Data analysis at CMS
-- Exotic hadron study as an example

Jingqing Zhang

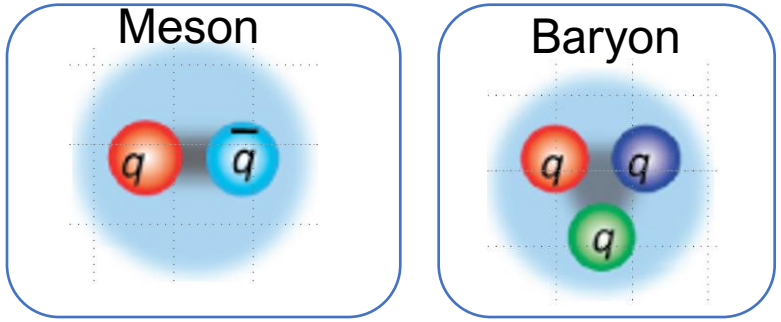
Nanjing Normal University

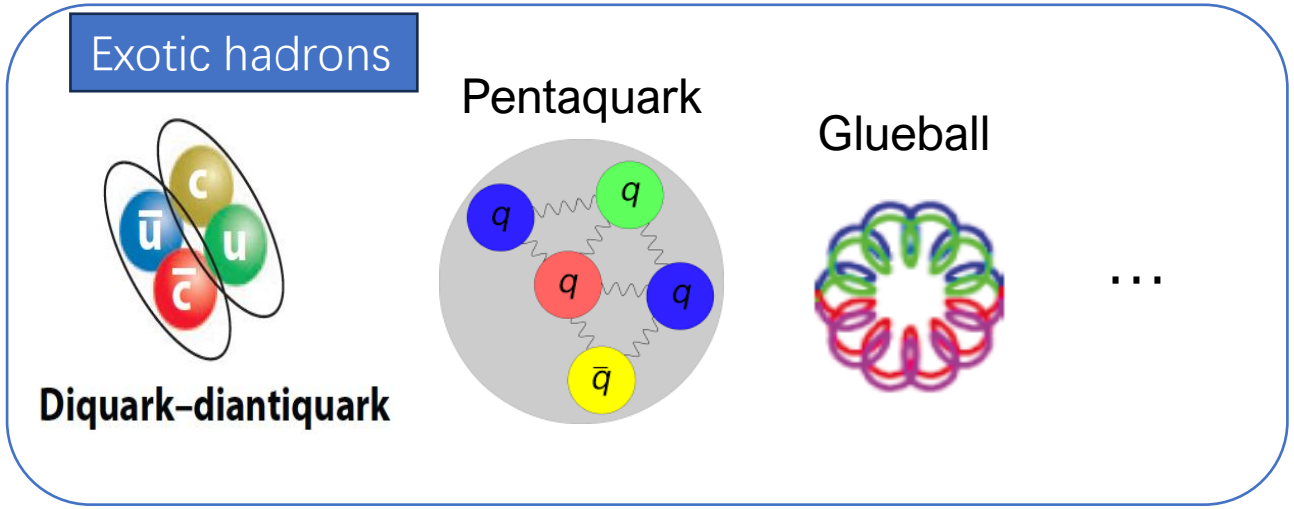
Outline

- Motivation of the analysis
- Steps of an analysis: from scratch to the end
- Dataset to ntuple
  - Analyzer to make ntuple
- Analyze ntuple
  - Background
  - Fit the mass spectrum
  - Significance

Motivation

- What is the exotic hadron





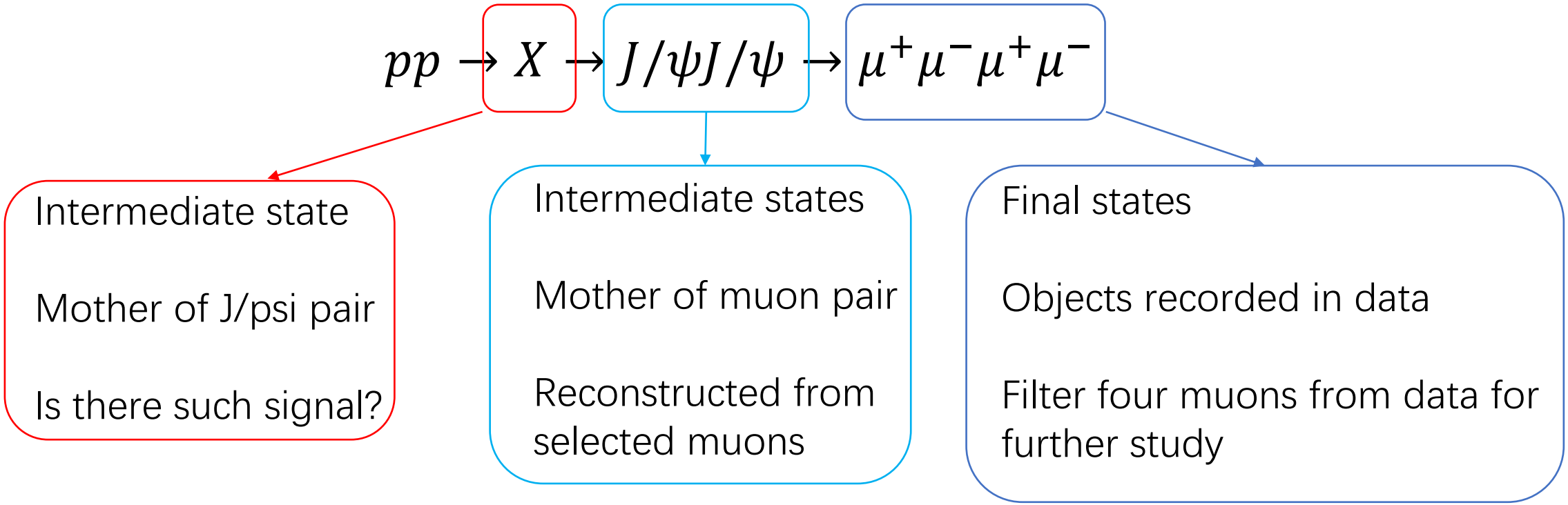
- Why we want to study exotic hadron
- Why at CMS
  - CMS can provide data with high production energy
  - And can provide more spin possibility than ee collider

What channel we want to study

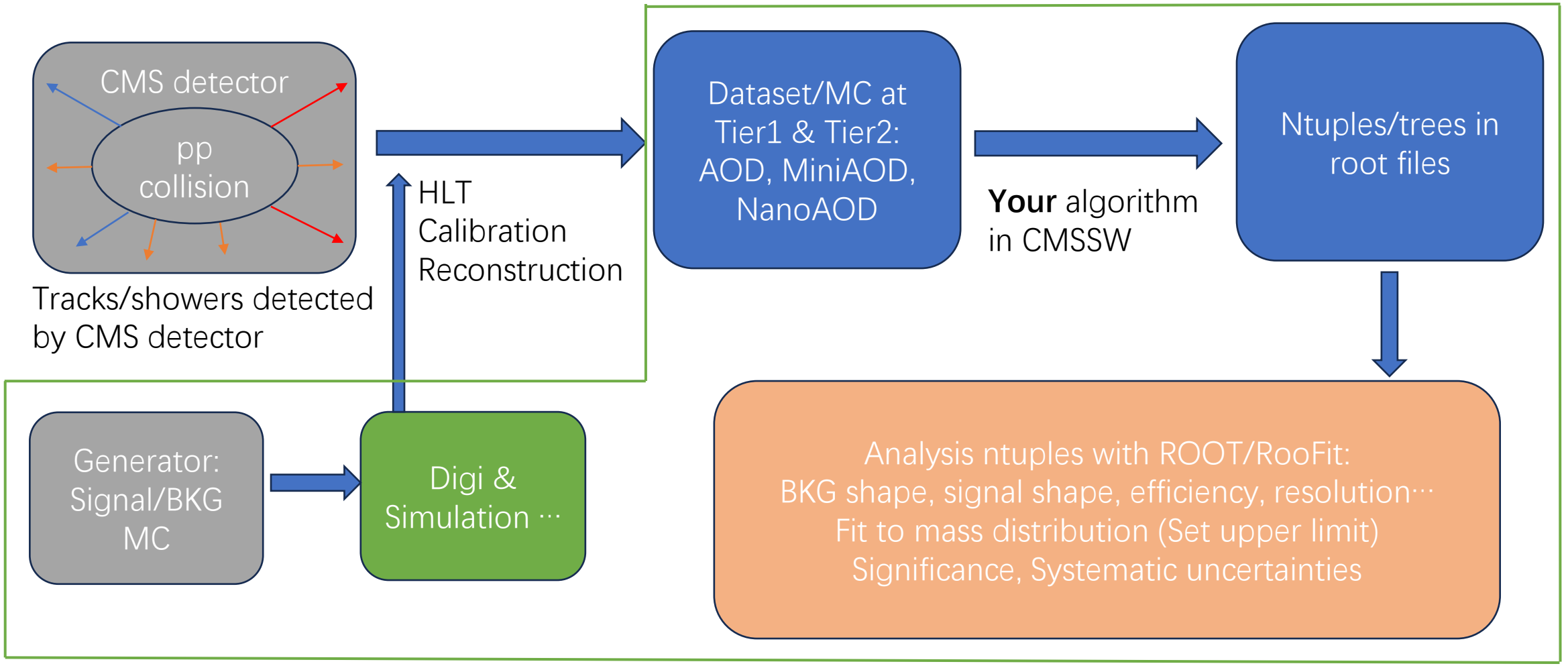
- J/ψ can be easily tagged/reconstructed
- Resonances in $J/\psi J/\psi$ most likely contains at least 4 charm quarks
  - Otherwise decays to $J/\psi J/\psi$ will be suppressed
- Study the channel: $J/\psi J/\psi \rightarrow \mu^+\mu^-\mu^+\mu^-$

How to do this analysis from scratch

- We now have chosen the channel:

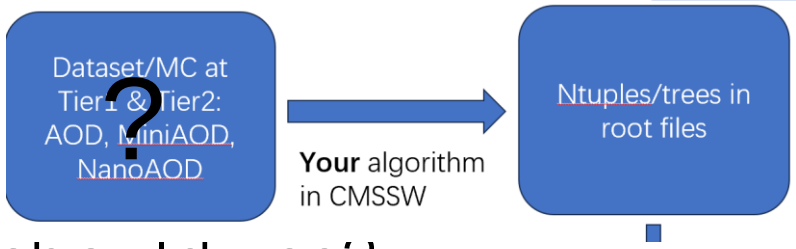


How to do the analysis from scratch

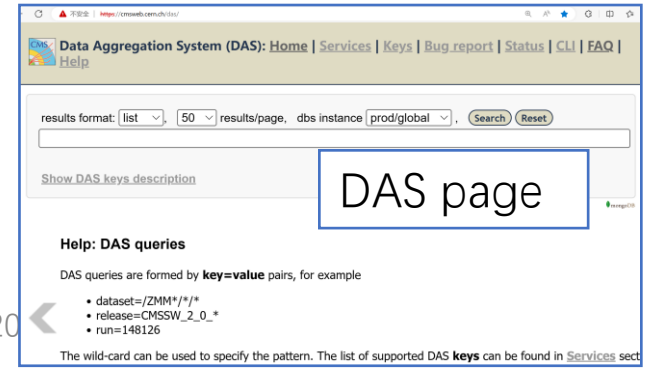


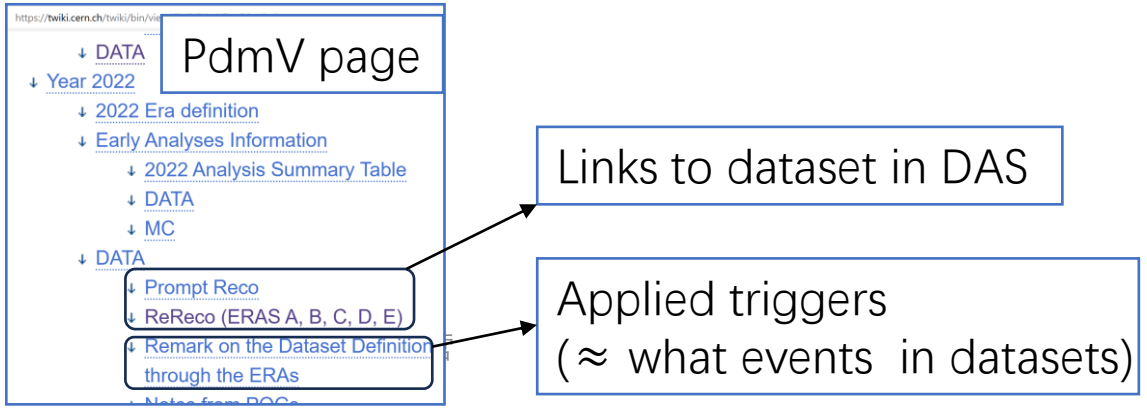
What/where is the data

- Our channel: $pp \rightarrow X \rightarrow J/\psi J/\psi \rightarrow \mu^+\mu^-\mu^+\mu^-$



- What data we should use?
  - Unfortunately, no webpage which contains all datasets and their triggers
  - Ask supervisor, colleagues, convenors
  - Lookup in DAS: CMS Data Aggregation Service (cern.ch)
  - Explore in PdmV: PdmVRun3Analysis < CMS < TWiki (cern.ch)





What/where is the data

- From PdmV:

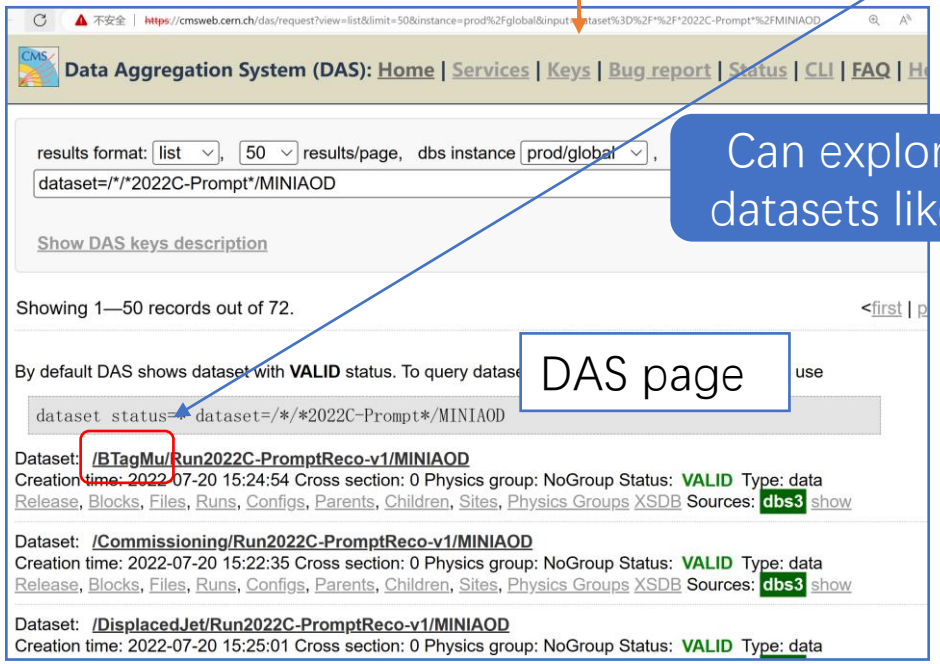
Prompt Reco

ERA	MINIAOD	NANOAOD
C	DAS	DAS
D	DAS	DAS
E	DAS	DAS
F	DAS	DAS
G	DAS	DAS

ERAs	!HLT menu	PDs set
A		
B		
C	v1.2.0, and v1.2.5	ZeroBias, MinimumBias, EGamma, BTagMu, DisplacedJet, JetHT, MET, Tau, DoubleMuon, MuonEG, SingleMuon, ScoutingPFMonitor, ScoutingPFRun3, ParkingBPH [1-5], ParkingDoubleElectronLowMass [0-5], ParkingDoubleMuonLowMass [0-7]
D	v1.2.5	ZeroBias, MinimumBias, EGamma, BTagMu

HLT Menu

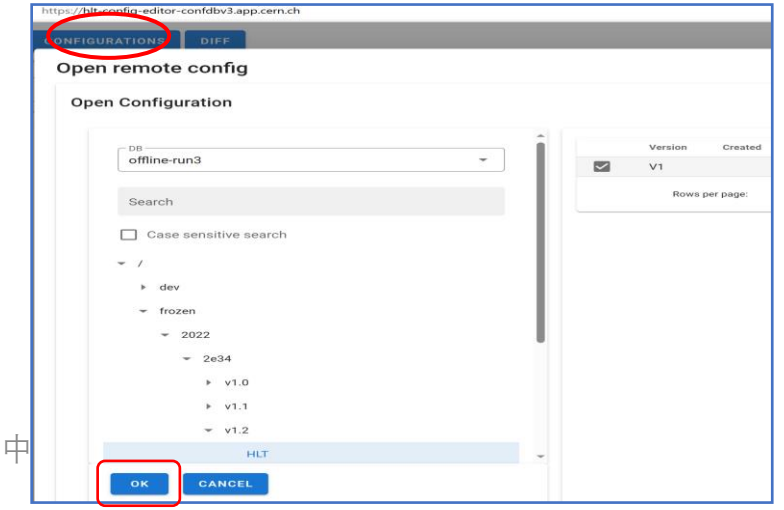
Primary name of dataset



Can explore all datasets like this

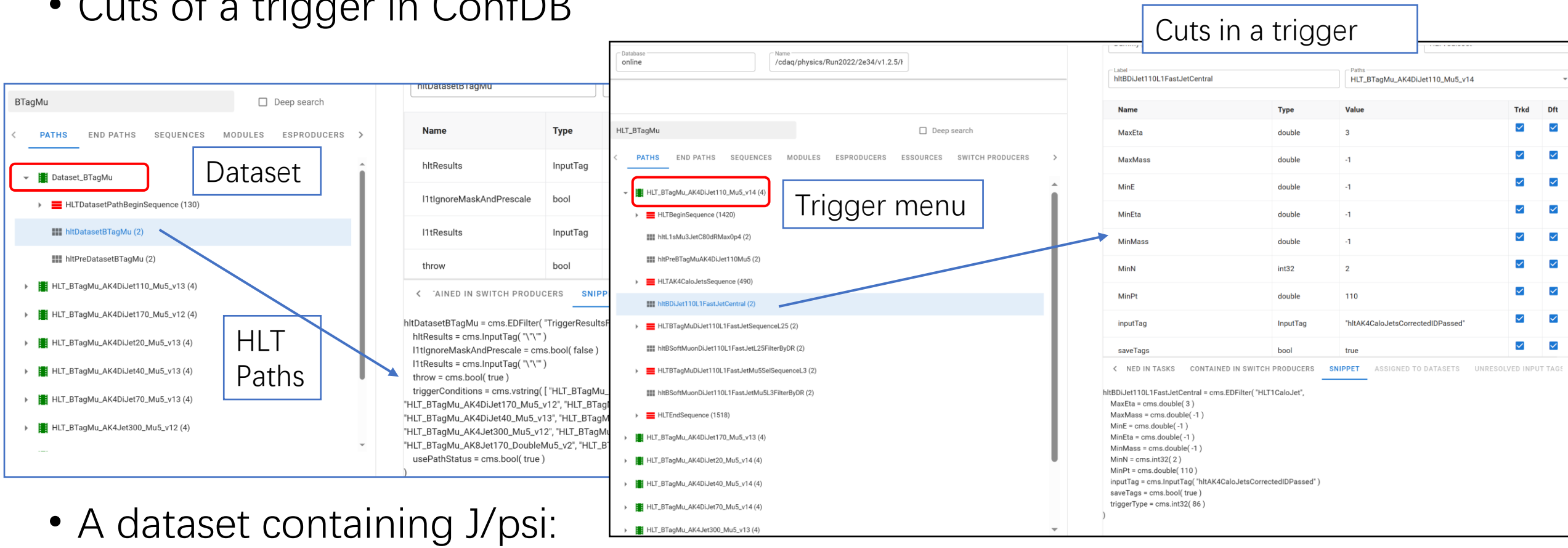
DAS page

Then check the triggers/cuts of a primary dataset in confDB: ConfDB < CMS < TWiki (cern.ch)
→ ConfDB Web interface



What/Where is the data

- Trigger menus of a dataset in ConfDB
- Cuts of a trigger in ConfDB



- A dataset containing J/psi:

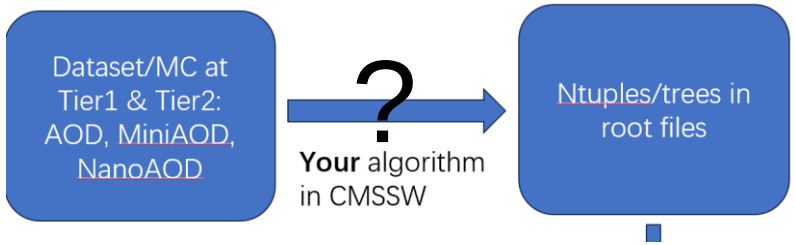
/ParkingDoubleMuonLowMass0/Run2022C-PromptReco-v1/MINIAOD **Exercise: how many trigger paths containing J/psi for this dataset?**

Exercise: how many trigger paths containing J/psi for this dataset?

- ConfDB Web interface → Configurations → Open remote →

From dataset to ntuple

- Signal channel: $pp \rightarrow X \rightarrow J/\psi J/\psi \rightarrow \mu^+\mu^-\mu^+\mu^-$
- Dataset: /ParkingDoubleMuonLowMass0/Run2022C-PromptReco-v1/MINIAOD



- Write an analyzer, analyze data and produce ntuples: WorkBookWriteFrameworkModule < CMSPublic < TWiki (cern.ch)
- Setup env

```
csh or tcsh users:
  source /cvmfs/cms.cern.ch/cmsset_default.csh
bash users:
  source /cvmfs/cms.cern.ch/cmsset_default.sh
```

- Setup CMSSW

```
# create a new project area
cmsrel CMSSW_12_0_0

cd CMSSW_12_0_0/src/
cmsenv
```

Write an analyzer

- Create an analyzer package

Write a Framework Module

First, create a subsystem area. The actual name used for the directory is not important, we'll use `Demo`. From the `src` directory, make and change to the `Demo` area:

```
mkdir Demo
cd Demo
```

Note that if you do not create the subsystem area and create you module directly under the `src` directory, your code will not compile. Create the "skeleton" of an EDAnalyzer module (see SWGuideSkeletonCodeGenerator for more information):

```
mkedanlzr DemoAnalyzer
```

Compile the code:

```
cd DemoAnalyzer
scram b
```

- Create a ConfFile_cfg.py in DemoAnalyzer/python, example link
- Run the job

```
cd into the DemoAnalyzer/python directory and do:
cmsRun ConfFile_cfg.py
```

Write analyzer

- Analyzer module is executed to filter data and make ntuple

cmsRun ConfFile_cfg.py

```
import FWCore.ParameterSet.Config as cms

process = cms.Process("Demo")

process.load("FWCore.MessageService.MessageLogger_cfi")

process.maxEvents = cms.untracked.PSet( input = cms.untracked.int32(-1) )

process.source = cms.Source("PoolSource",
                            # replace 'myfile.root' with the source file you want to use
                            fileNames = cms.untracked.vstring(
        'file:/afs/cern.ch/cms/Tutorials/workbook_twiki2021/MinBias_pythia8_14TeV_100events.root'
                            )

process.demo = cms.EDAnalyzer('DemoAnalyzer',
    tracks    = cms.untracked.InputTag('generalTracks')
                              )

process.p = cms.Path(process.demo)
```

Input data files

The analyzer module/package

To run the whole dataset, you need run crab job instead of running it locally.

This config file then is an input of the crab job config file, which uses the dataset name to identify data

- **Exercise: create a DemoAnalyzer following above instruction or the link**

Write an analyzer

- Write codes in the DemoAnalyzer then compile and cmsRun
- Save a histogram and a tree in output
- DemoAnalyzer/plugins/DemAnalyzer.cc

1 header files:

```
34 ////added to use service and save ntuple/histograms
35 #include "FWCore/ServiceRegistry/interface/Service.h"
36 #include "CommonTools/UtilAlgos/interface/TFileService.h"
37 #include "TH1.h"
38 #include "TTree.h"
39 #include <vector>
40 ////
```

1

2 Member data in class definition:

```
65   // ----------member data ---------------------------
66   edm::EDGetTokenT<TrackCollection> tracksToken_;
   configuration file
67   TH1F *h_pt; //to save in file
68   TTree *tree1;
69   unsigned int runNum;
70   unsigned int lumiNum;
71   unsigned int eventNum;
72   int nTrack;
     std::vector<double> *vec_pt;
     std::vector<double> *vec_px;
     std::vector<double> *vec_py;
     std::vector<double> *vec_pz;
```

2

3 Constructor

```
94 DemoAnalyzer::DemoAnalyzer(const edm::ParameterSet& iConfig)
95     : tracksToken_(consumes<TrackCollection>(iConfig.getUntrackedParameter<edm::InputTag>("tr
   acks"))), runNum(0), lumiNum(0), eventNum(0), nTrack(0), vec_pt(0), vec_px(0), vec_py(0), vec
   _pz(0) {
96   //to save histogrms/ntuples
97   edm::Service<TFileService> fs;
98   h_pt = fs->make<TH1F>("h_pt", "h_pt", 100, 0, 100);
99   tree1 = fs->make<TTree>("tree1", "my tree 1");
100  tree1->Branch("runNum", &runNum, "runNum/i");
101  tree1->Branch("lumiNum", &lumiNum, "lumiNum/i");
102  tree1->Branch("eventNum", &eventNum, "eventNum/i");
103  tree1->Branch("nTrack", &nTrack, "nTrack/I");
104  tree1->Branch("vec_pt", &vec_pt);
105  tree1->Branch("vec_px", &vec_px);
106  tree1->Branch("vec_py", &vec_py);
107  tree1->Branch("vec_pz", &vec_pz);
108 #ifdef THIS_IS_AN_EVENTSETUP_EXAMPLE
```

3

```
root [2] demo->cd()
(bool) true
root [3] .ls
TDirectoryFile*         demo     demo
 KEY: TH1F      h_pt;1  h_pt
 KEY: TTree     tree1;1 my tree 1
root [4]
```

Example at: /publicfs/cms/user/zhangjq/cms2024sysu/CMSSW_12_2_0/src/Demo/DemoAnalyzer

Write an analyzer

4 fill histograms and trees in analyze()

5 Modify DemoAnayzer/plugins/BuildFile.xml

```
126 void DemoAnalyzer::analyze(const edm::Event& iEvent, const edm::EventSetup& iSetup) {
127   using namespace edm;
128
129   ////event level
130   runNum = iEvent.id().run();
131   lumiNum = iEvent.id().luminosityBlock();
132   eventNum = iEvent.id().event();
133   nTrack = 0;
134   for (const auto& track : iEvent.get(tracksToken_)) {
135     // do something with track parameters, e.g, plot the charge.
136     // int charge = track.charge();
137     if (track.charge() < 0) continue;
138     nTrack++;
139     ////track/candidate level
140     h_pt->Fill(track.pt());
141     vec_pt->push_back(track.pt());
142     vec_px->push_back(track.px());
143     vec_py->push_back(track.py());
144     vec_pz->push_back(track.pz());
145   }
146   tree1->Fill();
147   //need to clear the vector after this event/candidate is fill to the tree
148   runNum = 0;
149   lumiNum = 0;
150   eventNum = 0;
151   nTrack = 0;
152   vec_pt->clear();
153   vec_px->clear();
154   vec_py->clear();
155   vec_pz->clear();
```

4

```
1 <use name="DataFormats/TrackReco"/>
2 <use name="FWCore/Framework"/>
3 <use name="FWCore/ParameterSet"/>
4 <use name="FWCore/PluginManager"/>
5 <use name="CommonTools/UtilAlgos"/>
6 <use name="FWCore/ServiceRegistry"/>
7 <flags EDM_PLUGIN="1"/>
```

5

Example at: /publicfs/cms/user/zhangjq/cms2024sysu/CMSSW_12_2_0/src/Demo/DemoAnalyzer

Run the analyzer

- Add output part in DemoAnalyzer/python/ConfFile_cfg.py
- cd DemoAnalyzer; scram b; cd python; cmsRun ConfFile_cfg.py

```python
import FWCore.ParameterSet.Config as cms

process = cms.Process("Demo")

process.load("FWCore.MessageService.MessageLogger_cfi")

process.maxEvents = cms.untracked.PSet( input = cms.untracked.int32(100) )

process.source = cms.Source("PoolSource",
                            # replace 'myfile.root' with the source file you want to use
                            fileNames = cms.untracked.vstring(
        'file:/afs/cern.ch/cms/Tutorials/workbook_twiki2021/MinBias_pythia8_14TeV_100events.root'
            )
                        )

process.TFileService = cms.Service("TFileService",
                                   fileName = cms.string('demo_out.root')
                                  )

process.demo = cms.EDAnalyzer('DemoAnalyzer',
    tracks    = cms.untracked.InputTag('generalTracks'),
                              )

process.p = cms.Path(process.demo)
```

- **Exercise: add above codes to save a tree and a histogram and run it**

```
[zhangjq@lxslc704 python]$ root -l demo_out.root
root [0]
Attaching file demo_out.root as _file0...
(TFile *) 0x5028ba0
root [1] .ls
TFile**         demo_out.root
 TFile*         demo_out.root
  KEY: TDirectoryFile   demo;1  demo
root [2] demo->cd()
(bool) true
root [3] .ls
TDirectoryFile*         demo    demo
 KEY: TH1F      h_pt;1  h_pt
 KEY: TTree     tree1;1 my tree 1
root [4] tree1->Print()
******************************************************************************
*Tree    :tree1     : my tree 1                                              *
*Entries :        0 : Total =            4540 bytes  File  Size =        701 *
*        :          : Tree compression factor =   1.00                       *
******************************************************************************
*Br    0 :runNum    : runNum/i                                               *
*Entries :        0 : Total  Size=        489 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    1 :lumiNum   : lumiNum/i                                              *
*Entries :        0 : Total  Size=        493 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    2 :eventNum  : eventNum/i                                             *
*Entries :        0 : Total  Size=        497 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    3 :nTrack    : nTrack/I                                               *
*Entries :        0 : Total  Size=        489 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    4 :vec_pt    : vector<double>                                         *
```

Output root file

Example at: /publicfs/cms/user/zhangjq/cms2024sysu/CMSSW_12_2_0/src/Demo/DemoAnalyzer

Event selection

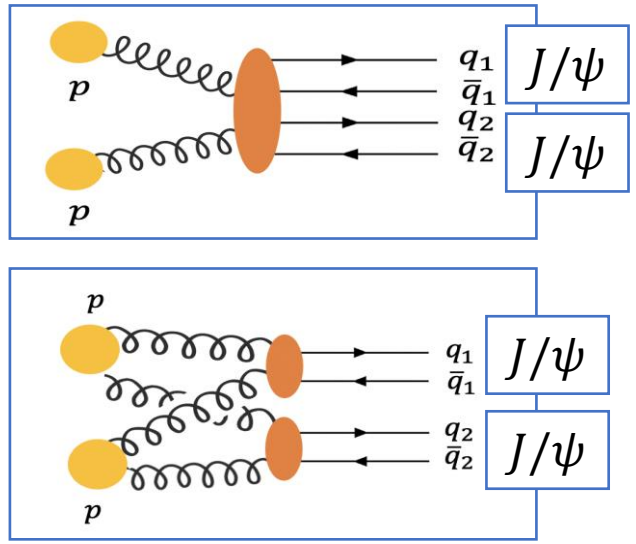
- To suppress bkg:
  - cuts in analyzer step and ntuple analysis step

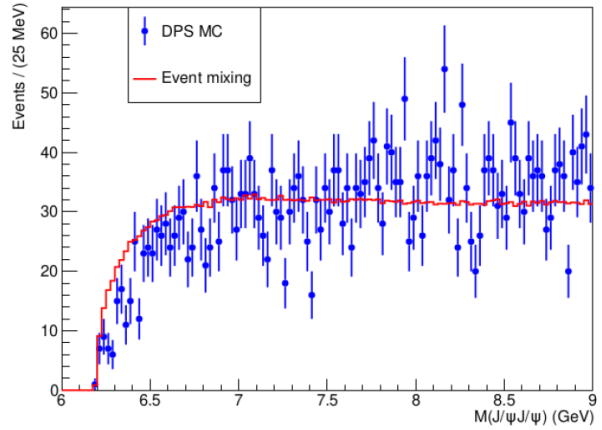
$$pp \to X \to J/\psi J/\psi \to \mu^+\mu^-\mu^+\mu^-$$

- Example cuts:
  - Detector acceptance/coverage: p_t and η of tracks
  - Tracks should be muon: MuonID
  - J/ψ candidate: mass window of muon pair; vertex fit of muon pair
  - $X \to J/\psi J/\psi \to \mu^+\mu^-\mu^+\mu^-$: vertex fit of two J/ψ and four muons

Background estimation

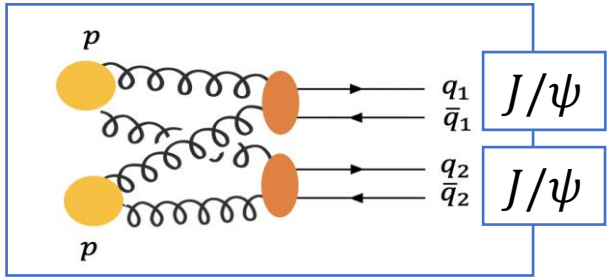
- Combinatorial background: non-$J/\psi J/\psi$ process
- Background from non-resonant process
  - Non-resonant single parton scattering, NRSPS
  - Double parton scattering, DPS
- MC simulation for NRSPS & DPS
- For DPS, another method is ‘event mixing’

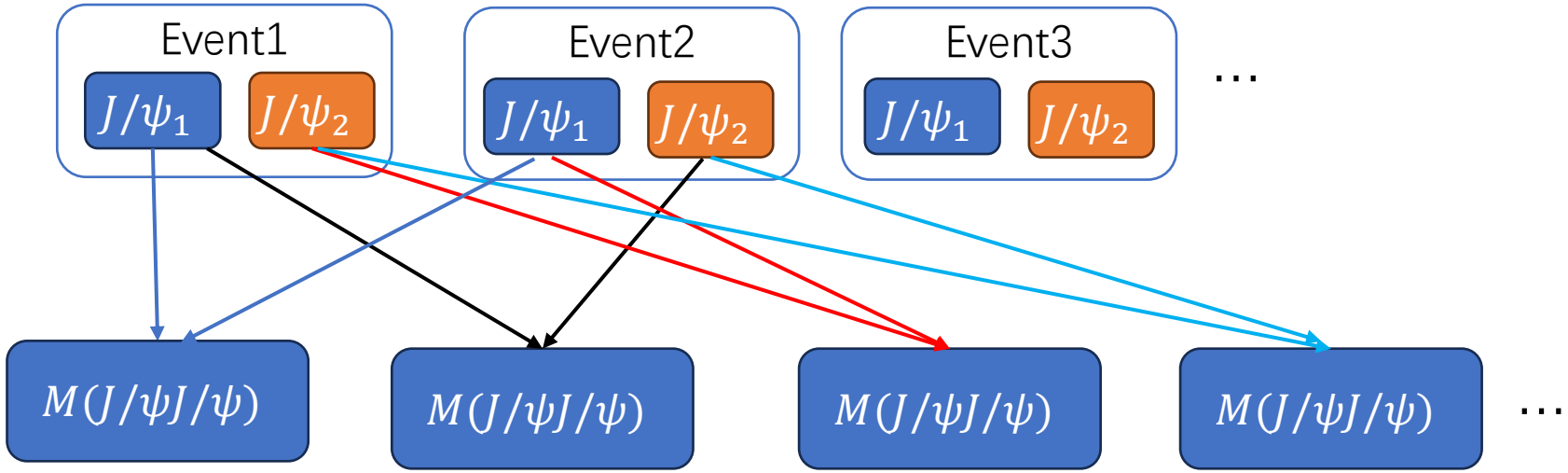


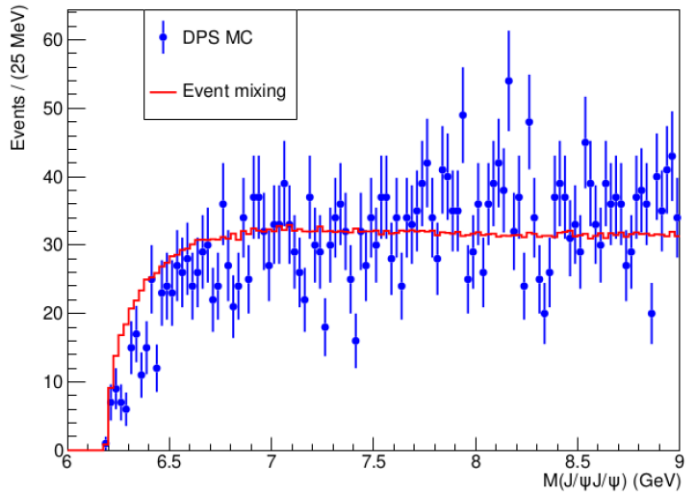


Event mixing for DPS shape

- DPS essentially contains two **independent** J/ψs
- Idea: artificially make events with independent J/ψs







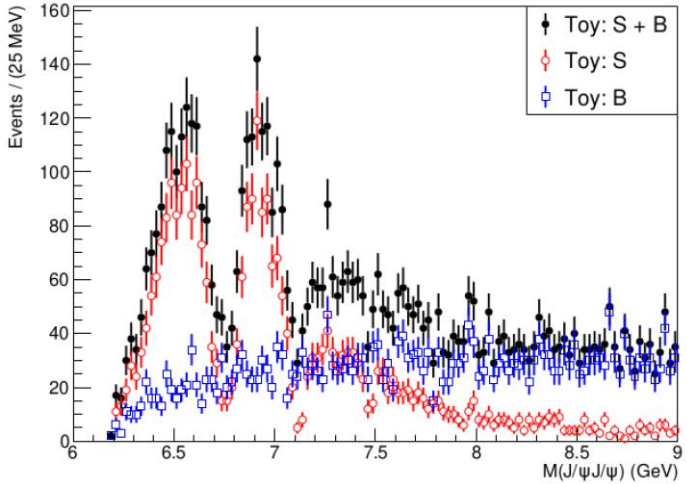
- **Exercise: get $M(J/\psi J/\psi)$ shape from event-mixing using MC in**

  /publicfs/cms/user/zhangjq/cms2024sysu/toy_dps/dps_mc.root

- **Copy mix_event.C in the same path, and add code in line 61-73**
- **Full example is mix_event_done.C in the same path**

Fit to mass distribution

- A toy $M(J/\psi J/\psi)$ distribution
  - Signal: three interfering relativistic BWs
  - Background: DPS



- Signal yields and parameters?
- Significance?

Fit to mass distribution

- DPS shape:

$$f_{NRDPS}(x) = \sqrt{x_t} \cdot \exp(-a \cdot x_t) \cdot (p_0 + p_1 \cdot x_t + p_2 \cdot x_t^2),$$
$$x_t = x - x_0, \quad x_0 = 2M_{J/\psi}.$$
$$a = 0.24358, p_0 = 0.23137, p_1 = -0.041952, p_2 = 0.012206$$

- Relativistic Breit-Wigner:
  - S-wave: $L = 0, B'_L = 1$

$$BW(m; m_0, \Gamma_0) = \frac{\sqrt{m\Gamma(m)}}{m_0^2 - m^2 - im\Gamma(m)},$$
$$\Gamma(m) = \Gamma_0 \left(\frac{q}{q_0}\right)^{2L+1} \frac{m_0}{m} \left(B'_L(q, q_0, d)\right)^2,$$

- Amplitude with interference:

$$f(m) = |r_1 \cdot \exp(i\phi_1) \cdot BW_1 + BW_2 + r_3 \cdot \exp(i\phi_3) \cdot BW_3|^2$$

- How to implement and use them in RooFit?

Fit mass distribution

- RooClassFactory::makePdf()
- Edit evaluate() function in MyDpsPdf.cxx

```
[zhangjq@lxslc704 test]$ root -l
root [0] RooClassFactory::makePdf("MyDpsPdf", "x,MTH,a,p0,p1,p2")
(bool) false
root [1] .ls
root [2] .q
[zhangjq@lxslc704 test]$ ls
MyDpsPdf.cxx  MyDpsPdf.h
[zhangjq@lxslc704 test]$
```

```
50  Double_t MyDpsPdf::evaluate() const
51  {
52    // ENTER EXPRESSION IN
53    double mx = x - MTH;
54    double fth = 0;
55    if (mx <= 0) return fth;
56    if (mx > 0) fth = TMath::Sqrt(mx);
57    double exponent = TMath::Exp(-1 * a * mx);
58    double poly = p0 + p1 * mx + p2 * mx * mx;
59    double res = fth * exponent * poly;
60    return res;
61  }
```

$$f_{NRDPS}(x) = \sqrt{x_t} \cdot \exp(-a \cdot x_t) \cdot (p_0 + p_1 \cdot x_t + p_2 \cdot x_t^2),$$
$$x_t = x - x_0, \quad x_0 = 2M_{J/\psi}.$$

- root –l –b –q “MyDpsPdf.cxx+”

```
[zhangjq@lxslc704 test]$ root -l -b -q "MyDpsPdf.cxx+"

Processing MyDpsPdf.cxx+...
Info in <TUnixSystem::ACLiC>: creating shared library /publicfs/cms/user/zhangjq/cms2024sysu/test/./MyDpsPdf_cxx.so
(MyDpsPdf) An instance of MyDpsPdf.
[zhangjq@lxslc704 test]$
```

- Then include .h, load .so in a root script, and use it as built-in pdf

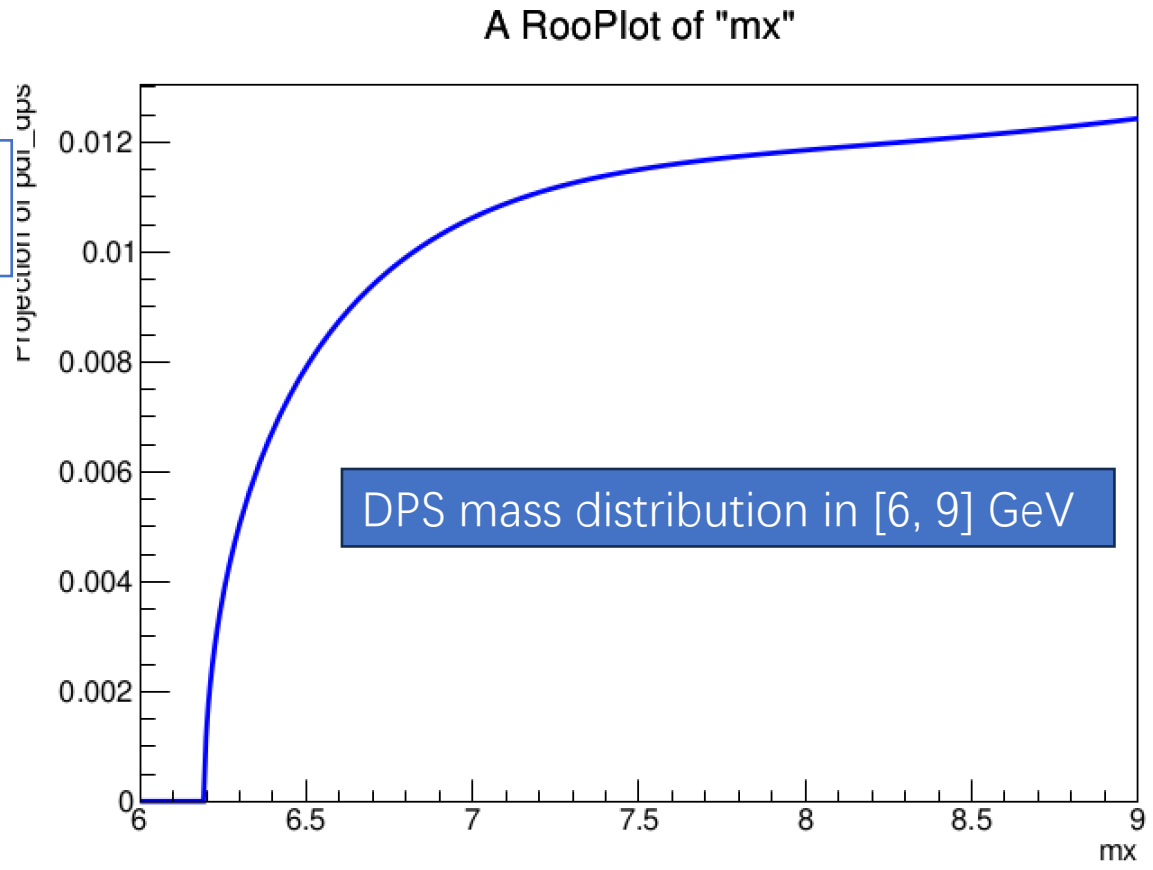
Note: here and after, we use the roofit version in CMSSW_11_1_1
So first: *cd some/path/CMSSW_11_1_1/src; cmsenv;*
Or: *cd some/path; cmsrel CMSSW_11_1_1/src; cmsenv;*

Fit mass distribution

- In root script, include .h and load _cxx.so
- Use MyDpsPdf as a built-in pdf

```
#include "RooRealVar.h"
#include "RooPlot.h"

#include "TSystem.h"
#include "MyDpsPdf.h"

using namespace RooFit;

void plotMyDpsPdf() {
  gSystem->Load("MyDpsPdf_cxx.so");

  RooRealVar mx("mx", "mx", 6.0, 9.0);
  RooRealVar R_MTH("R_MTH", "R_MTH", 2 * 3.0969);

  RooRealVar dpsA("dpsA", "dpsA", 0.24358);
  RooRealVar dpsP0("dpsP0", "dpsP0", 0.23137);
  RooRealVar dpsP1("dpsP1", "dpsP1", -0.041952);
  RooRealVar dpsP2("dpsP2", "dpsP2", 0.012206);
  MyDpsPdf pdf_dps("pdf_dps", "pdf_dps", mx, R_MTH, dpsA, dpsP0, dpsP1, dpsP2);

  RooPlot *frame = mx.frame();
  pdf_dps.plotOn(frame);

  TCanvas *c1 = new TCanvas("c1", "c1", 800, 600);
  c1->cd();
  frame->Draw();
  c1->Print("mydpspdf.pdf");

}
```

$a = 0.24358, p_0 = 0.23137,$
$p_1 = -0.041952, p_2 = 0.012206$



- **Exercise: wirte your own MyDpsPdf.cxx and use it to make a plot**
- **Example in the path:** /publicfs/cms/user/zhangjq/cms2024sysu/toy_dps

Fit mass distribution

- Pdf for interfering BWs

```
root [0] RooClassFactory::makePdf("MyRelBW1BW2BW3Square", "x,mass1,width1,r1,phi1,mass2,width2,r2,phi2,mass3,width3,r3,phi3")
```

```
 9 #include "Riostream.h"
10
11 #include "MyRelBW1BW2BW3Square.h"
12 #include "RooAbsReal.h"
13 #include "RooAbsCategory.h"
14 #include <math.h>
15 #include "TMath.h"
16
17 #include "ComplexRelBWFcn.h"
18
```

```
73 Double_t MyRelBW1BW2BW3Square::evaluate() const
74 {
75   // ENTER EXPRESSION IN TERMS OF VARIABLE ARGUMENTS HERE
76   std::complex<double> bw1 = ComplexRelBW(x, mass1, width1, r1, phi1);
77   std::complex<double> bw2 = ComplexRelBW(x, mass2, width2, r2, phi2);
78   std::complex<double> bw3 = ComplexRelBW(x, mass3, width3, r3, phi3);
79   std::complex<double> sum = bw1 + bw2 + bw3;
80   double res = std::norm(sum);
81   return res;
82 }
```

- Auxiliary files

```
 6 #ifndef COMPLEX_RELBW_FCN_H
 7 #define COMPLEX_RELBW_FCN_H
 8
 9 #include <iostream>
10 #include <complex>
11 #include "TMath.h"
12
13 void ComplexRelBWFcn();
14 double Q2(double sa, double sb, double sc);
15 double BlattWeisskopf(double q2, double d, double L);
16 std::complex<double> ComplexRelBW(double x, double mass, double width, double r, double phi);
17 #endif
```

ComplexRelBWFcn.h

```
 6 #include "ComplexRelBWFcn.h"
 7 #include "TMath.h"
 8
 9 void ComplexRelBWFcn() {
10   return;
11 }
12 double Q2(double sa, double sb, double sc) {
13   double res = (sa + sb - sc) * (sa + sb - sc) / 4.0 / sa - sb;
14   if (res <= 0) {
15     res = 0;
16   }
17   return res;
18 }
```

ComplexRelBWFcn.cxx

- First: root –l –b –q “ComplexRelBWFcn.cxx+”
- Then: root –l –b –q “MyRelBW1BW2BW3Square.cxx+”

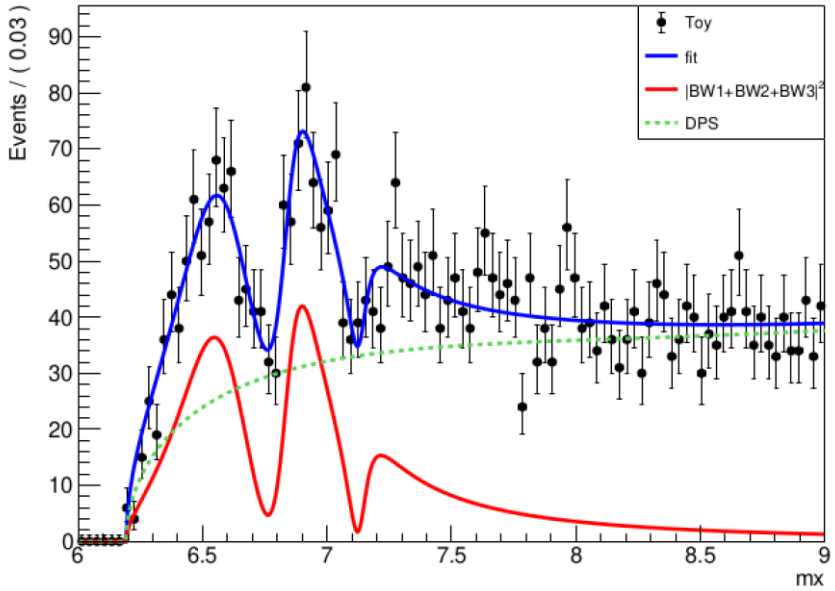
Example at: /publicfs/cms/user/zhangjq/cms2024sysu/myFit_test

Fit mass distribution

- Fit to the toy

```
93   MyRelBW1BW2BW3Square pdf_bw123("pdf_bw123", "pdf_bw123", mx,
94       mass_bw1, width_bw1, r_bw1, phi_bw1,
95       mass_bw2, width_bw2, r_bw2, phi_bw2,
96       mass_bw3, width_bw3, r_bw3, phi_bw3);
```

```
113   RooFitResult *fit_res_model = model.fitTo(*data, Save());
114   double nll_model = fit_res_model->minNll();
```



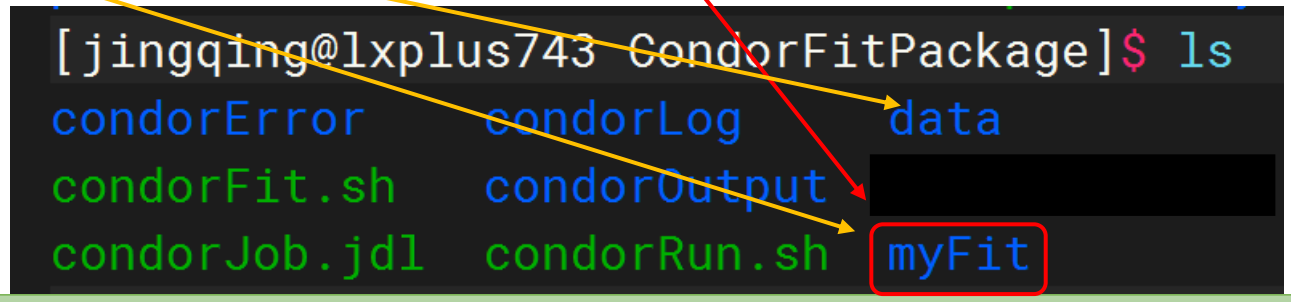
- Significance:
  1. Fit using **hypothesis 0** model, get negative log likelihood, nll0
  2. Fit using **hypothesis 1** model, get nll1
  3. $2\Delta nll$ follows χ^2 distribution
  4.
```
120   double pval = TMath::Prob(2 * delta_nll, 1); //assuming ndof = 1
121   double significance = RooStats::PValueToSignificance(pval / 2.0);
```

- **Question: significance of what, or between which two models?**

Example at: /publicfs/cms/user/zhangjq/cms2024sysu/myFit_test/fitToy.C

Fit using condor jobs

- When you need do many independent fits
  - Random sampling in parameter space to find global minimum
  - Closure test
  - Toy-experiment to evaluate significance
- Fit using condor jobs
  - A ‘template’ fit script and related files
  - Input & output transfer files
  - shell script [executable !] to run in condor worker node (condorRun.sh)
  - Condor job description file (condorJob.jdl)



Example at lxplus: /afs/cern.ch/user/j/jingqing/public/cms2024sysu/CondorFitPackage

Example at lxslc7: /publicfs/cms/user/zhangjq/cms2024sysu/CondorFitPackage

Fit using condor jobs

- Condor job description file: condorJob.jdl
  - **condor_submit condorJob.jdl**

```
universe = vanilla
executable = condorRun.sh
arguments = $(SEED)

####transfer_input_files = xxx, xxx, xxx,
######transfer_output_files can be files, directories
####transfer_output_files = yyy.log, yyy.txt, yyy.pdf
####when_to_transfer_output = ON_EXIT
#####output = output/$(ClusterId).$(ProcId).out
#####error = error/$(ClusterId).$(ProcId).err
#####log = log/$(ClusterId).log

should_transfer_files = yes
transfer_input_files = condorFit.sh, ./myFit, ./data

output = condorOutput/$(ClusterId).$(ProcId).out
error =  condorError/$(ClusterId).$(ProcId).err
log = condorLog/$(ClusterId).log

#### +MaRunTime in seconds 3600=1hour,7200=2hour,10800=3hour
####+MaxRunTime = 10800
####espresso: 20 minutes
####workday: 8 hours; tomorrow: 1 day; testmatch: 3 days; nextweek: 1 week
#### need the ""
+JobFlavour = "espresso"

#transfer_output_files = myFit/output, myFit/figure
#when_to_transfer_output = ON_EXIT

#submit multiple jobs, each to run a fitScript
#https://htcondor.readthedocs.io/en/latest/users-manual/submitting-a-job.html
## seed = 1, 2, 3
queue 1 SEED from seq 1 3 |
```

Executable shell script to be run in the server

Max job time

Submit 3 jobs for SEED = 1, 2, 3

Fit using condor jobs

- condorRun.sh
  - Run singularity for CMSSW_11_1_1 and execute **condorFit.sh**
  - If do not need singularity, can change .jdl

condorRun.sh

```
#!/bin/bash

if [ -e "/cvmfs/unpacked.cern.ch/registry.hub.docker.com/cmssw/el7:amd64" ]; then
  CONTAINER_NAME="el7:amd64"
elif [ -e "/cvmfs/unpacked.cern.ch/registry.hub.docker.com/cmssw/el7:x86_64" ]; then
  CONTAINER_NAME="el7:x86_64"
else
  echo "Could not find amd64 or x86_64 for el7"
  exit 1
fi
# Run in singularity container
# Mount afs, eos, cvmfs
# Mount /etc/grid-security for xrootd
export SINGULARITY_CACHEDIR="/tmp/$(whoami)/singularity"
singularity run -B /afs -B /eos -B /cvmfs -B /etc/grid-security -B /etc/pki/ca-trust --home $PWD:$PWD /cvmfs/unpacked.cern.ch/registry.hub.docker.com/cmssw/$CONTAINER_NAME $(echo "$(pwd)/condorFit.sh ${1}")
```

```
universe = vanilla
executable = condorFit.sh
arguments = $(SEED)
```

condorFit.sh

```
#!/bin/bash

fitPackage=myFit
fitScript="doFit.C"
SEED=${1}
echo "argument is ${1}"

cd ${fitPackage}
cp ../data/toy.root .
sed -i "s/SEED = 1/SEED = ${SEED}/g" ${fitScript}

#to use roofit in CMSSW, 11_1_1 and later for roofit with fft
export VO_CMS_SW_DIR=/cvmfs/cms.cern.ch
source ${VO_CMS_SW_DIR}/cmsset_default.sh
###cmsrel CMSSW_11_1_1 #this shortcut command is not avaialbe in shell script
export SCRAM_ARCH=slc7_amd64_gcc820
scramv1 project CMSSW CMSSW_11_1_1
cd CMSSW_11_1_1/src
eval `scramv1 runtime -sh`
cd ../..

##compile user defined fit functions
./makelib.sh

echo "will do root -l -b -q ${fitScript}"
root -l -b -q "${fitScript}+" > /dev/null
#root -l -b -q "${fitScript}+" > /eos/user/j/jingqing/test/log_${1}
echo "root -l -b -q ${fitScript}+ done"
```

- More materials for HTCondor
  - CondorFitPackage/TrainingSlides.pdf
  - HTCondor Version 23.3.0 Manual