

[下载文章](#)[视频教程](#)

## 0-序

### 观前提示

本篇note只会介绍程序的逻辑关系以及需要用到的变量设置，并会加入作者对其中物理含义的浅显理解，会有很多未涉及的细节部分，比如这个变量为什么要这样用，为什么这个函数要加这个变量等。对于这些问题一部分是我并不清楚该如何解释，还有大部分是底层代码的关系，这不是在这篇简短的笔记中能写明白的，你需要在网站的[manual](#)中查到对应的变量的用法。比如你可以在pat::Muon里面找到MuonID的函数，会发现isSoftMuon里面需要一个reco::Vertex的变量，而isLooseMuon则不需要，你还能在里面找到更详细的关于MuonID的介绍。你还可以在[WorkBookCMSSWFramework < CMSPublic < TWiki \(cern.ch\)](#)中找到更详细的关于CMSSW work。所以这篇笔记只能作为你的开始，要想成为真正的高能物理分析大师，你要做兴趣使然的物理学家。

```

Muons
Muons.Muon
Ht::Muons
reco::Muon
pat::Muon
muon
photon
ParticleFlow::ParticleFlowDecision::MUEON)
reco::PFBlockElement::MUEON)
EVT::CContainer::MUEON)
Container::MUEON)
reco::SoftLeptonProperties::Quality::Muon)
Deldt::Muon)
Ht::CContainer::Muon)
FW::CContainer::Muon)
Ht::MultiFilter::Muon)

```

```

bool IsLooseMuon () const
bool IsMediumMuon () const
bool IsSoftMuon (const reco::Vertex &) const
bool IsTightMuon (const reco::Vertex &) const

```

```

bool pat::Muon::isTightMuon (const reco::Vertex & ) const

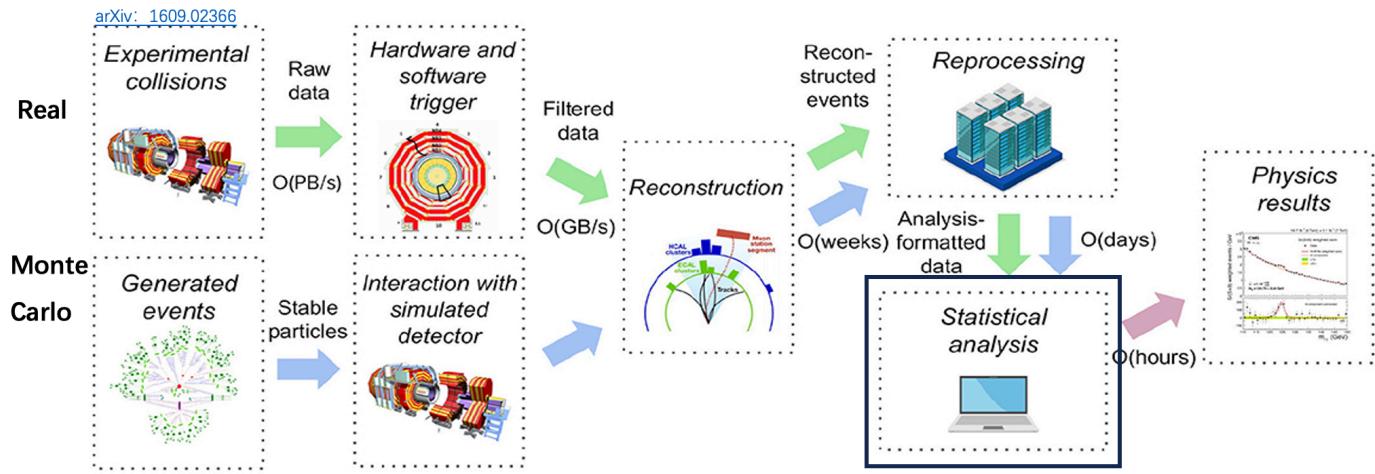
```

if muon id results are ever extracted from muon id value maps then the IsMuonIDAvailable method will be defined Muon  
Selectors as specified in <https://twiki.cern.ch/twiki/bin/view/CMSPublic/SWGuideMuonId>

### CMS数据

在进入到CMS物理分析之前，你需要大致了解CMS的数据是怎么来的。通常我们有两种数据来源，一种是在真实世界中，通过收集LHC对撞产生的粒子信息得到的真实数据（Real Data），一种是用理论的模型，结合探测器的模拟得到的模拟数据（Monte Carlo Data）。这两种数据都经过了类似的处理过程，最终存储在CERN的数据中心([CERN Data Centre](#))中，方便高能物理学家们去进行离线的物理分析。这些处理过程包括GENSIM-> RAW -> RECO ->SKIM

- **GENSIM (Generation & Simulation)** 指探测器直接探测的粒子撞击事例 (指通过Monte Carlo模拟的粒子撞击事例)
- **RAW** 事实上，大部分的撞击事例是无趣的，而且撞击频率远高于计算机能够处理的极限，通过探测器本身的硬件层面或软件程序会对这些事例进行初步的实时筛选，只把我们感兴趣的事例留下
- **RECO (Reconstruction)** 对撞产生的粒子在探测器中会留下电信号，这一步会将这些信号转变为粒子空间位置和具有物理意义的物理量以供分析使用。“重建”的意思在于，我们像是反拨时间的轴，推理出粒子之前的样子。
- **SKIM** 这一步正是这篇笔记所介绍的，与Trigger这样实时筛选相对应的离线筛选。



■ GENSIM -> RAW -> RECO -> SKIM

We are here!

**PS:** 在经过RECO的数据其实还会经过一次处理，生成名为**AOD (Analysis Object Data)**的root格式的数据存储下来，之后因为存储的信息过多，对AOD进行了瘦身而有了**mini-AOD**和**nano-AOD**的数据类型，他们之间会有一些差别，就需要你在具体的分析中去体会了。

### 术语

- **网格证书**

CMS 的数据分布在全球的计算中心。想要访问这些数据，需要通过安全认证。用户需要加载个人证书并生成临时代理：

```
1 | voms-proxy-init --voms cms
```

这样才能使用网格资源（如提交任务、读取数据等）。

- **CMSSW (CMS Software)**

CMS 的官方软件框架，用来进行数据处理、模拟和物理分析。所有分析代码通常在 CMSSW 环境下运行，需要先通过 `cmsrel`、`cmsenv` 等命令建立并进入工作区。

- **Condor Jobs**

HTCondor 是常用的任务调度系统，可以把大量任务分发到计算节点上运行。用户编写提交脚本，Condor 负责任务的排队、分发和状态管理。

- **CRAB Jobs**

CRAB (CMS Remote Analysis Builder) 是 CMS 提供的高层工具，基于 Condor 和网格系统封装而成。它简化了用户在分布式计算资源上运行分析任务的流程，只需提供配置文件即可自动提交、监控和管理任务。

## 前期准备

如果你还没准备好jump in，或许这篇手册能带你体验一下数据分析的快乐[Getting Started with CMS MiniAOD Open Data](#)，如果你想畅玩完整版，这个[CMS WorkBook](#)应该能让你从入门到精通。

Here are the empty files in following path, you can copy it to your work directory using `cmsrel CMSSW_13_0_13` cms-software version because of the root file we will use. I also dropped some hints in the corresponding folder named as '\_hint'. So, if and only if you are stuck, you can get some helps with it. Never forget `cmsenv` before your work.

ps: heptu集群在cmsRun时会出现问题，推荐在lxplus集群上进行下面的工作

```
/home/storage0/users/zhufeng/formymultilep_learning/UserCode/
/afs/cern.ch/user/z/zhuf/public/UserCode/
/nnucms/home/zhufeng/formutilep_learning/UserCode
```

## 1-文件结构

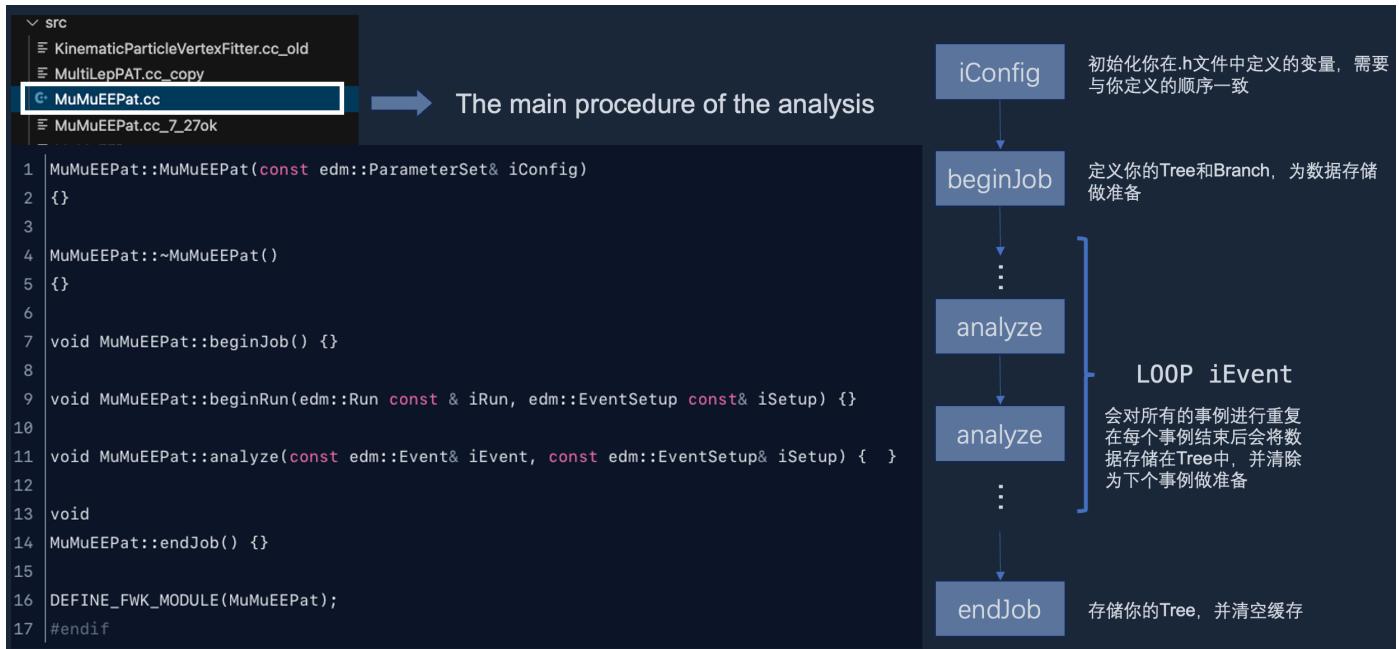
```
1 CMSSW_13_0_13
2 /src
3   └UserCode
4     └ MuMuEEPat
5       ├── BuildFile.xml
6       ├── doc
7       ├── interface
8       │   ├── MuMuEEPat.h
9       │   └── MuMuEEPat.h_hint
10      ├── mumueepat_cfg.py
11      ├── python
12      │   ├── mumueepat_cfi.py
13      │   └── __pycache__
14      │       └── mumueepat_cfi.cpython-39.pyc
15      └── src
16      │   ├── MuMuEEPat.cc
17      │   └── MuMuEEPat.cc_hint
18     └── test
19     └── f8e93985-e14c-4a8a-b28b-f8cceb3c878e.root
20     └── myntuple.C_hint
21     └── myntuple.C.rootmap
22     └── myntuple.h
23     └── Runjobs.C
24     └── runMultiLepPAT_dataRun3_miniAOD.py
```

在开始工作之前请对照上面的文件树检查自己的文件结构是否正确，特别是确保 `CMSSW_13_0_13/src/UserCode/MuMuEEPat` 的结构。其中的 `_hint` 文件是作为提示的完整代码，请适当得使用他。

当你对这个程序进行修改之后你需要编译他，在编译通过之后就可以进入test文件夹用一些文件去测试你的代码了。你需要在UserCode的上一级目录下用下面的命令编译你的代码，即在 `CMSSW_13_0_13/src` 这个路径下，并且你要确保 `UserCode/MuMuEEPat/src` 和 `UserCode/MuMuEEPat/interface` 下只能有一个 `.cc` 和 `.h` 后缀的文件（指内容重复的文件，你当然可以拥有不同内容的多个 `.cc` 和 `.h` 后缀的文件），因此你在备份自己的代码的时候请不要用 `.cc` 和 `.h` 作为后缀。

```
1 scramv1 b ProjectRename
2 scramv1 b clean #当你进入到一个新的工作目录的时候你需要上面两行去清空一些缓存
3 scramv1 b      #当你修改了你的代码的时候用这一句就可以编译了
```

其中的 `.cc` 和 `.h` 文件就是最主要的分析程序了，这里对程序的结构作一个简单的说明。对于输入的数据文件而言，即之前提到的AOD的root格式的文件，其中包含了多次碰撞事例(Events)中的物理信息，而事例与事例之间必然是没有物理关系的。因此我们只需要写一个事例(Events)的筛选流程，对其他事例只要不断循环就可以了，而这样的循环在这个框架下已经预设好了，即下图中的analyze的部分。其他部分则是为存储数据做的准备了。



`MuMuEEPat.h` 头文件用于声明函数变量等，不包含具体实现的细节

```
1 class MuMuEEPat : public edm::one::EDAnalyzer<edm::one::SharedResources>{
2
3     public://public成员可以从类的外部访问,任何对象或者函数都可以访问类的public成员
4     explicit MuMuEEPat(const edm::ParameterSet&); //初始化
5     ~MuMuEEPat();
6
7     private://private成员只能从类的内部访问。只有类的成员函数可以访问类的private成员
8     virtual void beginJob();
9     virtual void beginRun(edm::Run const & iRun, edm::EventSetup const& iSetup);
10    virtual void analyze(const edm::Event&, const edm::EventSetup&);
11    virtual void endJob();
12    //你可以不用理解上面在干什么, copy it directly!
13 };
14 #endif
```

`MuMuEEPat.cc` 源文件中就是具体分析过程

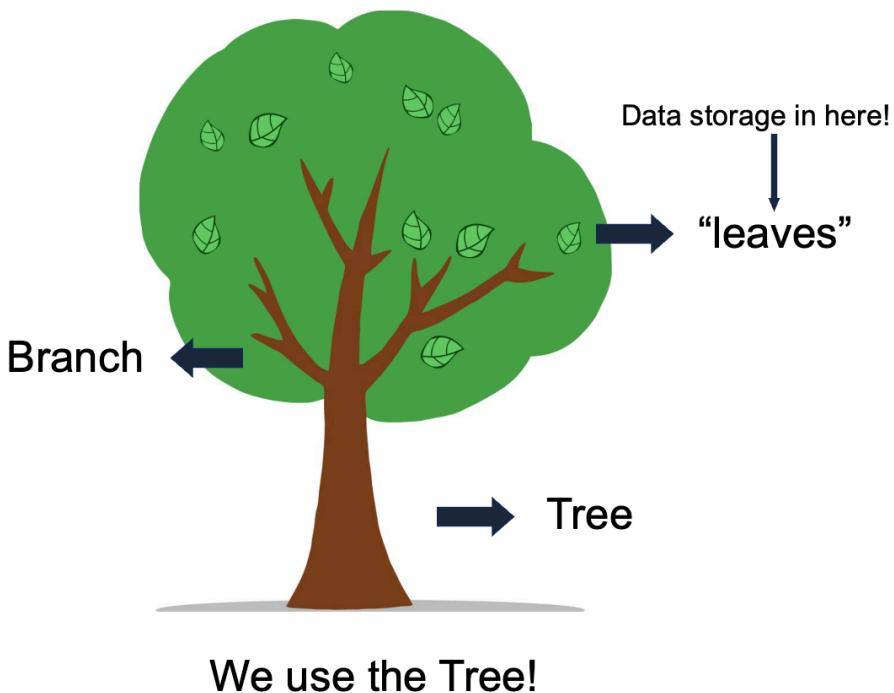
```
1 MuMuEEPat::MuMuEEPat(const edm::ParameterSet& iConfig)
2 {
3     //进行变量的初始化
4 }
5
6 MuMuEEPat::~MuMuEEPat()
7 {
8     //清除缓存?
```

```

9 }
10
11 void MuMuEEPat::beginJob() {
12 //在进行analyze的循环之前运行的部分, 这里就需要定义你需要的变量
13 }
14
15 void MuMuEEPat::beginRun(edm::Run const & iRun, edm::EventSetup const& iSetup) {
16 }
17
18
19 void MuMuEEPat::analyze(const edm::Event& iEvent, const edm::EventSetup& iSetup) {
20 //主要的分析程序, 会自动循环
21 }
22
23 void
24 MuMuEEPat::endJob() {
25 //analyze的循环过程中会不断在你定义的tree中填入数据, 循环结束后你就需要把这个tree给写入到一个文件中, 并进行必要的缓存清理
26 }
27
28 DEFINE_FWK_MODULE(MuMuEEPat);

```

## 2-创建与使用tree



我们需要在一棵树上放上许多枝桠, 将数据分类存储到枝桠对应的树叶中。正如之前所说, 种一棵树的最好时机是在之前(beginjob)。对应的在.h文件中要对tree和branch中的vector进行定义。在进行analyze之后, 树上已经枝繁叶茂了, 我们就需要在endjobs里面把这棵树给输出出来。至于如何在analyze中在这棵树上塞入树叶, 接着往下看。

```

1 //MuMuEEPat.h
2 ...
3 //define token begin
4 edm::EDGetTokenT<edm::View<pat::Muon> > gtpatmuonToken_;
5 edm::ESGetToken<MagneticField, IdealMagneticFieldRecord> magneticFieldToken_;
6 //define token end
7
8 TTree* X_One_Tree_;//Tree是必须的, 便利起见, 这里我们只用一个Tree存储所有变量
9 vector<double> *muononlyMass, *muononlyMassErr;//在.h文件中定义你的tree和变量, 这里我们用向量数组存储数据
10 ...

```

```

1 //MuMuEEPat.cc
2 MuMuEEPat::MuMuEEPat(const edm::ParameterSet& iConfig)
3 :magneticFieldToken_(esConsumes<MagneticField, IdealMagneticFieldRecord>()),
4 ...
5 mumuonlyMass(0),mumuonlyMassErr(0),
6 ...//这里是所有你在.h定义的变量初始化参数的地方, 初始化时需要与定义的顺序一致, 注意最后的变量没有逗号
7 {
8 //extract toke begin
9 gtpatmuonToken_ = consumes<edm::View<pat::Muon> >(edm::InputTag("slimmedMuons"));
10 //extract toke end
11 }
12 void MuMuEEPat::beginJob() {
13 edm::Service < TFileService > fs;
14 X_One_Tree_ = fs->make < TTree > ("X_data", "X_Data");//定义tree的名字, 相当于你种下一棵树命名为X_Data
15 X_One_Tree_->Branch("mumuonlyMass",&mumuonlyMass);
16 X_One_Tree_->Branch("mumuonlyMassErr",&mumuonlyMassErr);//定义重名的branch, 指向对应的vector, 这里意味着你之前的种下的树长出了枝桠
17 }
18 void MuMuEEPat::analyze(const edm::Event& iEvent, const edm::EventSetup& iSetup)//值得留意的是, 这个部分的程序按理
来说只会对一个对撞事例运行进行筛选, 并没有写对所有事例循环的语句, 不用担心, 在运行的时候会自动对所有的事例循环的, 这也是程序会分不同部分的原因
19 {
20 ...
21 for(...){//对一个事例中的所有muon进行循环, 可以设置条件去筛选你想要的muon并获取这些muon的物理量
22 ...
23   mumuonlyMass->push_back(...);
24   mumuonlyMassErr->push_back(...); //将数据填入数组的主要方式
25 ...
26 }
27 ...
28
29 //clear
30 X_One_Tree_->Fill();//树叶在之前的for循环中已经获取了, 用这一句话就能让这些数据到对应的枝桠上, 去塞满你的树吧
31 mumuonlyMass->clear();mumuonlyMassErr->clear();//因为接下来你需要对下一个事例中的muon再进行analyze部分的程序, 因而需要
把你数组进行清空, 避免数据重复
32 }
33 void MuMuEEPat::endJob() {
34 X_One_Tree_->Write();//当运行完analyze部分之后, 用这个命令就能将你的Tree保存下来
35 delete X_One_Tree_;//删除清理缓存
36 }

```

summary: 如果你想新加一个变量, 你需要:

- 1.在.h文件中定义你的变量名, 注意顺序和数据类型
- 2.在.cc文件中的iConfig中初始化你的变量, 注意顺序
- 3.在.cc文件中的beginJob中定义你要存储变量在Tree上对应的branch, 通常是与变量同名的
- 4.在.cc文件中的analyze中填入你的数据进变量中, 并在最后的clear部分进行必要的清除



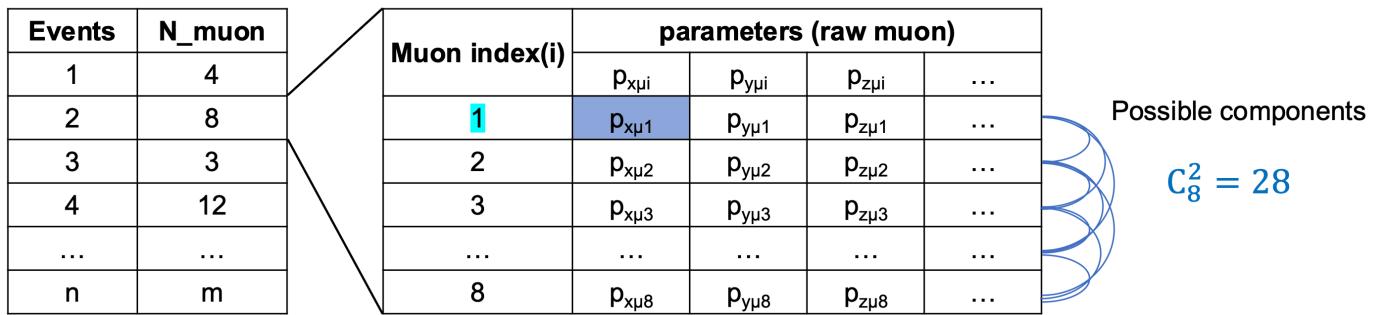
接下来两节我们将尝试输出事例中的mumu对的物理信息,第一步搭建for循环,对事例中的muon进行配对,第二步拟合muon的track获取物理信息并输出。

### 3-for循环起始

在理解程序的逻辑之前,我们需要在物理上理解即将要做的事情。对于我们要进行分析的root文件而言,里面包含了某一段探测器运行的时间内生成的所有对撞信息,而我们把一次对撞称为一次事件(Event),所以我们可以一个root文件里面包含了许多Events。每次对撞都可能产生很多个粒子,但这些粒子大部分我们都不能直接观测到,而是通过他们衰变产生的末态粒子,经过探测器被我们探测到,然后通过动力学计算反推出衰变前的母粒子的动力学信息。例如我们现在想找一个这样的过程 $J/\psi \rightarrow \mu^+ \mu^-$ ,其中 $J/\psi$ 粒子并不能直接被探测器观测,并且他的衰变时间限制了他的飞行时间,也来不及打到探测器上。所以我们就会研究他衰变出来的末态粒子 $\mu$ 子,通过 $\mu$ 子的动力学信息来探测 $J/\psi$ 粒子的产生。

我们可以先拿其中一次对撞事例来看,例如在第二次对撞事例中,我们探测到了8个 $\mu$ 子。这8个 $\mu$ 子在探测器上留下了位置信息被称为径迹,通过径迹和磁场信息我们就能获得他们的动力学信息。因为我们需要研究的是一对 $\mu$ 子,现在我们就需要把8个 $\mu$ 子两两组合,按理来说就会有28种可能的组合,但是我们考虑到 $J/\psi$ 粒子是电中性的,这就要求两个 $\mu$ 子的电荷加在一起为0,这样的条件(以及其他需要满足的条件)就会导致最终能配对组合数的会小于28个。

除了电荷的筛选,还有一个比较重要的条件是顶点拟合。因为就物理而言,一个粒子衰变成两个粒子,衰变出来的那两个粒子的轨迹肯定是从一个点出来的。探测器上肯定也是如此,那就需要要求假定是 $J/\psi$ 衰变出来的两个 $\mu$ 子也是从同一个点出来的,我们通常称之为顶点(vertex),但受限于探测器的精度,我们会用顶点概率(vertex probability)去衡量他们是来自同一顶点的可能性。



Muon pair		parameters					
		J/ψ (muon pair)		Fit muon 1		Fit muon 2	
Index(n)	comp	$p_{xJ/\psi n}$	...	$p'_{x\mu n}$	...	$p''_{x\mu n}$	...
1	[1,3]	$p_{xJ/\psi 1}$	...	$p'_{x\mu 1}$	...	$p''_{x\mu 1}$	...
2	[2,6]	$p_{xJ/\psi 2}$	...	$p'_{x\mu 2}$	...	$p''_{x\mu 2}$	...
3	[2,8]	$p_{xJ/\psi 3}$	...	$p'_{x\mu 3}$	...	$p''_{x\mu 3}$	...
...	...	...	...	...	...	...	...
14	[5,8]	$p_{xJ/\psi 14}$	...	$p'_{x\mu 14}$	...	$p''_{x\mu 14}$	...

理解了上面的物理关系，接下来就是在程序中的实现。

以下部分省略了一些变量的定义、初始化和清理过程，请参照之前的章节进行对应操作

```

1 //MuMuEEPat.h
2 private:
3 //define token begin
4 edm::EDGetTokenT<edm::View<pat::Muon> > gtpatmuonToken_; //获取pat::Muon的token
5 edm::ESGetToken<MagneticField, IdealMagneticFieldRecord> magneticFieldToken_; //获取MagneticField的token
6 //define token end

```

```

1 //MuMuEEPat.cc
2 MuMuEEPat::MuMuEEPat(const edm::ParameterSet& iConfig)
3 :magneticFieldToken_(esConsumes<MagneticField, IdealMagneticFieldRecord>())
4 {
5 //extract token
6 gtpatmuonToken_ = consumes<edm::View<pat::Muon> >(edm::InputTag("slimmedMuons")); //在.h文件中导入了pat::Muon的
token在这里就可以提取出来
7 //extract token
8 }

```

这里的gtpatmuonToken\_是一个成员变量，它保存了一个Token。这个Token表示模块需要pat::Muon类型的对象，数据来源是一个名为“slimmedMuons”的数据集（由edm::InputTag指定）。这个数据集名称是在数据处理链中定义的，用于标识数据源。

可以用下面的方式查看这个tag是什么（通常不用Displaced，至于其他的rootfile的情况不清楚是怎样的）

```

1 $ edmDumpEventContent f8e93985-e14c-4a8a-b28b-f8cceb3c878e.root
2 ...
3 vector<pat::Muon>           "slimmedDisplacedMuons"      " "      "PAT"
4 vector<pat::Muon>           "slimmedMuons"           " "      "PAT"
5 ...

```

pat::Muon是一个代表μ子的对象类型，它是分析中使用的“物理对象”（Physics Object）的一种。PAT（Physics Analysis Toolkit）是CMSSW中的一个包，提供了一组标准的高层次的物理对象类型，如电子、μ子、喷流（jets）等。pat::Muon对象包含了关于μ子的各种信息，例如动量、能量、轨迹等。

```

1 //MuMuEEPat.cc
2 void MuMuEEPat::analyze(const edm::Event& iEvent, const edm::EventSetup& iSetup)

```

```

3 // analyze
4 edm::Handle<edm::View<pat::Muon>> thePATMuonHandle; // 使用edm::Handle可以确保在访问数据时, 这些数据对象是有效的并且已被
5 正确初始化。定义一个智能指针 thePATMuonHandle
6 iEvent.getByToken(gtptmuonToken_, thePATMuonHandle); // 将之前的token提取出来并存储到 thePATMuonHandle
7 edm::View <pat::Muon>::const_iterator iMuon1; // 定义两个指向 edm::View<pat::Muon> 中 pat::Muon 对象常量迭代器, 只读
8 而不能修改
9 edm::View <pat::Muon>::const_iterator iMuon2;
10 for (iMuon1 = thePATMuonHandle->begin(); iMuon1 != thePATMuonHandle->end(); ++iMuon1) {
11     TrackRef muTrack1 = iMuon1->track(); // 与下面的 reco::Track 区别在于, muTrack1 可以用于轻量地操作, 比如为了看这个 track
12     是否是有效的, 下面的那个就直接重建出来值, 用于 track 数据的存储, 这样做是为了节省运算量, 总之, muTrack1 更适合进行一些操作运算
13     if (muTrack1.isNull()) {continue;} // 检测两个 muon 的 track 是否有效, 如果 track 是无效的, 跳过下面的步骤
14     reco::Track recoMul = *iMuon1->track(); // 有效的 muon 的 track 进行重建并存储到 recoMul
15     for (iMuon2 = iMuon1 + 1; iMuon2 != thePATMuonHandle->end(); ++iMuon2) { // 为了寻找两个配对在一起的 muon, 就需要
16     循环 n(n-1)/2 次
17         TrackRef muTrack2 = iMuon2->track();
18         if (muTrack2.isNull()) {continue;}
19         reco::Track recoMu2 = *iMuon2->track();
20         if ((iMuon1->charge() + iMuon2->charge()) == 0) { // 需要 mu+ mu- 电荷相加等于 0
21             // next part
22         }
23     }
24 }
25 } // analyze

```

## 4-track 拟合

```

1 //MuMuEEPat.cc
2 void MuMuEEPat::analyze(const edm::Event& iEvent, const edm::EventSetup& iSetup)
3 // analyze
4 const MagneticField &bFieldHandle = iSetup.getData(magneticFieldToken_);
5 const double myMumass = 0.1056583745;
6 const double myMumasserr = myMumass * 1e-6; // 后面会用到的妙妙工具
7 /////////////////
8 if ((iMuon1->charge() + iMuon2->charge()) == 0) {
9     TransientTrack muonPTT(muTrack1, &(bFieldHandle));
10    TransientTrack muonMTT(muTrack2, &(bFieldHandle)); // 获取 muon 的 track 信息, 需要 track 的数据和磁场信息, 命名为 minus
11    KinematicParticleFactoryFromTransientTrack pmumuFactory;
12    ParticleMass muon_mass = myMumass;
13    float muon_sigma = myMumasserr;
14    float chi = 0.;
15    float ndf = 0.;
16    vector < RefCountedKinematicParticle > muonParticles;
17    muonParticles.push_back(pmumuFactory.particle(muonPTT, muon_mass, chi, ndf, muon_sigma));
18    muonParticles.push_back(pmumuFactory.particle(muonMTT, muon_mass, chi, ndf, muon_sigma));
19    // 以上是对 track 的拟合
20    KinematicParticleVertexFitter fitter;
21    RefCountedKinematicTree psiVertexFitTree;
22    psiVertexFitTree = fitter.fit(muonParticles);
23    // 以上是对顶点的拟合
24    if (psiVertexFitTree->isValid()) { // 顶点拟合有效, 则
25        psiVertexFitTree->movePointerToTheTop(); // 这个 tree 的结构是 psi->mu mu 所以 Top 就是 psi, 后面还会用到 FirstChild,
26        NextChild 指的就是 mu mu 了, child 排序按照 muonParticles.push_back 的填入顺序决定的
27        RefCountedKinematicParticle psi_vFit_noMC = psiVertexFitTree->currentParticle();
28        RefCountedKinematicVertex psi_vFit_vertex_noMC = psiVertexFitTree->currentDecayVertex();
29        KinematicParameters mymumupara = psi_vFit_noMC->currentState().kinematicParameters();
30        // 以上是对这个 tree 提取数据
31        muumuonlyMass->push_back(psi_vFit_noMC->currentState().mass());
32    }
33 } // analyze

```

试着将下面的这些变量都在 Tree 里面存储下来吧!

```

1 //MuMuEEPat.cc
2 float mymuonlyctau=GetcTau(psi_vFit_vertex_noMC,psi_vFit_noMC,theBeamSpotV);
3 float mymuonlyctauerr=GetcTauErr(psi_vFit_vertex_noMC,psi_vFit_noMC,theBeamSpotV);
4 std::cout << "mumuonlyChg" << " " << (iMuon1->charge() + iMuon2->charge()) << std::endl;
5 std::cout << "mumuonlyctau" << " " << mymuonlyctau << std::endl;
6 std::cout << "mumuonlyctauerr" << " " << mymuonlyctauerr << std::endl;
7 std::cout << "mumuonlyVtxCL" << " " << ChiSquaredProbability((double)(psi_vFit_vertex_noMC->chiSquared()),
8 (double)(psi_vFit_vertex_noMC->degreesOfFreedom())) << std::endl;
9 mymuonlyMass->push_back(psi_vFit_noMC->currentState().mass());
10 std::cout << "mumuonlyMass" << " " << psi_vFit_noMC->currentState().mass() << std::endl;  no mass constrain
11 if( psi_vFit_noMC->currentState().kinematicParametersError().matrix()(6,6)>0) {
12     std::cout << "mumuonlyMassErr" << sqrt(psi_vFit_noMC->currentState().kinematicParametersError().matrix()
13 (6,6)) << std::endl;
14 } else {
15     mymuonlyMassErr->push_back(-9);
16 }
17 std::cout << "mumuonlyPx" << " " << mymumupara.momentum().x() << std::endl;
18 std::cout << "mumuonlyPy" << " " << mymumupara.momentum().y() << std::endl;
19 std::cout << "mumuonlyPz" << " " << mymumupara.momentum().z() << std::endl;
20 std::cout << "mumuonlymu1Idx" << " " << std::distance(thePATMuonHandle->begin(), iMuon1) << std::endl;
21 std::cout << "mumuonlymu2Idx" << " " << std::distance(thePATMuonHandle->begin(), iMuon2) << std::endl;

```

summary:

以上部分可以理解为，我们对探测器中的 $\mu$ 子进行重建，并尝试将重建的 $\mu$ 子对的顶点进行拟合，选择那些由一个粒子衰变出来两个正负 $\mu$ 子的过程，将这个重建出来的粒子的物理信息输出出来。

现在你已经掌握了如何在一个事例中筛选并提取简单的muon对的物理信息了，这代表着你已经掌握了绝大部分代码的逻辑关系！接下来就是要输出更多有用的物理信息，不过在此之前，为什么不看一眼你的 $J/\psi$ 是什么样子的呢？

## 5-myntuple 分析root文件

runMuMuEEPAT\_data\_Run2012CMSSW53XJan.py -> ivars.outputFile='mymultilep.root' 会根据你之前写的.cc文件进行筛选输出这个root文件， cmsenv 配置环境之后，用下面的命令运行这个py文件

```
1 cmsRun runMultiLepPAT_dataRun3_miniAOD.py
```

一个ntuple的job需要 myntuple.C.rootmap, myntuple.C, myntuple.h和一个运行的Runjobs.py文件

.rootmap是通用的

```

1 //myntuple.C.rootmap
2 { decls }
3 [ myntuple_C.so ]
4 namespace myntuple
5 class myntuple::myntuple
6 header myntuple.h

```

当你没有.C与.h文件时，可以通过mymultilep.root生成，他是根据你之前编辑的MuMuEEPat.h中的变量，去生成一个对应的myntuple.h文件，以及一个空的myntuple.C文件，当然你可以在 MakeClass 那里取一个自己的名字，但你也需要对应的在rootmap中修改，如果你是新手的话用相同的命名是最好的。注：这一步会覆盖掉之前你生成的myntuple.C等文件，当你没有对MuMuEEPat.h添加额外的变量的时候就不需要重复这一步。

```

1 $ root -l mymultilep.root
2 .ls //mkcands
3 mkcands->cd()
4 .ls //X_data
5 X_data->MakeClass("myntuple")

```

```

1 //会生成这样的.C文件和一个包含你在之前.cc里面定义过的所有变量的.h文件
2 #define myntuple_cxx
3 #include "myntuple.h"
4 #include <TH2.h>
5 #include <TStyle.h>

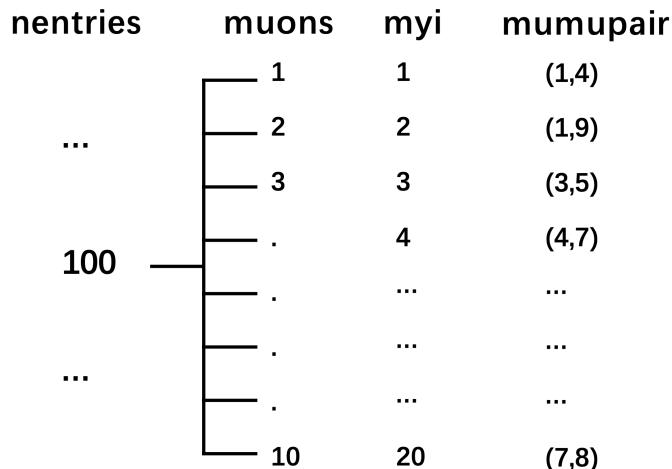
```

```

6 #include <TCanvas.h>
7
8 #include <iostream>
9 void myntuple::Loop()
10 {
11     if (fChain == 0) return;
12     Long64_t nentries = fChain->GetEntries();
13     Long64_t nbytes = 0, nb = 0;
14
15     TFile* myhbk = new TFile ("myhbk.root", "recreate"); //创建一个root, 将所有你想画出来的直方图都放进去
16     TH1F* mumumass = new TH1F("mumumass", "mumumass", 5000, 0, 25); //创建一个一维直方图
17     TH1F* mumumasserr = new TH1F("mumumasserr", "mumumasserr", 5000, 0, 25);
18     for (Long64_t jentry=0; jentry<nentries; jentry++) { //对每个事例进行loop
19         Long64_t ientry = LoadTree(jentry);
20         if (ientry < 0) break;
21         nb = fChain->GetEntry(jentry);   nbytes += nb;
22         // if (Cut(ientry) < 0) continue;
23         for (unsigned int myi = 0; myi < mumuonlyMass->size(); myi++) { //对一个事例中的mumu进行loop
24             mumumass->Fill((*mumuonlyMass)[myi]); //将你的变量填入到你定义的直方图中
25             mumumasserr->Fill((*mumuonlyMassErr)[myi]);
26         }
27         if (jentry%10000 == 0) std::cout << "I am running " << jentry << " th entries out of " << nentries << " total
28         entries" << std::endl;
29     }
30     myhbk->Write(); //这一句就能将所有你填入的直方图保存下来
31 } //end

```

Tips: 这里需要说明几个概念。nentries可以看作是对撞事例的编号，所以第一个for循环是为了对所用的对撞事例进行循环的，而第二个for循环是对一次事例中的mumuonlyMass->size()进行循环，这里取的编号是mumu对的编号，也就是对所有的μ子对进行循环。在之后你还会尝试对单个μ子进行操作，逻辑也是如此。



写好myntuple.C之后，需要进行编译

```

1 $ root -l
2 .L myntuple.C++

```

编译完成后就能运行Runjobs.C

```

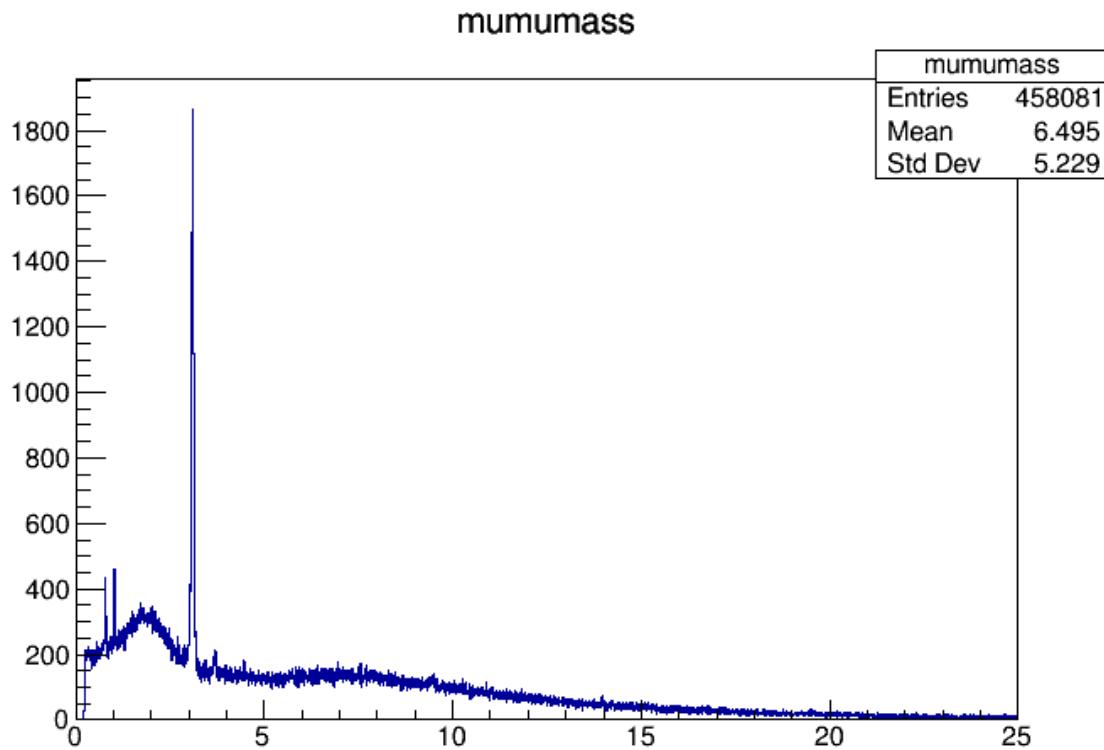
1 void Runjobs()
2 {
3     gSystem->Load("myntuple_C.so");
4     TChain * chain = new TChain("/mkcands/X_data","");
5     chain->Add("./mymultilep.root"); //这里放入你在cmsRun生成的文件
6     myntuple a(chain);
7     a.Loop();
8 }

```

```
1 $ root -l -b -q Runjobs.C # -l 是不显示root启动界面, -b 以批处理模式运行, 不显示图形界面, -q 处理完命令行宏文件后退出。这个命令通常在需要跑很多文件的时候配合脚本使用
```

运行完成后会生成一个root文件, 就是你在myntuple.C中命名的那个myhbk.root, 你可以用一些命令去看里面的直方图, 不过更推荐用vscode和root file viewer这个插件去看。

```
1 $ root -l myhbk.root
2 root [0]
3 Attaching file myhbk.root as _file0...
4 (TFile *) 0x1cc6e10
5 root [1] .ls
6 TFile** myhbk.root
7 TFile* myhbk.root
8 KEY: TH1F mumumass;1 mumumass
9 root [2] mumumass->GetXaxis()->SetRangeUser(0.5, 5)
10 root [3] mumumass->Draw()
```



## 6-Muon ID的输出

Muon ID 本质上是对 $\mu$ 子的一些筛选条件的整合, 将其打上Loose, soft, medium, tight等标签, 以方便我们的分析, 如果你完成了这一章, 你就能在你的myntuple.C中用MuonID设置你自己的筛选条件了。这一节会介绍以下这几个ID的输出方式以及必要的物理信息。

```
1 muIsPatLooseMuon->push_back(iMuonP->isLooseMuon());
2 muIsPatTightMuon->push_back(iMuonP->isTightMuon(thePrimaryV));
3 muIsPatSoftMuon->push_back(iMuonP->isSoftMuon(thePrimaryV));
4 muIsPatMediumMuon->push_back(iMuonP->isMediumMuon());
```

需要注意的有两点:

- 1.与之前输出mu子对不同, 这次是输出单个mu子的信息
- 2.有些muon ID需要PrimaryVetex的信息, 而PrimaryVetex需要获取beamSpot的信息,

下面是需要用到的妙妙工具, 把他们放入到合适的地方吧!

```

1 // .h
2 edm::EDGetTokenT<BeamSpot> gtbeamspotToken_;
3 edm::EDGetTokenT<VertexCollection> gtprimaryVtxToken_;
4 bool addXlessPrimaryVertex_; bool resolveAmbiguity_;
5
6 // .cc
7 MuMuEEPat::MuMuEEPat(const edm::ParameterSet& iConfig)
8 :addXlessPrimaryVertex_(iConfig.getUntrackedParameter < bool > ("addXlessPrimaryVertex", true)),
9 resolveAmbiguity_(iConfig.getUntrackedParameter < bool > ("resolvePileUpAmbiguity", true))
10 {
11 gtbeamspotToken_ = consumes<BeamSpot>(edm::InputTag("offlineBeamSpot"));
12 gtprimaryVtxToken_ = consumes<VertexCollection>(edm::InputTag("offlineSlimmedPrimaryVertices"));
13 }

```

## Beamspot and PrimaryVertex

```

1 // .cc ->analyze
2 //Beamspot
3 Vertex theBeamSpotV; //声明一个Vertex类型的对象, 表示一个顶点
4 BeamSpot beamSpot; //声明一个BeamSpot类型的对象, 表示束流点
5
6 edm::Handle < reco::BeamSpot > beamSpotHandle; //在之前的代码中也用到, 用于存储束流信息
7 iEvent.getByToken(gtbeamspotToken_, beamSpotHandle); //提取信息存储在beamSpotHandle中
8 if (beamSpotHandle.isValid()) { //如果beamSpotHandle是有效的则把对应的信息存储起来
9   beamSpot = *beamSpotHandle;
10  theBeamSpotV = Vertex(beamSpot.position(), beamSpot.covariance3D()); //用beamSpot的位置信息, 和协方差矩阵来初始化
11  theBeamSpotV对象
12 } else std::cout << "No beam spot available from EventSetup" << std::endl;
13 //Beamspot end
14
15 //PrimaryVertex
16 Vertex thePrimaryV;
17 math::XYZPoint RefVtx; //用于表示三维点或向量, 存储PrimaryVertex
18
19 edm::Handle < VertexCollection > recVtxs;
20 iEvent.getByToken(gtprimaryVtxToken_, recVtxs);
21
22 unsigned int nVtxTrks = 0; //无符号整数即正整数避免负值
23 int mynGoodPrimVtx=0;
24
25 for(unsigned myi=0;myi<recVtxs->size();myi++) { //遍历所有recVtxs取到的顶点
26   if((*recVtxs)[myi].ndof()>=5 && fabs((*recVtxs)[myi].z())<=24 && fabs((*recVtxs)[myi].position().rho())
27   <=2.0) { //顶点条件有ndof自由度, z坐标, rho径向距离
28     mynGoodPrimVtx++; //符合条件的为goodPrimaryVertex, 计数+1
29   }
30 }
31 nGoodPrimVtx = mynGoodPrimVtx; //这个变量记得输出出来
32
33 if (recVtxs->begin() != recVtxs->end()) { //检查是否为空
34   if (addXlessPrimaryVertex_ || resolveAmbiguity_) { //如果为真, 选择第一个顶点为thePrimaryV
35     thePrimaryV = Vertex(*recVtxs->begin());
36   } else { //如果不真, 选择轨迹数量最多的顶点为thePrimaryV
37     for (reco::VertexCollection::const_iterator vtx = recVtxs->begin(); vtx != recVtxs->end(); ++vtx) {
38       if (nVtxTrks < vtx->tracksSize()) {
39         nVtxTrks = vtx->tracksSize();
40         thePrimaryV = Vertex(*vtx);
41       }
42     }
43   }
44 } else { //如果为空, 用之前的beamspot信息进行初始化
45   thePrimaryV = Vertex(beamSpot.position(), beamSpot.covariance3D());
46 } //这一部分确保在顶点重建和物理分析中, 有一个合理的主顶点可供使用
47
48 //接下来导出初始顶点的位置信息

```

```

47 RefVtx = thePrimaryV.position();
48 priVtxX = (thePrimaryV.position().x());
49 priVtxY = (thePrimaryV.position().y());
50 priVtxZ = (thePrimaryV.position().z());
51 priVtxXE = (thePrimaryV.xError());
52 priVtxYE = (thePrimaryV.yError());
53 priVtxZE = (thePrimaryV.zError());
54 priVtxChiNorm = (thePrimaryV.normalizedChi2());
55 priVtxChi = thePrimaryV.chi2();
56 priVtxCL = ChiSquaredProbability((double) (thePrimaryV.chi2()), (double) (thePrimaryV.ndof()));
57 //PrimaryVertex end

```

## Muon ID and Muon block

```

1 //.cc ->如果之前更复杂的情况你已经完全掌握的话,下面这部分对你来说应该是非常简单的:)
2 edm::Handle< edm::View<pat::Muon> > thePATMuonHandle; //同样定义handle,然后提取信息
3 iEvent.getByToken(gtpatmuonToken_, thePATMuonHandle);
4 edm::View<pat::Muon>::const_iterator iMuon;
5 for (iMuonP = thePATMuonHandle->begin(); iMuonP != thePATMuonHandle->end(); ++iMuonP) {
6     ++nMu; //muon计数器,用于之后的效率计算
7     muIsPatLooseMuon->push_back(iMuonP->isLooseMuon());
8     muIsPatTightMuon->push_back(iMuonP->isTightMuon(thePrimaryV));
9     muIsPatSoftMuon->push_back(iMuonP->isSoftMuon(thePrimaryV));
10    muIsPatMediumMuon->push_back(iMuonP->isMediumMuon());
11    muPx->push_back(iMuonP->px());
12    muPy->push_back(iMuonP->py());
13    muPz->push_back(iMuonP->pz());
14    muCharge->push_back(iMuonP->charge());
15 }

```

## 7-处理数据

如果一切顺利的话,你应该会在mumu的质量谱中3Gev附近看到一个很尖的峰,没错这就是你重建出来的 $J/\psi$ 粒子! Congratulations! 但像这样粗浅的看一眼就说,我找到的这个峰就是3.0969Gev的 $J/\psi$ 峰,很显然是会被丁肇中先生打死的。我们还需要对这些数据点进行拟合,用数学的方式去确定我们找到的峰是什么粒子的衰变函数。

```
1 TH1F* mumumass = new TH1F("mumumass", "mumumass", 5000, 0, 25); //5000为bin的数量
```

还记得之前我们在myntuple.C文件中定义的名为mumumass一维直方图吗,里面我们设置了他的范围0-25,以及一个5000的bin。这里的bin可以理解为成条形统计图的柱子,这也就意味着我们会损失掉一些精细的数据,比如我们就无法分辨质量为3.000和3.0002的两个mumu,因为他们都会被丢到3.000~3.005这一个bin中,当然如果你的bin足够多也可以称之为“精细”,这也是微积分的方法了。但为什么不直接获取每一个mumu对的质量数值呢!所以,你这里输出的直方图虽说可以用于拟合,但如果想更加自由一点,你需要输出每一个mumu对的质量,这样你就能在你的拟合程序中随心所欲地改变你的bin了!

输出成txt部分比较简单,直接在myntuple.C中用下面的方式就能输出了,相信你应该能把这两个拼图放在正确的地方了,记得在输出多个变量的时候在两个变量之间加入空格,这样在后续需要输入多个变量的时候可以通过空格分成不同的列。

```

1 fstream myoutfile;
2 myoutfile.open("mydata.txt", std::ios::out); //放在循环外面
3 ...
4 myoutfile << std::fixed
5     << (*mumuonlyMass)[myi] << " " << (*mumuonlyMassErr)[myi]
6     << std::endl;
7 //放在循环里面

```

当你成功输出一个txt文件之后,就可以新做一个拟合程序。

下面正式进行拟合部分,以及如果你对使用的变量有任何疑问我没能包含的,你可以在这个链接中找到解答[ROOT:RooFit\(cern.ch\)](http://ROOT:RooFit(cern.ch)),你也可以参考我另外一篇笔记去详细了解RooFit的用法->[roofit manual](#)

```

RooRealVar::RooRealVar ( const char * name,
                        const char * title,
                        double      minValue,
                        double      maxValue,
                        const char * unit = ""
)

```

比如你可以找到 `RooRealVar` 的用法, 善用查找功能, 你也能成为master!

```

1 #include <TFile.h>
2 #include <TH1.h>
3 #include "RooAbsReal.h"
4 #include <RooRealVar.h>
5 #include <RooDataHist.h>
6 #include <RooGaussian.h>
7 #include <RooPlot.h>
8 #include <RooFitResult.h>
9 #include "RooChebychev.h"
10 #include "RooAddPdf.h"
11 #include "TCanvas.h"
12 #include "RooCrystalBall.h"
13 #include <RooFit.h>
14 using namespace RooFit;
15 using namespace std;
16 void fitroot() {//这个需要与你的文件名相同
17     RooRealVar mass("mass", "Invariant Mass", 0.5, 4); //定义变量, 是最常用的一个类
18     RooArgSet variables;
19     variables.add(mass); //创建了一个变量集合并在里面添加了mass这个变量
20
21     RooDataSet *data = RooDataSet::read("./mydata.txt", variables, "Q"); //读入数据, 当你需要多个变量输入的时候, 请确保你的变量集的顺序和你在myntuple.c的输出顺序是一致的, 也就是说你的变量需要跟你txt文件中每列(以空格分割)相对应
22     mass.setBins(300); //这里并不是对数据分bin, 而是对变量分bin, 相当于先分好一些空箱子
23     RooDataHist datahist("datahist", "binned data", RooArgSet(mass), *data); //这一步才是把数据分到之前的箱子中
24     //下面这些可以忽略
25     //TFile *file = TFile::Open("myhbk.root");
26     //TH1F *hist = (TH1F*)file->Get("mumumassSoft"); //这两步是直接从root提取你的直方图
27     //TH1F *rebinHist = (TH1F*)hist->Rebin(2, "rebinHist"); //这个意味着将直方图的两个bin合并为一个
28     //RooDataHist data("data", "Dataset with mass", mass, hist);
29     //
30
31     RooRealVar c0("c0", "c0", 0.5, -1., 1.);
32     RooRealVar c1("c1", "c1", 0.1, -1., 1.);
33     RooRealVar c2("c2", "c2", 0.1, -1., 1.);
34     RooRealVar c3("c3", "c3", 0.1, -1., 1.);
35     RooChebychev chev("chev", "chev", mass, RooArgList(c0, c1, c2, c3)); //Chebyshev polynomials
36
37
38     RooRealVar psimean("psimean", "Mean", 3.1, 2.9, 3.3);
39     RooRealVar psisigma("psisigma", "Sigma", 0.05, 0.001, 0.1);
40     RooRealVar psialpha("psialpha", "psialpha", 1., 0.0, 2.0);
41     RooRealVar psin("psin", "psin", 1., 0, 15);
42     RooCrystalBall psicb("psicb", "psicb", mass, psimean, psisigma, psialpha, psin, false); //Crystal Ball function
43     //RooGaussian psigauss("psigauss", "Gaussian PDF", mass, psimean, psisigma);
44
45     RooRealVar njpsi("njpsi", "signal fraction", 500, 0., 10000.);
46     RooRealVar nbkg("nbkg", "background fraction", 500, 0., 100000.);
47
48     RooExtendPdf epsisig("esig", "esig", psicb, njpsi);
49     RooExtendPdf epsibkg("ebkg", "ebkg", chev, nbkg); //这部分有一些知识前提, 总之将pdf extend之后可以直接获得成分的数
50     目
51     RooAddPdf model("fsig", "fsig", RooArgList(epsisig, epsibkg)); //model(x) = (njpsi/(njpsi+nbkg))*psicb(x) +
52     (1-(nbkg/(njpsi+nbkg)))*bkg(x)
53     RooFitResult* result = model.fitTo(*data, Save());

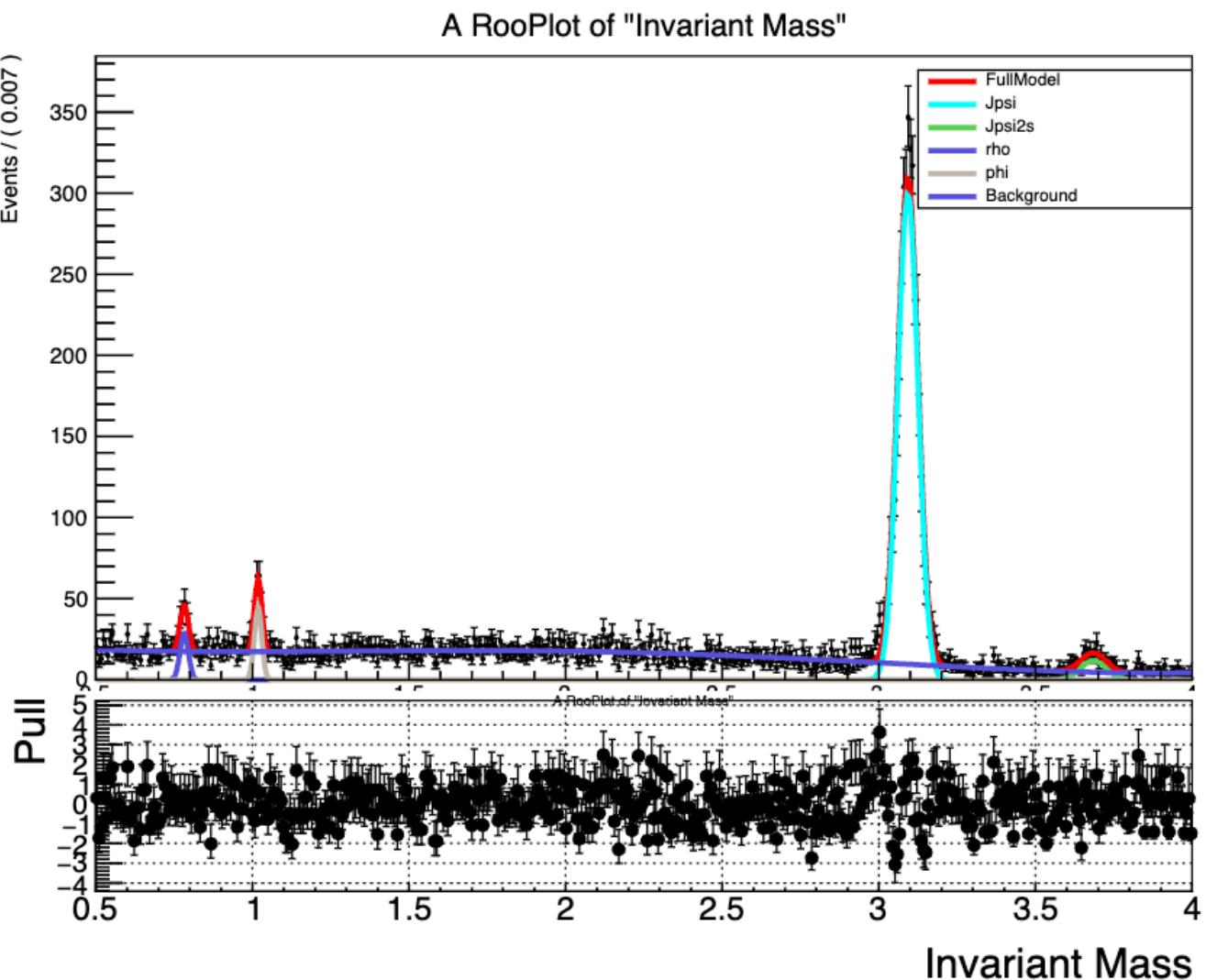
```

```

53 //RooFitResult* result = model.fitTo(datahist, Save());
54
55
56 RooPlot* frame = mass.frame(); //以mass创建一个坐标轴相当于x轴
57 data->plotOn(frame,MarkerStyle(20),MarkerSize(0.4),Name("data"));
58 //datahist.plotOn(frame,MarkerStyle(20),MarkerSize(0.4),Name("data"));
59 model.plotOn(frame,LineColor(2),LineWidth(3),Name("model"));
60 model.plotOn(frame,Components(epsisig),LineColor(7),LineStyle(1),LineWidth(3),Name("Jpsi"));
61 model.plotOn(frame,Components(epsiBkg),LineColor(9),LineStyle(1),LineWidth(3),Name("BKG"));
62 //将对应部分画到这个坐标轴上
63
64 TLegend yleg(0.7,0.7,0.9,0.9); //在一个画布的相对于左下角的(0.7,0.7)位置开始到(0.9,0.9)画出一个矩形
65 yleg.AddEntry(frame->FindObject("model"), "FullModel", "L");
66 yleg.AddEntry(frame->FindObject("Jpsi"), "Jpsi", "L");
67 yleg.AddEntry(frame->FindObject("BKG"), "Background", "L");
68 //添加图例
69
70 RooPlot *xpull=mass.frame(); //画pull分布
71 RooHist *pullx=frame->pullHist("data", "model"); //数据点在拟合的model图线上的偏离情况
72 xpull->addPlotable(pullx, "p");
73 //xpull->GetXaxis()->SetTitle("J/#psi J/#psi");
74 xpull->GetXaxis()->SetTitleSize(0.15);
75 xpull->GetXaxis()->SetLabelSize(0.12);
76 xpull->GetYaxis()->SetTitle("Pull");
77 xpull->GetYaxis()->SetTitleSize(0.15);
78 xpull->GetYaxis()->SetLabelSize(0.1);
79 xpull->GetYaxis()->SetTitleOffset(0.2);
80
81 TCanvas c ("c", "Fit with RooFit", 800, 600); //创建一个画布
82 c.cd();
83 TPad pad11("pad11", "pad11", 0, 0.3, 1, 1.0); //创建(0,0.3)到(1,1)的范围
84 pad11.SetTopMargin(0.08);
85 pad11.SetBottomMargin(0.017);
86 pad11.Draw();
87 pad11.cd(); //进入这个区域画你想画上去的plot和legend
88 frame->Draw();
89 yleg.Draw();
90 c.cd();
91 TPad pad12("pad12", "pad12", 0, 0.0, 1, 0.3);
92 pad12.SetTopMargin(0.03);
93 pad12.SetBottomMargin(0.325);
94 pad12.SetGridx();
95 pad12.SetGridy(2);
96 pad12.Draw();
97 pad12.cd();
98 xpull->Draw();
99 c.Update();
100 c.cd();
101
102 c.SaveAs("fitResult.pdf");
103
104 }

```

不出意外的话，你最终会得到一个很漂亮的拟合图形。或许你会发现了一些较小的峰，他们分别是 $\rho, \phi, \psi(2S)$ ，那么试着在[PDG](#)上找到这些粒子的质量，用 `RooBreitWigner` 或者 `RooGaussian` 函数去拟合这些峰吧！相信你会能得到一个看起来非常棒的结果！



现在喝杯可乐休息一下吧！后面的内容会更加复杂！

## pre 1-CRAB jobs submit

到此为止，你大概率只使用了我放在文件夹中的一个root文件，虽然对于目前的练习来说足够获取可观的结果了，但是随着筛选条件和分析需求的增加，一个root文件可能满足不了我们的需求，因此需要批量处理很多root文件。然而这个处理不需要在本地进行，而是提交一份“作业”，让分布在全球各地的计算集群一起完成你需要处理的任务，然后再存储到你指定的服务器上（这很酷！），这就是CMS的CRAB (CMS Remote Analysis Builder)。

接下来我会教你如何提交crab job，需要注意**CRAB**依赖你的证书，下面的所有操作都建立在你已经拥有证书，并在终端中使用 `voms-proxy-init --voms cms` 命令去获取证书代理。

- **DAS(Data Aggregation System)**

首先你要找到你想要处理的数据标签。实际上，CMS的对撞数据会经过多次的重建和分类，每次都会对应一个数据标签，至于你的研究需要用到那个数据标签，需要和相应的人员确认。然后你就能在DAS网站上查到你需要用的所有数据了，比如 `/ParkingDoubleMuonLowMass*/Run2024*-PromptReco-v1/MINIAOD` 这个标签你就能看到对应有64个数据标签，每个标签对应了某个时段运行的数据，这些都是你需要处理的。

 Data Aggregation System (DAS): [Home](#) | [Services](#) | [Keys](#) | [Bug report](#) | [Status](#) | [CLI](#) | [FAQ](#) | [Help](#)

results format:   results/page, dbs instance ,

/ParkingDoubleMuonLowMass\*/Run2024\*-PromptReco-v1/MINIAOD

[Show DAS keys description](#)

Showing 1—50 records out of 64.

By default DAS shows dataset with **VALID** status. To query datasets regardless of their status please use

```
dataset status=* dataset=/ParkingDoubleMuonLowMass*/Run2024*-PromptReco-v1/MINIAOD
```

Dataset: [/ParkingDoubleMuonLowMass0/Run2024B-PromptReco-v1/MINIAOD](#)  
 Creation time: 2024-04-08 18:23:22 Cross section: 0 Physics group: NoGroup Status: **VALID** Type: data  
[Release](#), [Blocks](#), [Files](#), [Runs](#), [Configs](#), [Parents](#), [Children](#), [Sites](#), [Physics Groups](#) [XSDB](#) Sources: **dbs3** [show](#)

Dataset: [/ParkingDoubleMuonLowMass0/Run2024C-PromptReco-v1/MINIAOD](#)  
 Creation time: 2024-04-16 20:14:39 Cross section: 0 Physics group: NoGroup Status: **VALID** Type: data  
[Release](#), [Blocks](#), [Files](#), [Runs](#), [Configs](#), [Parents](#), [Children](#), [Sites](#), [Physics Groups](#) [XSDB](#) Sources: **dbs3** [show](#)

Dataset: [/ParkingDoubleMuonLowMass0/Run2024D-PromptReco-v1/MINIAOD](#)  
 Creation time: 2024-05-04 22:45:05 Cross section: 0 Physics group: NoGroup Status: **VALID** Type: data  
[Release](#), [Blocks](#), [Files](#), [Runs](#), [Configs](#), [Parents](#), [Children](#), [Sites](#), [Physics Groups](#) [XSDB](#) Sources: **dbs3** [show](#)

Dataset: [/ParkingDoubleMuonLowMass0/Run2024E-PromptReco-v1/MINIAOD](#)  
 Creation time: 2024-05-24 00:17:31 Cross section: 0 Physics group: NoGroup Status: **VALID** Type: data  
[Release](#), [Blocks](#), [Files](#), [Runs](#), [Configs](#), [Parents](#), [Children](#), [Sites](#), [Physics Groups](#) [XSDB](#) Sources: **dbs3** [show](#)

Dataset: [/ParkingDoubleMuonLowMass0/Run2024F-PromptReco-v1/MINIAOD](#)  
 Creation time: 2024-06-21 16:55:02 Cross section: 0 Physics group: NoGroup Status: **VALID** Type: data  
[Release](#), [Blocks](#), [Files](#), [Runs](#), [Configs](#), [Parents](#), [Children](#), [Sites](#), [Physics Groups](#) [XSDB](#) Sources: **dbs3** [show](#)

看起来好像不是很多？还记得我给你的root文件吗，他的位置在哪呢？你可以点击下面的 [File](#) 按钮，每个数据标签里面包含的就是root文件了，数量和大小都会有所不同，比如下面的标签里面包含了894个root文件，这样如果把所有标签里面的root文件都加在一起想必是一个很大的数字。

results format:   results/page, dbs instance ,

file dataset=/ParkingDoubleMuonLowMass3/Run2024H-PromptReco-v1/MINIAOD

[Show DAS keys description](#)

Showing 1—50 records out of 894.

File name: [/store/data/Run2024H/ParkingDoubleMuonLowMass3/MINIAOD/PromptReco-v1/000/385/836/00000/0b80e21c-c3b4-4976-a9e2-a2c839cd6d08.root](#)  
 File size: 2.552272661e+09 (2.6GB) File size: 2552272661 (2.6GB) File type: EDM Number of events: 44239  
[Dataset](#), [Block](#), [Sites](#), [Runs](#), [Parents](#), [Children](#), [Lumis](#) Sources: **dbs3 rucio** [show](#)

File name: [/store/data/Run2024H/ParkingDoubleMuonLowMass3/MINIAOD/PromptReco-v1/000/385/836/00000/1d4819bf-4083-4c1f-b8cd-7f088460d698.root](#)  
 File size: 8.43176008e+08 (843.2MB) File size: 843176008 (843.2MB) File type: EDM Number of events: 14078  
[Dataset](#), [Block](#), [Sites](#), [Runs](#), [Parents](#), [Children](#), [Lumis](#) Sources: **dbs3 rucio** [show](#)

File name: [/store/data/Run2024H/ParkingDoubleMuonLowMass3/MINIAOD/PromptReco-v1/000/385/836/00000/388c6bec-1858-48cf-bfc1-f2c4a18b1483.root](#)  
 File size: 4.040535215e+09 (4.0GB) File size: 4040535215 (4.0GB) File type: EDM Number of events: 67995  
[Dataset](#), [Block](#), [Sites](#), [Runs](#), [Parents](#), [Children](#), [Lumis](#) Sources: **dbs3 rucio** [show](#)

如果你想测试别的root文件，可以将DAS上的root数据文件拷到本地：

```
xrdcp -d 1 -f root://xrootd-cms.infn.it//store/data/Run2024H/ParkingDoubleMuonLowMass3/MINIAOD/PromptReco-v1/000/385/836/00000/0b80e21c-c3b4-4976-a9e2-a2c839cd6d08.root .
```

也可以在py文件中直接将这个地址放进去，多个root之间需要用 , 连接(当然做到这些需要一个证书)。

```

1 ivars.inputFiles='root://xrootd-
2 cms.infn.it//store/data/Run2024H/ParkingDoubleMuonLowMass3/MINIAOD/PromptReco-
v1/000/385/836/00000/0b80e21c-c3b4-4976-a9e2-a2c839cd6d08.root',
3           'root://xrootd-
4 cms.infn.it//store/data/Run2024I/ParkingDoubleMuonLowMass0/MINIAOD/PromptReco-
v1/000/386/478/00000/1b1f1417-5da2-40be-b5da-3ab3daaf52cb.root')

```

其他的标签里面的信息同样也很重要，这里我简单介绍一下我们能用到的：

- release CMSSW版本信息，一般我们会使用这个数据标签下对应的最新的CMSSW版本，就像我在序中说明的一样，每年都会有很多不同的CMSSW版本，因此需要做出对应。

results format: list 50 results/page, dbs instance prod/global, Search Reset

release dataset=/ParkingDoubleMuonLowMass0/Run2024B-PromptReco-v1/MINIAOD

Show DAS keys description

Showing 1–1 records out of 1.

Release name: CMSSW\_14\_0\_4

Datasets Sources: dbs3 show

Showing 1–1 records out of 1.

- config 这里我们通常会获取数据对应的Global Tag (GT)，这相当于一个“配置标签”，用来告诉CMSSW在运行的时候应该使用哪一套校准、对齐、探测器状态等条件信息。这个版本也会不断变化，和CMSSW一样使用这个数据标签下对应的最新的GT版本。GT需要在cmsRun的py文件中修改，找到py文件中下面的部分，if里是MC的GT，else里是data的GT，对应修改数据的GT就好。

```

1 if runOnMC:
2     process.GlobalTag.globaltag = cms.string( 'MCRUN2_74_V9::All' )
3 else:
4     process.GlobalTag = GlobalTag(process.GlobalTag, '132X_dataRun3_Prompt_v4', '')

```

results format: list 50 results/page, dbs instance prod/global, Search Reset

config dataset=/ParkingDoubleMuonLowMass0/Run2024B-PromptReco-v1/MINIAOD

Show DAS keys description

Showing 1–1 records out of 1.

<first | prev | next | last>

Request name: cmsRun

Created by: /DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=cmst0/CN=658085/CN=Robot: CMS Tier0 Creation time: 2024-04-02 23:05:39 Global Tag: 140X\_dataRun3\_Prompt\_v2 Pset hash: GIBBERISH Release: CMSSW\_14\_0\_4 Request urls:

ReqMgr info Sources: dbs3 show

Showing 1–1 records out of 1.

<first | prev | next | last>

## • CRAB 配置文件

经过上面的查询和确认，并且已经完成对PAT程序包的debug，那么接下来就需要编辑一个crab的配置文件，用来提交crab作业。接下来我会介绍这6个需要修改的地方的意思：

```

1 from WMCore.Configuration import Configuration
2 config = Configuration()
3

```

```

4 config.section_('General')
5 config.General.transferOutputs = True
6 config.General.requestName = '2024JEta_Run2024I-PromptReco-v1' #modify 1
7
8 config.section_('JobType')
9 config.JobType.psetName = './runMultiLepPAT_dataRun3_miniAOD.py' #modify 2
10 config.JobType.pluginName = 'Analysis'
11 config.JobType.outputFiles = ['mymultilep.root']
12 config.JobType.allowUndistributedCMSSW = True
13
14 config.section_('Data')
15 config.Data.inputDataset = '/ParkingDoubleMuonLowMass0/Run2024I-PromptReco-v1/MINIAOD' #modify 3
16 config.Data.inputDBS = 'global'
17 config.Data.unitsPerJob = 50
18 config.Data.splitting = 'LumiBased'
19 #config.Data.outLFNDirBase = '/store/group/lpcbphy/noreplica/miniAOD_zhuf/JEta_July22/'
20 config.Data.outLFNDirBase = '/store/user/zhuf/JEta_July22/' #modify 4
21 config.Data.outputDatasetTag = '2024JEta_Run2024I-PromptReco-v1' #modify 5
22
23 config.section_('User')
24 config.section_('Site')
25 config.Site.storageSite = 'T3_CH_CERNBOX' # T3_US_FNALLPC #modify 6

```

- M1: `config.General.requestName`

- 这是提交作业的名字 (task name)。它会成为 CRAB 在服务器上和数据库中识别的唯一 ID。
- 一般建议包含 物理过程/数据集名 + 版本号/日期，方便区分不同的提交。
- 不能重复，否则 CRAB 会报错（即便你删了本地的任务目录，服务器上还会冲突），因此需要对每一个的数据标签编辑对应的crab config文件。

- M2: `config.JobType.psetName`

- 指定你要运行的 CMSSW 配置文件 (pset)。通常是 .py 文件，比如这里的 `runMultiLepPAT_dataRun3_miniAOD.py`，需要提供这个文件的相对/绝对路径。

- M3: `config.Data.inputDataset`

- 这是你要处理的官方数据集名称 (从 DAS 查询得到)。

- M4: `config.Data.outLFNDirBase`

- 指定你的 输出文件保存路径 (在 CERN EOS 或者 FNAL dCache 等存储系统下)。一般用 `/store/user/<username>/...`。你需要在这里写一个子文件夹，以对不同的分析做区分。

- M5: `config.Data.outputDatasetTag`

- 这是输出数据集的标签，类似一个后缀。它会被附加到你的输出数据集名字后面，方便在 DAS 里区分。一般和 `requestName` 保持一致，或者稍微简化。
- 如果重复提交同一个crab时，crab返回的输出文件会根据你提交的时间在同样路径下生成一个对应时间的文件夹，最中生成的输出文件夹路径类似

```

outLFNDirBase + inputDataset(1) + requestName :
/store/user/<username>/JEta_July22/ParkingDoubleMuonLowMass0/2024JEta_Run2024I-PromptReco-
v1/250602_020604

```

- M6: `config.Site.storageSite`

- 指定你的输出数据要写到哪个 存储站点 (SE, Storage Element)。比如: `T3_CH_CERNBOX`, `T3_US_FNALLPC`
- 需要选择一个你有写权限的站点，否则提交会报错。下面是一些常用的站点和检查权限的命令。事实上一般情况下都是有权限的，但是会出现你占用的空间实在太多，相应的管理员会找到你让你删除掉不用的，因此你需要熟练掌握下面的check和remove的操作。

```

T3_CH_CERNBOX
T2_CN_Beijing
T3_US_FNALLPC
cmsenv
voms-proxy-init --rfc --voms cms

```

```
crab checkwrite --site=T2_CN_Beijing
which scram >/dev/null 2>&1 && eval `scram unsetenv -sh` (这一步通常能找到下面命令里文件位置的前缀:
davs://cceos.ihep.ac.cn:9000/eos/ihep/cms/store/user/zhuf/)
```

```
1 | gfal-ls -lH davs://cceos.ihep.ac.cn:9000/eos/ihep/cms/store/user/zhuf/`  
2 | gfal-rm -r 'davs://cceos.ihep.ac.cn:9000/eos/ihep/cms/store/user/zhuf/SPStosingleJ_2023HI/'
```

一句话总结：

- M1 是作业的名字
- M2 是运行的配置文件
- M3 是输入的数据集
- M4 是输出文件存放路径
- M5 是输出数据集标签
- M6 是存储站点

## • CRAB 提交、状态查询和重新提交

当你编辑好你的config文件，并命名为 `crab3_xxx.py` 之后，就能使用 `crab submit crab3_xxx.py` 将你的CRAB job提交上去，正确的输出类似下面：

```
● test$crab submit crab3_2024Jpipi.py
Will use CRAB configuration file crab3_2024Jpipi.py
Importing CMSSW configuration ./runMultiLepPAT_dataRun3_miniAOD.py
Finished importing CMSSW configuration ./runMultiLepPAT_dataRun3_miniAOD.py
Warning: The following output files will not be published, as they are not EDM files: ['mymultilep.root']
Sending the request to the server at cmsweb.cern.ch
Success: Your task has been delivered to the prod CRAB3 server.
Task name: 250831_142639:zhuf_crab_2024Jpipi_Run2024I-PromptReco-v1
Project dir: ./crab_2024Jpipi_Run2024I-PromptReco-v1
Please use 'crab status -d ./crab_2024Jpipi_Run2024I-PromptReco-v1' to check how the submission process
proceeds.
Log file is /uscms_data/d3/fengzhu/Mar6_FourMuTwoPion/CMSSW_14_0_16/src/UserCode/FourMuTwoPion/test/crab_2
024Jpipi_Run2024I-PromptReco-v1/crab.log
```

然后会在你提交crab的地方生成一个名为 `crab_+requestName` 的文件夹，你可以使用 `crab status crab_2024Jpipi_Run2024I-PromptReco-v1` 去查询你的作业的状态：

```
fengzhu@cmslpc312:~/nobackup/Mar6_FourMuTwoPion/CMSSW_14_0_16/src/UserCode/FourMuTwoPion/test$crab status
-d ./crab_2024Jpipi_Run2024I-PromptReco-v1
Rucio client initialized for account zhuf
CRAB project directory: /uscms_data/d3/fengzhu/Mar6_FourMuTwoPion/CMSSW_14_0_16/src/UserCode/FourMuTwoPion/test/crab_2024Jpipi_Run2024I-PromptReco-v1
Task name: 250831_142639:zhuf_crab_2024Jpipi_Run2024I-PromptReco-v1
Grid scheduler - Task Worker: crab3@vocms0196.cern.ch - crab-prod-tw01
Status on the CRAB server: SUBMITTED
Task URL to use for HELP: https://cmsweb.cern.ch/crabserver/ui/task/250831_142639%3Azhuf_crab_2024Jpipi_Run2024I-PromptReco-v1
Dashboard monitoring URL: https://monit-grafana.cern.ch/d/cmsTMDetail/cms-task-monitoring-task-view?orgId=11&var-user=zhuf&var-task=250831_142639%3Azhuf_crab_2024Jpipi_Run2024I-PromptReco-v1&from=1756646799000&to=now
Status on the scheduler: SUBMITTED
Jobs status: unsubmitted 100.0% (1405/1405)

No publication information available yet
Log file is /uscms_data/d3/fengzhu/Mar6_FourMuTwoPion/CMSSW_14_0_16/src/UserCode/FourMuTwoPion/test/crab_2024Jpipi_Run2024I-PromptReco-v1/crab.log
```

Status on the scheduler:	SUBMITTED
Jobs status:	failed 63.8% ( 897/1405) idle 16.1% ( 226/1405) running 13.1% ( 184/1405) toRetry 7.0% ( 98/1405)

No publication information available yet

Error Summary: (use `crab status --verboseErrors` for details about the errors)

897 jobs failed with exit code 8001

Have a look at <https://twiki.cern.ch/twiki/bin/viewauth/CMSPublic/JobExitCodes> for a description of the exit codes.

我们通常会关注红框中的状态，其中 `Status on the scheduler` 是这个 job 的大致状态，`Jobs status` 则是一些具体的状态，`finished` 是指成功的，`failed` 是指失败的（废话），其他的就是一些中间状态不是很重要，当然成功的也可能不是很重要，主要的要解决可能出现的 `failed` 状态，你可以做这样一些尝试（下面需要用到 crab 文件夹的地方我做了简略替换）：

- `crab resubmit crab_xx` 偶尔几个文件 failed，通常是某些站点的问题，这时只需要 resubmit，让站点重新运行一下，多次 resubmit 或许就能解决问题。
- 查询 exit codes 根据输出你可以看到他会返回一个错误代码，你可以在他提示的网址中找到对应的代码，可以大致了解出错的原因

## Error codes currently sent from CMS jobs to the dashboard

⚠ indicates site error

- **Error exit code of the cmsRun application itself - range 0-10000**
- **Exit codes in 1-255 are standard ones in Unix and indicate a CMSSW abort that the cmsRun did not catch as exception**
  - 1 - 255 those exit codes are usually returned by `bash` when it is interrupted by a fatal signal. The exit code is set to `signal`
  - For convenience we have collected a list of most common such exit codes: [StandardExitCodes](#)
- **cmsRun (CMSSW) exit codes. range 7000-9000 \* These codes may depend on specific CMSSW version, the list**
  - // The first three are specific categories of CMS Exceptions.
  - 7000 - Exception from command line processing
  - 7001 - Configuration File Not Found
  - 7002 - Configuration File Read Error
  - 8001 - Other CMS Exception
  - 8002 - `std::exception` (other than `bad_alloc`)
  - 8003 - Unknown Exception
  - 8004 - `std::bad_alloc` (memory exhaustion)
  - 8005 - Bad Exception Type (e.g throwing a string)

- 查询 Dashboard 里的 log 在 CRAB status 中的红框上面一行会有一个 `Dashboard monitoring URL`，这个网址就显示了你 CRAB 任务的完整 log，其中 `JobLog` 里面就是运行时输出的 log。

比如我在经过之前的步骤发现都没啥用之后，在这里我就知道是我的程序出现了问题。

▼ Jobs Table - last retry only

Job details for 250831\_142639:zhuf\_crab\_2024Jpipi\_Run2024I-PromptReco-v1 -last retry only ⓘ

Id	Retry	Status	ExitCode	Submit		Start		Finish	Wall time	Site	Job Log	PostJob Log
				Submit	Start	Finish						
1	0	failed	8001	8月-31 14:28:39	8月-31 14:32:36	8月-31 14:34:12	00:01:35	T2_UK_London_IC		<a href="#">JobLog</a>	<a href="#">PostJob</a>	
10	0	failed	8001	8月-31 14:28:39	8月-31 14:29:42	8月-31 14:32:18	00:02:36	T1_US_FNAL		<a href="#">JobLog</a>	<a href="#">PostJob</a>	
100	0	failed	8001	8月-31 14:28:41	8月-31 14:32:34	8月-31 14:33:55	00:01:21	T2_CH_CSCS		<a href="#">JobLog</a>	<a href="#">PostJob</a>	
101	0	failed	8001	8月-31 14:28:46	8月-31 14:32:35	8月-31 14:33:52	00:01:17	T2_CH_CSCS		<a href="#">JobLog</a>	<a href="#">PostJob</a>	
102	0	failed	8001	8月-31 14:28:41	8月-31 14:32:33	8月-31 14:33:52	00:01:19	T2_CH_CSCS		<a href="#">JobLog</a>	<a href="#">PostJob</a>	
103	0	failed	8001	8月-31 14:28:46	8月-31 14:32:34	8月-31 14:34:07	00:01:33	T2_CH_CSCS		<a href="#">JobLog</a>	<a href="#">PostJob</a>	
104	0	failed	8001	8月-31 14:28:46	8月-31 14:32:35	8月-31 14:34:16	00:01:41	T3_UK_London_QMUL		<a href="#">JobLog</a>	<a href="#">PostJob</a>	

```

== CMSSW: 31-Aug-2025 15:33:54 BST Initiating request to open file
root://gfe02.grid.hep.ph.ic.ac.uk:1094//pnfs/hep.ph.ic.ac.uk/data/cms/store/data/Run2024I/ParkingDoubleMuonLowMass0/MINIAOD/b0384ec4f574.root
== CMSSW: 31-Aug-2025 15:34:03 BST Successfully opened file
root://gfe02.grid.hep.ph.ic.ac.uk:1094//pnfs/hep.ph.ic.ac.uk/data/cms/store/data/Run2024I/ParkingDoubleMuonLowMass0/MINIAOD/b0384ec4f574.root
== CMSSW: 31-Aug-2025 15:34:08 BST Closed file
root://gfe02.grid.hep.ph.ic.ac.uk:1094//pnfs/hep.ph.ic.ac.uk/data/cms/store/data/Run2024I/ParkingDoubleMuonLowMass0/MINIAOD/b0384ec4f574.root
== CMSSW: ----- Begin Fatal Exception 31-Aug-2025 15:34:08 BST -----
== CMSSW: An exception of category 'PluginNotFound' occurred while
== CMSSW: [0] Constructing the EventProcessor
== CMSSW: Exception Message:
== CMSSW: Unable to find plugin 'FourMuTwoPion' in category 'CMS EDM Framework Module'. Please check spelling of name.
== CMSSW: ----- End Fatal Exception -----
== CMSSW:
== CMSSW:      Scram Command Diagnostic:
== CMSSW:      Command : scramv1
== CMSSW:      Architecture: el9_amd64_gcc12
== CMSSW:      Executed: stdbuf -oL -eL cmsRun -j FrameworkJobReport.xml PSet.py > cmsRun-stdout.log.tmp 2>&1
== CMSSW:      Exit Status: 5
== CMSSW:      Stdout:
== CMSSW:      Stderr: None
===== CMSSW OUTPUT FINISHING =====
==== Report file creation STARTING at Sun Aug 31 14:34:08 2025 UTC ====
Sanitize FJR

```

这里既然我知道是我程序的问题，我修改完我的程序需要重新提交的时候，需要注意这些问题：

- 清理之前可能的输出文件 虽然crab在输出的时候会有一个按照提交时间分的文件夹，但还是建议在重新提交的时候将之前的输出删除，不仅仅是在存储空间的考虑，也有在之后的处理过程导致的数据重复的问题考虑。
- 删除之前提交CRAB时自动生成的crab\_xx文件夹 当你不做这一步时，会提示你已经存在了这个文件夹而导致不能提交。当然你也可以通过修改 `requestName` 以此来生成不一样的crab\_xx文件夹。
- 再次用 `crab submit crab3_xxx.py` 提交你的CRAB任务 注意和 `crab resubmit` 命令不同，resubmit不会因为你修改了你的程序或者别的一些配置而使用你最新的文件。

CRAB是一个很好用的工具，当你确保你的程序没问题之后，提交CRAB的时候的感觉是顺畅的，但是也要注意自己的存储空间是否够用哦！

## pre 2-Condor Jobs submit

当你交了很多CRAB之后就会输出很多的root文件，在进行下一步的处理的时候，也会遇到数量太多，没办法一次性全部处理的困境，那么就需要提交Condor Jobs让计算机并行处理。和CRAB能调用全球的计算资源不同，Condor使用的是当前集群的计算资源，他能输入的文件也只能是在当前集群上的文件。编写Condor提交所需要的文件其实更像编写一个自动化处理的脚本，因此写出一个能全自动的脚本是最重要的，下面我将用一个常用的例子来说明。如果下面的看不明白，可以参考[这个链接](#)获取更加完整的学习流程。

我们经常提交condor任务用来批量处理需要使用myntulpe.C程序处理的任务。在提交condor之前需要准备好所有要用到的程序文件，主要分为myntulpe.C以及runjobs.C程序相关，自动化脚本Run.sh，和condor jdl (job description language) file。

Run.sh中使用CMSSW的方式和在本地的使用方式不一样，但是逻辑是一样的，我的run jobs.C的全部输出都放到output文件夹里面了，所以要在脚本里要使用 `mkdir output` 创建这个文件夹之后再运行后面的程序。

```

1 #!/bin/bash
2 source /cvmfs/cms.cern.ch/cmsset_default.sh
3 export SCRAM_ARCH=el9_amd64_gcc12
4 eval `scramv1 project CMSSW CMSSW_14_0_4` 
5 cd CMSSW_14_0_4/src/
6 eval `scramv1 runtime -sh` 
7 cd ../../
8 mkdir -p output
9 root -b -q -l runjobs.C

```

下面是condor jdl文件 runOnCondor，里面指出运行脚本的文件 `Executable = Run.sh`，上传到condor的文件 `Transfer_Input_Files`，传输到本地的文件 `transfer_output_files`，condor运行时的log文件输出位置 `Output Error Log`，这里我都输出到 `logfile` 这个文件夹中，所以也需要先在本地 `mkdir logfile` 这个文件夹。

```

1 universe = vanilla
2 Executable = Run.sh
3 Should_Transfer_Files = YES
4 Transfer_Input_Files = "myntulpe.C, myntulpe.C.rootmap, myntulpe.h, myntulpe_C.d, myntulpe_C.so,
5 myntulpe_C_ACLiC_dict_rdict.pcm, runjobs.C"
6 transfer_output_files = output
7 WhenToTransferOutput = ON_EXIT
8 Output = logfile/condor_test_${Cluster}_${Process}.stdout
9 Error = logfile/condor_test_${Cluster}_${Process}.stderr
10 Log = logfile/condor_test_${Cluster}_${Process}.log
11 Queue 1

```

之后就能使用 `condor_submit runOnCondor` 提交这个文件。查询状态时用 `condor_q`，其中 `held` 是异常状态；想要移除condor任务使用 `condor_rm` 后面跟 `-all` 是移除所有的任务，跟对应任务的ID就能移除对应的任务了，`condor_q -better-analyze 680724.207` 跟对应任务的ID查看condor hold的原因。

```
[zhufeng@node00:~/condor-ce-test$condor_q

-- Schedd: node00.hepnnu.com : <192.168.40.2:9618?... @ 09/01/25 13:35:21
OWNER  BATCH_NAME  SUBMITTED  DONE  RUN  IDLE  TOTAL JOB_IDS
zhufeng ID: 675979  9/1 13:35  -  2  -  2 675979.0-1

Total for query: 2 jobs; 0 completed, 0 removed, 0 idle, 2 running, 0 held, 0 suspended
Total for zhufeng: 2 jobs; 0 completed, 0 removed, 0 idle, 2 running, 0 held, 0 suspended
Total for all users: 4 jobs; 0 completed, 0 removed, 0 idle, 2 running, 2 held, 0 suspended

[zhufeng@node00:~/condor-ce-test$condor_rm -all
All jobs have been marked for removal
[zhufeng@node00:~/condor-ce-test$condor_submit sleep-condor.jdl
Submitting job(s)..
2 job(s) submitted to cluster 675980.
[zhufeng@node00:~/condor-ce-test$condor_q

-- Schedd: node00.hepnnu.com : <192.168.40.2:9618?... @ 09/01/25 13:35:42
OWNER  BATCH_NAME  SUBMITTED  DONE  RUN  IDLE  TOTAL JOB_IDS
zhufeng ID: 675980  9/1 13:35  -  2  -  2 675980.0-1

Total for query: 2 jobs; 0 completed, 0 removed, 0 idle, 2 running, 0 held, 0 suspended
Total for zhufeng: 2 jobs; 0 completed, 0 removed, 0 idle, 2 running, 0 held, 0 suspended
Total for all users: 4 jobs; 0 completed, 0 removed, 0 idle, 2 running, 2 held, 0 suspended

[zhufeng@node00:~/condor-ce-test$condor_rm 675980
All jobs in cluster 675980 have been marked for removal
```

事实上，每个人都有自己提交condor的方式，相比于CRAB会更加自由一点，你可以先从学习使用别人的脚本开始，然后再加入一些自己的小创意。熟练使用这两个工具能让你的研究效率大大提高！

## 8-two muons & two tracks

在开始第九章之前，我们可以先做一个小工程帮助后面的理解。这次我们来重建 $\psi(2S) \rightarrow J/\psi + \pi^+\pi^-$ ,  $J/\psi \rightarrow \mu^+\mu^-$ 的过程，相比于之前two muons的重建，新加入了两个带电径迹pion，要想让他们都从同一个顶点即母粒子 $\psi(2S)$ 中衰变出来，就需要在之前的基础上再加两个track做顶点拟合，下面是两个部分对比，接下来就只需要对track进行类似的循环似乎就能找到我们想要的母粒子了！

```
vector < RefCountedKinematicParticle > muonParticles;
muonParticles.push_back(pmumuFactory.particle(muonPTT,
muon_mass, chi, ndf, muon_sigma));
muonParticles.push_back(pmumuFactory.particle(muonMTT,
muon_mass, chi, ndf, muon_sigma));
KinematicParticleVertexFitter fitter;
RefCountedKinematicTree psiVertexFitTree;
psiVertexFitTree = fitter.fit(muonParticles);
if ( !(psiVertexFitTree->isValid()) ) {continue;}
psiVertexFitTree->movePointerToTheTop();
RefCountedKinematicParticle psi_vFit_noMC =
psiVertexFitTree->currentParticle();
RefCountedKinematicVertex psi_vFit_vertex_noMC =
psiVertexFitTree->currentDecayVertex();

vector < RefCountedKinematicParticle > JPiPiParticles;
JPiPiParticles.push_back(JPiPiFactory.particle(trackTT1,
pion_mass, chi, ndf, pion_sigma));
JPiPiParticles.push_back(JPiPiFactory.particle(trackTT2,
pion_mass, chi, ndf, pion_sigma));
JPiPiParticles.push_back(pmumuFactory.particle(muon1TT,
muon_mass, chi, ndf, muon_sigma));
JPiPiParticles.push_back(pmumuFactory.particle(muon2TT,
muon_mass, chi, ndf, muon_sigma));
KinematicParticleVertexFitter fitter;
RefCountedKinematicTree JPiPiVertexFitTree;
JPiPiVertexFitTree = fitter.fit(JPiPiParticles);
if ( !(JPiPiVertexFitTree->isValid()) ) {continue;}
JPiPiVertexFitTree->movePointerToTheTop();
RefCountedKinematicParticle JPiPi_vFit_noMC =
JPiPiVertexFitTree->currentParticle();
RefCountedKinematicVertex JPiPi_vFit_vertex_noMC =
JPiPiVertexFitTree->currentDecayVertex();
```

对于track的token你需要这样获取，请考虑他们应该放入的地方，参考muon的token

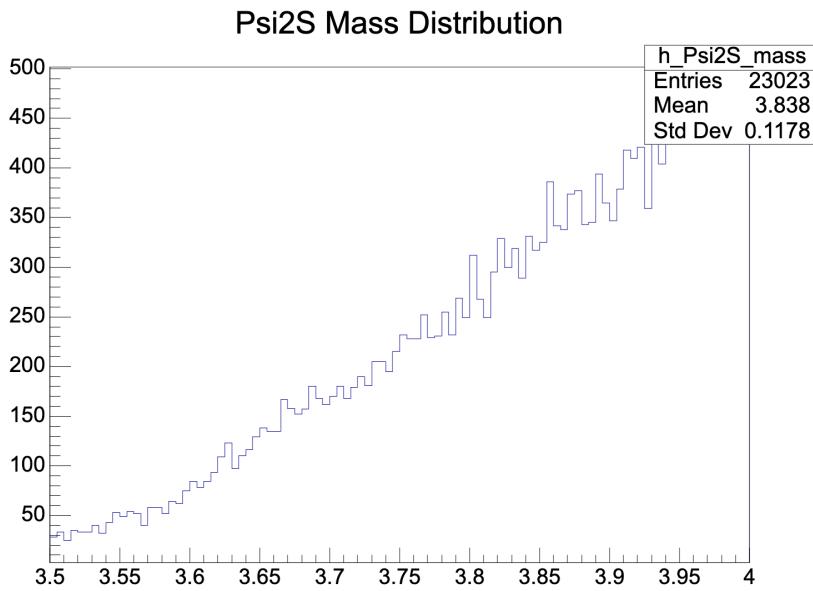
```
1 // in .h file
2 edm::EDGetTokenT<edm::View<pat::PackedCandidate>> trackToken_;
3 //edm::EDGetTokenT<edm::View<reco::Track>> trackToken_;
```

```

4  //////
5
6
7  /////in .cc file
8  trackToken_ = consumes<edm::View<pat::PackedCandidate>>(edm::InputTag("packedPFCandidates"));
9  //trackToken_ = consumes<edm::View<reco::Track>>(edm::InputTag("displacedTracks"));
10 //////
11 edm::Handle<edm::View<pat::PackedCandidate>> theTrackHandle;
12 // edm::Handle<edm::View<reco::Track>> theTrackHandle;
13 iEvent.getByToken(trackToken_,theTrackHandle );
14
15 //////
16 // for (edm::View<reco::Track>::const_iterator iTrack1 = theTrackHandle->begin(); iTrack1 != theTrackHandle-
17 //>end(); ++iTrack1) {
18 for (edm::View<pat::PackedCandidate>::const_iterator iTrack1 = theTrackHandle->begin(); iTrack1 !=
19 theTrackHandle->end(); ++iTrack1) {
    if (((iTrack1->px() == iMuon2->track()->px() && iTrack1->py() == iMuon2->track()->py() &&
    iTrack1->pz() == iMuon2->track()->pz()) || ( iTrack1->px() == iMuon1->track()->px() && iTrack1->py() == iMuon1-
    >track()->py() && iTrack1->pz() == iMuon1->track()->pz())) { continue; } //避免这个带电径迹是muon留下的
}

```

当你类似two muons那样去输出 `JPiPi_vFit_noMC->currentState().mass()` 就能得到  $J/\psi\pi^+\pi^-$  的不变质量谱,  $\psi(2S)$  的质量为 3.686GeV, 那么你就能在  $J/\psi\pi^+\pi^-$  的不变质量谱的 3.6GeV 位置看到一个峰了 (吗?)



当你用上面的代码处理了多个dataset之后终于得到一个拥有足够统计量的结果之后, 发现不变质量谱在3.6GeV位置并没能如愿出现  $\psi(2S)$  的峰, 类似上图, 是代码问题还是条件不够紧? 一个主要的原因是, 我们希望找到的是  $\psi(2S) \rightarrow J/\psi + \pi^+\pi^-$ ,  $J/\psi \rightarrow \mu^+\mu^-$  的过程, 那随便两个  $\mu$  和两个  $\pi$  能拟合到一个顶点并不意味着, 两个  $\mu$  就是从  $J/\psi$  衰变出来的, 这就导致很大部分都只是本底。要想避免这种情况, 有两种可以想到的方法, 一种是我选取的两个  $\mu$  是不变质量为  $J/\psi$  质量附近的, 根据之前做的mumuonly的质量谱可以大致选一个  $3.0GeV < M_{\mu^+\mu^-} < 3.2GeV$  的质量范围的两个  $\mu$ , 让他们和两个  $\pi$  做顶点拟合, 似乎能让其中的本底事件减少很多。但总觉得其中还有一些奇怪的地方.....当你去 [PDG](#) 查询  $J/\psi$  粒子的信息的时候会发现, 他的衰变宽度只有 0.0000926GeV! 而在我们的不变质量谱上, 位于 3.096GeV 的峰的宽度很显然比这个值要大得多, WHY? 我们的探测器出现问题了吗?

事实上这受到探测器的分辨率的影响, 按理来说在真实的物理世界中, 如果我们能精密测量  $\mu$  的动量的话, 我们重建出来的  $J/\psi$  峰的形状应该像一个  $\delta$  函数。但我们的探测器做不到这一点, 他能测量出来的动量是有误差的, 而且因为这个过程是:  $\mu$  子打在探测器上->留下电信号的空间位置->拟合出多个电信号组成轨迹->根据轨迹半径计算  $\mu$  的动量。其中的探测器就像是一个一个像素点, 如果点亮了就说明  $\mu$  子来过, 这个像素点很显然不能无限精细, 这就有了分辨率这一说, 就像 4K 与 360P 的区别。既然有分辨率导致的不准确, 也就会导致  $J/\psi$  质量附近变得“模糊”了 (当你摘下眼镜看向月亮, 就会看到一坨月亮), 那么如何修正这一点呢? 我们需要做 mass constraint! 关于探测器分辨率的影响效果请参考 [这里](#)。

正如其名, 我们要将  $J/\psi$  附近的模糊的  $\mu\mu$  动量通过一些拟合修正, 把他们的不变质量重新约束到 3.096GeV! 这里要说明一点的是, 并不是所有的粒子都适合做 mass constraint, 像衰变宽度很大的粒子就不适合, 这会改变原有的物理性质。了解了这些之后, 让我们开始代码的书写吧!

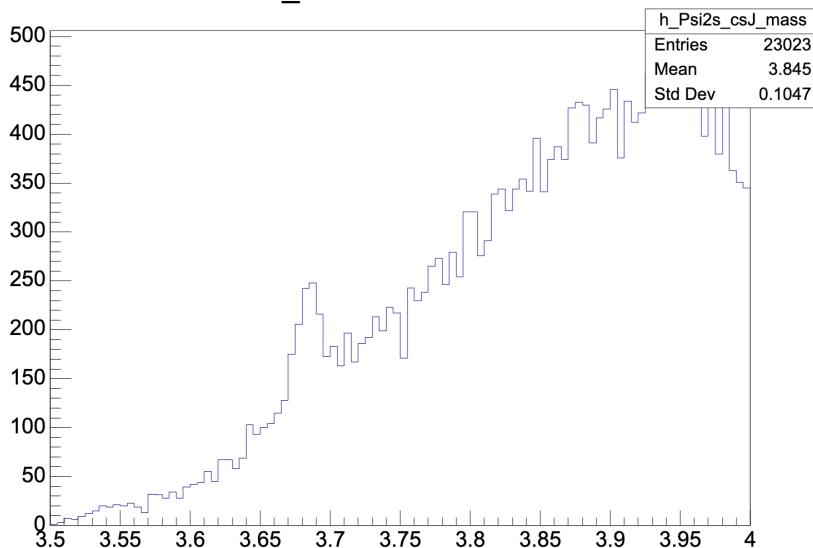
```

1 TransientTrack muon1TT(muTrack1, &(bFieldHandle));
2 TransientTrack muon2TT(muTrack2, &(bFieldHandle));
3 KinematicParticleFactoryFromTransientTrack pmumuFactory;
4 ParticleMass muon_mass = myMumass; // pdg mass
5 float muon_sigma = myMumasserr;
6 float chi = 0.;
7 float ndf = 0.;
8 vector < RefCountedKinematicParticle > muonParticles;
9 muonParticles.push_back(pmumuFactory.particle(muon1TT, muon_mass, chi, ndf, muon_sigma));
10 muonParticles.push_back(pmumuFactory.particle(muon2TT, muon_mass, chi, ndf, muon_sigma));
11 KinematicParticleVertexFitter fitter;
12 RefCountedKinematicTree jpsi1VertexFitTree;
13 jpsi1VertexFitTree = fitter.fit(muonParticles);
14
15 if (!jpsi1VertexFitTree->isValid()) {continue;}
16 jpsi1VertexFitTree->movePointerToTheTop();
17 RefCountedKinematicParticle jpsi1_vFit_noMC = jpsi1VertexFitTree->currentParticle();
18 RefCountedKinematicVertex jpsi1_vFit_vertex_noMC = jpsi1VertexFitTree->currentDecayVertex();
19 KinematicParameters mymumupara = jpsi1_vFit_noMC->currentState().kinematicParameters();
20 double jpsi1_vtxprob=ChiSquaredProbability( (double)(jpsi1_vFit_vertex_noMC->chiSquared()),(double)
(jpsi1_vFit_vertex_noMC->degreesOfFreedom()));
21 //if(jpsi1_vFit_noMC->currentState().mass()>3.4 || jpsi1_vFit_noMC->currentState().mass()<2.8) {continue;}
22 //if( jpsi1_vtxprob<vtxprobprecut) {continue;}
23 //以上和mumuonly一致
24
25 //mass constrain for jpsi1 from psi2s:
26 const double myJmass=3.0969, myJmasserr=0.00004;
27 KinematicConstraint *jpsi1_cs = new MassKinematicConstraint(myJmass,myJmasserr); //设置你想constraint的粒子参数
28 KinematicParticleFitter csfitter;
29 jpsi1VertexFitTree = csfitter.fit(jpsi1_cs,jpsi1VertexFitTree);
30 jpsi1VertexFitTree->movePointerToTheTop();
31 RefCountedKinematicParticle jpsi1_vFit_cs = jpsi1VertexFitTree->currentParticle();
32 RefCountedKinematicVertex jpsi1_vFit_vertex_cs = jpsi1VertexFitTree->currentDecayVertex();
33 KinematicParameters mymumupara_cs = jpsi1_vFit_cs->currentState().kinematicParameters();
34
35
36 //////////////////////////////////////////////////////////////////
37 vector < RefCountedKinematicParticle > csJ_PiPiParticles;
38 csJ_PiPiParticles.push_back(JPiPiFactory.particle(trackTT1, pion_mass, chi, ndf, pion_sigma));
39 csJ_PiPiParticles.push_back(JPiPiFactory.particle(trackTT2, pion_mass, chi, ndf, pion_sigma));
40 csJ_PiPiParticles.push_back(jpsi1_vFit_cs);
41
42 KinematicParticleVertexFitter fitter_csJ;
43 RefCountedKinematicTree JPiPiVertexFitTree_csJ;
44 JPiPiVertexFitTree_csJ = fitter_csJ.fit(csJ_PiPiParticles);
45 if( !(JPiPiVertexFitTree_csJ->isValid()) ) {continue;}
46 JPiPiVertexFitTree_csJ->movePointerToTheTop();
47 RefCountedKinematicParticle JPiPi_vFit_csJ = JPiPiVertexFitTree_csJ->currentParticle();
48 RefCountedKinematicVertex JPiPi_vFit_vertex_csJ = JPiPiVertexFitTree_csJ->currentDecayVertex();

```

如此一来，我们再输出经过mass constraint的 $J/\psi \pi\pi$ 的不变质量谱之后，就能很明显在3.686GeV附近看到一个尖峰了，当然要想压低本底信号还需要进一步的研究，如果你感兴趣的话你或许能在3.872GeV找到另外一个小峰，这就是著名的X(3872)。但那又是另一个故事了，我们的故事还要继续.....

### Psi2s\_csJ Mass Distribution



## 9-four muons 拟合

p.s. 下面的内容如果直接拷贝可能会出现一些变量没提及或未定义的情况，因为我只会介绍的逻辑，一些细枝末节可能会因为我的粗心未能完全包含，不过，相信你经过之前的练习已经能完全对付这些小问题了！

在之前的分析中，我们成功的对两个muon进行拟合并得到一些令人兴奋的结果。但我们的征途并不会止步于此，在接下来的章节里我们将研究四个muon，多了一倍的数量的muon带来的是更多的变量和更复杂的筛选，以及一些更有趣的结果！

```

1 if (thePATMuonHandle->size()>=4 ) {
2     for (iMuon1 = thePATMuonHandle->begin(); iMuon1 != thePATMuonHandle->end(); ++iMuon1) {
3         TrackRef muTrack1 = iMuon1->track();
4         if (muTrack1.isNull()) {continue;}
5         reco::Track recoMu1 = *iMuon1->track();
6         for (iMuon2 = iMuon1 + 1; iMuon2 != thePATMuonHandle->end(); ++iMuon2) {
7             TrackRef muTrack2 = iMuon2->track();
8             if (muTrack2.isNull()) {continue;}
9             reco::Track recoMu2 = *iMuon2->track();
10            for (iMuon3 = iMuon2 + 1; iMuon3 != thePATMuonHandle->end(); ++iMuon3) {
11                TrackRef muTrack3 = iMuon3->track();
12                if (muTrack3.isNull()) {continue;}
13                reco::Track recoMu3 = *iMuon3->track();
14                for (iMuon4 = iMuon3 + 1; iMuon4 != thePATMuonHandle->end(); ++iMuon4) {
15                    TrackRef muTrack4 = iMuon4->track();
16                    if (muTrack4.isNull()) {continue;}
17                    reco::Track recoMu4 = *iMuon4->track();
18                    if ( (iMuon1->charge() + iMuon2->charge() + iMuon3->charge() + iMuon4->charge() ) == 0 ) {

```

循环和之前的类似，通过遍历一个事例中的所有 $\mu$ 来进行后续的配对操作。你可以在此之前添加一个条件 `if (thePATMuonHandle->size()>=4)` 以确保你这个事例中确实是有四个或四个以上的 $\mu$ 的，这样你才能进行后面的操作。同样的，如果我们想重建两个中性粒子的话，需要要求四个 $\mu$ 的电荷加和为零，这样能经过这个条件的 $\mu$ 必定是 $\mu^+\mu^-\mu^+\mu^-$ ，但是在程序中我们得到的只是 $\mu_1\mu_2\mu_3\mu_4$ ，不同于两个 $\mu$ ，四个 $\mu$ 就代表着如果两两组合的话，我们就会有三种不同的组合

$$\mu_1\mu_2, \mu_3\mu_4 \mid \mu_1\mu_3, \mu_2\mu_4 \mid \mu_1\mu_4, \mu_2\mu_3$$

而在这三种组合中，再考虑两两组合里的两个 $\mu$ 的电荷之和也应该为0，这样其实最终只有两种组合是正确的，请注意， $\mu_1\mu_2\mu_3\mu_4$ 的电荷排序并不一定就是 $\mu^+\mu^-\mu^+\mu^-$ ，所以不代表 $\mu_1\mu_3, \mu_2\mu_4$ 就一定是错误的组合（你需要思考！）那么为了剔除其中错误的组合，你可以选择在myntuple.C进行筛选，也可以在.cc这里就进行筛选。前者是我们最后主要的处理方式，这里为了逻辑的连续性我会展示在.cc的处理方式，也为后续更高级的处理方式做一个铺垫。

其实很简单，就是把电荷之和为零的填入到一个二维数组里。

```

1 if ( (iMuon1->charge() + iMuon2->charge() + iMuon3->charge() + iMuon4->charge() ) == 0 ) {
2     std::vector<std::pair<const edm::View<pat::Muon>::const_iterator, const
      edm::View<pat::Muon>::const_iterator>> muonPairs;

```

```

3  if ((iMuon1->charge() + iMuon2->charge()) == 0) {
4      muonPairs.push_back(std::make_pair(iMuon1, iMuon2));
5      muonPairs.push_back(std::make_pair(iMuon3, iMuon4));
6  }
7  if ((iMuon1->charge() + iMuon3->charge()) == 0) {
8      muonPairs.push_back(std::make_pair(iMuon1, iMuon3));
9      muonPairs.push_back(std::make_pair(iMuon2, iMuon4));
10 }
11 if ((iMuon1->charge() + iMuon4->charge()) == 0) {
12     muonPairs.push_back(std::make_pair(iMuon1, iMuon4));
13     muonPairs.push_back(std::make_pair(iMuon2, iMuon3));
14 }
15 for (unsigned int i = 0; i < muonPairs.size(); i += 2) {
16     const auto& muon1 = muonPairs[i].first;
17     const auto& muon2 = muonPairs[i].second;
18     const auto& muon3 = muonPairs[i+1].first;
19     const auto& muon4 = muonPairs[i+1].second;
20     //下面的代码和两个muon的部分十分相似，相信你能很容易看懂其中的意义
21     TransientTrack muonPTT1(muon1->track(), &(bFieldHandle));
22     TransientTrack muonMTT2(muon2->track(), &(bFieldHandle));
23     TransientTrack muonPTT3(muon3->track(), &(bFieldHandle));
24     TransientTrack muonMTT4(muon4->track(), &(bFieldHandle));
25     KinematicParticleFactoryFromTransientTrack pmumuFactory;
26     ParticleMass muon_mass = myMumass;
27     float muon_sigma = myMumasserr;
28     float chi = 0.;
29     float ndf = 0.;
30     vector < RefCountedKinematicParticle > muonParticles1;
31     muonParticles1.push_back(pmumuFactory.particle(muonPTT1, muon_mass, chi, ndf, muon_sigma));
32     muonParticles1.push_back(pmumuFactory.particle(muonMTT2, muon_mass, chi, ndf, muon_sigma));
33     muonParticles1.push_back(pmumuFactory.particle(muonPTT3, muon_mass, chi, ndf, muon_sigma));
34     muonParticles1.push_back(pmumuFactory.particle(muonMTT4, muon_mass, chi, ndf, muon_sigma));
35     KinematicParticleVertexFitter fitter;
36     RefCountedKinematicTree psiVertexFitTree1;
37     psiVertexFitTree1 = fitter.fit(muonParticles1);
38
39 if (psiVertexFitTree1->isValid()) {
40     //这里我删去了输出fourmuonmass的部分，直接输出各muon的动量
41     //child 的顺序与填入muonParticles1的顺序一致，所以与muon1的排序一致
42     psiVertexFitTree1->movePointerToTheFirstChild();
43     RefCountedKinematicParticle mu1CandMC = psiVertexFitTree1->currentParticle();
44     KinematicParameters myFourMuonMu1KP= mu1CandMC->currentState().kinematicParameters();
45     psiVertexFitTree1->movePointerToTheNextChild();
46     RefCountedKinematicParticle mu2CandMC = psiVertexFitTree1->currentParticle();
47     KinematicParameters myFourMuonMu2KP= mu2CandMC->currentState().kinematicParameters();
48     psiVertexFitTree1->movePointerToTheNextChild();
49     RefCountedKinematicParticle mu3CandMC = psiVertexFitTree1->currentParticle();
50     KinematicParameters myFourMuonMu3KP= mu3CandMC->currentState().kinematicParameters();
51     psiVertexFitTree1->movePointerToTheNextChild();
52     RefCountedKinematicParticle mu4CandMC = psiVertexFitTree1->currentParticle();
53     KinematicParameters myFourMuonMu4KP= mu4CandMC->currentState().kinematicParameters();
54
55     //下面需要输出四个muon的动量，这样我们就能获得这些muon的四动量
56     MyFourMuonMu1Px->push_back(myFourMuonMu1KP.momentum().x());
57     MyFourMuonMu1Py->push_back(myFourMuonMu1KP.momentum().y());
58     MyFourMuonMu1Pz->push_back(myFourMuonMu1KP.momentum().z());
59     MyFourMuonMu2Px->push_back(myFourMuonMu2KP.momentum().x());
60     MyFourMuonMu2Py->push_back(myFourMuonMu2KP.momentum().y());
61     MyFourMuonMu2Pz->push_back(myFourMuonMu2KP.momentum().z());
62     MyFourMuonMu3Px->push_back(myFourMuonMu3KP.momentum().x());
63     MyFourMuonMu3Py->push_back(myFourMuonMu3KP.momentum().y());
64     MyFourMuonMu3Pz->push_back(myFourMuonMu3KP.momentum().z());
65     MyFourMuonMu4Px->push_back(myFourMuonMu4KP.momentum().x());
66     MyFourMuonMu4Py->push_back(myFourMuonMu4KP.momentum().y());
67     MyFourMuonMu4Pz->push_back(myFourMuonMu4KP.momentum().z());
68 }
```

```

69     mumuonlymu1Idx->push_back(std::distance(thePATMuonHandle->begin(), muon1));
70     mumuonlymu2Idx->push_back(std::distance(thePATMuonHandle->begin(), muon2));
71     mumuonlymu3Idx->push_back(std::distance(thePATMuonHandle->begin(), muon3));
72     mumuonlymu4Idx->push_back(std::distance(thePATMuonHandle->begin(), muon4));

```

如果你不想在.cc中进行这一步筛选，直接把对应 `muon1` 替换为 `iMuon1` 就好了。但你同时还需要输出对应 $\mu$ 的电荷以便于之后在myntuple.C中进行筛选，在 [Muon ID and Muon block](#) 章节中其实已经对所有的 $\mu$ 按照顺序输出了他们的电荷，那么你就可以用 `(*muCharge)[(*muon1Idx)[myi]]` 的方式在myntuple.C中获取对应 $\mu$ 的电荷了（这里只是作为提示，请思考这里的myi一个是取哪个变量的 `size()`）

如此一来在myntuple.C中的处理变得心应手了，请利用下面的提示去获取你的 $\mu\mu$ 组合的质量，这样就能画出两个不变质量谱，甚至可以画出一个二维分布图！

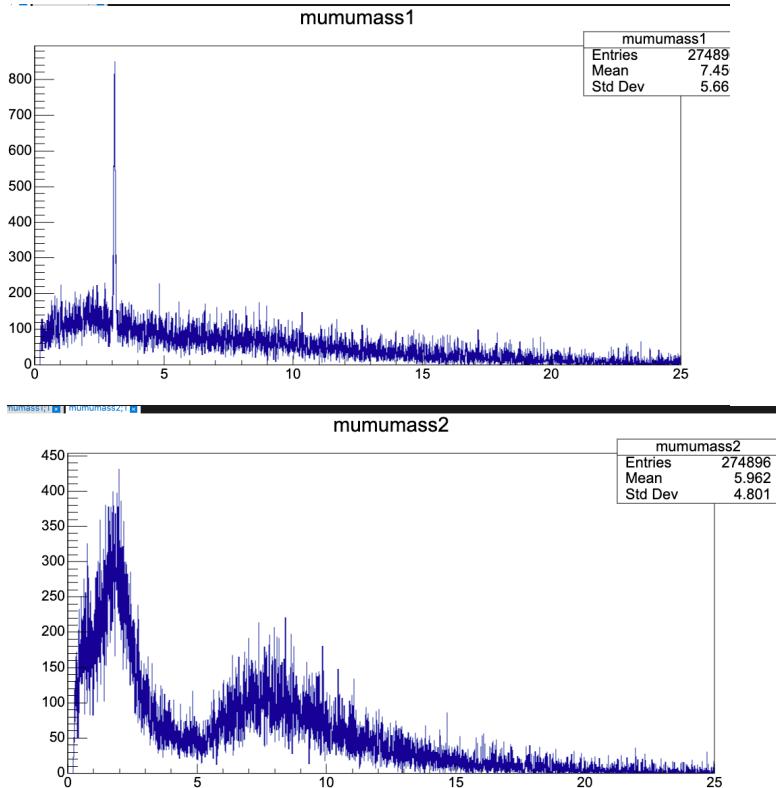
```

1 #include "TLorentzVector.h"
2 TH2F* mumumassJJ = new TH2F("mumumassJJ", "mumumassJJ", 60, 2.8, 3.4, 60, 2.8, 3.4);
3 ...
4 TLorentzVector Muon1FourVector;
5 Muon1FourVector.SetXYZM((*muPx)[(*MyFourMuonMulIdx)[myi]], (*muPy)[(*MyFourMuonMulIdx)[myi]], (*muPz)
6 [(*MyFourMuonMulIdx)[myi]], muonMass);
7 std::cout << Muon1FourVector.M() << Muon1FourVector.Pt() << std::endl;
8 ...
9 mumumassJJ->Fill(mumumass1, mumumass2);

```

但是，however, だがしかし、你或许会发现你的第二对 $\mu$ 的不变质量看起来很奇怪，为什么第一对可以清楚地看到峰，而第二对却不行？下面是我的初步解释，你可以用一些方法验证我说的是否正确

一个事例中的 $\mu$ 子的动量是按动量从大到小进行排序的，第一对的两个 $\mu$ 子往往会得到更大的横动量pt，从而拥有更低的本底，因此要想在第二对 $\mu$ 中看到峰可以对 $\mu$ 的pt进行限制



然而当对本底进行压低的时候也意味着信号会有相应的损失，因此一个root文件的数据已经不太够了，你需要尝试多run几个文件，在 [cms Data Aggregation System](#) 中我已经筛选出所需要的root文件，你可以用类似下面的命令下载到你的目录下

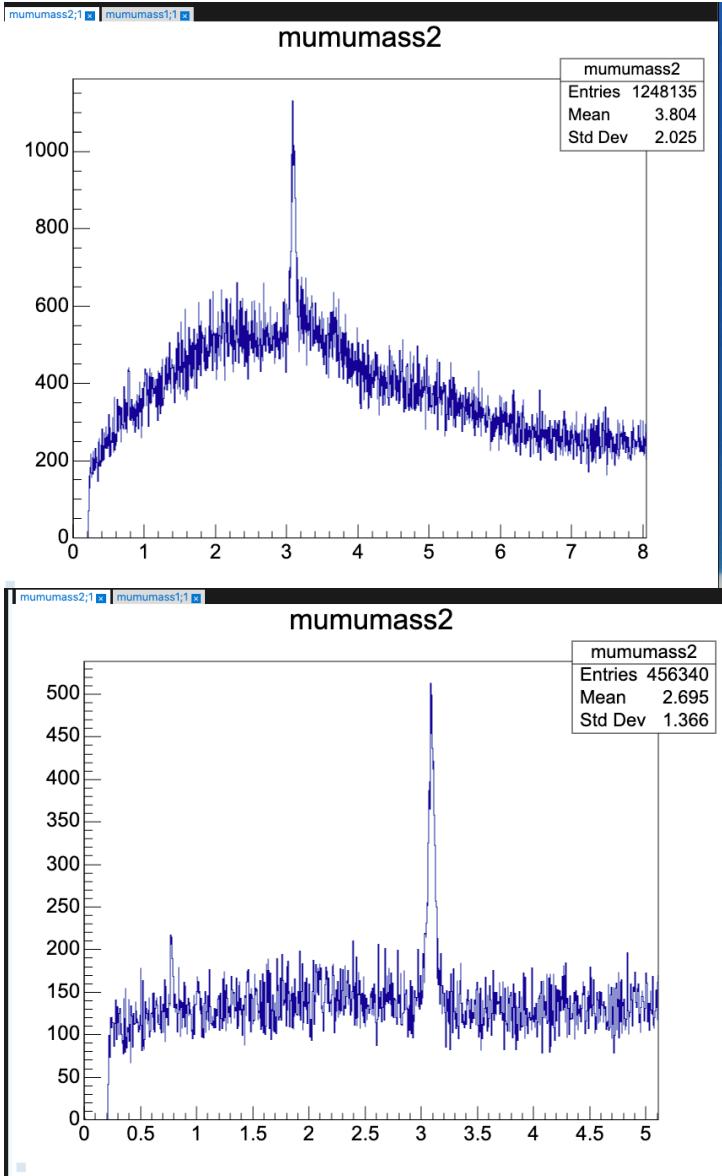
```
xrroot -d 1 -f root://xrootd-cms.infn.it//store/data/Run2023B/ParkingDoubleMuonLowMass0/MINIAOD/22Sep2023-
v1/2560000/0cb0075f-5308-4849-a0ee-f451012a0c7c.root .
```

下载合适数量的文件后（10个左右）在runMultiLepPAT\_dataRun3\_miniAOD.py中修改inputFiles，拿到数据之后，在myntuple.C中可以设置每个 $\mu$ 子的pt大于1.5或者2应该就能看到峰啦！下图中左边是 `pt>1.5` 右边是 `pt>2`

```

1 ivars.inputFiles=(
2   'file:f8e93985-e14c-4a8a-b28b-f8cceb3c878e.root', #可以通过添加逗号并行多个root文件
3   'file:847912fd-20d4-4cbe-80bf-3eeade7bcb1f.root',
4   'root://xrootd-cms.infn.it//store/data/Run2023B/ParkingDoubleMuonLowMass0/MINIAOD/22Sep2023-
5   v1/2560000/0cb0075f-5308-4849-a0ee-f451012a0c7c.root'#也可以不下载直接写进das上的路径
)

```



## 10-mass constraint

**Note:** 在此之前, 在下面的文件夹内包含相关的文件, .cc是截止到4muon部分的所有代码, 你可以把他覆盖掉你原先自己写的.cc, .h是包含所有所需的变量, 这些变量即使没用到也没有关系, 另外还有一个作为提示的完整代码。这部分重复的结构很多, 在理解其中的逻辑之后请仔细确认变量名, be patient!

用下面的路径下的文件吧!

```

/afs/cern.ch/user/z/zhuf/public/UserCode/massconst
--->MuMuEEPat.cc
--->MuMuEEPat.h
--->MultiLepPAT.cc_hint

```

新的.cc文件中调整了一些循环结构让整体逻辑更简洁明了

```

-- for iMuonP 这里对所有的 $\mu$ 遍历, 输出所有的信息
-> raw_muonPx, muonID, Index...

```

```

if Handle->size()>=2: 如果有大于2个的μ的话
-- for iMuon1 iMuon2
    -> mumuonly fit_muonPx
if Handle->size()>=4: 如果有大于4个的μ的话
-- for iMuon3 iMuon4
    -> fourmuon
    -> mass constraint for each pairs (we are here!)

```

在之前four muon 拟合我们提到有三种不同的配对组合：

$\mu_1\mu_2, \mu_3\mu_4 \mid \mu_1\mu_3, \mu_2\mu_4 \mid \mu_1\mu_4, \mu_2\mu_3$

那么如果我们需要对每对μ重建出来的 $J/\psi$ 做mass constraint 的话就需要按照不同的电荷条件重复3次，比如我们先做第一组：

```

1  if(muon1TT.charge()+muon2TT.charge() == 0)  {
2      muonP12.push_back(pFactory.particle(muon1TT, muon_mass, chi, ndf, muon_sigma));
3      muonP12.push_back(pFactory.particle(muon2TT, muon_mass, chi, ndf, muon_sigma));
4      muonP34.push_back(pFactory.particle(muon3TT, muon_mass, chi, ndf, muon_sigma));
5      muonP34.push_back(pFactory.particle(muon4TT, muon_mass, chi, ndf, muon_sigma));
6      RefCountedKinematicTree Jpsi1 = kpvFitter.fit(muonP12);
7      RefCountedKinematicTree Jpsi2 = kpvFitter.fit(muonP34);
8
9      RefCountedKinematicTree Jpsi1noMCJJ = kpvFitter.fit(muonP12);
10     RefCountedKinematicTree Jpsi2noMCJJ = kpvFitter.fit(muonP34);
11     if (Jpsi1->isValid() && Jpsi2->isValid()){
12         .....//这部分是在分别取两个J/\psi的物理信息，和之前的类似，不做过多解释
13         if(doJPSIMassCost) //这部分做mass constraint，你可以把这个if去掉不用判断
14             RefCountedKinematicTree Chi1_bTree;
15             RefCountedKinematicParticle MyChi1_part;
16
17             //JJ assumption
18             jp1 = myJmass;  jp_m_sigma1 = myJmasserr;
19             jp2 = myJmass;  jp_m_sigma2 = myJmasserr;
20             jpsi_c1 = new MassKinematicConstraint(jp1,jp_m_sigma1);
21             jpsi_c2 = new MassKinematicConstraint(jp2,jp_m_sigma2); //设置你想constraint的粒子参数
22             try{ Jpsi1 = csFitter.fit(jpsi_c1,Jpsi1noMCJJ);} catch (VertexException const& x) { cout<<"mu12 vertex
exception with mass constrained to J!"<<endl; }
23             try{ Jpsi2 = csFitter.fit(jpsi_c2,Jpsi2noMCJJ);} catch (VertexException const& x) { cout<<"mu34 vertex
exception with mass constrained to J!"<<endl; } //尝试将之前的mumu组合constraint到你设置的参数上，并将结果覆盖进Jpsi1
中
24             if(Jpsi1->isEmpty() != true && Jpsi2->isEmpty() != true) //如果Jpsi1和Jpsi2拟合成功且非空，提取当前粒子并将其添加
到Chi_1列表中
25             Jpsi1->movePointerToTheTop();
26             Jpsi2->movePointerToTheTop();
27             Jpsi1_part = Jpsi1->currentParticle();
28             Jpsi2_part = Jpsi2->currentParticle();
29             Chi_1.push_back(Jpsi1_part);
30             Chi_1.push_back(Jpsi2_part);
31             bool isagoodfit=true;
32             try{ Chi1_bTree = kpvFitter.fit(Chi_1); } catch (VertexException const& x) {isagoodfit=false;
cout<<"mu12 and mu34 vertex exception with mu12 and mu34 constrained to JJ"<<endl; } //相当于先分别对两个muon拟合顶
点到J/\psi，再对两个经过mass constraint后的两个J/\psi再进行顶点拟合
33             if(Chi1_bTree->isValid() && isagoodfit) {//如果拟合成功就提取J/\psi物理信息
34                 .....//省略
35             }
36         }
37         delete jpsi_c1; delete jpsi_c2;
38     //end JJ assumption

```

Note: 关于Ctau的定义

需要一个洛伦兹不变量表示粒子的寿命。

实验室系下:  $\tau = \gamma\tau_0$ ;  $\tau_0$ 为粒子的静止寿命  $\gamma = \frac{1}{\sqrt{1-\beta^2}}$ ,  $\beta = \frac{v}{c}$

衰变长度:  $L = v\tau = \gamma\beta c\tau_0$ ; 其中 $c\tau_0$ 就可认为是洛伦兹不变的

其中 $p = \gamma m_0 \beta$ , 带入可得 $c\tau_0 = \frac{L}{\gamma\beta} = \frac{Lm_0}{p}$

指得注意的是，这里的 $c\tau_0$ 是在动量方向和位移方向平行的情况下，当有呈现一定角度的时候也只有在平行于位移方向上才会有洛伦兹收缩的现象，因此在实际的计算中还需要有一个夹角 $\cos\alpha$ ，下面是在.h中找到的函数定义

```

1 virtual double GetcTau(RefCountedKinematicVertex& decayVrtx, RefCountedKinematicParticle& kinePart, Vertex&
2   bs)
3   {
4     TVector3 vtx;
5     TVector3 pvtx;
6     vtx.SetXYZ((*decayVrtx).position().x(), (*decayVrtx).position().y(), 0);
7     pvtx.SetXYZ(bs.position().x(), bs.position().y(), 0);
8     VertexDistanceXY vdistXY;
9     TVector3 pperp(kinePart->currentState().globalMomentum().x(),
10                  kinePart->currentState().globalMomentum().y(), 0);
11
12     TVector3 vdiff = vtx - pvtx;
13     double cosAlpha = vdiff.Dot(pperp) / (vdiff.Perp() * pperp.Perp());
14     Measurement1D distXY = vdistXY.distance(Vertex(*decayVrtx), Vertex(bs));
15     double ctauPV = distXY.value() * cosAlpha * kinePart->currentState().mass() / pperp.Perp();
16     return ctauPV;
17   }

```

以上对于.cc的内容基本上就结束了。我们查看完整版的代码会发现，里面除了JJ的mass constraint 还有很多其他衰变道的代码，而那些代码之间并无太多区别，在这里就不再重复。然而学习并不会就此终止，.cc中的筛选条件只是一些最最基础的，他最主要的作用还是做一些拟合，和输出对应的物理量，而更细致的筛选则是在myntuple.C中完成的。那么在接下来的章节中我会继续介绍myntuple.C的内容。

(以上内容完成于13/09/2024)

## 11-myntuple 进阶

**Note:** 以下的任务推荐在清华集群中完成，你可以使用下面路径中的mymultilep.root文件进行 MakeClass 的操作，生成和编写你的myntuple.C文件

Data path: 这里是你可以用的数据文件，你需要在之前完整版的cc找到你需要的变量名

/home/storage0/users/llchen/dataMINI/Run3/ReReco2023/2023B/ParkingDoubleMuonLowMass0

myntuple\_hint: 这是所有衰变道的整合文件，里面很乱，选取你需要的部分

/home/storage0/users/zhufeng/formymultilep\_learning/myntuple.C\_hint

### 数据输入的管理和排序

在之前的myntuple.C中我们学会了一些简单的画图操作，这很显然不能满足我们日渐增长的筛选需求，并且随着four muon和mass constraint 的加入，带来了更多的变量，我们也需要对这些变量进行管理，下面是一个思路：

在.cc中 $\mu$ 的变量是以四个为一组进行输出的，同样的在.C中我们也会以同样的思路获取，比如一个 $\mu$ 子的charge和fourLorentzVector，就可以把他们四个为一组输入到一个数组里：（下面不是代码，只是作为逻辑展示）

我们可以把 $\mu$ 物理量按照顺序填入到数组里：

MuCharge: {charge\_mu1, charge\_mu2, charge\_mu3, charge\_mu4}

Mu4vect: {4vect\_mu1, 4vect\_mu2, 4vect\_mu3, 4vect\_mu4}

如果所有的变量都按照这个顺序填入到一个对应的数组里的话，那么我们只需要 变量名+序号 就能获取对应 $\mu$ 子的物理量了，比如我们想知道第3个 $\mu$ 的Px，那只需要用到 (MuPx)[2] 就能得到，前提是你创建了 MuPx 这个数组并按照正确的顺序填入了这个变量。

既然我们用合适的方式管理了4个 $\mu$ 子的物理量，接下来就是对这4个 $\mu$ 子进行配对了。之前我介绍了用 if 条件判断电荷相加是否为零的方式来配对他们，如果为零则输出配对之后的两个 $\mu$ 的不变质量，而且也已经知道三种配对方式中只有两对是合适的。在条件很少的情况下这种方式确实能很方便的得到我们想要的东西，但我们需要找到一个公用的标签去筛选和组合这些物理量，上面的管理就是为了这一步做准备的。

还记得刚刚提到的序号吗，事实上我们如果对序号进行组合和排序的话，这样我们在完成操作之后只需要找需要对应的物理量输出出来就好了！比如：

对序号进行组合： myComblIdx[3] = {0,1,2,3}, {0,2,1,3}, {0,3,1,2},

分别对应  $\mu_1\mu_2, \mu_3\mu_4$  |  $\mu_1\mu_3, \mu_2\mu_4$  |  $\mu_1\mu_4, \mu_2\mu_3$  的组合方式

比如我们想看 $\mu_1\mu_3$ 这个组合是否满足电荷相加为零的条件，如果满足则输出 $\mu_1\mu_3$ 的不变质量：

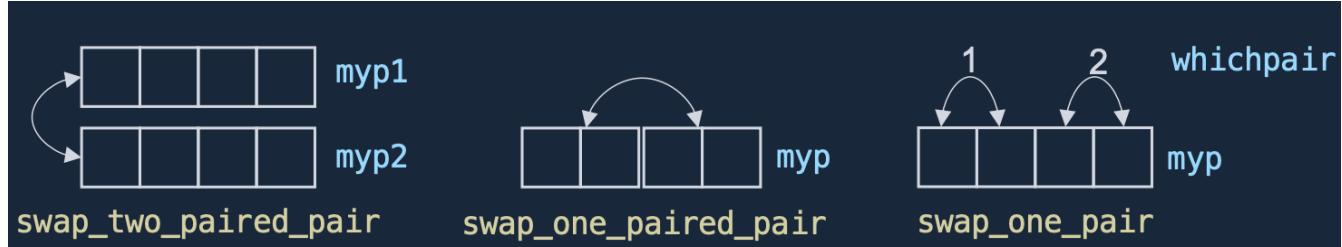
```

1 | if ( (MuCharge)[(myCombIdx)[1][0]] + (MuCharge)[(myCombIdx)[1][1]] == 0)
2 | { (Mu4vect)[(myCombIdx)[1][0]] + (Mu4vect)[(myCombIdx)[1][1]] ).M()

```

接下来是排序，排序除了想解决规范问题之外，还会有比如在 $\Upsilon J/\psi$ 这样的衰变道中，我们需要让第一对的两个 $\mu$ 子的不变质量大于第二对的，这样才比较合理。下面我会介绍对于电荷和质量大小的排序：

首先找出错误的电荷组合，将其放入到myCombIdx[0]序号为零的里面，这样如果我们不想研究带电的态的话就可以只对[1][2]进行循环了，然后比较两对的不变质量，把较大的放在前面即 $([0]+[1]).M() > ([2]+[3]).M()$ ，之后将 $\mu$ 子的电荷按照 $+ - -$ 的顺序排列。实现以上需求，你需要下面几个交换函数。



除了这些基础的排序，我们还可以根据他们的质量和你想找的粒子的质量的相近程度来排序，这里就会用到 $\chi^2$ 检验。

卡方值 $\chi^2$  (Chi-Square)

卡方值是一种统计量，用于衡量观测值与期望值之间的偏差程度。卡方值越大，观测值与期望值之间的偏差越大。

比如我想研究 $\psi(2S)J/\psi$ 这个衰变道可以用 $\chi^2$ 的公式是：

$$\chi^2 = \left( \frac{M_{1\_obs} - M_{\psi(2S)}}{\sigma_{M_1}} \right)^2 + \left( \frac{M_{2\_obs} - M_{J/\psi}}{\sigma_{M_2}} \right)^2$$

具体的代码中还会为了调整质量误差的影响而在质量误差后乘上一个缩放因子

除了用于排序的函数之外还有一个用于对应 $\mu\mu$ 对的不变质量的函数`setFourMuPairsVars`，这个函数的作用是，将已经排序的`myCombIdx`与在`cc`中进行过拟合的 $\mu\mu$ 对的不变质量对应上。比如排序完成后的`myCombIdx[1]`为{3,1,4,2}，那么我输出的`myFourMuVars[3][2]`的数组中`myFourMuVars[1][0]=(*MyJpsi3Var_Mu13)[fourMuIdx] myFourMuVars[1][1] = (*MyJpsi4Var_Mu24)[fourMuIdx]`

## 结果输出的命名与自动化

你可以修改下面三个地方自定义输出的root文件名，这样你就可以通过脚本程序生成多个Runjobs.C同时运行，并且会输出为不同文件，最后再用`hadd`将这些文件合成为一份就可以了

脚本可以参考`/home/storage0/users/zhufeng/formymultilep_learning/shell/`

```

1 // .c
2 void myntuple::Loop(TString outputname){
3 .....
4 TString myroot = outputname + "_zhuf.root";
5 TFile* myhbk = new TFile (myroot, "recreate");
6
7 // .h
8 virtual void Loop(TString outputname);
9
10 // Runjobs
11 a.Loop("youroutputdir/youroutputname");

```

了解这一步之前，我们首先要了解`myntuple.C` `myntuple.h` 和 `Runjobs.C`之间的关系。当你用`MakeClass`去创建一个新的`myntuple.C`的时候，大部分的成员函数已经创建完成了，像`GetEntry`这样的，这些函数的功能比较单一，所以都会放在`.h`文件中，而其中有一个`virtual void Loop();`这一`Loop`成员函数是我们主要需要编写的函数。如果我们编译之后得到的`myntuple_C.so`文件就能在`Runjobs.C`中加载使用了（下面代码中路径有点长省略了一部分）

```
1 void Runjobs()
2 {
3     gSystem->Load("myntuple_C.so");//加载库
4     TChain * chain = new TChain("/mkcands/X_data","");
5     chain->Add("../MuOnia/ReReco2016/FILE/mymultilep_*Number.root");//将路径中的root文件加载进TChain中, TChain就包含了
6     //所有的TTree的信息
7     myntuple a(chain);//使用chain 创建一个 myntuple 类的实例 a
8     a.Loop();//调用 myntuple 类的 Loop 方法, 处理 TChain 中的数据
9 }
```

可以发现我们在Runjobs中只会调用 `Loop` 函数所以我们在这个函数里面增加一个 `string` 变量就能在函数里面使用了，并将这个 `string` 运用在输出文件的命名中就大功告成了！

end-