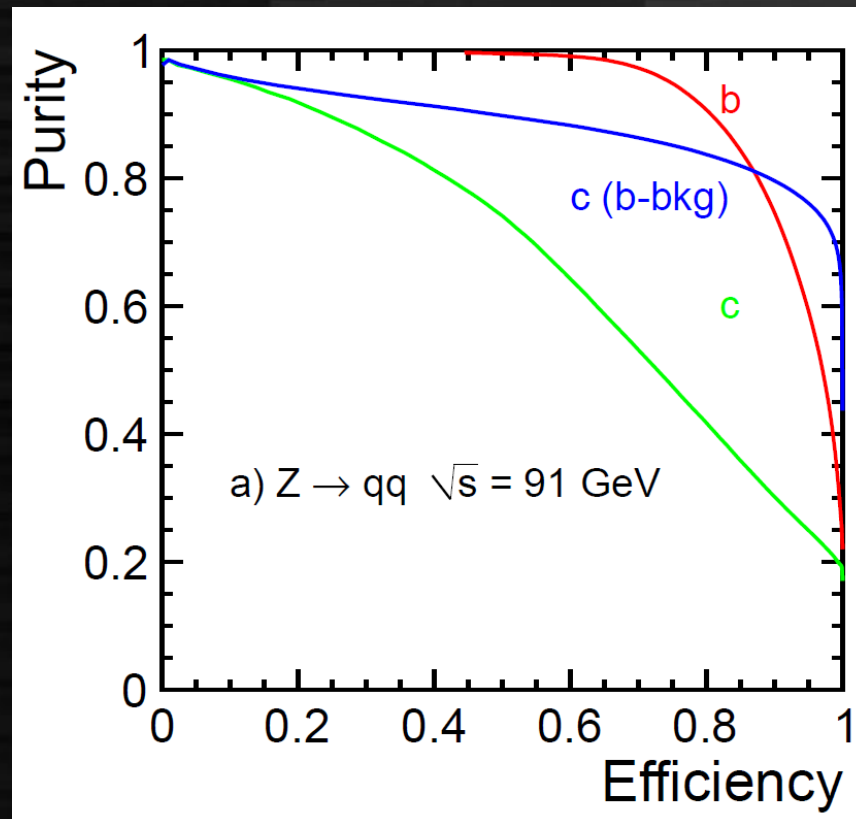
Kingston, 25 Oct. 2012

Marlin

# Performance: (old) LCFI vs LCFI+

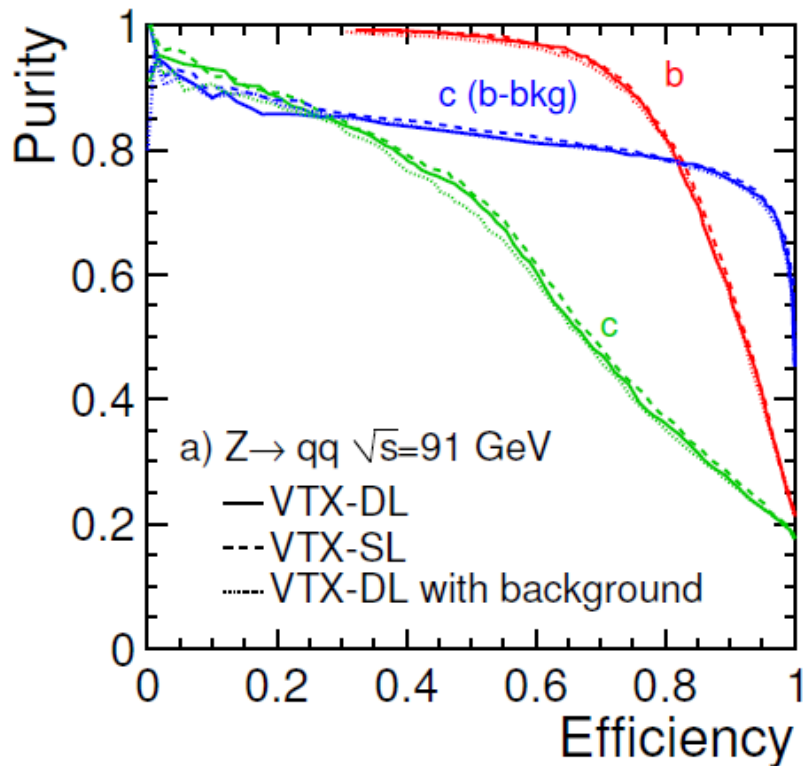


LCFIVertex performance  
in ILD Lol

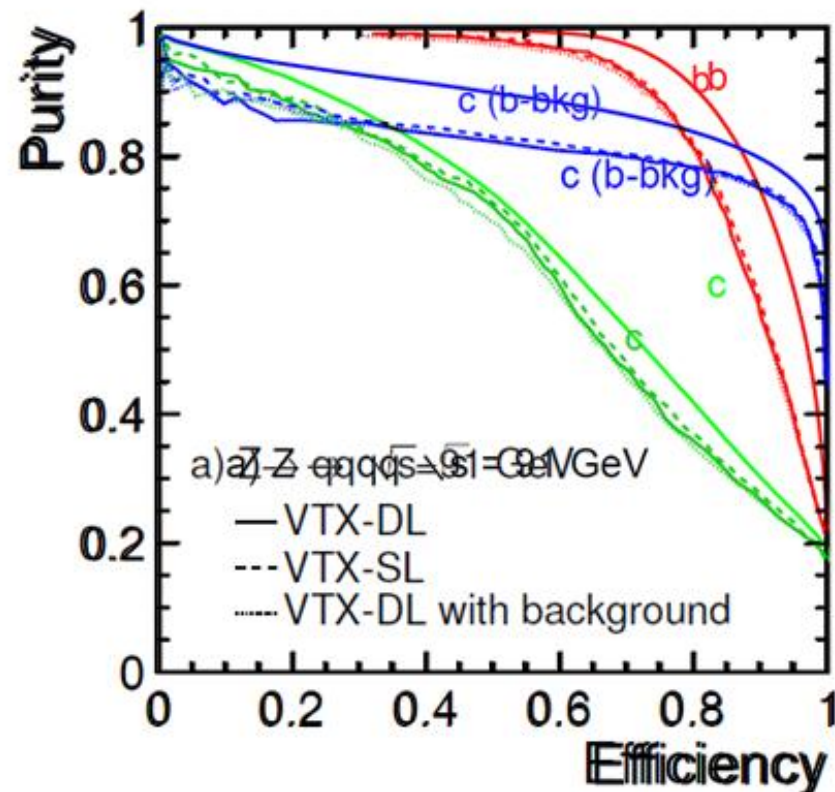


ILD\_o1\_v5  
LCFIPlus v02 variables

# Performance: (old) LCFI vs LCFI+



LCFIVertex performance  
in ILD Lol



ILD\_o1\_v5  
LCFIPlus v02 variables

# LCFIPlus processors

1. Primary vertex finder
2. Secondary vertex finder

DBD mass reconstruction up to here

3. Jet clustering

JetClustering + JetVertexRefiner

4. Training MVA

(can be omitted with existing weight files)

1. Making ntuples
2. Training

5. Flavor tagging

# Vertex Finders

- PrimaryVertexFinder
  - tear-down with beam vertex
- BuildUpVertex
  - Secondary vertex finder with build-up method
  - V0 rejection (original code, updated)

(a) $ZHH \rightarrow qqbbbb$	Track origin			
	Primary	$b$ hadron	$c$ hadron	Other
Number of all reconstructed tracks	67575	12912	15246	4087
Number of tracks used by ZVTOP	1162	8534	10404	999
...in <i>good</i> vertices	-	8248	10103	-
Number of tracks used by our original vertex finder	617	8717	10529	358
...in <i>good</i> vertices	-	8551	10333	-
(b) $t\bar{t} \rightarrow bbqqqq$	Track origin			
	Primary	$b$ hadron	$c$ hadron	Other
Number of all reconstructed tracks	74504	8945	12602	4219
Number of tracks used by ZVTOP	920	5999	8353	1024
...in <i>good</i> vertices	-	5830	8137	-
Number of tracks used by our original vertex finder	420	6161	8447	341
...in <i>good</i> vertices	-	6060	8279	-

Better than LCFIVertex vertex finder in ZHH/tt sample!

# Jet Clustering

- Should be used in user analysis (not included in DBD prod)
- Jet clustering with vertex information
- Various configuration possible
  - Ordinal Durham method (vertex = “0”, UseMuonID = 0)
  - Durham with vertex, but no enhancement for separation of vertex-jets (YAddedForJetVertexVertex = 0, etc)
  - Durham with vertex with separation of vertex-jets (default)
  - Using jet muons as vertex (with UseMuonID = 1)
- Multiple output collections possible
  - ex. NJetsRequested = 8 6 4, (must be descending order),  
OutputJetCollectionName = Jets8 Jets6 Jets4
- Problem of enhancement of ttg->ttbb
  - Should be updated for ZHH analysis (but not soon)



# Jet Vertex Refiner

- Should be used in user analysis after jet clustering

- Consists of two algorithms

- SingleTrackVertexFinder & VertexCombiner

- SingleTrackVertexFinder

- reconstruct single-track vertices using existing vertex directions

- VertexCombiner

- combine vertices into two at most aiming at combining multi+single vertices which are from same b or c – tuned for b/c separation

- Jet & vertex collection are specified separately, so this can be used after other jet clustering method (Durham, anti- $k_T$  etc.)

Event	1+1 vtx	2 vtx
bb	20.4%	22.2%
cc	0.73%	0.16%
qq	0.06%	0.04%

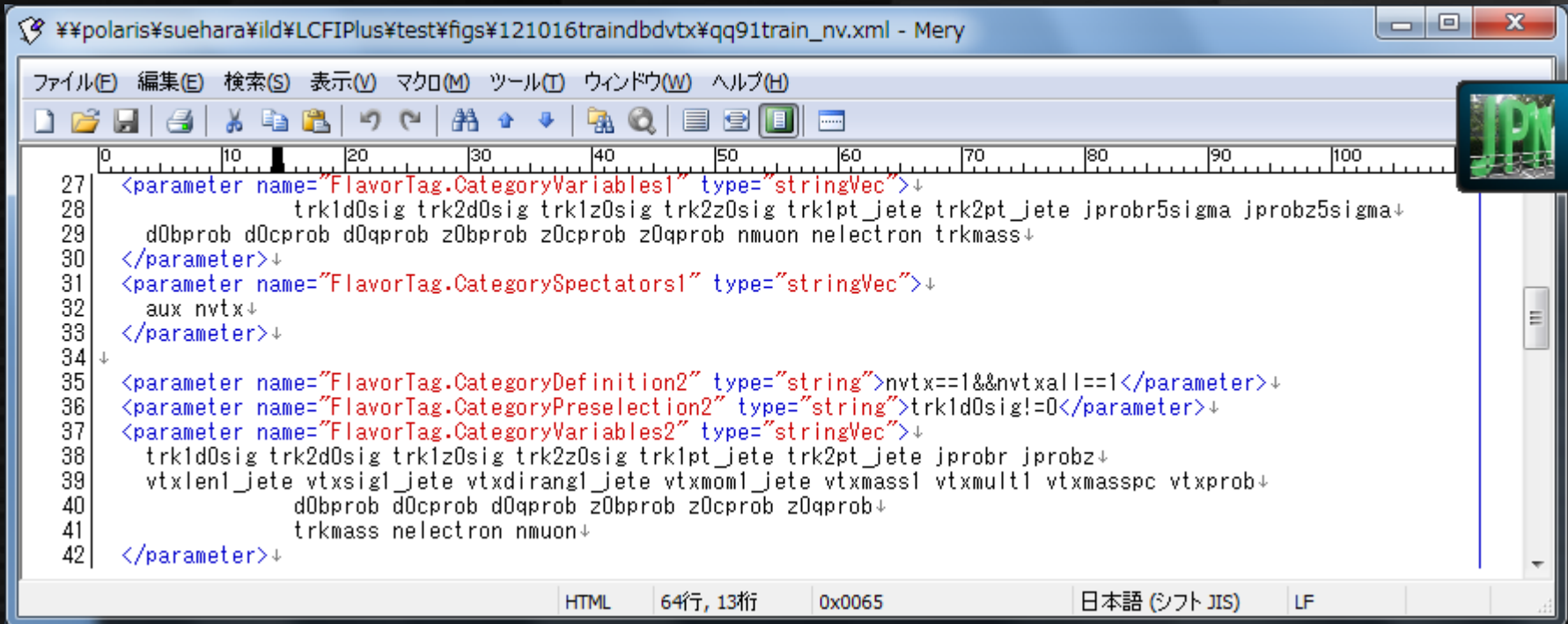
# Flavor Tagging

- Based on TMVA Boosted Decision Trees
  - Four categories: #vtx = 0, 1, 1+singletrack, 2
  - Output: Category, BTag, CTag ( $+\alpha$ ) in LCIO PID
- Procedure (after jet clustering/vertex refiner)
  1. FlavorTag + MakeNtuple for each training sample
  2. TrainMVA with all ntuples (output: weight file)
  3. FlavorTag + ReadMVA with the weight file
    - 1 + 2 can be omitted for use of existing weight files

# Standard Training Sample (ILD)

- ILDConfig/LCFIPlusConfig/lcfiweights
- qq samples (91 GeV / 250 GeV)
  - 100 kjets each
    - qq(91/250)\_v(01/02)\_p01
  - 1 Mjets each
    - qq91\_v(01/02)\_p11 (released very soon)
    - 250 GeV coming (need to run Mokka)
- 6q samples (500 GeV / 1 TeV)
  - bbbbbb/cccccc/qqqqqq, mainly from ZZZ
  - 500k/500k/1500k jets
    - 6q(500/1000)\_v(01/02)\_p01 (1 TeV soon)
- 4q samples planned (500 GeV / 1 TeV)

# New variables (v02)



```
27 <parameter name="FlavorTag.CategoryVariables1" type="stringVec">↓
28     trk1d0sig trk2d0sig trk1z0sig trk2z0sig trk1pt_jete trk2pt_jete jprobr5sigma jprobz5sigma↓
29     d0bprob d0cprob d0qprob z0bprob z0cprob z0qprob nmuon nelectron trkmass↓
30 </parameter>↓
31 <parameter name="FlavorTag.CategorySpectators1" type="stringVec">↓
32     aux nvtx↓
33 </parameter>↓
34 ↓
35 <parameter name="FlavorTag.CategoryDefinition2" type="string">nvtx=1&&nvtxall==1</parameter>↓
36 <parameter name="FlavorTag.CategoryPreselection2" type="string">trk1d0sig!=0</parameter>↓
37 <parameter name="FlavorTag.CategoryVariables2" type="stringVec">↓
38     trk1d0sig trk2d0sig trk1z0sig trk2z0sig trk1pt_jete trk2pt_jete jprobr jprobz↓
39     vtxlen1_jete vtxsig1_jete vtxdirang1_jete vtxmom1_jete vtxmass1 vtxmult1 vtxmasspc vtxprob↓
40     d0bprob d0cprob d0qprob z0bprob z0cprob z0qprob↓
41     trkmass nelectron nmuon↓
42 </parameter>↓
```

Vertex probability

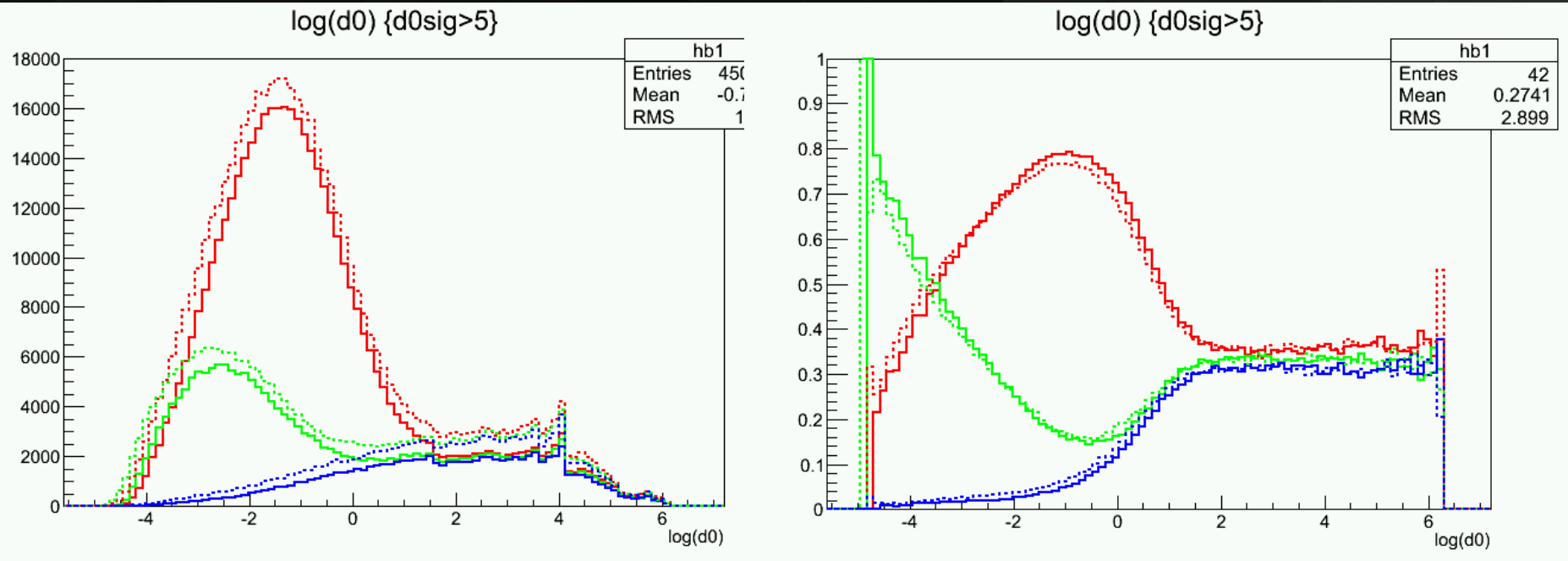
(using b/c/q d0/z0 distributions in data/vtxprob/ )

Mass of secondary tracks

# electrons, # muons

# New input variables

- product of  $d_0/z_0$  b/c/q likeness over all secondary tracks ( $d_0z_{sig}/z_0sig > 5$ )

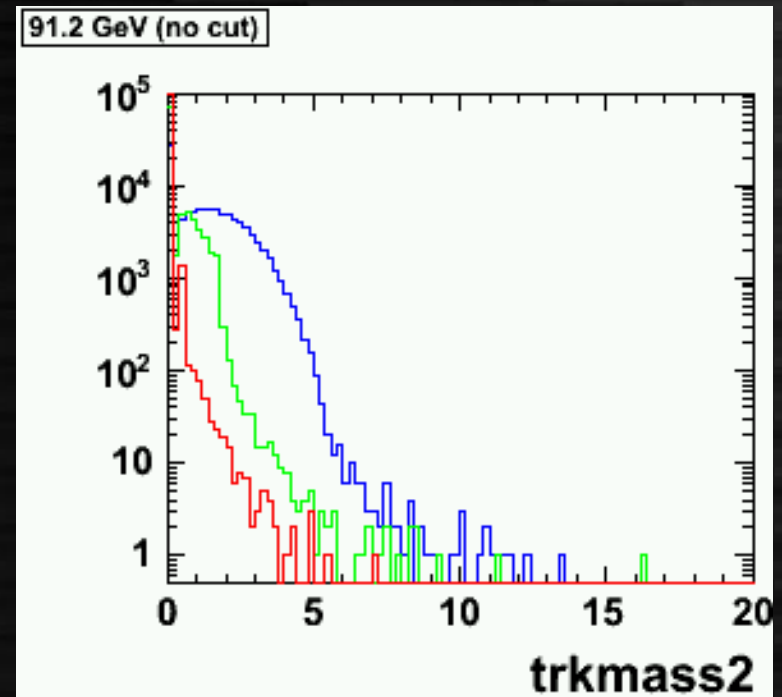
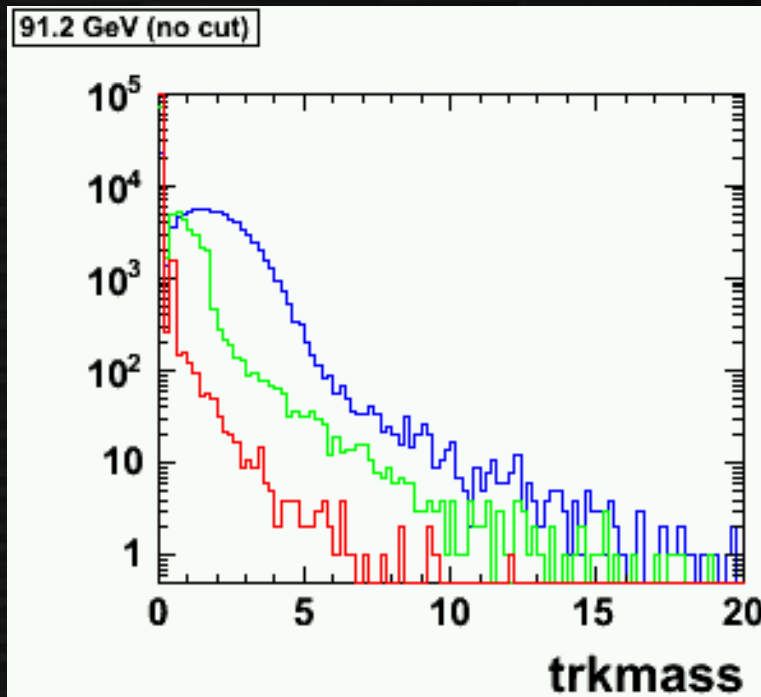


- (existing) joint probability is modified to use  $d_0/z_0sig < 5$  tracks only (for independency)

ROOT files in ILDConfig/LCFIPlusConfig/vtxprob/ needed:  
Please check the error message if you plan to use v02 variables

# New input variables(2)

- Mass with all secondary tracks
  - loose selection: trkmass
  - tight selection: trkmass2 (currently not used)

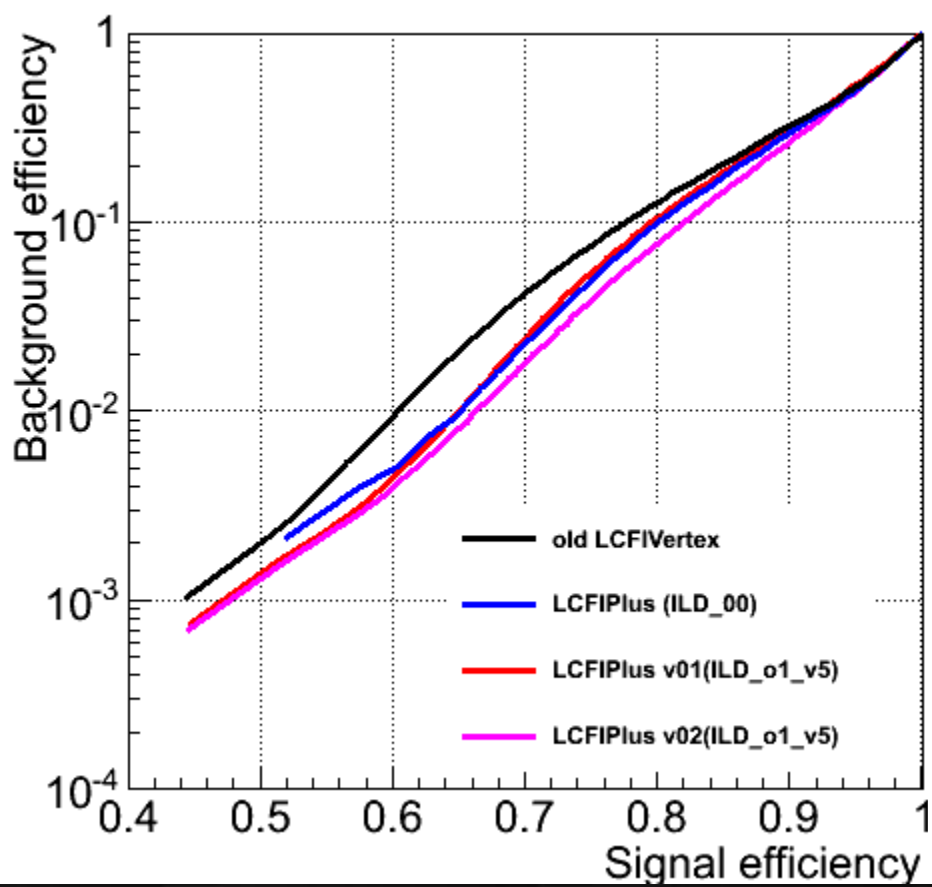


# New input variables(3)

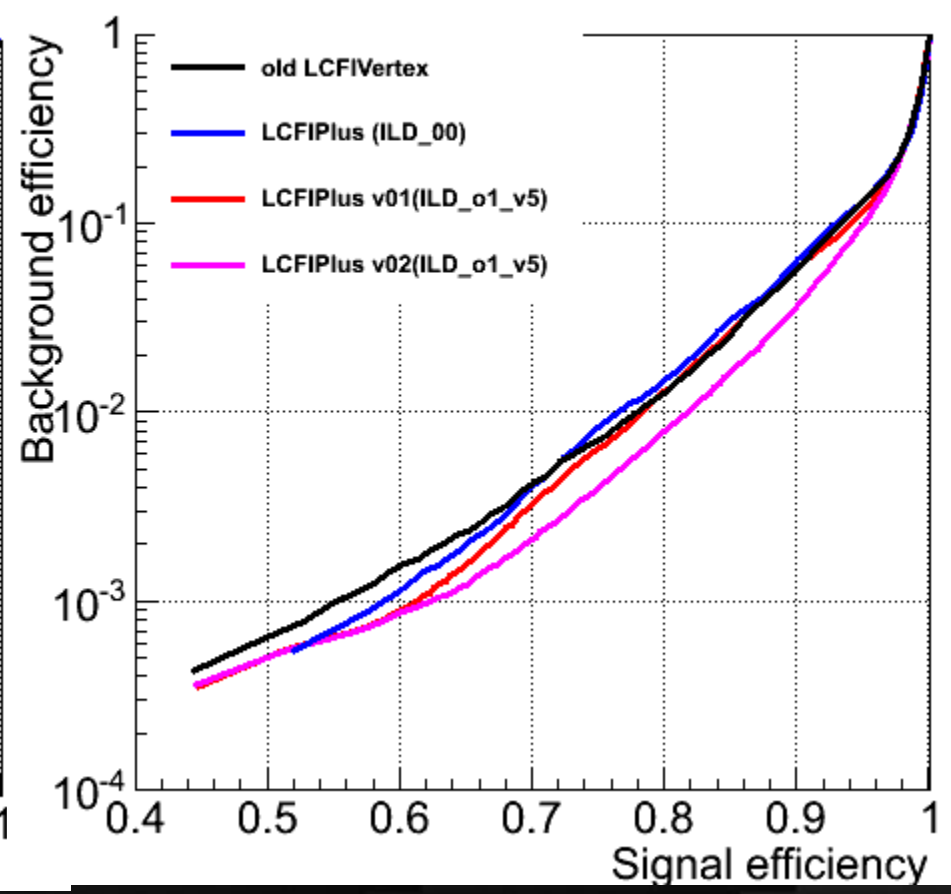
- # muons, # electrons
  - Tuned to  $> 3\text{-}4$  GeV muons/electrons
    - require off-IP, muon hit, ECal/Hcal energy deposit
  - Efficiency (overall):  $\sim 25\%$   
(rejected leptons)
    - Energy  $< 3$  GeV: about 60%
    - secondary cut (5 sigma): about 10%
    - Suffered from mis-PFA: about 30%
  - Electron purity decreases for larger energies

# b-tag performance: Z-pole qq

b-tag: Z->qq at 91.2 GeV, c bkg



b-tag: Z->qq at 91.2 GeV, uds bkg

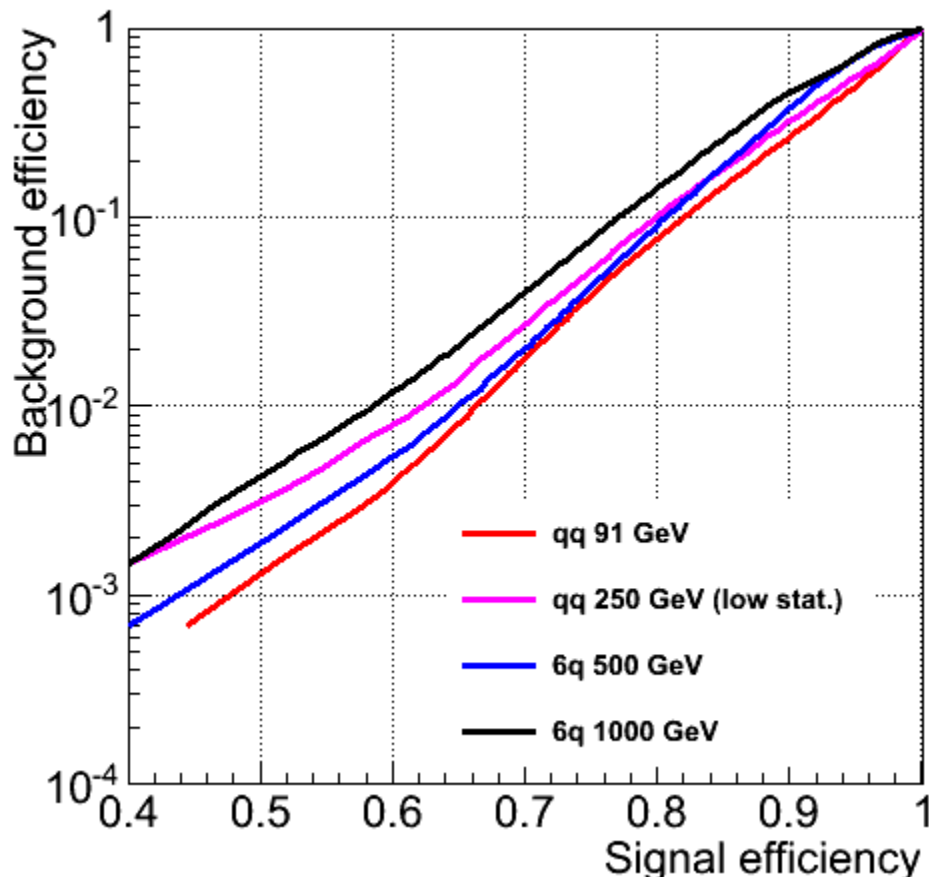


old LCFIVertex -> LCFIPlus improvement seen in all region  
ILD\_00 & ILD\_o1\_v5 give similar performance  
v02 is better than v01 in all region: use v02!

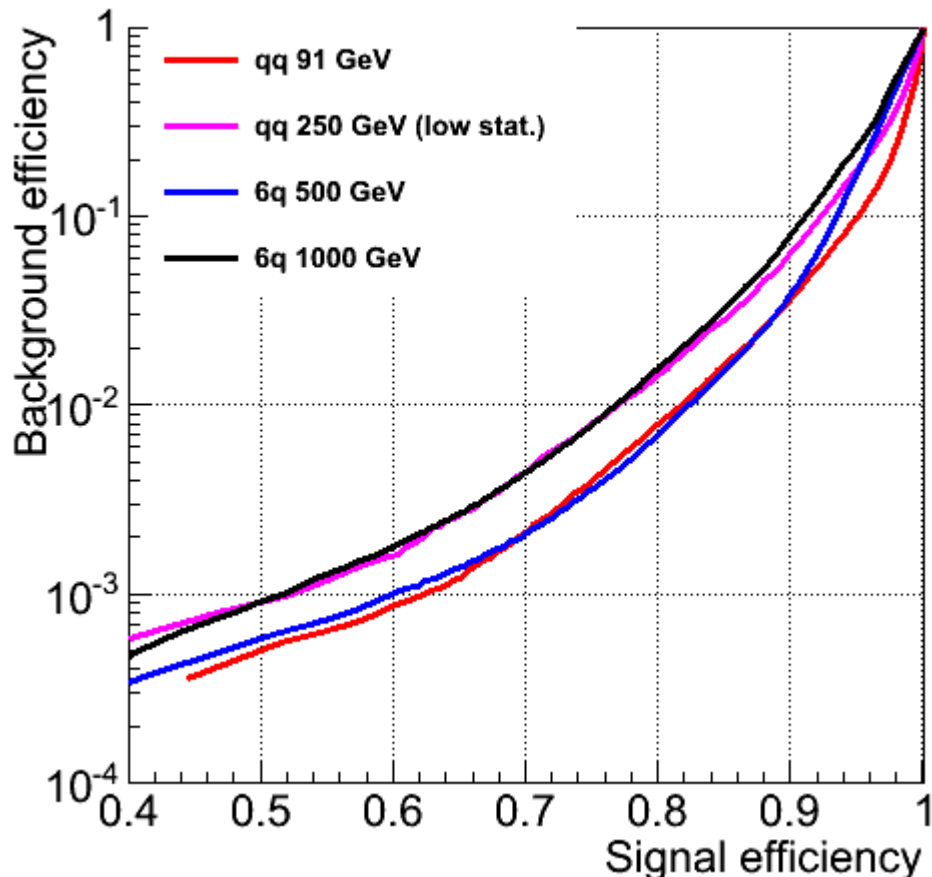


# Dependence on Process

b-tag: LCFIPlus v02 variables, c bkg



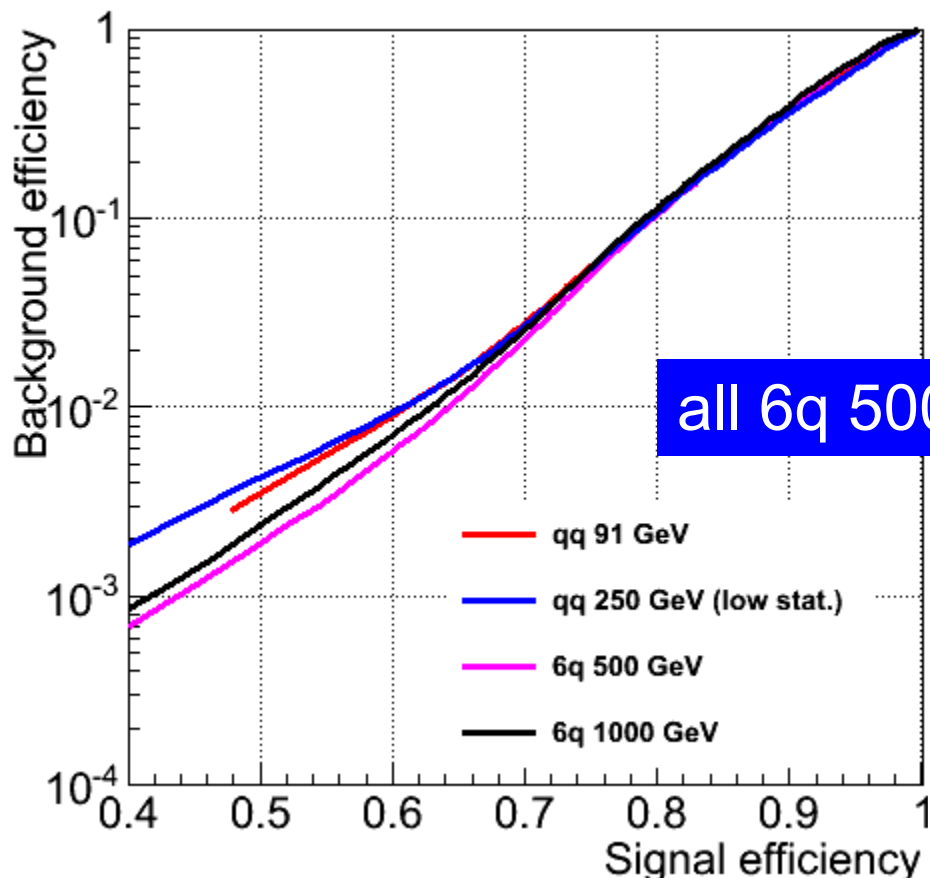
b-tag: LCFIPlus v02 variables, uds bkg



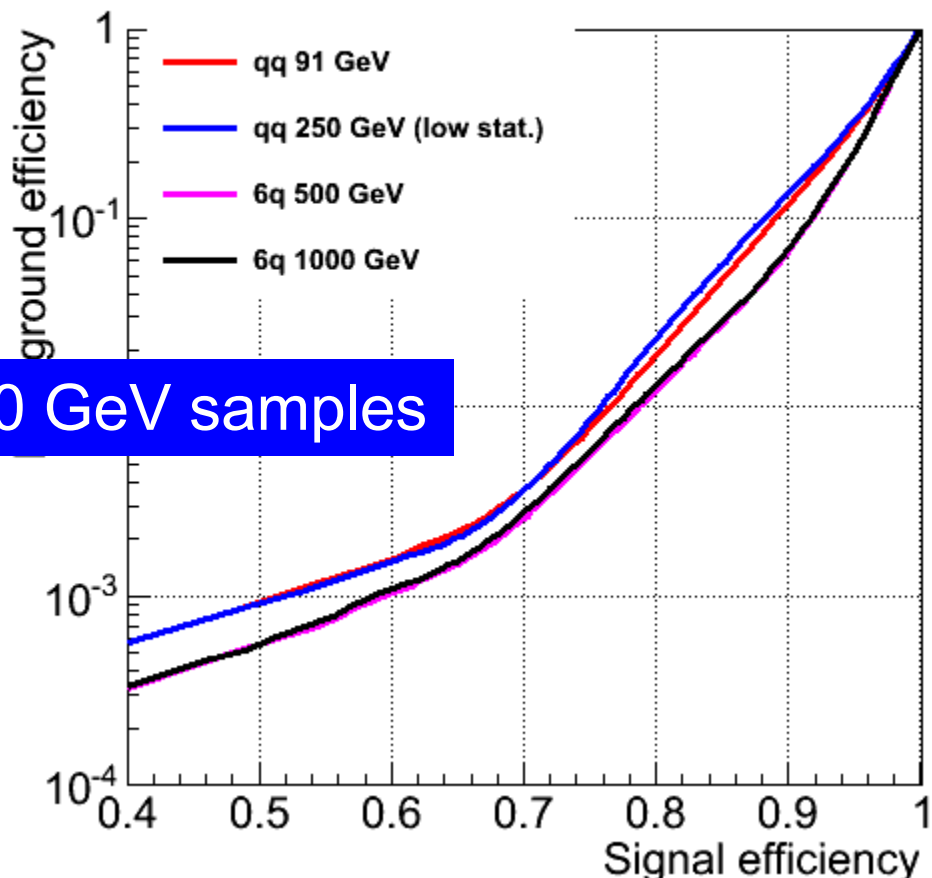
use the same process (each) for training  
worse in higher energy jets: need to tune v0 rejection?

# Dependence on Weight Files

b-tag: LCFIPlus v02, 6q 500 GeV events, c bkg



b-tag: LCFIPlus v02, 6q 500 GeV events, uds bkg

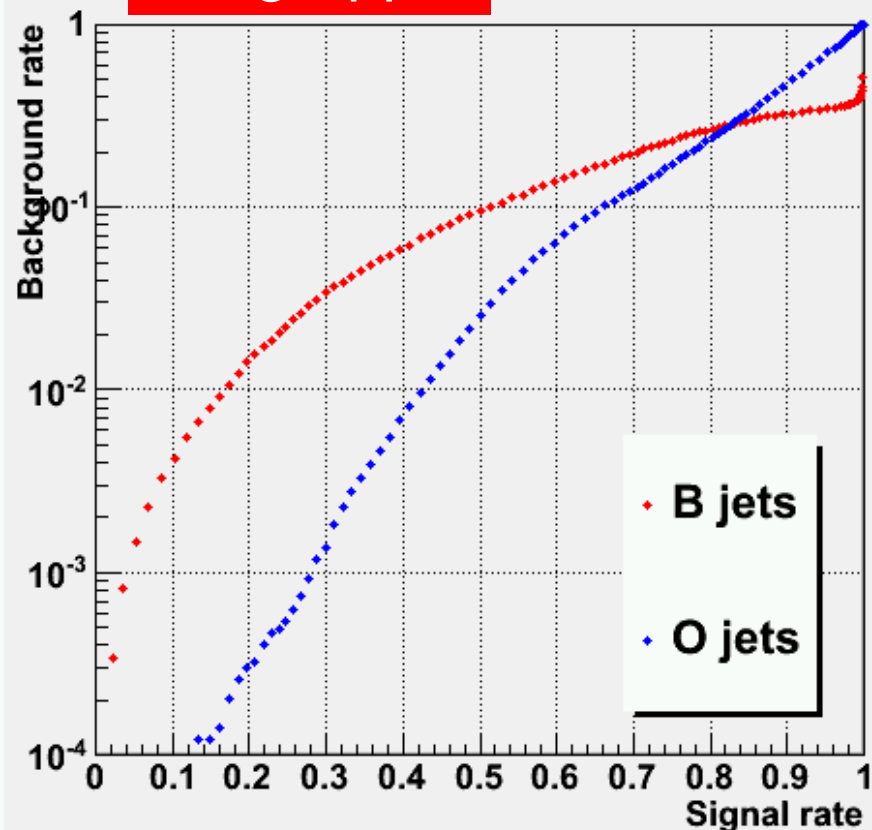


all 6q 500 GeV samples

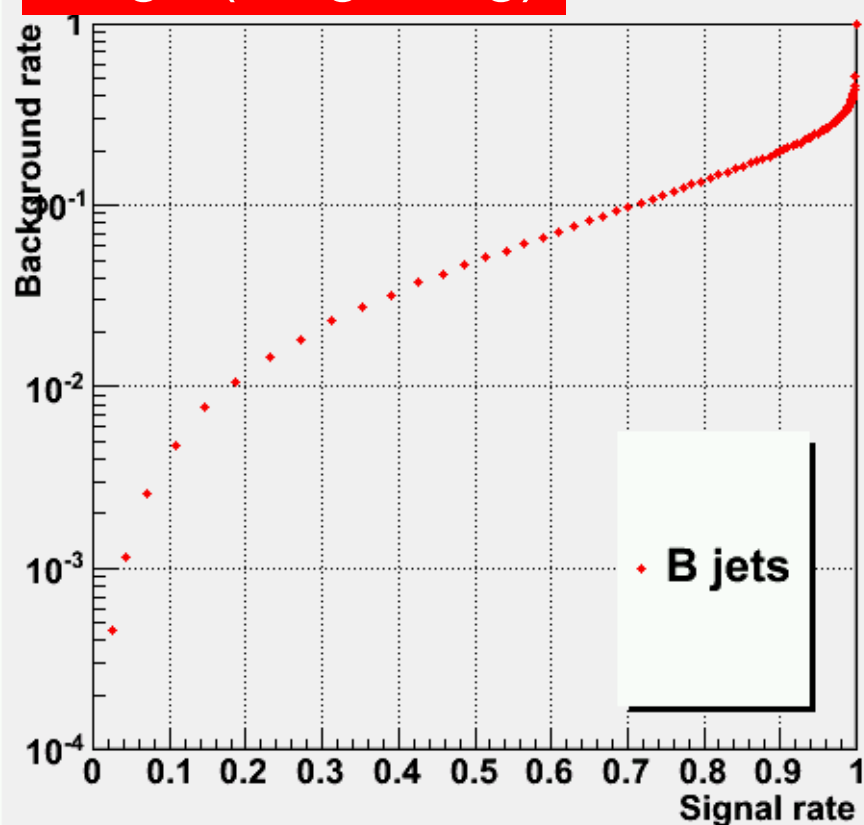
For selecting weight files, # of quarks affects more than energy!

# C-tag vs BC-tag

c-tag, qq91



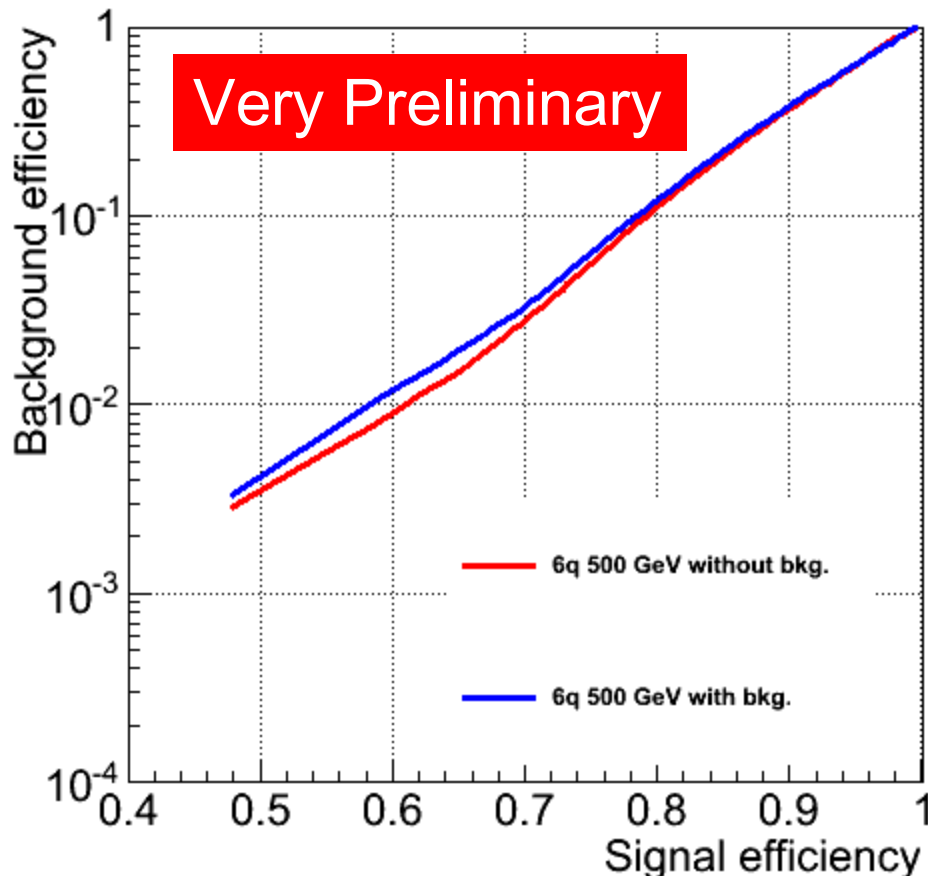
ctag / (btag+ctag)



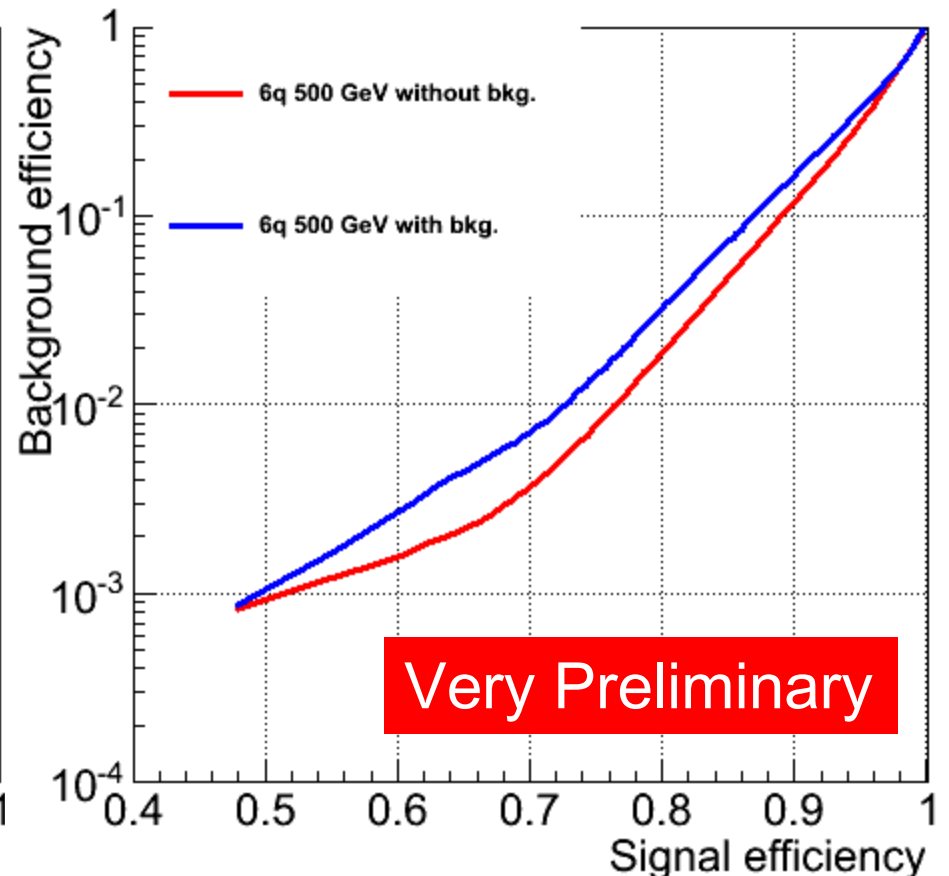
Use  $ctag/(btag+ctag)$  as previous 'bc-tag'  
Performance is identical to 'bc-only' training

# background

b-tag: LCFIPlus v02, qq 91 GeV training, c bkg



b-tag: LCFIPlus v02, qq 91 GeV training, uds bkg



some effects on beam background seen: may need to tune...

# Plans

- Short term (1-2 weeks)
  - release 6q1000, 4q, qq250 (better stat.)
  - found a minor issue in v02 – will be updated
  - using ttbar for training, using MC information
    - 6-category tagging: B, C, O, BB, BC, CC
    - Code has been ready: need sanity check
  - Investigating pileup effect
- Mid term
  - Jet clustering re-optimization for ZHH
  - More variables, more performance

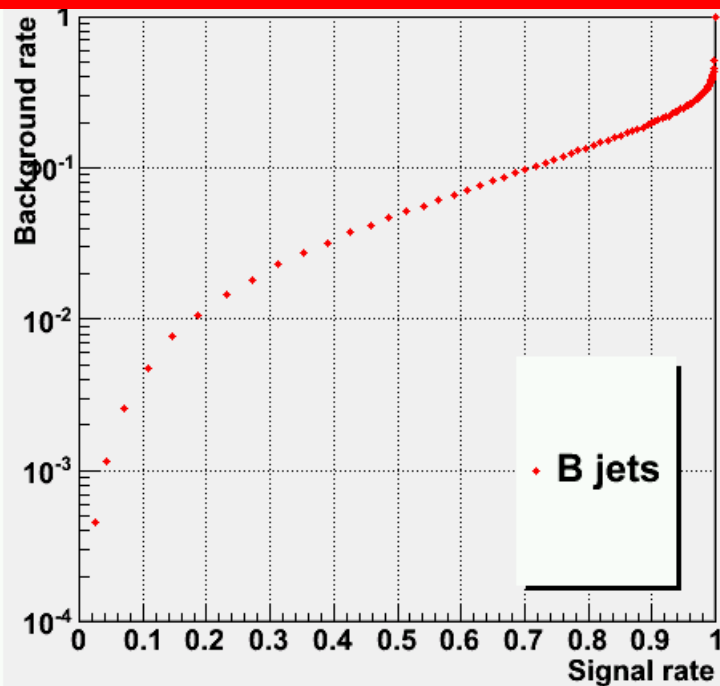
# Summary

- LCFIPlus (almost) ready for DBD analysis
- Impressive performance improvement seen!!
- Various weight files supplied, more coming
  - number of quarks seem to be important for choosing weight file
- Use  $\text{ctag}/(\text{btag}+\text{ctag})$  for bc-tag
- Performance of v02 is better:  
we encourage to use it
- Some effect of beam background seen
  - need more investigation

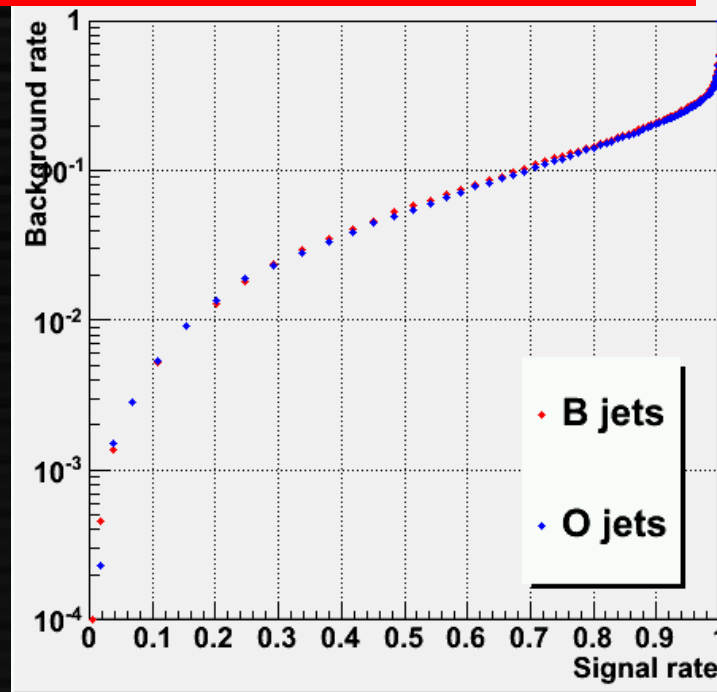


# BC-tag??

ctag / (btag+ctag), qq91



ctag, training with b/c/b



In our sample btag + ctag + other is normalized to 1  
Use  $\text{ctag}/(\text{btag}+\text{ctag})$  as previous 'bc-tag'