Flow Package
AliRoot/PWG2/FLOW

- Structure of PWG2/FLOW
- How to use it

Naomi van der Kolk

In PWG2/FLOW you'll find

- Several methods for flow analysis
- Within the task framework
- Using the Correction Framework for event and particle cuts
- For local, proof or grid

The code is divided into 2 libraries

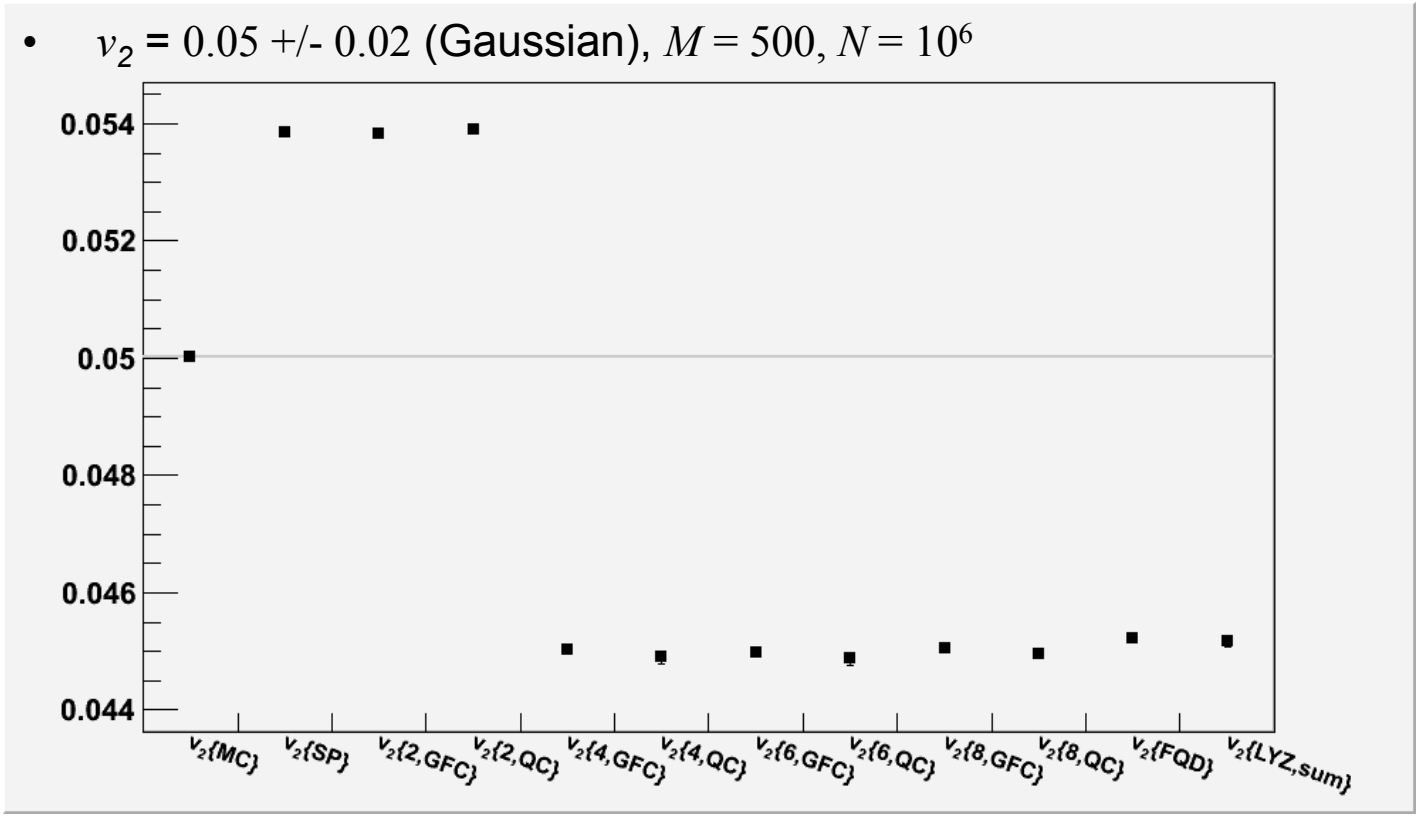
- AliFlowCommon:
  holds all analysis code
- AliFlowTasks:
  holds all the tasks (one for each analysis class)
- Independent analysis classes
- Easy to adapt to changes in the Analysis Framework

Available methods

- Scalar Product (SP)
- Cumulants (GFC, QC)
- Fitting of Q-vector (FQD)
- Lee-Yang Zeroes (LYZ)
- Monte-Carlo event plane (MCEP)

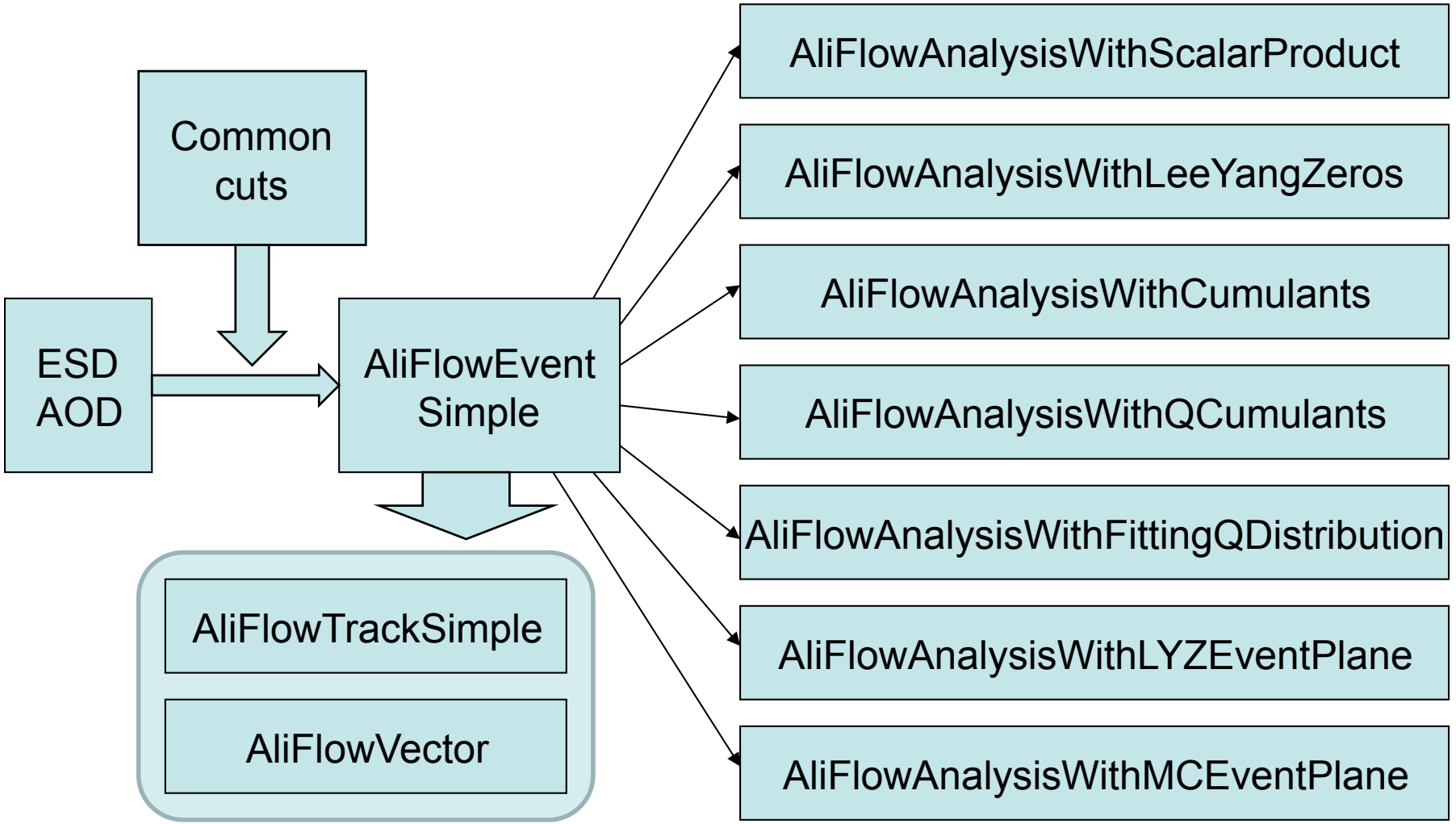
Different methods are needed for a decent analysis

- v_2 = 0.05 +/- 0.02 (Gaussian), $M = 500$, $N = 10^6$



- Need a direct comparison of all methods

Common input

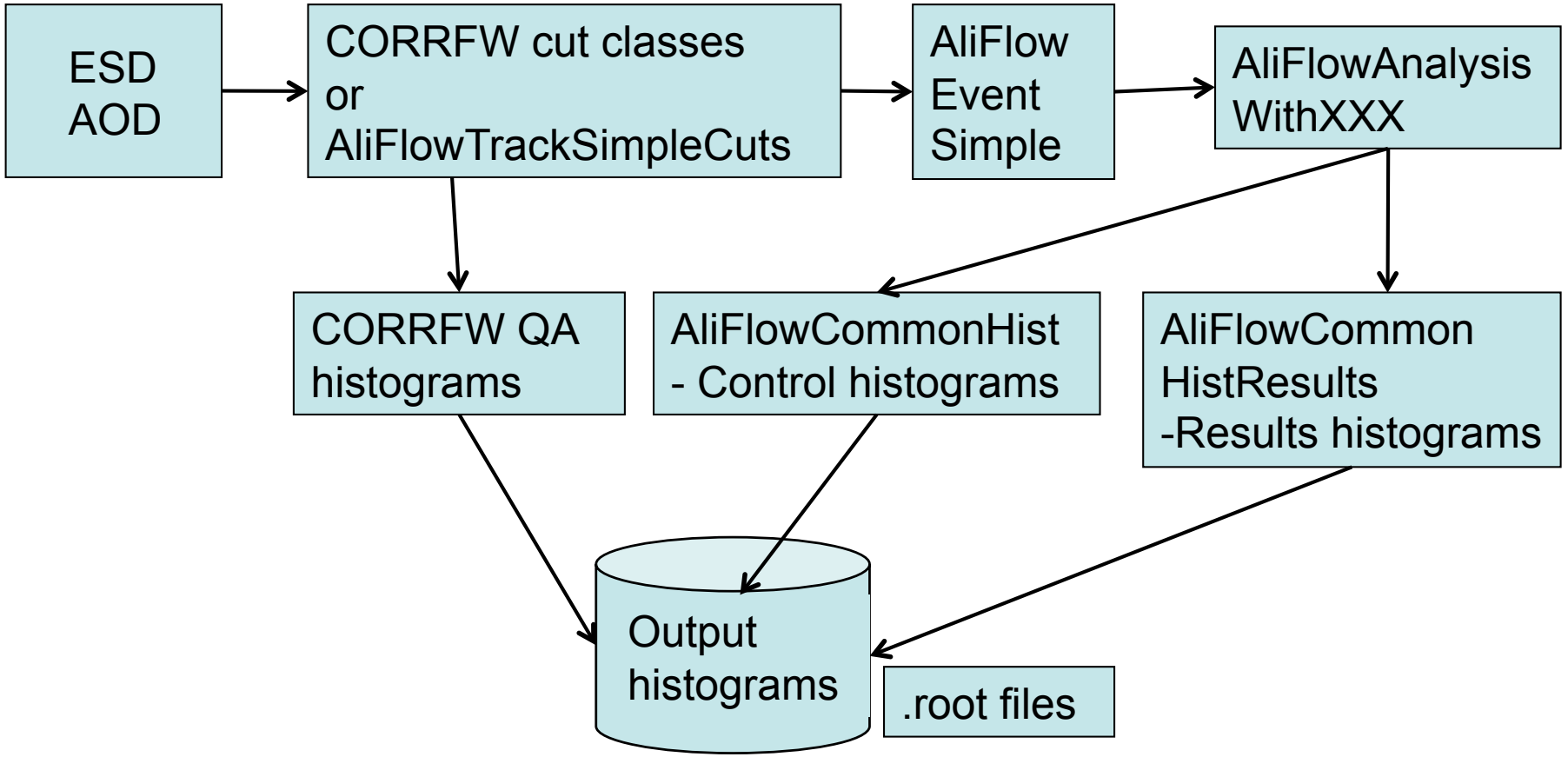


AliFlowEventSimpleMaker can take any input

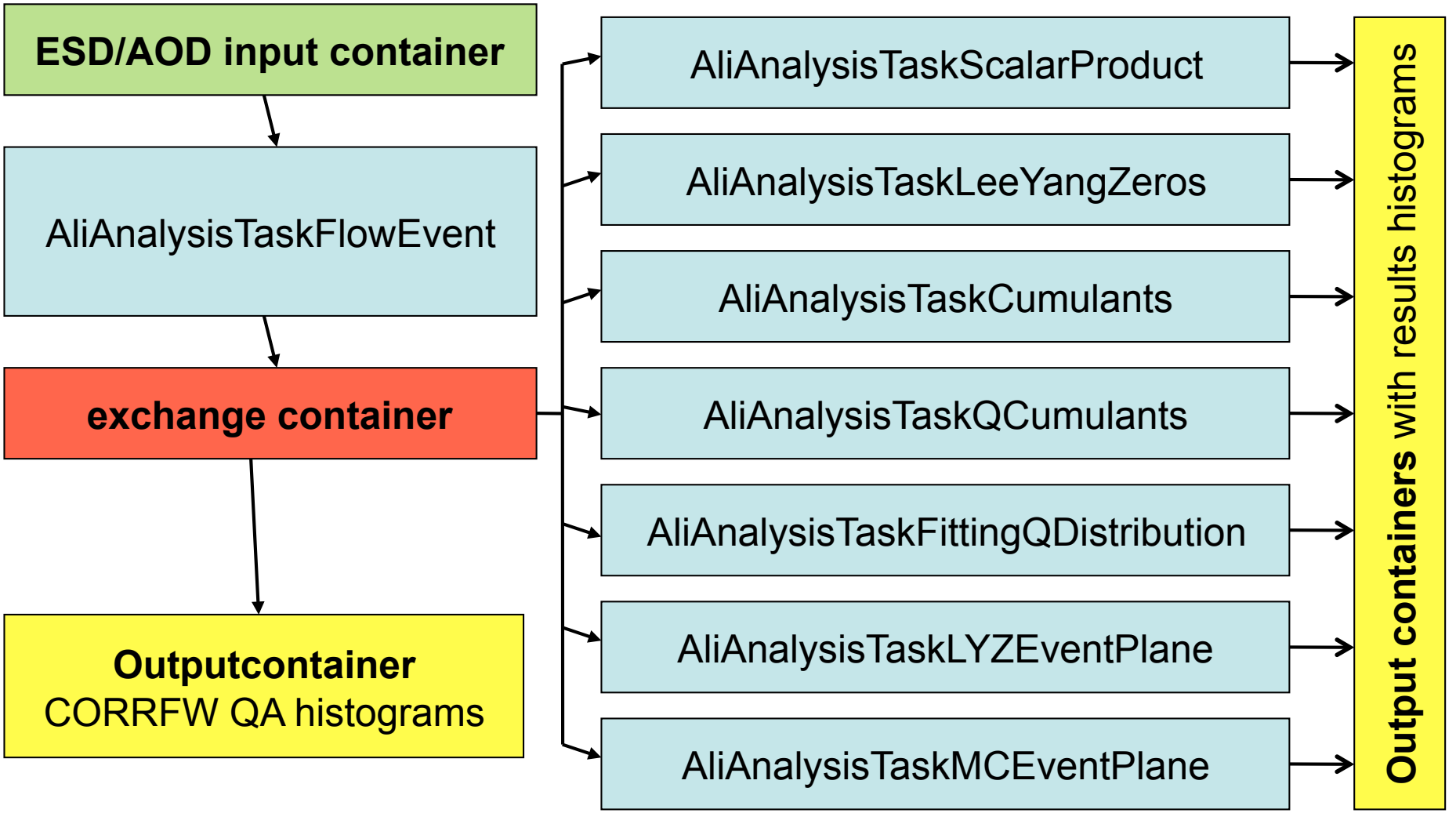
- Adaptable for other input types, e.g. tracklets, by overloading the method FillTracks()

```
27 class AliFlowEventSimpleMaker {
28
29  public:
30
31   AliFlowEventSimpleMaker();              //constructor
32   virtual ~AliFlowEventSimpleMaker();     //destructor
33
34   void SetMCReactionPlaneAngle(Double_t fPhiRP)  { this->fMCReactionPlaneAngle = fPhiRP; }
35   //TTree
36   AliFlowEventSimple* FillTracks(TTree* anInput, AliFlowTrackSimpleCuts* rpCuts, AliFlowTrackSimpleCuts* poiCuts);   //use own cut class
37   //AliMCEvent
38   AliFlowEventSimple* FillTracks(AliMCEvent* anInput);   //use own cuts
39   AliFlowEventSimple* FillTracks(AliMCEvent* anInput, AliCFManager* rpCFManager, AliCFManager* poiCFManager ); //use CF(2x)
40   //AliESDEvent
41   AliFlowEventSimple* FillTracks(AliESDEvent* anInput);   //use own cuts
42   AliFlowEventSimple* FillTracks(AliESDEvent* anInput   AliCFManager* rpCFManager, AliCFManager* poiCFManager); //use CF(2x)
```

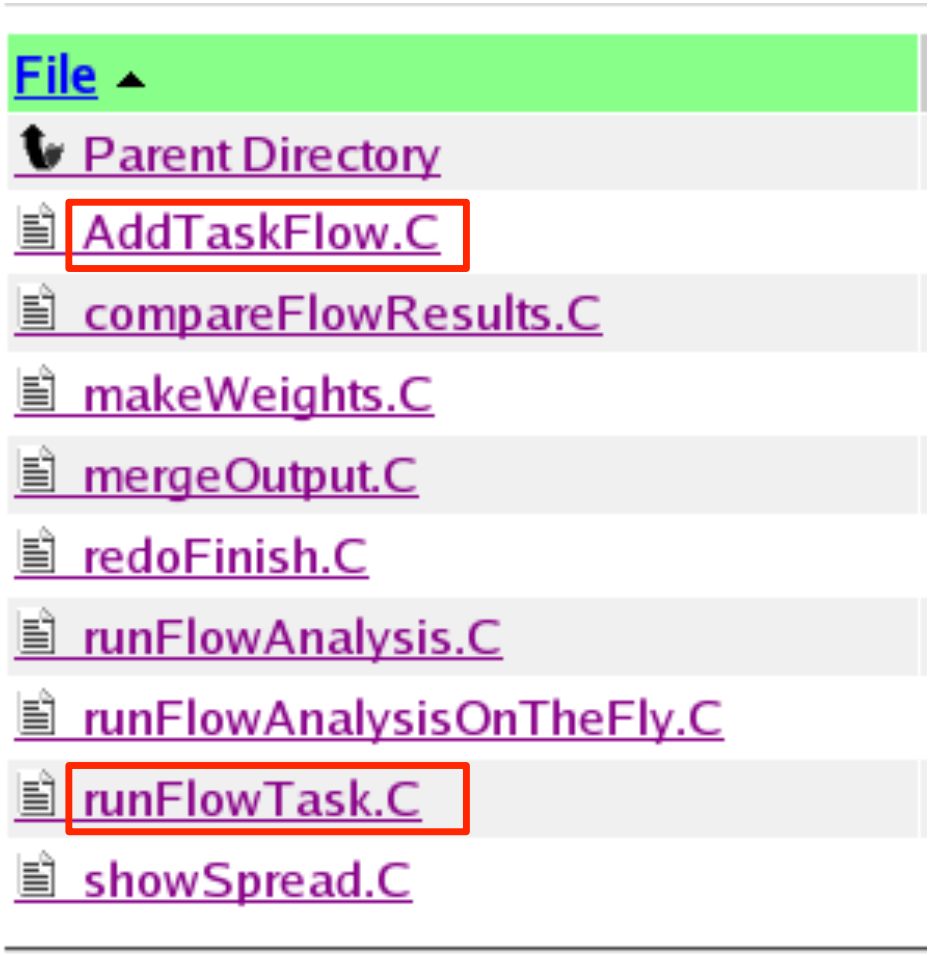
Common output



AliFlowTasks



Content of PWG2/FLOW/macros



- Macros to run within the task framework:
  - runFlowTask.C
    AddTaskFlow.C

runFlowTask.C

```
6  // RUN SETTINGS
7
8  // Flow analysis method can be:(set to kTRUE or kFALSE)
9  Bool_t SP       = kTRUE;
10 Bool_t LYZ1SUM  = kTRUE;
11 Bool_t LYZ1PROD = kTRUE;
12 Bool_t LYZ2SUM  = kFALSE;
13 Bool_t LYZ2PROD = kFALSE;
14 Bool_t LYZEP    = kFALSE;
15 Bool_t GFC      = kTRUE;
16 Bool_t QC       = kTRUE;
17 Bool_t FQD      = kTRUE;
18 Bool_t MCEP     = kTRUE; //not for pp
19
20 Bool_t METHODS[] = {SP,LYZ1SUM,LYZ1PROD,LYZ2SUM,LYZ2PROD,LYZEP,GFC,QC,FQD,MCEP};
21
22 // Analysis type can be ESD, AOD, MC, ESDMC0, ESDMC1
23 const TString type = "ESD";
24
25 // Boolean to fill/not fill the QA histograms
26 Bool_t QA = kTRUE;
27
28 // Boolean to use/not use weights for the Q vector
29 Bool_t WEIGHTS[] = {kFALSE,kFALSE,kFALSE}; //Phi, v'(pt), v'(eta)
30
```

- Set which methods you want to run
- Set input type
- Set CORRFW QA histograms
- Set weights usage

runFlowTask.C

```
31
32 //void runFlowTask(Int_t mode=mLocal, Int_t nRuns = 100,
33                   //const Char_t* dataDir="/data/alice2/kolk/PP/LHC09a4/81119", Int_t offset = 0)
34                   //const Char_t* dataDir="/data/alice2/kolk/Therminator_midcentral", Int_t offset = 0)
35                   //const Char_t* dataDir="/Users/snelling/alice_data/Therminator_midcentral", Int_t offset = 0)
36 void runFlowTask(Int_t mode=mPROOF, Int_t nRuns = 1000000,
37                   //const Char_t* dataDir="/COMMON/COMMON/LHC09a14_0.9TeV_0.5T", Int_t offset = 0)
38                   //const Char_t* dataDir="/COMMON/COMMON/LHC08c11_10TeV_0.5T", Int_t offset = 0)
39                   //const Char_t* dataDir="/PWG2/akisiel/Therminator_midcentral_ESD", Int_t offset=0)
40                   const Char_t* dataDir="/COMMON/COMMON/LHC09a4_run8101X", Int_t offset = 0)
```

- Set the analysis mode (local, proof, grid)
- Set the number of files to run over
- Set the location of the data

AddTaskFlow.C

```
10
11 // SETTING THE CUTS
12
13 // event selection
14 const Int_t multminESD = 10;  //used for CORRFW cuts
15 const Int_t multmaxESD = 1000000; //used for CORRFW cuts
16
17 const Int_t multmin = 10;     //used for AliFlowEventSimple (to set the centrality)
18 const Int_t multmax = 1000000;     //used for AliFlowEventSimple (to set the centrality)
19
20 // For RP selection
21 const Double_t ptmin1 = 0.0;
22 const Double_t ptmax1 = 10.0;
23 const Double_t ymin1  = -1.;
24 const Double_t ymax1  = 1.;
```

```
33 // For for POI selection
34 const Double_t ptmin2 = 0.0;
35 const Double_t ptmax2 = 10.0;
36 const Double_t ymin2  = -1.;
```

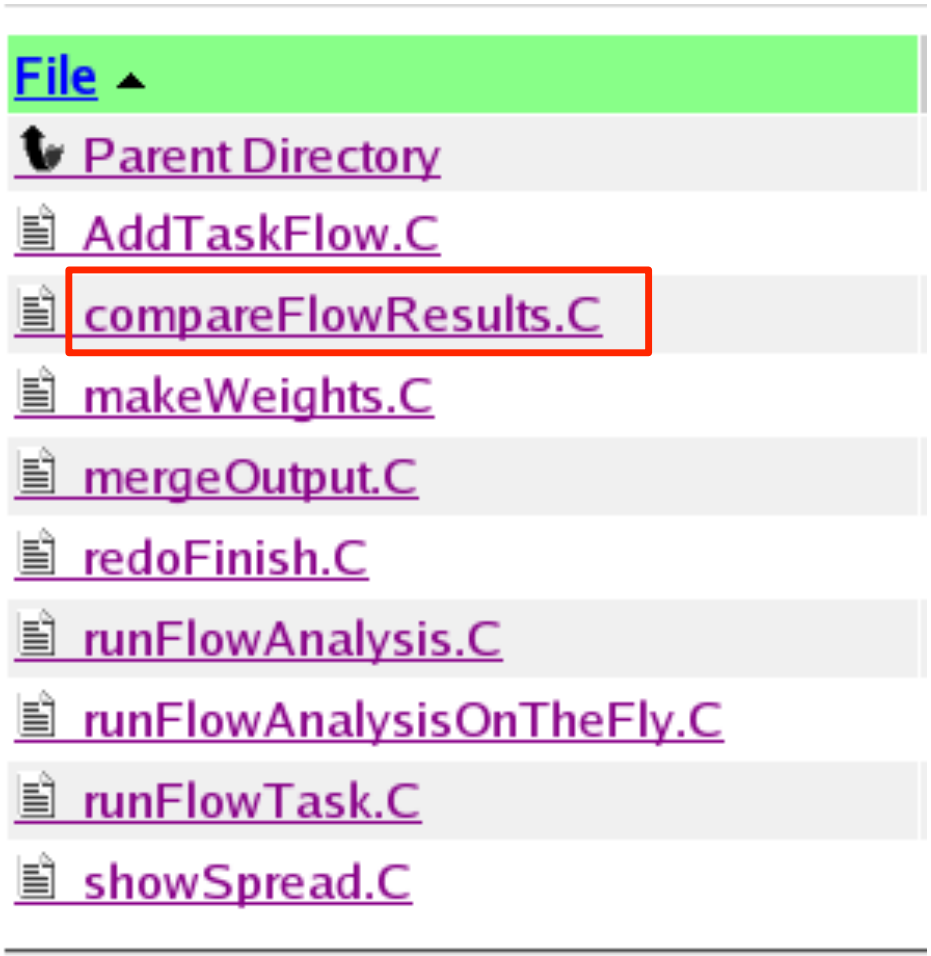
- Use all available cuts in CORRFW
- Set particle cuts for two selections: RP and POI

- RP - reference particles with which to calculate the EP
- POI – particles of interest of which to calculate flow

Selected Particles

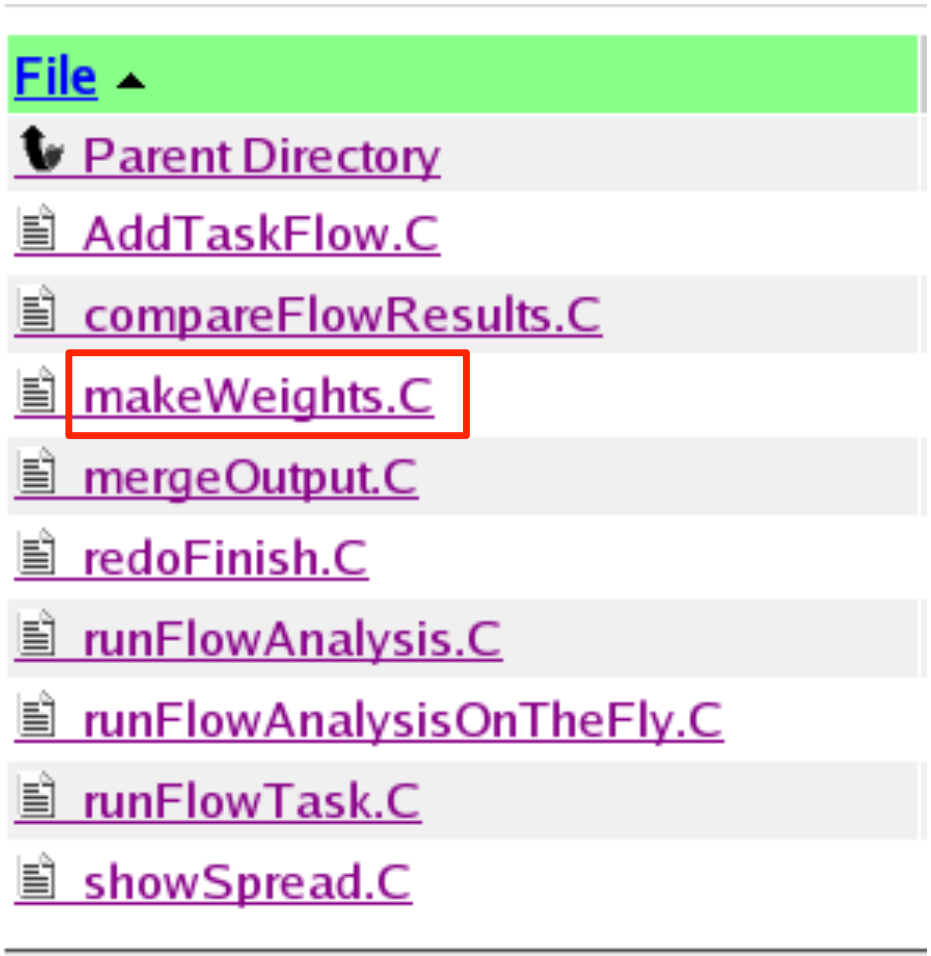
- Define the RP’s and POI's for each analysis.
- Add candidates (jets, charm, resonances) to POI selection
  - This could be done with AODs
  - An interface is needed for this

compareFlowResults.C



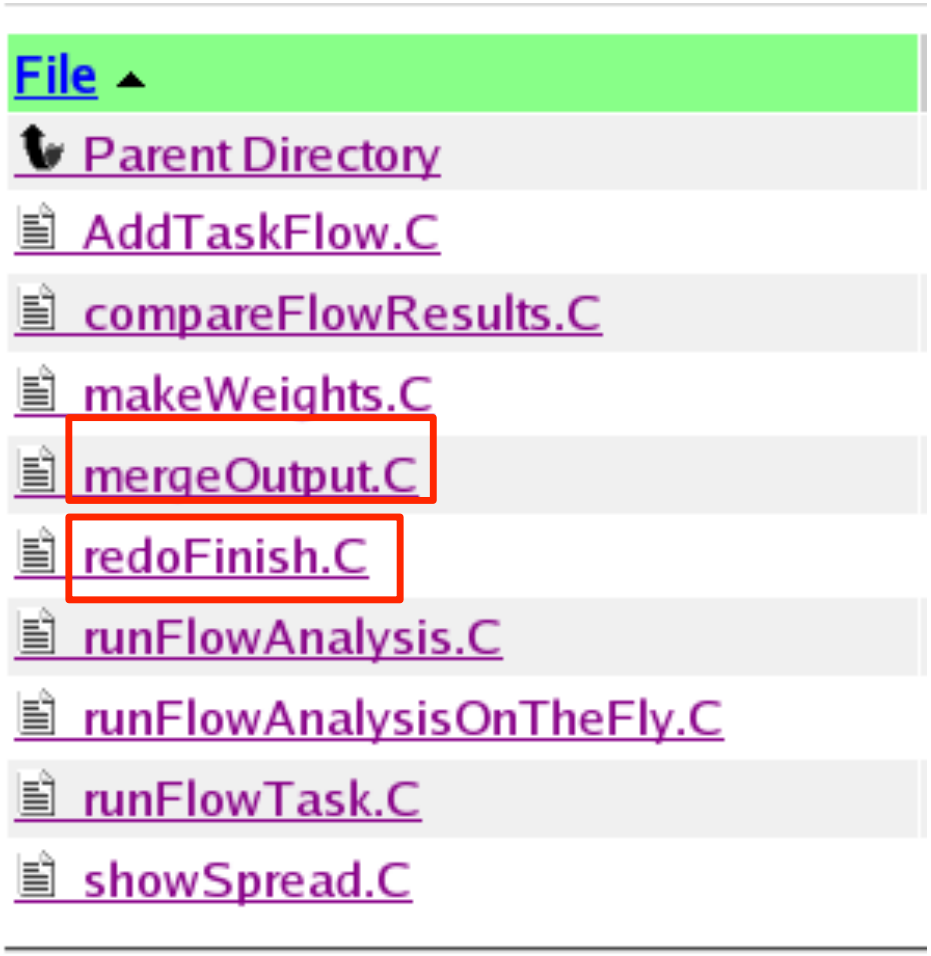
- Macro to plot the results histograms together in order to compare them

makeWeights.C



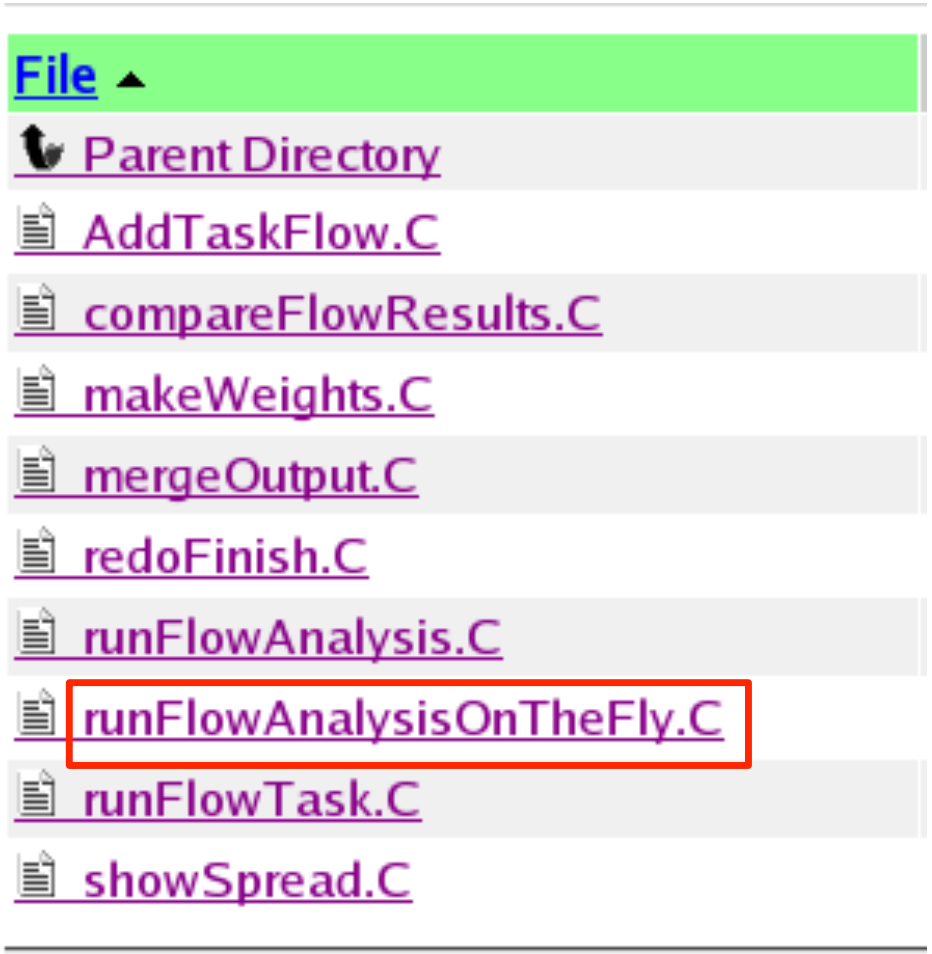
- Macro to create phi, pt or eta weights from the analysis output or based on a function
- Presently only for QC, GFC and FQD

mergeOutput.C / redoFinish.C



- Macros for combining results of many subjobs (for grid or local)
  - mergeOutput.C merges histograms filled in Make()
  - RedoFinish.C recalculated and fills the results histograms

runFlowAnalysisOnTheFly.C



- Macro for quick testing of the analysis methods
- Simulates ideal events “on the fly”
- Controlled way of checking your analysis

The flow package is ready for use

- You are encouraged to use the flow classes for your analysis
- To interface to your favorite particle candidate or detector
- And also to report encountered problems, possible bugs and missing functionality