



# python steering fileと 解析用ファイルの出力

今野智之  
KEK素核研

Belle II analysis tutorial, 2017/6/24, KEK

# 方針

講義の目標 : **basf2 script (python)**でどんなことが出来るかMCを使って体験する

- 今日の内容はconfluenceにあるtutorialを基にしています。  
<https://confluence.desy.de/display/BI/Physics+HandsOnAnalysisTutorialJune2017>
- basf2 scriptの機能を全て網羅する(できる)気はさらさないので解析を始めるときのreference(の足がかり)になれば良いかなと思います。

## 内容

- KEK CCでbasf2を使うには
  - basf2 環境設定
  - MC7サンプルの所在
- basf2 script (python)を使って物理解析
  1. データファイルを読む
  2. ntupleに落とし込む
  3. イベント再構成とセレクション
- basf2のpython コードはKEKCC以下のパスからコピーしてください
  - /gpfs/fs02/belle2/users/tkonno/basf2samples

**KEK CCでbasf2を使うには**

# 環境設定

- KEKCCのユーザーアカウントを取得する(groupはbelle2)
- KEKCCにログインする(<youraccount>はKEKCCのアカウント名)

```
[tkonno@cw06 ~]$ ssh <youraccount>@login.cc.kek.jp -XY
```

- 環境変数を読み込む

```
[tkonno@cw06 ~]$ source /sw/belle2/tools/setup_belle2  
Belle II software tools set up at: /sw/belle2/tools
```

- 使用するbasf2リリースバージョンを選択する

```
[tkonno@cw06 ~]$ setuprel release-00-08-00  
Environment setup for release: release-00-08-00  
Central release directory      : /cvmfs/belle.cern.ch/sl6/releases  
/release-00-08-00
```

※ 今回は安定版のrelease-8を使用します

ここまででbasf2を動かす環境設定は完了です

# 設定を確認

Belle IIのロゴ(アスキーアート)

- 下記の設定されていれば、設定は完了(のはず)

```
[tkonno@cw06 ~]$ basf2 --info
```

```
BASF2 (Belle Analysis Software Framework 2)
```

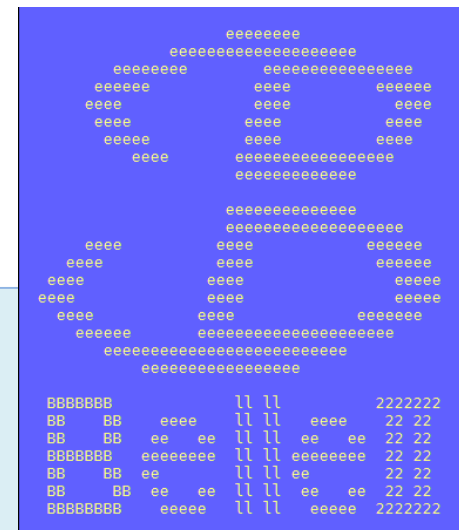
```
Copyright(C) 2010-2016 Belle II Collaboration
```

```
Version release-00-08-00
```

```
-----
BELLE2_RELEASE:           release-00-08-00
BELLE2_RELEASE_DIR:       /cvmfs/belle.cern.ch/sl6/releases/release-00-08-00
BELLE2_LOCAL_DIR:
BELLE2_SUBDIR:            Linux_x86_64/opt
BELLE2_EXTERNALS_VERSION: v01-03-01
BELLE2_ARCH:             Linux_x86_64
Kernel version:           2.6.32-642.15.1.el6.x86_64
Python version:           3.5.2
ROOT version:             6.06/08
```

```
basf2 module directories:
```

```
/cvmfs/belle.cern.ch/sl6/releases/release-00-08-00/modules/Linux_x86_64/opt
-----
```



# MC samplesにアクセスする

- MC production 7 (MC7)を使用します

- MC8以降はgbasf2 (GRID)からのアクセスのみです  
=> 詳細は早坂さんの講義で(この講義の意義とは?)

- MC7 sampleの詳細:

- <https://confluence.desy.de/display/BI/MC7+samples+for+analysis+users>

- Phase III (4S) generic samplesを使います

- <https://confluence.desy.de/display/BI/MC7+phase+III+-+Y%284S%29+generic+samples>

ページ / ... / MC7 samples for analysis users

## MC7 phase III - Y(4S) generic samples

Jake Bennett posted on 02. 12. 2016 04:14h - last edited by Pablo Goldenzweig on 24. 4. 2017 21:25h

- mixed
- charged
- uubar
- ddbar
- ssbar
- ccbar
- taupair
- mupair
- bhabha
- photon

MC sample fileの在処 in KEKCC:  
**/ghi/fs01/belle2/bdata/MC/release-00-07-02/DBxxxxxxxx/MC7**

Note that the first production of these samples had an incorrect number of events for the mixed and charged samples, so each individual sample is not quite 1 ab<sup>-1</sup>, but the total is 2 ab<sup>-1</sup>. Similarly, the first set of some of the continuum samples were generated with a truncated precision, so the number of events for the two samples are not quite the same, but the total number for 2 ab<sup>-1</sup> is correct.

\*\*\*All file locations have a base of /ghi/fs01/belle2/bdata/MC/release-00-07-02/DBxxxxxxxx/MC7

Key: All jobs submitted Production finished (including merge steps) Transferred to KEKCC Not produced

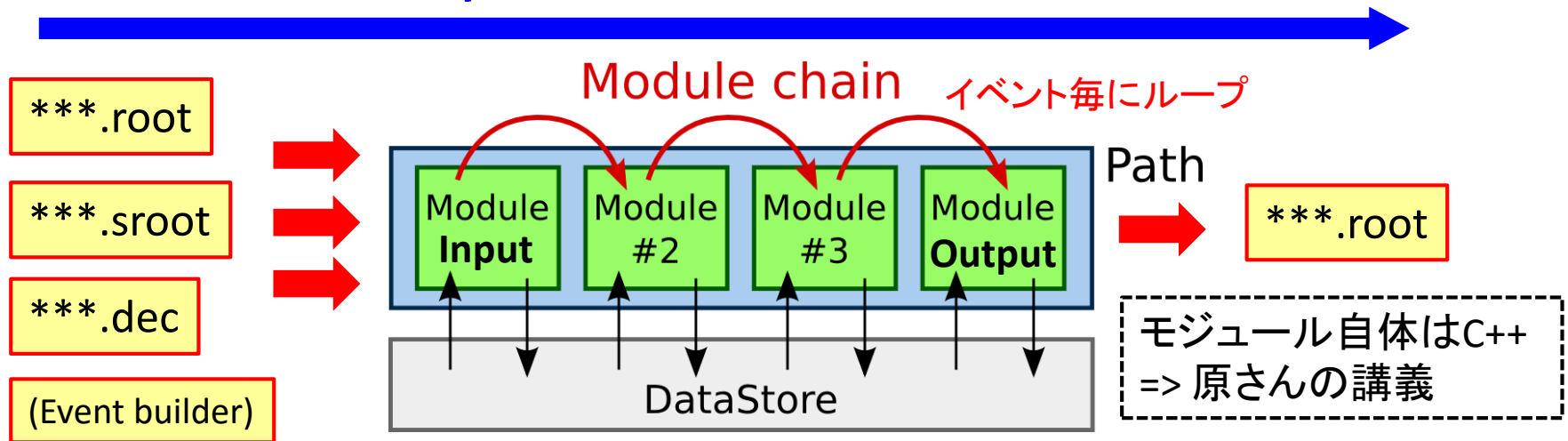
mixed

Sample type	Production ID	BG level	Expected Events (10 <sup>6</sup> )	Data Block	Files	Events/block (10 <sup>6</sup> )	Location at KEKCC local storage ***
mixed	00000786	BGx1	452 (1.06 ab <sup>-1</sup> )	sub00	1000	89.93	prod00000786/s00/e00000/4S/r00000/mixed/sub00

**basf2 scriptを使って物理解析**

# basf2 scriptで出来ること

Pythonスクリプトに工程を記述



basf2 = 解析ルーチンをモジュール化

- 実行スクリプト(python)にプログラムモジュールの組み合わせたパスを記述
  - モジュールの実行順
  - モジュールごとのパラメータ設定
  - パスの分岐
- Input モジュールに始まり、Outputモジュールで終わる
  - 今回はInput: Mass pro MC fileを読んでOutput: ntupleに出力することが目標

パスを作るまでの設定  
だけでも実はかなり大変



# モジュールを調べる

- 利用可能なモジュールの一覧を取得することができます
- モジュール毎の詳細も調べることができます

```
[tkonno@cw05 ~]$ basf2 --modules
```

```
.....
```

```
.....
```

```
EvtGenInput      .....
```

```
....
```

```
[tkonno@cw05 ~]$ basf2 --modules EvtGenInput
```

```
=====
```

```
    EvtGenInput
```

```
=====
```

```
Description: ...
```

```
Found in:     ...
```

```
Package:      generators
```

```
-----
```

Parameter	Type	Default	Description
-----------	------	---------	-------------

```
-----
```

DECFile	str	....	....
---------	-----	------	------

```
....
```

- すごくいっぱいある
- パラメータもいっぱい



全部を自力で設定するのは  
大変なので

**basf2 標準のラッパー関数**  
を使ってやりたいことだけを  
やります

# データファイルを読み込む

- **MC generic samples** を読み込んでみます : BGx1=> beam backgroundあり
  - <https://confluence.desy.de/display/BI/MC7+phase+III+-+Y%284S%29+generic+samples>
- sample01.py => 赤字の関数がラッパー関数

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from basf2 import *
from modularAnalysis import *

# input mdst file
inputMdst('default','/ghi/fs01/belle2/bdata/MC/release-00-07-02/DBxxxxxxxx/MC7/prod00000786/¥
s00/e0000/4S/r00000/mixed/sub00/mdst_000046_prod00000786_task00000046.root')

#-- start analysis routine --#
fillParticleList('pi+:all','')
fillParticleList('gamma:all','')

printVariableValues('pi+:all',['p'])
printVariableValues('gamma:all',['E'])
#-- end analysis routine --#

# process the events
process(analysis_main)

# print out the summary
print(statistics)
```

# データファイルを読み込む

- ラッパー関数の詳細は(細かい情報は足りない気がする...)

<https://confluence.desy.de/display/BI/Physics+AnalysisSteering>

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
```

```
from basf2 import *
from modularAnalysis import *   ラッパー関数をimport
```

```
# input mdst file
```

```
inputMdst('default','/ghi/fs01/belle2/bdata/MC/release-00-07-02/DBxxxxxxxx/MC7/prod00000786/¥
s00/e0000/4S/r00000/mixed/sub00/mdst_000046_prod00000786_task00000046.root')
```

```
#-- start analysis routine --#
```

```
fillParticleList('pi+:all','')
```

```
fillParticleList('gamma:all','')
```

```
printVariableValues('pi+:all',['p'])
```

```
printVariableValues('gamma:all',['E'])
```

```
#-- end analysis routine --#
```

```
# process the events
```

```
process(analysis_main)
```

```
# print out the summary
```

```
print(statistics)
```

## 1. 読み込むMCファイル選択

### 2-1. 再構成粒子の選別と選別条件

### 2-2. 選別粒子の物理量情報の表示設定

## 3. 解析プログラムを実行！

統計情報を表示

# 実行する(sample01)

- 全イベント処理すると時間がかかるので先頭10イベント分を見ます
  - "-n <nevt>" : <nevt>イベント分だけ実行

```
[tkonno@cw05 ~]$ basf2 sample01.py -n 10
....
[INFO] [ParticlePrinterModule] START -----
[INFO] ParticleList : gamma:all (0+28)
[INFO] - 20 = 22[0]
[INFO]   o) E = 0.126282
[INFO] - 21 = 22[1]
[INFO]   o) E = 0.328425
[INFO] - 22 = 22[2]
[INFO]   o) E = 0.0121352
[INFO] - 23 = 22[3]
[INFO]   o) E = 0.46734
[INFO] - 24 = 22[4]
[INFO]   o) E = 0.0953682
[INFO] - 25 = 22[5]
[INFO]   o) E = 0.0087153
...
[INFO] - 47 = 22[27]
[INFO]   o) E = 0.0621889
[INFO] [ParticlePrinterModule] END -----
```

`printVariableValues('gamma:all',['E'])`  
の部分

# 実行する(sample01)

- 先頭10イベント分をしてみる

```
...
[INFO] [ParticlePrinterModule] START -----
[INFO] ParticleLists: pi+:all (10+0) + pi-:all (10+0)
[INFO] - 0 = 211[0]
[INFO]   o) p = 0.710637
[INFO]   o) M = 0.13957
[INFO]   o) dr = 0.00609163
[INFO]   o) mcPDG = 211
[INFO]   o) piid = 1
[INFO] - 4 = 211[1]
[INFO]   o) p = 0.604811
[INFO]   o) M = 0.13957
[INFO]   o) dr = 0.00291447
[INFO]   o) mcPDG = -13
[INFO]   o) piid = 1
...
[INFO] - 18 = -211[19]
[INFO]   o) p = 0.250571
[INFO]   o) M = 0.13957
[INFO]   o) dr = 2.74259
[INFO]   o) mcPDG = -13
[INFO]   o) piid = 0.969363
[INFO] [ParticlePrinterModule] END -----
```

pion以外の粒子  
が混じっている

`printVariableValues('pi+:all',...`  
の部分

# 実行する(sample01)

- 最後に実行時の統計情報を表示
    - 実行されたモジュール毎に消費リソース(PC)
- => パフォーマンスチューニングに便利！

```
...
=====
Name                |      Calls | VMemory(MB) |      Time(s) |      Time(ms)/Call
=====
RootInput            |          10 |           0 |         0.00 |         0.14 +- 0.06
ProgressBar          |          10 |           0 |         0.00 |         0.01 +- 0.02
Gearbox              |          10 |           0 |         0.00 |         0.00 +- 0.00
Geometry             |          10 |           0 |         0.00 |         0.00 +- 0.00
ParticleLoader_pi+:all |          10 |           0 |         0.00 |         0.32 +- 0.50
ParticleLoader_gamma:all |          10 |           0 |         0.00 |         0.16 +- 0.20
ParticlePrinter_pi+:all |          10 |           0 |         0.00 |         0.30 +- 0.44
ParticlePrinter_gamma:all |          10 |           0 |         0.00 |         0.25 +- 0.04
=====
Total                |          10 |           0 |         0.01 |         1.21 +- 1.11
=====
```

- inputMDSTの指定はコマンドんのオプションからも出来ます
  - "-i <filepath>" : <filepath> に読み込み先のファイルパスを置き換え

```
$ basf2 -n 10 sample01.py -i <mdtsfile>
```

# ParticleLoader

- `fillParticleList('<particle>:<label>', '<cutCondition>')`の実体
    - `<particle>` : 粒子の種類 => e+/e-/mu+/mu-.....  
名前のルール=> ``cat $BELLE2_EXTERNALS_DIR/share/evtgen/evt.pdl``  
(name listのありかを初めて知りました...)
    - `<label>` : リストにつける任意のラベル(all, all, good等など)  
Predefined なラベル(+cut 条件)は↓のstdXXX.pyを参照  
<https://stash.desy.de/projects/B2/repos/software/browse/analysis/scripts>
    - `<cutCondition>` : 選別条件 => 難しい
  - 選別された粒子は**ParticleList**に格納
    - ParticleListとは: <https://confluence.desy.de/display/BI/Physics+ParticleList>  
=> 選別条件に従って集めた再構成粒子のリスト(イベント毎)
    - ('pi+:all', '') => 任意の荷電粒子
    - ('gamma:all', '') => 任意の光子
- 生成粒子を全部取ってくるときに重宝

# 変数の取扱い

- Paritcleから物理量を取り出す => 物理量を表す変数名が必要！
  - カット条件
  - 再構成条件
- 詳細はVariable Manager/Particle Selector Functionsを参照
  - <https://confluence.desy.de/display/BI/Physics+VariableManager>

(本文抜粋)

The Variables registered in the [VariableManager](#) can be used inside C++ (namespace Variable)

=>変数名を知りたいければ「コードを読め」と。調べるスクリプトもあるようです。

```
[tkonno@cw06 ~]$ basf2 ${BELLE2_RELEASE_DIR}/analysis/scripts/variables.py
```

- <https://confluence.desy.de/display/BI/Physics+ParticleSelectorFunctions>
- よく使うもの:

Particle毎:

- M : the invariant mass
- E : the total energy
- p : the total momentum
- px : the x component of the momentum
- Mbc : the beam-constrained mass

Event毎

- nTracks : the number of tracks
- evtNum : the event number
- runNum : the run number
- expNum : the experiment number



# Cutを入れる

- sample02.py (※変更部分を抜粋)

```
##-- start analysis routine --#
fillParticleList('pi+:all','')
fillParticleList('gamma:all','')

applyCuts('gamma:all','E>0.3') #1
cutAndCopyList('pi+:good','pi+:all','piid>0.1') #2
applyCuts('pi+:all','Kid>0.1') #3

printVariableValues('gamma:all',['E'])
printVariableValues('pi+:good',['p','M','dr','piid','Kid','mcPDG'])
printVariableValues('pi+:all',['p','M','dr','piid','Kid','mcPDG'])
##-- end analysis routine --#
```

# Cutを入れる

- sample02.py (※変更部分を抜粋)

```
-- start analysis routine --#
fillParticleList('pi+:all','')
fillParticleList('gamma:all','')
```

```
applyCuts('gamma:all','E>0.3') #1
cutAndCopyList('pi+:good','pi+:all','piid>0.1') #2
applyCuts('pi+:all','Kid>0.1') #3
```

```
printVariableValues('gamma:all',['E'])
printVariableValues('pi+:good',['p','M','dr','piid','Kid','mcPDG'])
printVariableValues('pi+:all',['p','M','dr','piid','Kid','mcPDG'])
-- end analysis routine --#
```

1.  $E > 0.3$  のphotonを残す
2.  $PID(\pi) > 0.1$  の $\pi$ をpi+:goodにコピー
3.  $PID(K) > 0.1$  の $\pi$  (?)を残す  
=> 荷電粒子からKを選び出す

- copyListでParticleを変えるエラーになる

```
-- start analysis routine --#
fillParticleList('pi+:all','')
cutAndCopyList('K+:good','pi+:all','Kid>0.1')
printVariableValues('K+:good',['p'])
-- end analysis routine --#
```

ERROR

```
-- start analysis routine --#
fillParticleList('pi+:all','')
fillParticleList('K+:all','')
cutAndCopyList('K+:good','K+:all','Kid>0.1')
printVariableValues('K+:good',['p'])
-- end analysis routine --#
```

OK

# 実行する

- 先頭10イベント分を実行します

```
[tkonno@cw05 ~]$ basf2 -n 10 sample02.py
```

```
....
```

```
[INFO] [ParticlePrinterModule] START -----
```

```
[INFO] ParticleList : gamma:all (0+4)
```

```
[INFO] - 21 = 22[0]
```

```
[INFO]      o) E = 0.328425
```

```
[INFO] - 23 = 22[1]
```

```
[INFO]      o) E = 0.46734
```

```
[INFO] - 31 = 22[2]
```

```
[INFO]      o) E = 0.680739
```

```
[INFO] - 37 = 22[3]
```

```
[INFO]      o) E = 0.322397
```

```
[INFO] [ParticlePrinterModule] END -----
```

`printVariableValues('gamma:all',['E'])`

の部分

=> だいぶ減っている

# 実行する(続き)

- sample01.pyは先頭10イベント分しか実行しません

```
...
[INFO] [ParticlePrinterModule] START -----
[INFO] ParticleLists: pi+:good (8+0) + pi-:good (9+0)
[INFO] - 0 = 211[0]
[INFO]   o) p = 0.710637
[INFO]   o) M = 0.13957
[INFO]   o) dr = 0.00609163
[INFO]   o) piid = 1
[INFO]   o) Kid = 6.44191e-28
[INFO]   o) mcPDG = 211
...
[INFO] - 18 = -211[16]
[INFO]   o) p = 0.250571
[INFO]   o) M = 0.13957
[INFO]   o) dr = 2.74259
[INFO]   o) piid = 0.969363
[INFO]   o) Kid = 0.0306374
[INFO]   o) mcPDG = -13
[INFO] [ParticlePrinterModule] END -----
```

`printVariableValues('pi+:good',...`  
の部分  
pion以外の粒子がまだ混じっている

# Ntupleに書き出す

basf2は開始までの処理が長い => ntupleに落としていじくりたい

- sample03.py (※抜粋)

```
#-- start analysis routine --#
fillParticleList('pi+:all','')
fillParticleList('gamma:all','')

ntupleFile('sample03.root') #1

tools = ['EventMetaData', 'gamma']
tools += ['Kinematics', '^gamma']
tools += ['MCKinematics', '^gamma']
tools += ['MCTruth', '^gamma']
tools += ['Cluster', '^gamma']
tools += ['CustomFloats[goodGamma]', '^gamma']
ntupleTree('photon', 'gamma:all', tools) #2-1

tools = ['EventMetaData', 'pi+']
tools += ['Kinematics', '^pi+']
tools += ['Track', '^pi+']
tools += ['PID', '^pi+']
tools += ['Charge', '^pi+']
tools += ['MCKinematics', '^pi+']
tools += ['MCTruth', '^pi+']
tools += ['CustomFloats[dr]', '^pi+']
ntupleTree('pion', 'pi+:all', tools) #2-2
#-- end analysis routine --#
print(statistics)
```

1. 書き出すroot fileを設定

2-1. 光子のtree: photon

2-2.  $\pi$ のtree : pion

# Ntupleに書き出す

- `ntupleFile('<filename>')` : ntuple (TTree)をrootファイルに出力
  - <filename> : ROOTファイルパス
- `ntupleFile('<treename>', '<particlelist>', <tools>)`
  - <treename> : tree の名前
    - 'gamma'などは内部で名称がかぶるようでバグります
  - <particlelist> : Particle List名 => filleParticleListなどで作ったリスト名を渡す
  - <tools> : Ntupleにbranchを作るツールを指定する
    - <https://confluence.desy.de/display/BI/Physics+NtupleTool>  
=> branch名がcutなどで使う名前から微妙に変わるので紛らわしい...
    - EventMetaData : experiment #, run #, event #
    - Kinematics : 実験室系の4次元運動量 (P, P4)
    - InvMass : Invariant mass
    - PID : PIDk, PIDpi,... => Kid, piid,...
    - MCTruth: MC truth 情報
    - CustomFloats[varname]: cut変数を直接指定できる

# 実行する(sample03)

- 出来たroot fileを覗いてみる

```
$ basf2 -n 100 sample03.py
...
$ root -l sample03.root
..
root [1] .ls
TFile**          sample03.root
TFile*           sample03.root
KEY: TTree       photon;6
KEY: TTree       photon;5
KEY: TTree       pion;3
KEY: TTree       pion;2
```

photonとpionは  
別々のtreeに出力



イベントの同期に  
EventMetaData(evt\_no)  
が必要

```
root [2] pi->Print()
*Br   0 :exp_no      : exp_no/I      *
*Br   1 :run_no      : run_no/I      *
*Br   2 :evt_no      : evt_no/I      *
*Br   3 :pi_P        : pi_P/F        *
*Br   4 :pi_P4       : pi_P4[4]/F    *
*Br   5 :pi_d0       : pi_d0/F        *
*Br   6 :pi_z0       : pi_z0/F        *
*Br   7 :pi_TrPval   : pi_TrPval/F    *
*Br   8 :pi_PIDk     : pi_PIDk/F      *
*Br   9 :pi_PIDpi    : pi_PIDpi/F     *
*Br  10 :pi_PIDe     : pi_PIDe/F      *
*Br  11 :pi_PIDmu    : pi_PIDmu/F     *
*Br  12 :pi_PIDp     : pi_PIDp/F      *
*Br  13 :pi_charge   : pi_charge/I     *
*Br  14 :pi_TruthP   : pi_TruthP/F    *
*Br  15 :pi_TruthP4  : pi_TruthP4[4]/F *
*Br  16 :pi_TruthM   : pi_TruthM/F     *
*Br  17 :pi_mcPDG    : pi_mcPDG/I     *
*Br  18 :pi_mcErrors : pi_mcErrors/I  *
*Br  19 :pi__dr      : pi__dr/F       *
*Br  20 :nCands      : m_nCands/I     *
*Br  21 :iCand       : m_iCand/I      *
```

# 実行する(sample03)

- 出来たroot fileを覗いてみる

```
root [3] photon->Print()  
*Br   0 :exp_no      : exp_no/I          *  
*Br   1 :run_no      : run_no/I          *  
*Br   2 :evt_no      : evt_no/I          *  
*Br   3 :gamma_P      : gamma_P/F         *  
*Br   4 :gamma_P4     : gamma_P4[4]/F     *  
*Br   5 :gamma_TruthP  : gamma_TruthP/F    *  
*Br   6 :gamma_TruthP4 : gamma_TruthP4[4]/F *  
*Br   7 :gamma_TruthM  : gamma_TruthM/F    *  
*Br   8 :gamma_mcPDG   : gamma_mcPDG/I     *  
*Br   9 :gamma_mcErrors : gamma_mcErrors/I *  
*Br  10 :gamma_clusterReg : gamma_clusterReg/I *  
*Br  11 :gamma_clusterE9E25 : gamma_clusterE9E25/F *  
*Br  12 :gamma_clusterNHits : gamma_clusterNHits/I *  
*Br  13 :gamma_clusterTrackMatch : gamma_clusterTrackMatch/I *  
*Br  14 :gamma_clusterUncorrE : gamma_clusterUncorrE/F *  
*Br  15 :gamma_clusterHighE : gamma_clusterHighE/F *  
*Br  16 :gamma_clusterTiming : gamma_clusterTiming/F *  
*Br  17 :gamma_clusterTheta : gamma_clusterTheta/F *  
*Br  18 :gamma_clusterPhi  : gamma_clusterPhi/F *  
*Br  19 :gamma_clusterR    : gamma_clusterR/F *  
*Br  20 :nCands          : m_nCands/I     *  
*Br  21 :iCand           : m_iCand/I      *
```



# Decay reconstruction

$B^0 \rightarrow J/\psi K_S^0$  のsignal サンプルを使ってB0再構成をします。

<https://confluence.desy.de/display/BI/MC7+phase+III+-+Y%284S%29+signal+samples>

- sample04.py
  - Backgroundは何も考慮していません

```
# input mdst file
inputMdstList('default', ['/ghi/fs01/belle2/bdata/MC/release-00-07-02/DBxxxxxxx/MC7/prod00000223/¥
s00/e0000/4S/r00000/signal/sub00/mdst_000001_prod00000223_task00000001.root',
                    '/ghi/fs01/belle2/bdata/MC/release-00-07-02/DBxxxxxxx/MC7/prod00000627/¥
s00/e0000/4S/r00000/signal/sub00/mdst_000001_prod00000627_task00000001.root'])

#-- start analysis routine --#
fillParticleList('mu+:all', 'eid > 0.1 and chiProb > 0.01')
fillParticleList('e+:all', 'muid > 0.1 and chiProb > 0.01')
fillParticleList('K_S0+:all', '0.3 < M < 0.7')

reconstructDecay('J/psi:ee -> e+:all e-:all', '2.7 < M < 3.2', 1)

reconstructDecay('J/psi:mm -> mu+:all mu-:all', '2.7 < M < 3.2', 2)

copyLists('J/psi:ll', ['J/psi:ee', 'J/psi:mm'])

reconstructDecay('B0:jpsiks -> J/psi:ll K_S0:all', 'M > 5.2 and abs(deltaE)<0.25')
```

再構成の対応

$$K_S^0 \rightarrow \pi^+ \pi^-$$

$$J/\psi \rightarrow e^+ e^-$$

$$J/\psi \rightarrow \mu^+ \mu^-$$

$$B^0 \rightarrow J/\psi K_S^0$$

# Decay reconstruction

- sample04.py のつづき

```
matchMCTruth('J/psi:11')
matchMCTruth('B0:jpsiks')

ntupleFile('sample04.root')

toolsB = ['EventMetaData', '^B0']
toolsB += ['InvMass', '^B0 -> ^J/psi ^K_S0']
toolsB += ['Kinematics', '^B0 -> ^J/psi ^K_S0']
toolsB += ['Track', '^B0 -> ^J/psi ^K_S0']
toolsB += ['DeltaEMbc', '^B0']
toolsB += ['MCTruth', '^B0 -> ^J/psi ^K_S0']
ntupleTree('b0', 'B0:jpsiks', toolsB)
```

- matchMCTruth('<listname>')
    - MC truthと再構成結果を関連付ける
  - ^B0 -> ^J/psi ^K\_S0
    - '^'が付いた粒子の物理量を保存する
    - 再構成に使われた粒子の情報が同じエントリに保持される
- => InvMassならB0\_M, B0\_Jpsi\_M, B0\_K\_S0i\_Mが作られる

# 実行する(sample04)

- スクリプトを実行して出来たroot fileを覗いてみる

```
[tkonno@cw11 ~]$ basf2 -n 100 sample04.py
...
[tkonno@cw06 ~]$ root -l sample04.root
root [1] b0->Scan("evt_no:B0_mbc")
*****
*      Row      *      evt_no *      B0_mbc *
*****
*          0 *      700001 * 5.2808809 *
*          1 *      700002 * 5.2809023 *
*          2 *      700003 * 5.2754497 *
*          3 *      700004 * 5.2757582 *
*          4 *      700014 * 5.2723460 *
*          5 *      700017 * 5.2832436 *
*          6 *      700021 * 5.2775831 *
*          7 *      700025 * 5.2669687 *
*          8 *      700034 * 5.2802038 *
*          9 *      700039 * 5.2661099 *
*         10 *      700039 * 5.2811393 *
...

```

- 同じイベント内に複数のB0が再構成されている
- 3個以上再構成されないはず



- Miss reco.を減らす
- **B0候補をランク付け**

# Decay candidateをソートする

- `rankByLowest('<particlelist>', '<variable>', '<ncandidates>', '<label>')`
  - `<variable>`が小さい順に`<particlelist>`をソート
  - `<ncandidates> = 0` => candidate を全て保持
  - `<label>` : ソート順のインデックス
- カット変数に種々の操作を行う
  - `daughter(0/1, SigM)` : 娘粒子の物理量=Invariant massのズレ
  - `abs(V)` : 絶対値
  - `formula(v0 + v1)` : 四則演算
- 先程のスクリプトに数行追加 : `sample04b.py`

```
variables.addAlias('myRank', 'extraInfo(myRank)')

# input mdst file
...
reconstructDecay('B0:jpsiks -> J/psi:ll K_S0:all', 'Mbc > 5.2 and abs(deltaE) < 0.250')
rankByLowest('B0:jpsiks', 'formula(abs(daughter(0, SigM)) + abs(daughter(1, SigM)))', 0, 'myRank')

...

toolsB += ['CustomFloats[myRank]', '^B0']
ntupleTree('b0', 'B0:jpsiks', toolsB)
```

# 実行する(sample04b)

- 出来たroot fileを覗いてみる

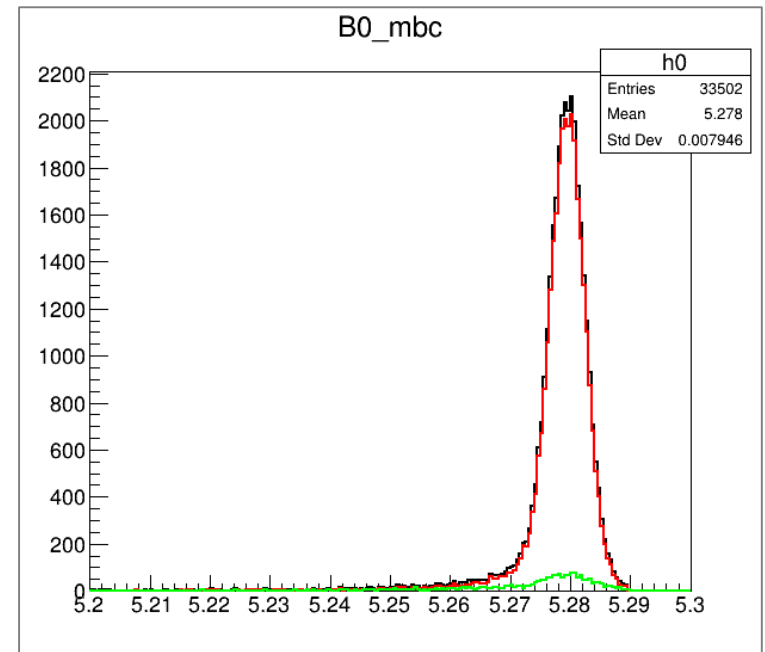
```
[tkonno@cw11 ~]$ basf2 sample04b.py

[tkonno@cw11 ~]$ root -l sample04b.root
root [0]
Attaching file sample04b.root as _file0...
(TFile *) 0x3692d80
root [1] b0->Scan("evt_no:B0_mbc:B0__myRank")
*****
*      Row      *      evt_no *      B0_mbc * B0__myRan *
*****
*          0 *      700001 *  5.2808809 *          1 *
*          1 *      700002 *  5.2809023 *          1 *
*          2 *      700003 *  5.2754497 *          1 *
*          3 *      700004 *  5.2757582 *          1 *
*          4 *      700014 *  5.2723460 *          1 *
*          5 *      700017 *  5.2832436 *          1 *
*          6 *      700021 *  5.2775831 *          1 *
*          7 *      700025 *  5.2669687 *          1 *
*          8 *      700034 *  5.2802038 *          1 *
*          9 *      700039 *  5.2661099 *          1 *
*         10 *      700039 *  5.2811393 *          2 *
...

```

```
root [3] b0->Draw("B0_mbc>>h0", "")
root [4] b0->Draw("B0_mbc>>h1", "B0_myRank==1", "same")
root [5] b0->Draw("B0_mbc>>h2", "B0__myRank>1", "same")

```



B0のBeam constraint mass (GeV)

# まとめ

- basf2 scriptを使ってデータファイルを読み込む手順を紹介しました
  - KEKCCの解析環境は一通りあり自分で用意する必要はありません
  - C++コンパイラがなくても解析できるのはすごいなと思いました
  - ドキュメントもかなり整備されています

<https://confluence.desy.de/display/BI/Physics+AnalysisSoftware>
- basf2 scriptは実行時の初期設定(データベース読み込みなど)が長い
  - 1度ntupleに落としてカット条件など解析の方針を決めてから再度 basf2 scriptに戻すのが現実的なアプローチではないかと思います
- 今日はPython scriptからできる多彩な機能の殆どを紹介できませんでした
  - ForEach / RestOfEvent
  - HowToVeto / SkimFilter
- 過去のものも含めチュートリアルをやってみると良いと思います
  - <https://confluence.desy.de/display/BI/Physics+HandsOnAnalysisTutorialJune2017>
  - 結局よくわからんからソースコードを読むことになるかも...
- 端末からbasf2を動かせる様になったらGRIDに投げてみましょう=>早坂さん
- 既存のモジュールに限界を感じたら自力で作ってください=>原さん

# basf2 release versionを探すには

- 利用可能なrelease versionの一覧を取得することが出来ます
  - release : 安定版(通常の解析はreleaseを使う)
  - prerelease : プレリリース版
  - build : コンパル済みの開発版

```
[tkonno@cw06 ~]$ setuprel --help
~
Usage: setuprel [release]
~
The following releases are available:
~
    prerelease-00-08-00b
    release-00-08-00
~
    prerelease-00-09-00a
    prerelease-00-09-00b
    build-2017-06-14
```

- 自分でコンパイルしたいという人は:  
<https://confluence.desy.de/display/BI/Software+SoftwareInstallation>