Event plane measurements in MPD using evPlane wagon

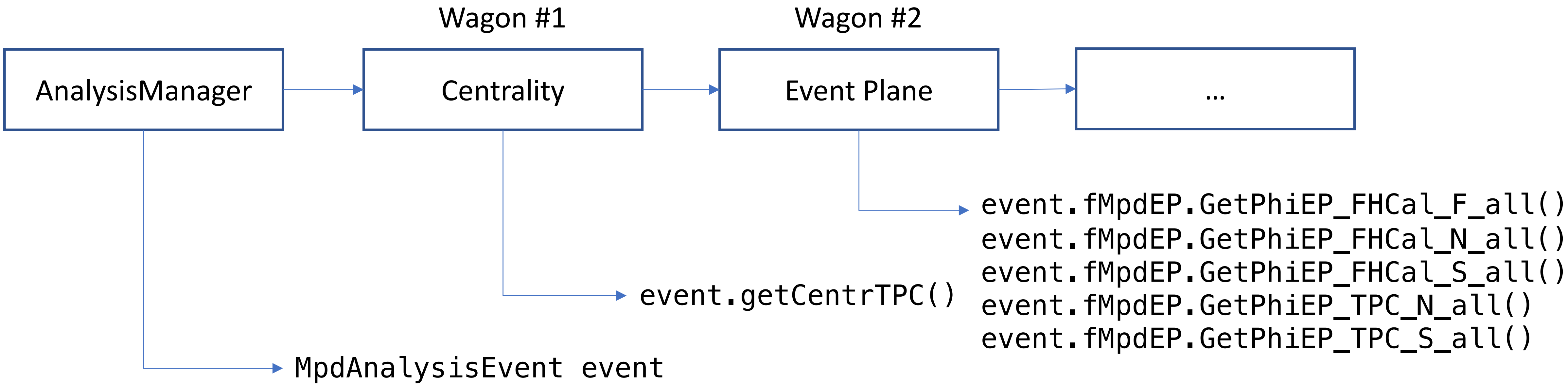
Petr Parfenov

NRNU MEPhI, INR RAS

Main idea of the wagons

- All analyses are packed into wagons within the Analysis Framework
  - All wagons have similar structure, provide consistency among all analyses
  - All info from the wagons is being stored into the main class `MpdAnalysisEvent`

Example:



Wagon #1

Wagon #2

AnalysisManager

Centrality

Event Plane

...

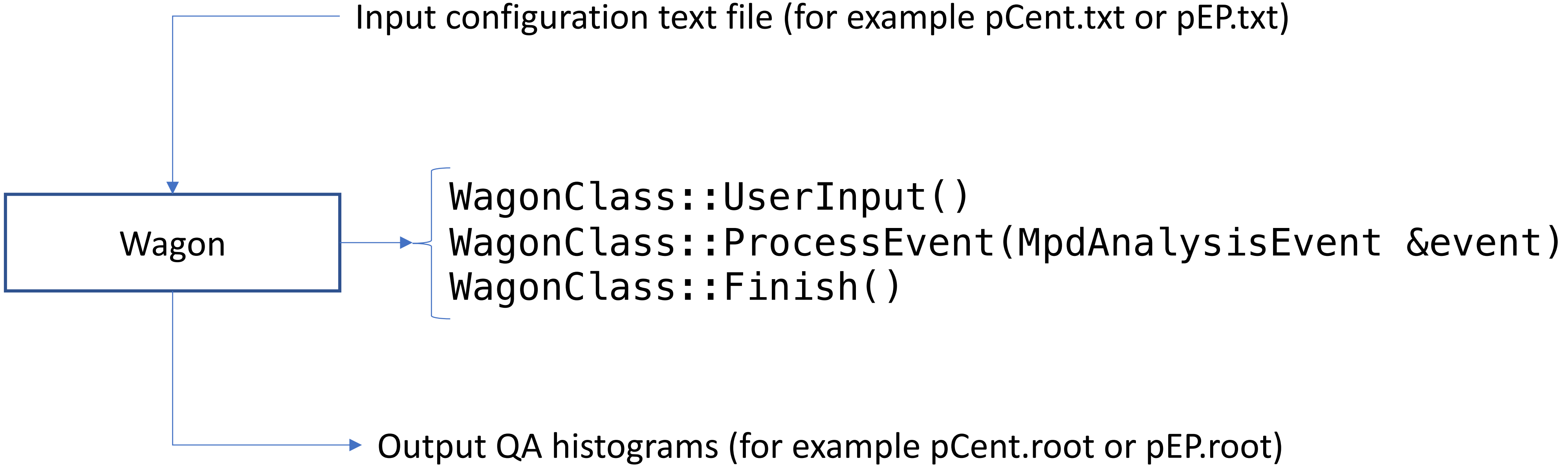
```
event.fMpdEP.GetPhiEP_FHCal_F_all()
event.fMpdEP.GetPhiEP_FHCal_N_all()
event.fMpdEP.GetPhiEP_FHCal_S_all()
event.fMpdEP.GetPhiEP_TPC_N_all()
event.fMpdEP.GetPhiEP_TPC_S_all()
```

```
event.getCentrTPC()
```

```
MpdAnalysisEvent event
```

Structure of the wagons

- Same basic structure of the wagons:
  - Similar Input/Output treatment
  - Similar methods (UserInput(), ProcessEvent(…), Finish())



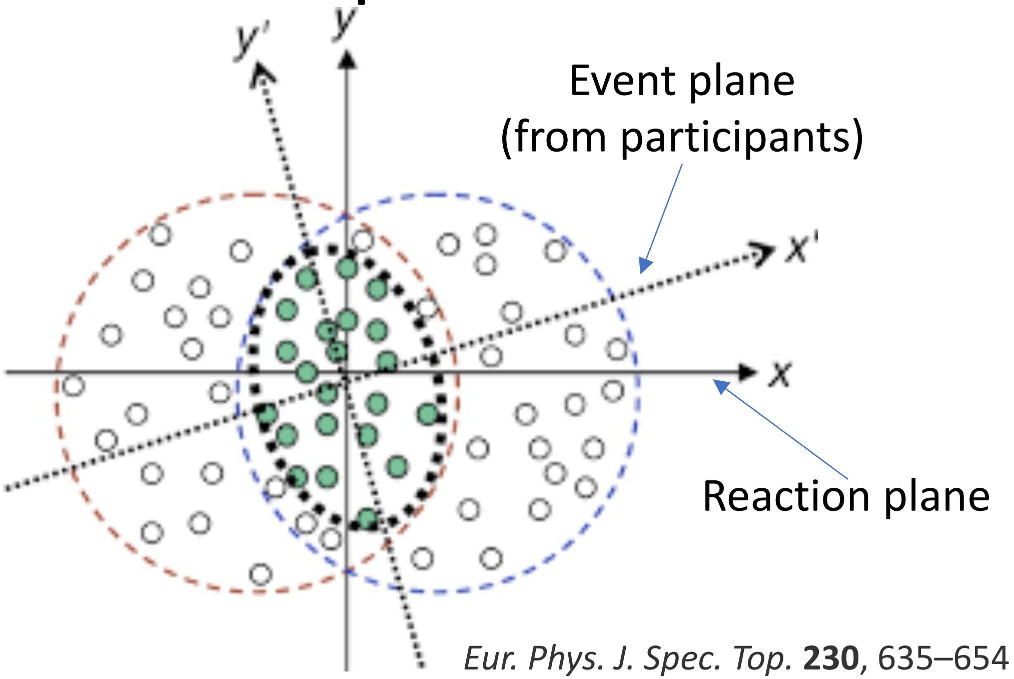
Input configuration text file (for example pCent.txt or pEP.txt)

Wagon

```
WagonClass::UserInput()
WagonClass::ProcessEvent(MpdAnalysisEvent &event)
WagonClass::Finish()
```

Output QA histograms (for example pCent.root or pEP.root)

Event plane measurements



Eur. Phys. J. Spec. Top. **230**, 635–654 (2021)

- Reaction plane (RP) – plane formed by impact parameter *b* and beam line
  - RP cannot be measured in the experiment since we cannot measure *b*
- Event plane (EP) is the observable estimation of the reaction plane

To measure EP one can use Q-vector (a.k.a. “event flow” vector):

$$Q_{n,x} = \sum \omega_i \cos n\varphi_i \, , Q_{n,y} = \sum \omega_i \sin n\varphi_i \, , \Psi_n = \frac{1}{n} \tan^{-1} \frac{Q_{n,y}}{Q_{n,x}}$$

where ω_i is the weight, φ_i is particle’s azimuthal angle, and n is the harmonics

Event plane measurements in MPD

- EP angle is measured using Q-vectors from FHCal and TPC:

$$Q_{1,x}^{\mathrm{FHCal}} = \frac{1}{\sum E_{dep,i}} \sum E_{dep,i} \cos\phi_i \,, Q_{1,y}^{\mathrm{FHCal}} = \frac{1}{\sum E_{dep,i}} \sum E_{dep,i} \sin\phi_i$$

$$Q_{2,x}^{\mathrm{TPC}} = \sum p_{T,i} \cos 2\varphi_i \,, Q_{2,y}^{\mathrm{TPC}} = \sum p_{T,i} \sin 2\varphi_i$$

$$\Psi_1^{\mathrm{FHCal}} = \tan^{-1}\frac{Q_{1,y}^{\mathrm{FHCal}}}{Q_{1,x}^{\mathrm{FHCal}}}, \qquad \Psi_2^{\mathrm{TPC}} = \frac{1}{2}\tan^{-1}\frac{Q_{2,y}^{\mathrm{TPC}}}{Q_{2,x}^{\mathrm{TPC}}}$$

Here:

$E_{dep,i}$ - energy deposition in *i*-th module of the FHCal

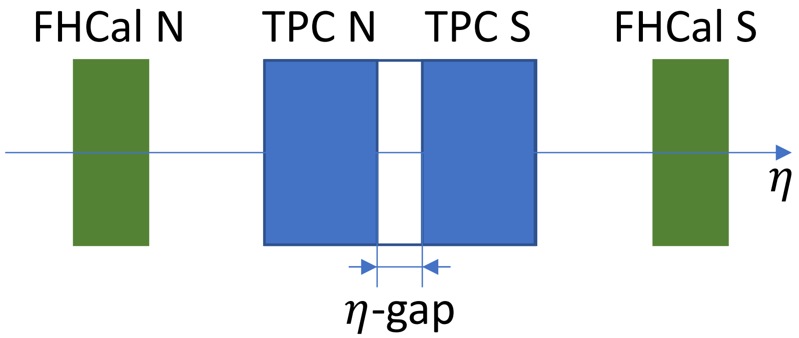
ϕ_i - azimuthal angle of the center of *i*-th module of the FHCal

$p_{T,i}$ - transverse momentum of the *i*-th track in the TPC

φ_i - azimuthal angle of the *i*-th track in the TPC ($\varphi_i = \tan^{-1}\frac{p_{y,i}}{p_{x,i}}$)

Ψ_1^{FHCal} and Ψ_2^{TPC} are the event plane angles (from FHCal and TPC)

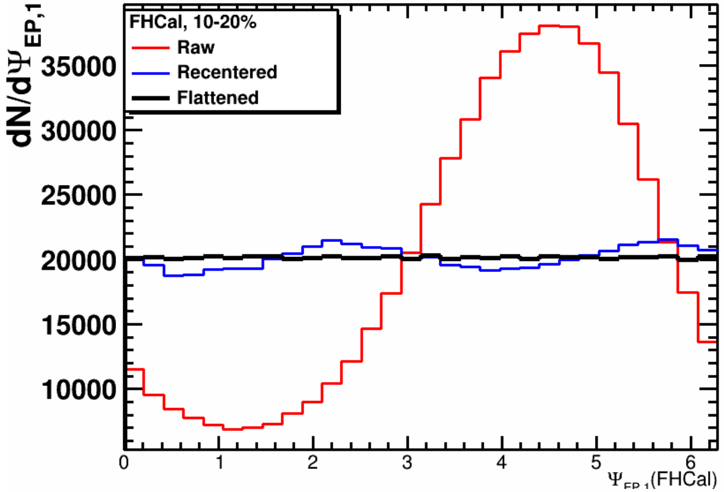
4 sub-events were chosen:



+1 additional sub-event:
FHCal F = FHCal N + FHCal S

Event plane corrections for non-uniform acceptance

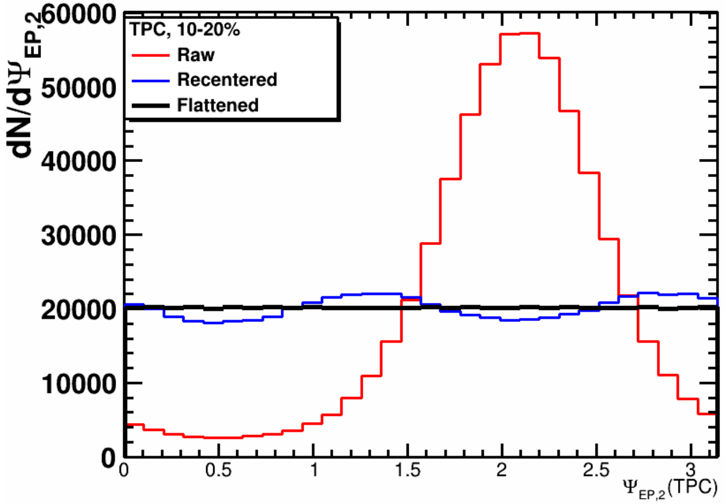
- EP distribution $\frac{dN}{d\Psi_n}$ is uniform
- Non-uniform acceptance of the detector (TPC, FHCal) might introduce anisotropy to $\frac{dN}{d\Psi_n}$ and bias to the observables (v_n, P_Λ, ...)



In that case one has to apply EP corrections:

- **Recentering**: $Q' = Q - \langle Q \rangle$
- **Shift (Flattening)**: $\Psi''_n = \Psi'_n + \Delta\Psi'_n$,

$$\mathrm{n}\Delta\Psi'_n = \sum_{i}^{i_{max}} \frac{2}{i}\left[\langle \cos in\Psi'_n \rangle \sin in\Psi'_{\mathrm{n}} - \langle \sin in\Psi'_n \rangle \cos in\Psi'_{\mathrm{n}}\right]$$



In MPD EP corrections will be required once we start working with real experimental data

evPlane wagon: how to run

- evPlane wagon is implemented in MpdRoot (mpdroot/physics/evPlane/)
  - README file there has a quick explanation and how-to for the wagon
- Example of RunAnalyses.C macro (in evPlane/macros/):

```
void RunAnalyses(int nEvents = -1){

            gSystem->Load("libZdc.so");
            gSystem->Load("libMpdPhysics.so");

            MpdAnalysisManager man("ManagerAnal", nEvents);
            man.InputFileList("list.txt");
            man.ReadBranches("*");
            man.SetOutput("histos.root");

            MpdCentralityAll pCentr("pCentr","pCentr");
            man.AddTask(&pCentr);

            MpdEventPlaneAll pEP("pEP","pEP");
            man.AddTask(&pEP);

            man.Process();
}
```

Loading needed mpdroot libraries (detectors that will be used, etc.)

List of input MpdDst files

wagon’s name

Set I/O names (input: pCentr.txt, output: pCentr.root)

Add Centrality wagon to the train

Add Centrality wagon to the train

Run the train (process MpdDst data)

evPlane wagon: input file configuration

- Example of the input file configuration can be found in evPlane/macros/:

```
#-------Parameters used for analysis------
# Event selection:
mZvtxCut 130 # cut on vertex z coordinate

```

Cuts for TPC-based EP

```
# Track selection:
mNofHitsCut 16 # minimal number of hits to accept track
mEtaCut 1.5 # maximal pseudorapidity accepted
mEtaGapCut 0.1 # minimal pseudorapidity accepted: abs(eta)>0.05 for mEtaGap=0.1
mPtminCut 0.1 # minimal pt used in analysis
mPtmaxCut 2.0 # maximal pt used in analysis
mDcaCut 2.0 # maximal DCA accepted
# Event plane corrections:
mInFileEpCorr ANY # input file with QA histograms and EP corrections profiles
```

evPlane wagon: UserInit()

- The main class in the evPlane wagon is MpdEventPlaneAll. **MpdEventPlaneAll::UserInit()** performs procedures that are needed before the event loop:
    - Read input config file
    - Read TProfiles for the EP correction (if the mInFileEpCorr file provided)
    - Initialize output QA histograms:
        - Basic QA (event count, vtxZ, track's parameters: p_T, η, N_{hits}, DCA)
        - mhCorrStep - histogram that keeps track of the correction step (0 – raw, 1 – recentered, 2 –shifted)
        - Q-vectors and EP angle distributions
        - $\left\langle \cos\left(\Psi_1^{\mathrm{FHCal}\,N} - \Psi_1^{\mathrm{FHCal}\,S}\right)\right\rangle$ and $\left\langle \cos\left(2\left(\Psi_2^{\mathrm{TPC}\,N} - \Psi_2^{\mathrm{TPC}\,S}\right)\right)\right\rangle$ correlations vs. *b*
        - TProfiles for EP corrections

evPlane wagon: ProcessEvent(…)

- MpdEventPlaneAll::ProcessEvent(MpdAnalysisEvent &event) performs all analysis in the event (EP measurement and EP corrections). It consists of:
  - Applying event selection (vtxZ cut from the input file)
  - Getting centrality from evCentrality wagon (from event.getCentrTPC())
  - Defining Q-vectors from FHCal and TPC
  - Applying recentering and shift EP correction (depending on iteration)
  - Filling QA histograms
  - Filling $\left\langle\cos\left(\Psi_1^{\mathrm{FHCal}\,N}-\Psi_1^{\mathrm{FHCal}\,S}\right)\right\rangle$ and $\left\langle\cos\left(2\left(\Psi_2^{\mathrm{TPC}\,N}-\Psi_2^{\mathrm{TPC}\,S}\right)\right)\right\rangle$ correlations
  - Filling TProfiles for EP corrections (depending on iteration)
  - Writing resulting EP angles in MpdAnalysisEvent event

evPlane wagon: Output file

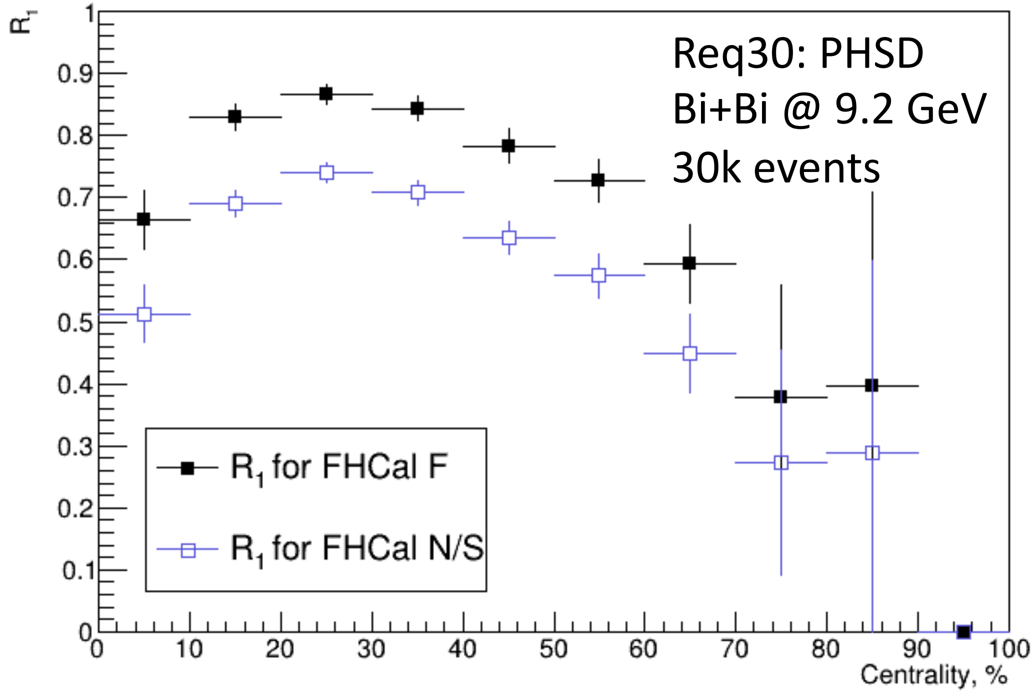
```
$ root –l pEP.root
root [0]
Attaching file pEP.root as _file0...
(TFile *) 0x1a41590
root [1] .ls
TFile** pEP.root
 TFile* pEP.root
  KEY: TH1F mhEvents;1 Number of events
  KEY: TH1F hVertex;1 Event vertex distribution
  KEY: TH1F mhHits;1 Number of TPC hits
  KEY: TH1F mhEta;1 Eta
  KEY: TH1F mhDca;1 DCA
  KEY: TH1F mhPt;1 Pt
```

```
KEY: TH1F mhCorrStep;1 Correction step: 0 - raw, 1 - rec, 2 - shift
KEY: TH1F mhQxRawFHCalFAll;1 Q_{x}^{Raw} from FHCal F
KEY: TH1F mhQyRawFHCalFAll;1 Q_{y}^{Raw} from FHCal F
KEY: TH1F mhPhiEPRawFHCalFAll;1 #Psi_{EP}^{Raw} from FHCal F
        ... Same for FHCal N, FHCal S
KEY: TH1F mhQxRawTPCNAll;1 Q_{x}^{Raw} from TPC N
KEY: TH1F mhQyRawTPCNAll;1 Q_{y}^{Raw} from TPC N
KEY: TH1F mhPhiEPRawTPCNAll;1 #Psi_{EP}^{Raw} from TPC N
        ... Same for recentered (Rec) and shifted (Shf) results
KEY: TProfile mhCosFHCalNFHCalSAll;1
KEY: TProfile mhCosTPCNTPCSAll;1
        ... TProfiles for EP corrections
```

Additional macro to calculate EP resolution

In addition, evPlane/macros/ has a macro getResolution.C that calculates resolutions for measured event plane angles:

```
$ root -l pEP_resolution.root
root [0]
Attaching file pEP_resolution.root as _file0...
(TFile *) 0x36bda20
root [1] .ls
TFile** pEP_resolution.root
 TFile* pEP_resolution.root
  KEY: TH1F mhRes1FHCalNFHCalSAll;1 R_{1} for FHCal N/S
  KEY: TH1F mhRes2FHCalNFHCalSAll;1 R_{2} for FHCal N/S
  KEY: TH1F mhRes1FHCalFullAll;1 R_{1} for FHCal F
  KEY: TH1F mhRes2FHCalFullAll;1 R_{2} for FHCal F
  KEY: TH1F mhRes2TPCNTPCSAll;1 R_{2} for TPC N/S
```



Req30: PHSD
Bi+Bi @ 9.2 GeV
30k events

Usage:

```
root -l -b -q getResolution.C'("pEP.root","pEP_resolution.root")'
```

`"pEP.root"` – input file (output from evPlane wagon)

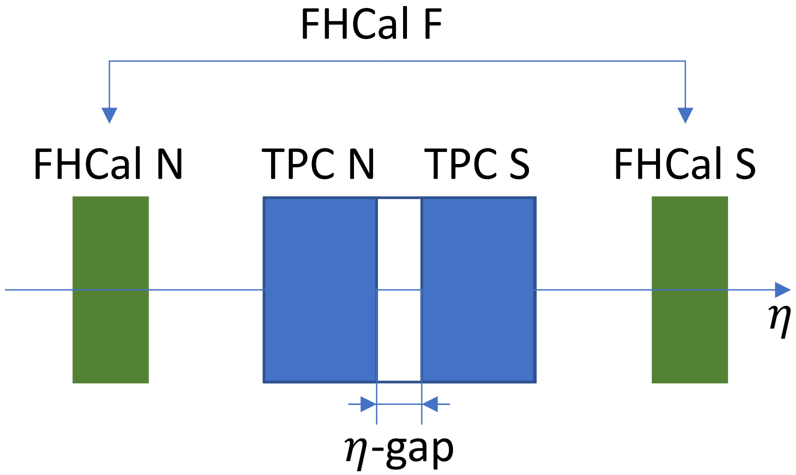
`"pEP_resolution.root"` – output file with resolutions

How to use EP from evPlane in your wagon

- One can use event plane angles in their wagon:
  - `event.fMpdEP.GetPhiEP_FHCal_F_all()` – returns EP angle from full FHCal
  - `event.fMpdEP.GetPhiEP_FHCal_N_all()` – returns EP angle from left (north) FHCal
  - `event.fMpdEP.GetPhiEP_FHCal_S_all()` – returns EP angle from right (south) FHCal
  - `event.fMpdEP.GetPhiEP_TPC_N_all()` – returns EP angle from left part (north) of TPC
  - `event.fMpdEP.GetPhiEP_TPC_S_all()` – returns EP angle from right part (south) of TPC
- Returned float value is in radians. If EP is not defined it will return –9999.

Examples:

- For v_1 and P_Λ one can use EP from FHCal
- For v_2 measurements EP angles from both FHCal and TPC can be used



Additional note: how to perform EP corrections

- In case one needs to do EP corrections, RunAnalyses.C macro has to be done 3 times (steps):
  - Step 1: put `mInFileEpCorr ANY` in the pEP.txt input file and run the macro. Rename output pEP.root -> pEP_it1.root. No corrections are applied.
  - Step 2: put `mInFileEpCorr pEP_it1.root` in the pEP.txt input file and run the macro. Rename output pEP.root -> pEP_it2.root. Recentering applied.
  - Step 3: `mInFileEpCorr pEP_it2.root` in the pEP.txt input file and run the macro. This iteration is final: all EP angles have been corrected.

After the evPlane wagon ran for the third time it provides fully corrected EP angles. During the run the Analysis Manager will print out what step you're on:

```
evPlane: Step (1/3). Recentering: collecting. Shift: waiting.
evPlane: Step (2/3). Recentering: applying.   Shift: collecting.
evPlane: Step (3/3). Recentering: applying.   Shift: applying.
```