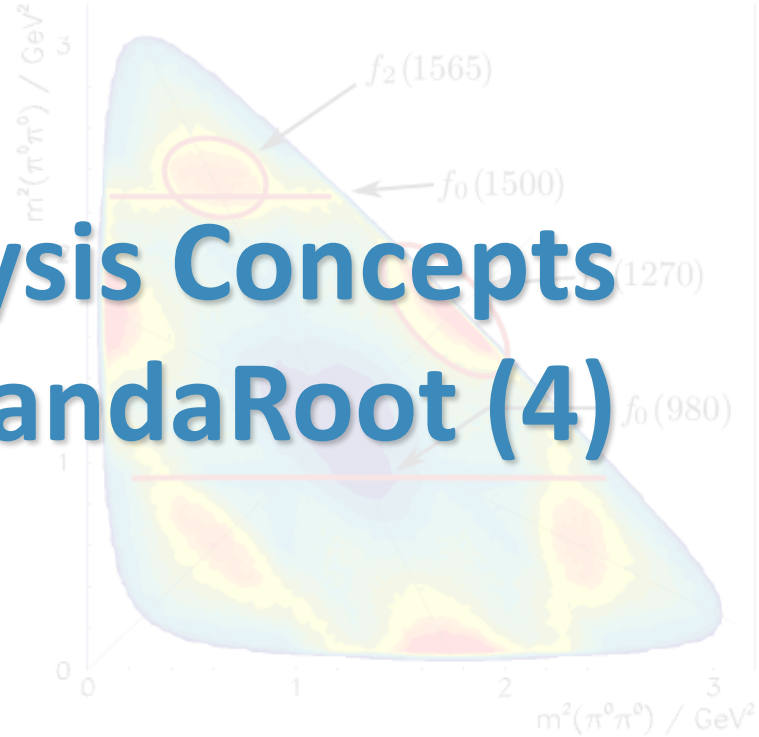
Physics Analysis Concepts with PandaRoot (4)



PANDA Lecture Week 2017

GSI, Dec 11 - 15, 2017

Klaus Götzen
GSI Darmstadt

Topics

- Event Filtering (*FairFilteredPrimaryGenerator*)
- Useful Event and Kinematic Variables (*PndEventShape*)
- Creating Output
  - Simplified N-Tuple output (*RhoTuple*)
  - QA-Tools (*PndRhoTupleQA*)
- Simplified Analysis Tools
  - Quick Analysis Tools (*quick(fsim)ana.C*)
- Multi Variate Analysis (*TMVATrainer*/*TMVATester*)

EVENT FILTERING

Event Filtering

- Usually $\sigma_{Background} \gg \sigma_{Signal}$ (e.g. ~mb vs. ~nb)
- Since (full) simulation of reactions computational intensive:
  - **Idea:** Reject events already at generator level likely being rejected at reco/analysis level
  - Saves a lot of computing power!
  - **Caveat**: Due to missing secondaries, rejecting criteria must be chosen carefully
- Comprehensive tutorial at
  https://panda-wiki.gsi.de/foswiki/bin/view/Computing/PandaRootEventFilterTutorial

Event Filter Usage - (Selected) Filters

- FairEvtFilterOnSingleParticleCounts
  - AndMinMaxAllParticles (min, max)
  - AndMinMaxCharged (min, max, type)
    - type: FairEvtFilter::kPlus / kMinus / kCharged / kNeutral
  - AndMinMaxPdg (min, max, pdg1 [, pdg2, ..., pdg8])
  - AndPRange / AndPtRange / AndPzRange (min, max)
  - AndThetaRange / AndPhiRange (min, max)
  - AndVzRange / AndVRhoRange / AndRadiusRange (min, max)

Each MinMax can also be Min or Max only, placing just one limit.

- PndEvtFilterOnInvMassCounts
  - SetPdgCodesToCombine (pdg1, pdg2 [,pdg3, pdg4, pdg5])
  - SetMinMaxInvMass (m_{min}, m_{max})
  - SetMinMaxCounts (min, max)
- FairFilteredPrimaryGenerator *(allows logical combination of filters)*
  - AndFilter / AndNotFilter (filterName)
  - OrFilter / OrNotFilter (filterName)
  - AddVetoFilter (filterName) *(has higher priority)*

Event Filter Usage

- Event filter setup in simulation macro (Full or Fast Simulation)

```
FairFilteredPrimaryGenerator* primGen = new FairFilteredPrimaryGenerator();

FairEvtFilterOnSingleParticleCounts* chrgFilter
          = new FairEvtFilterOnSingleParticleCounts("chrgFilter");
chrgFilter->AndMaxCharge(3, FairEvtFilter::kCharged);


FairEvtFilterOnSingleParticleCounts* neutFilter
          = new FairEvtFilterOnSingleParticlesCounts("neutFilter");
neutFilter->AndMaxCharge(6, FairEvtFilter::kNeutral);


PndEvtFilterOnInvMassCounts* eeInv= new PndEvtFilterOnInvMassCounts("eeInvMFilter");
eeInv->SetPdgCodesToCombine( 11, -11);
eeInv->SetMinMaxInvMass( 2.0, 4.0 );
eeInv->SetMinCounts(1);

primGen->AddVetoFilter(chrgFilter);
primGen->AndFilter(neutFilter);
primGen->AndFilter(eeInv);

primGen->SetFilterMaxTries(100000);
```

less than 4 charged

at most 6 neutral

at least on e^+e^- cand. with $2 < m < 4$ GeV/c^2

Vetos events with less than 4 charged

Combine all with logical AND

Maximum tries before counted as failed

Event Filter Example: $\bar{p}p \to \chi_{c1}\pi^+\pi^-$

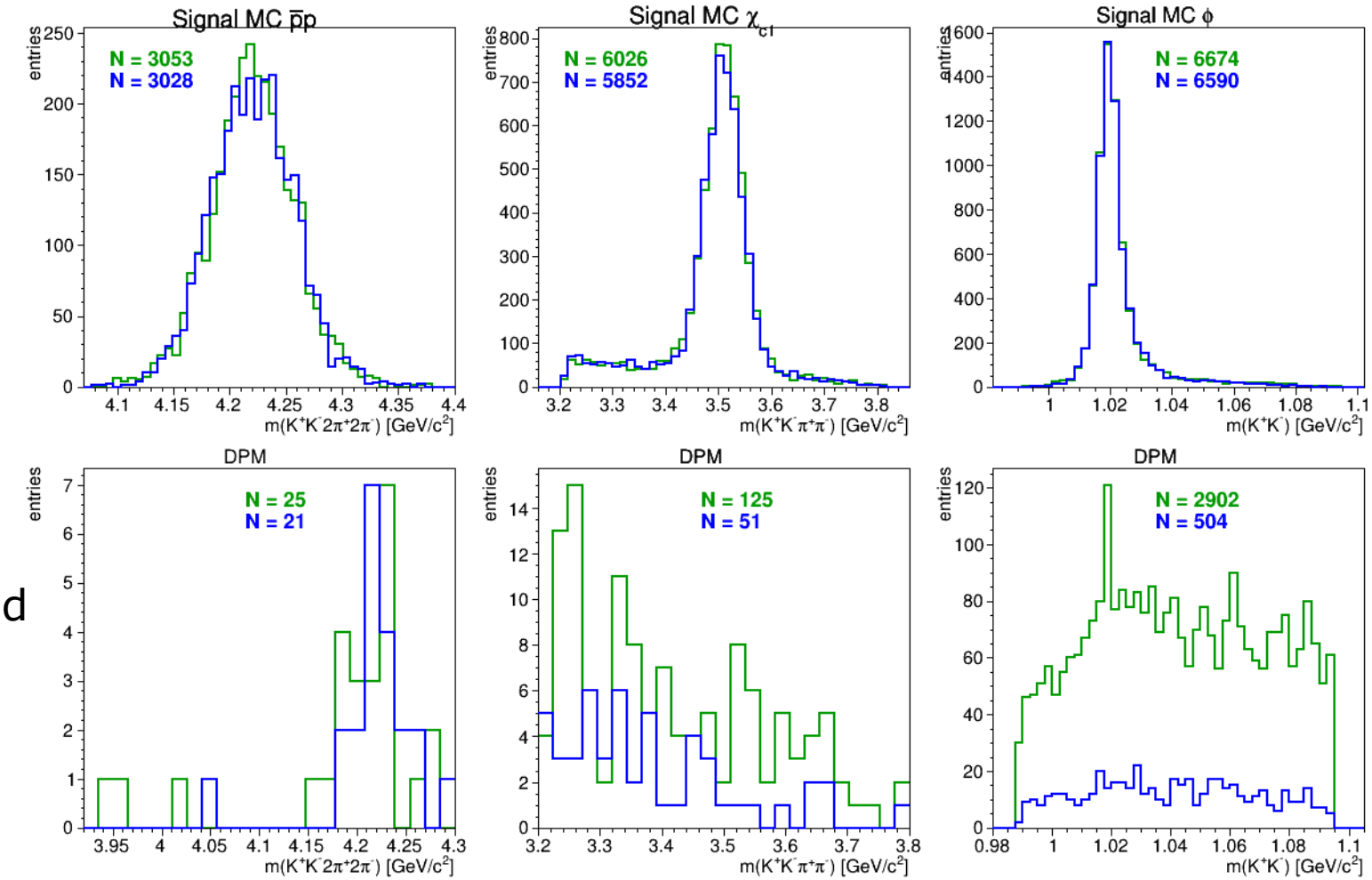
Reconstruct $\bar{p}p \to \chi_{c1}\pi^+\pi^- \to (\varphi\pi^+\pi^-)\pi^+\pi^- \to K^+K^-2\pi^+2\pi^-$

Filter: $N(t^+) \geq 3$, $N(t^-) \geq 3$, $|m_{KK} - m_\varphi| < 50$MeV, $|m_{2K2\pi} - m_{\chi c}| < 300$MeV

→ Unfiltered DPM: **1.1M** ev; filtered DPM: **0.1M** ev (→ 11x less simulation!!)

Signal MC

Background (DPM)



Event Filter Example: $\bar{p}p \to \chi_{c1}\pi^+\pi^-$

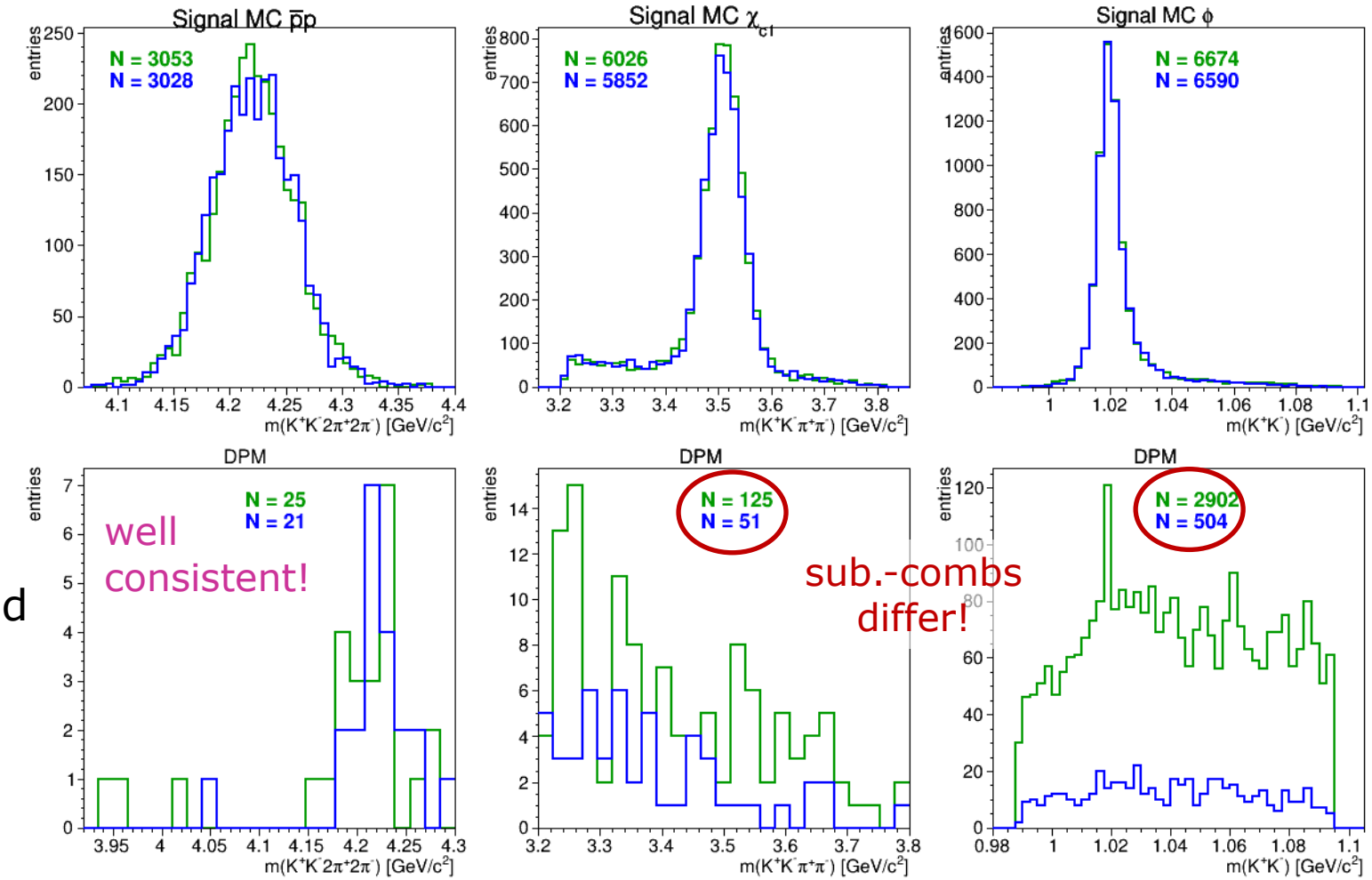
Reconstruct $\bar{p}p \to \chi_{c1}\pi^+\pi^- \to (\varphi\pi^+\pi^-)\pi^+\pi^- \to K^+\ K^-2\pi^+2\pi^-$

Filter: $N(t^+) \geq 3$, $N(t^-) \geq 3$, $|m_{KK} - m_\varphi| < 50$MeV, $|m_{2K2\pi} - m_{\chi c}| < 300$MeV

→ Unfiltered DPM: **1.1M** ev; filtered DPM: **0.1M** ev (→ 11x less simulation!!)

Signal MC

Background (DPM)



EVENT SHAPE VARIABLES

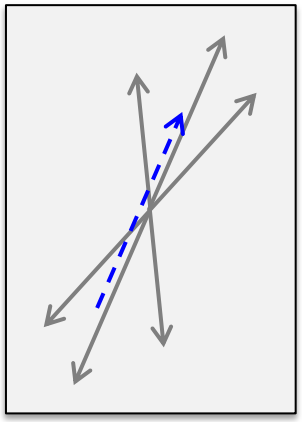
Event Shape Variables

- Sometimes signal and background differ in overall event shape
- Useful Event Shape variables (always computed in CMS) are
  - Thrust
  - Sphericity
  - (A)planarity
  - Circularity
  - Fox-Wolfram Moments

Thrust (- Vector)

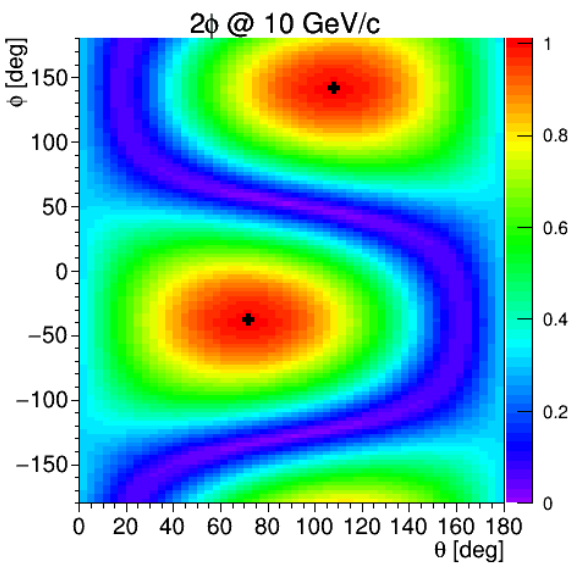
- Thrust-axis = "Jet direction" of event

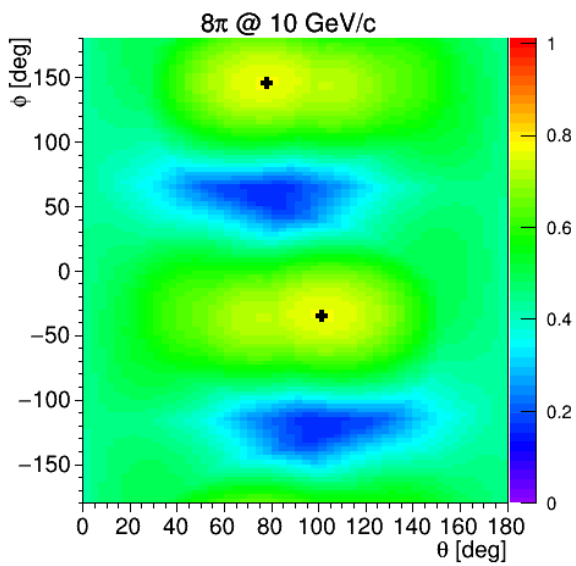
$$T = \max_{|n|=1} \frac{\sum_i |n \cdot p_i|}{\sum_i |p_i|}$$

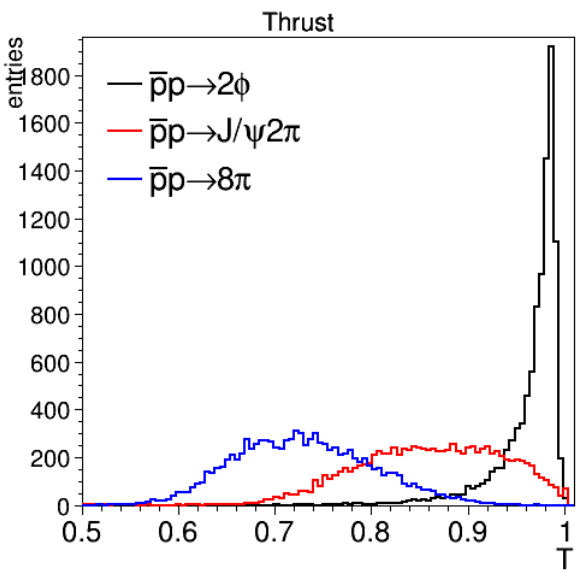


with thrust axis n and $\frac{1}{2} \leq T \leq 1$

isotropic
jet-like







Sphericity - (A)planarity - Circularity

- Base for all is the sphericity tensor $S^{\alpha\beta}$

$$S^{\alpha\beta} = \frac{\sum_i p_i^\alpha \cdot p_i^\beta}{\sum_i |p_i|^2}$$

with $\alpha, \beta = 1,2,3$ and eigenvalues $\lambda_1,\ \lambda_2,\ \lambda_3$

- Sphericity $S = \frac{3}{2}(\lambda_2 + \lambda_3)$ with $0 \leq S \leq 1$ isotropic / jet-like
- Aplanarity $A = \frac{3}{2}\lambda_3$ with $0 \leq A \leq \frac{1}{2}$ isotropic / planar
- Planarity $P = \lambda_2 - \lambda_3$ with $0 \leq P \leq \frac{1}{2}$
- Circularity $C = \frac{2 \cdot \min(\lambda_1, \lambda_2)}{\lambda_1 + \lambda_2}$ with $0 \leq C \leq 1$

Fox-Wolfram Moments

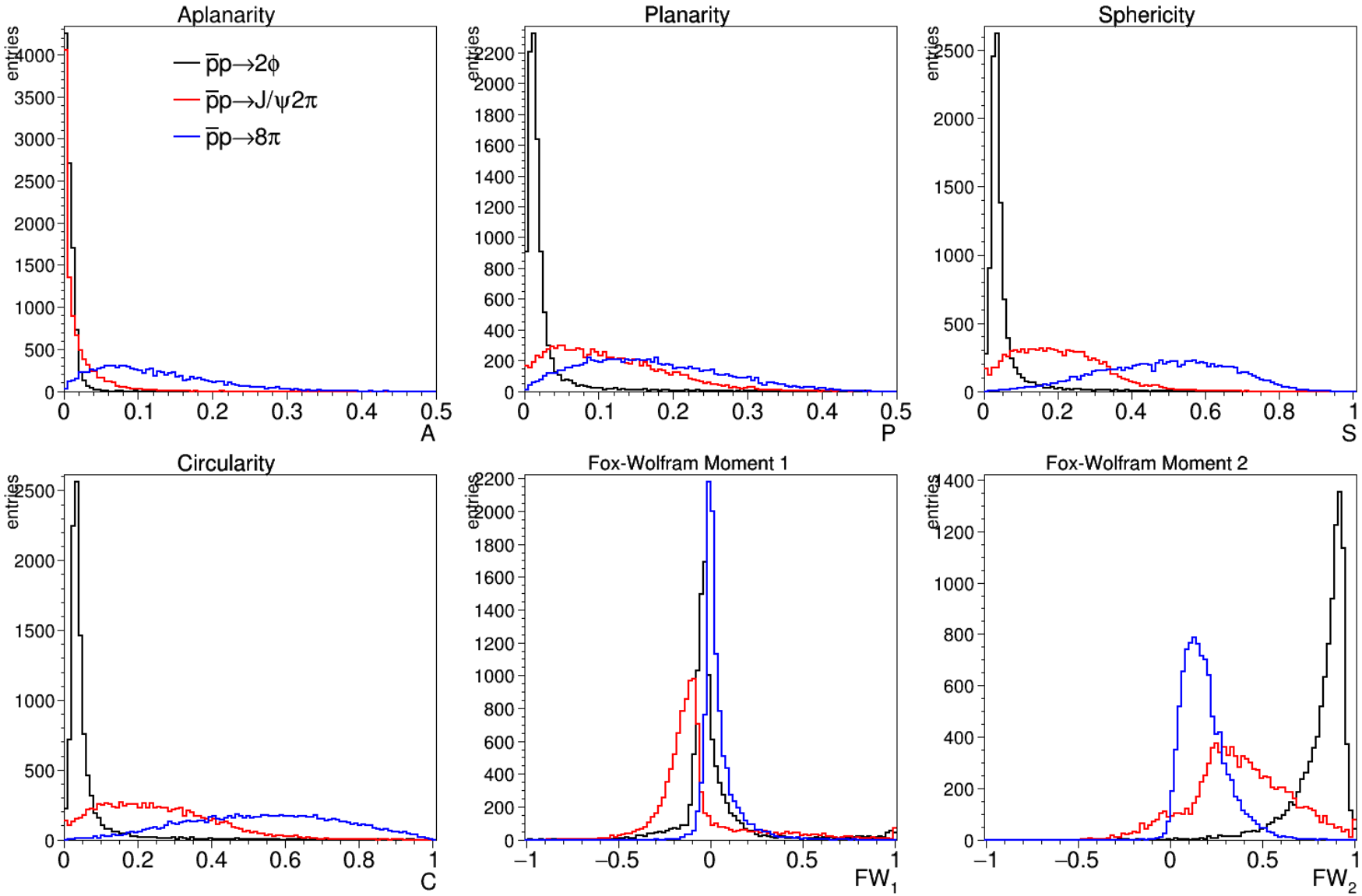
- Fox-Wolfram Moments are defined by

$$H_l = \sum_{i,j} \frac{|p_i| \cdot |p_j|}{E_{vis}^2} \cdot P_l\left(\cos\theta_{ij}\right)$$

with

  - θ_{ij} = opening angle of particles *i,j*
  - E_{vis} = visible energy
  - $P_l\left(\cos\theta_{ij}\right)$ = Legendre polynomials
- Balanced events: $H_1 = 0$
- Jet-like events: $H_l \approx 1$ for l=even and $H_l \approx 0$ for l=odd

Sphericity - (A)planarity - Circularity - Fox Wolfram Mom



Event (Shape) Variables in PandaROOT/Rho

- Class PndEventShape offers many variables

```
PndEventShape(RhoCandList &all, TLorentzVector pbp_sys, double neutMinE=0, double chrgMinP=0)
```

- Global multiplictities: NParticles, NCharged, NNeutral
- Event shape variables
  - Sphericity, Planarity, Aplanarity, Circularity, Thrust, ThrustVector
  - FoxWolfMomH(order), FoxWolfMomR(order)
- Minimum/maximum energy and (transverse) momentum (lab, cms)
  - PmaxLab/Cms, PminLab/Cms, EmaxNeutLab/Cms, PmaxChrgLab/Cms,...
- (Transverse) momentum and energy sums (lab, cms)
  - PtSumLab, NeutESumLab/Cms, ChrgPSumLab/Cms
- Multiplicities with (transv.) min/max momentum/energy (lab, cms)
  - MultPminLab/Cms, MultPmaxLab/Cms, MultPtminLab/Cms, MultPtmaxLab/Cms
  - MultChrgPminLab/Cms, MultChrgPmaxLab/Cms, MultNeutEminLab/Cms
- PID multiplicities with minimum probability and momentum (lab, cms)
  - MultElectronPminCms, MultMuonPminCms,....
- (Transv.) momentum and energy sums with min momentum/energy (lab, cms)
  - SumPminLab/Cms, SumNeutEminLab/Cms,...

SPECIAL KINEMATICS

Special Kinematics

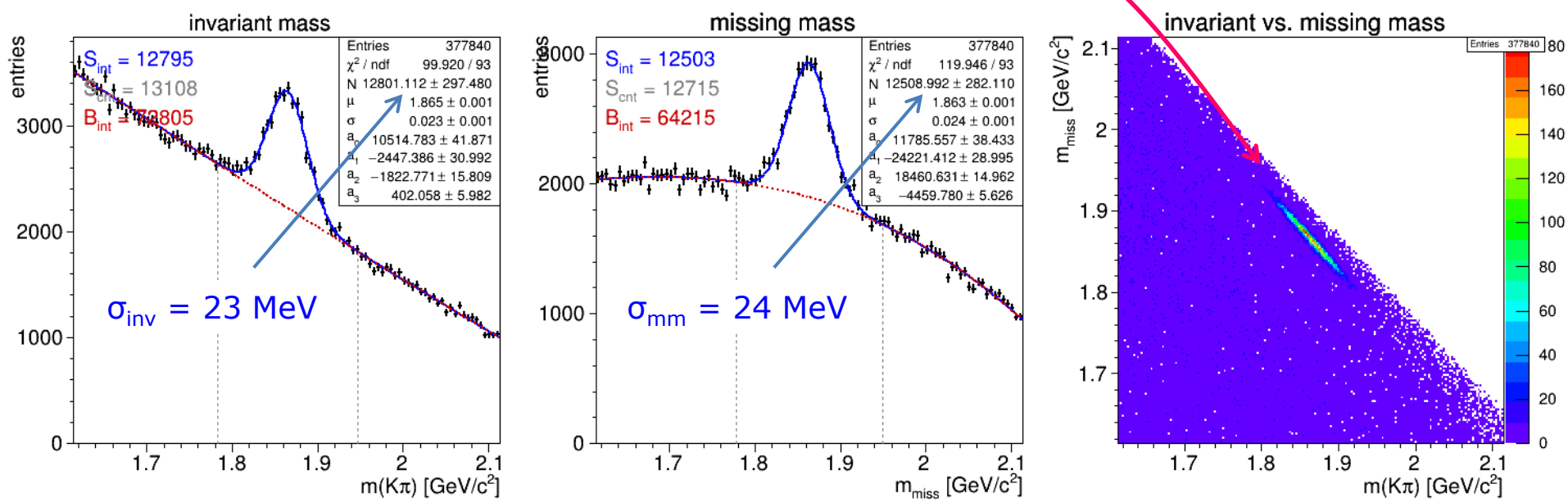
- Some signal reactions exhibt special properties for selection
- Prominent cases are
  - Double resonance production close to threshold
    Examples: $\bar{p}p \to D\overline{D}$, $\bar{p}p \to \Lambda_c^+\overline{\Lambda}_c^-$, $\bar{p}p \to D\overline{D^*}$, ...
  - Cascaded decays with low momentum particle emission
    Example: $D^{*+} \to D^0\pi^+$, $D_s^{*+} \to D_s^+\gamma$, $\chi_{c1} \to J/\psi\ \gamma$

Double Resonance Production at Threshold

- Fixed and known E_{cm}
  → resolution of invariant mass correlates with missing mass

$$m_{\text{miss}} = \sqrt{\left(E_{p\bar{p}} - E_{\text{rec}}\right)^2 - \left(\vec{p}_{p\bar{p}} - \vec{p}_{\text{rec}}\right)^2}$$

- Example: $\bar{p}p \rightarrow D\bar{D}, D/\bar{D} \rightarrow K^{\mp}\pi^{\pm}$ @ E_{cm} = 3.75 GeV
  - invariant mass and missing mass resolution about 24 MeV
  - but very narrow correlation ellipsis

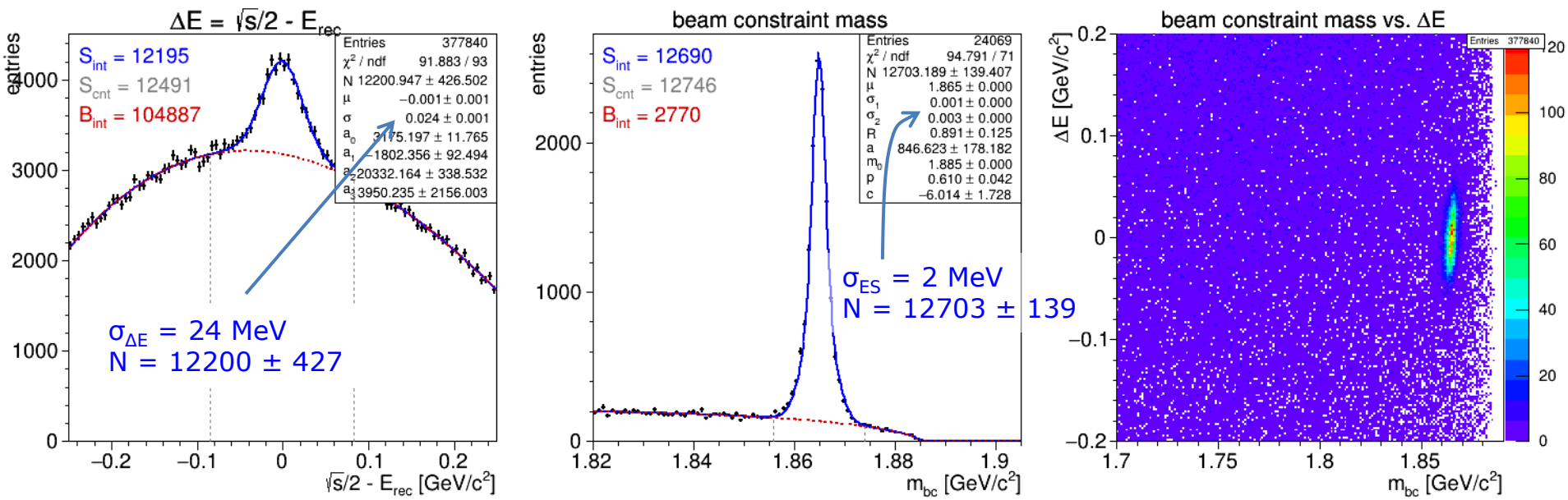


Beam Energy Substituted Mass m_{ES}

- In case of resonance - anti-resonance production (like $D\bar{D}$) → can infer the resonance's energy from the beam energy
- So called energy-subsituted mass is (* = in cm-frame)

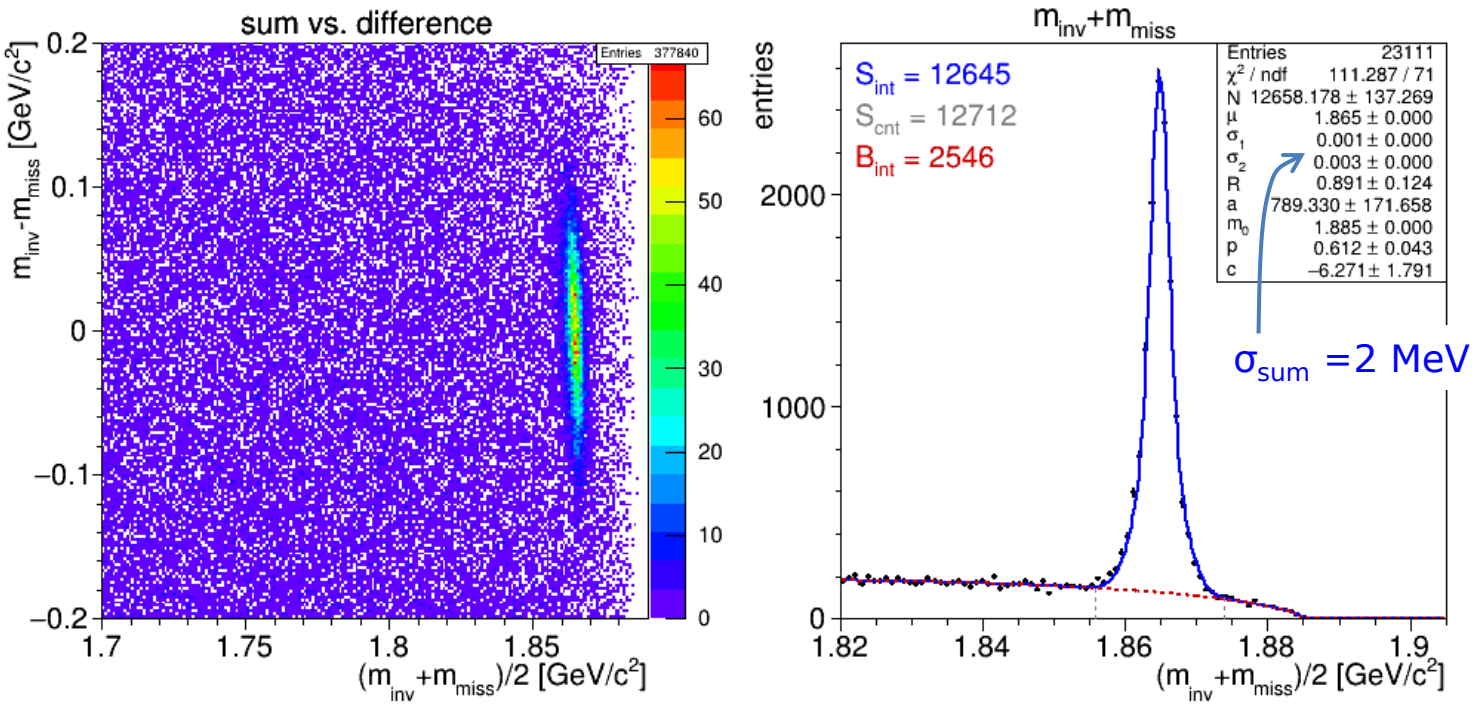
$$m_{\mathrm{ES}} = \sqrt{\left(\frac{E^*_{\mathrm{cm}}}{2}\right)^2 - (p^*_{\mathrm{rec}})^2}$$, combined with $$\Delta E = \left(\frac{E^*_{\mathrm{cm}}}{2}\right)^2 - E_{\mathrm{rec}}$$

- Much better S:N and thus relative error!



Double Resonance Production at Threshold

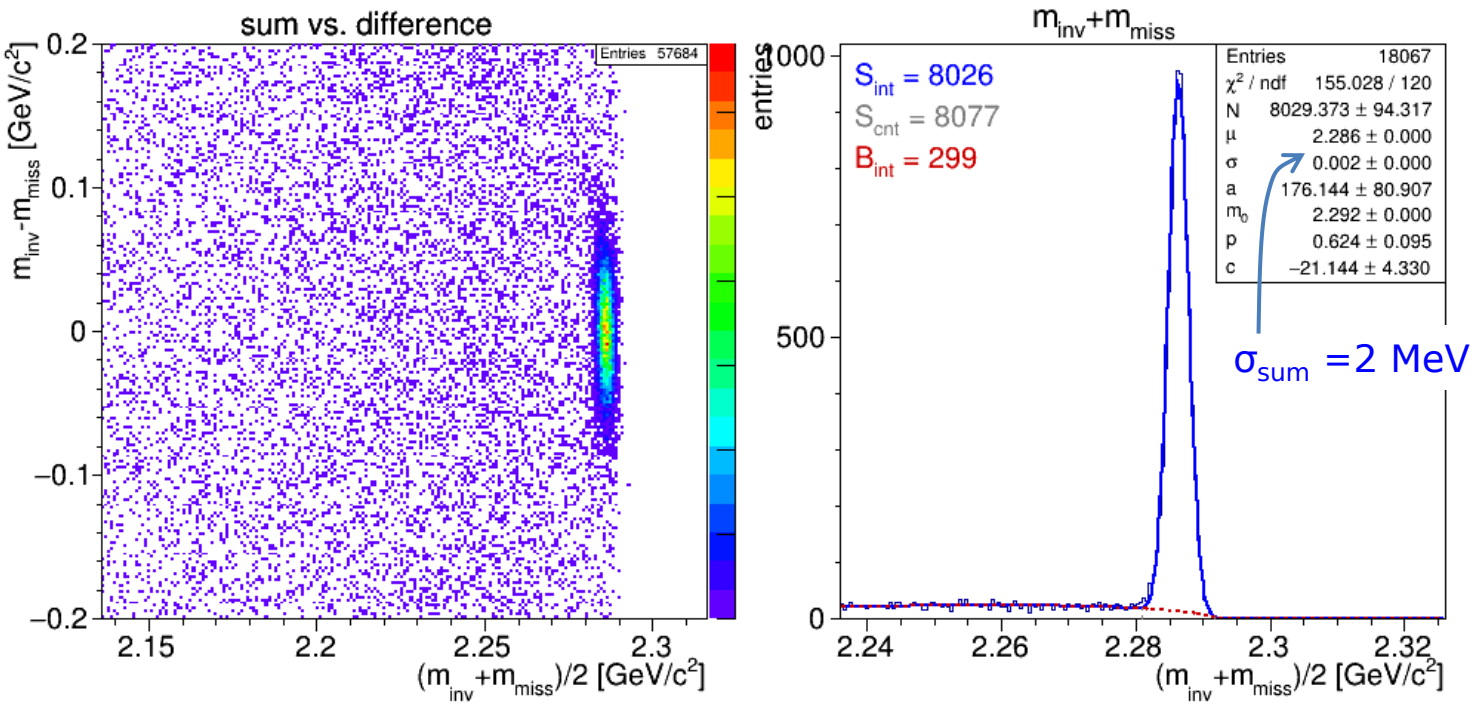
- Very similar result by using m_{inv} and m_{miss} directly
- Works also for $R\bar{R}'$ production (e.g. $D\bar{D}^*$) with different masses
- Plot difference vs. sum (or for $R\bar{R}$ sum/2; peaks at corr. mass)



Double Resonance Production at Threshold

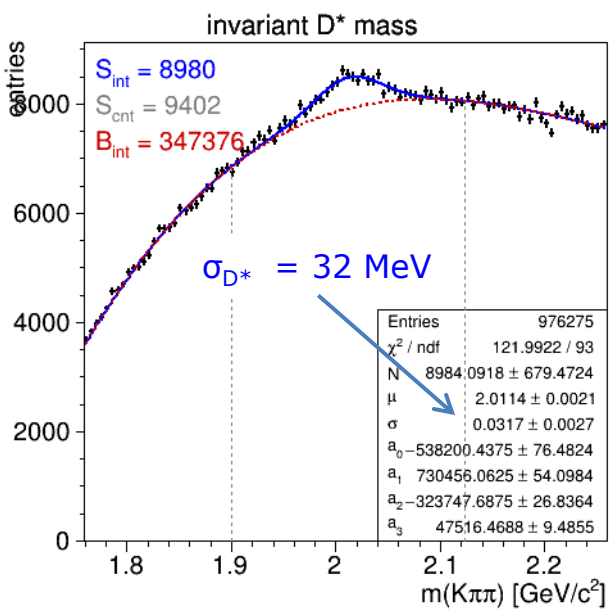
- Very similar result by using m_{inv} and m_{miss} directly
- Works also for $R\bar{R}'$ production (e.g. $D\bar{D}^*$) with different masses
- Plot difference vs. sum (or for $R\bar{R}$ sum/2; peaks at corr. mass)

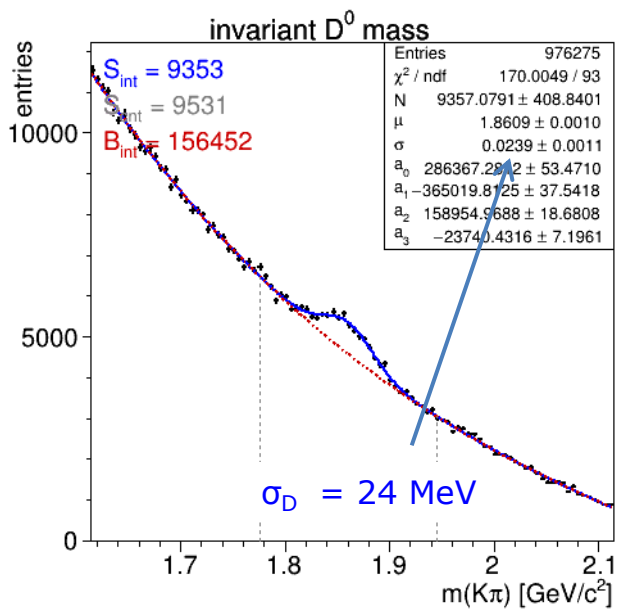
Also works for other cases like e.g. $\Lambda_c\overline{\Lambda_c}$

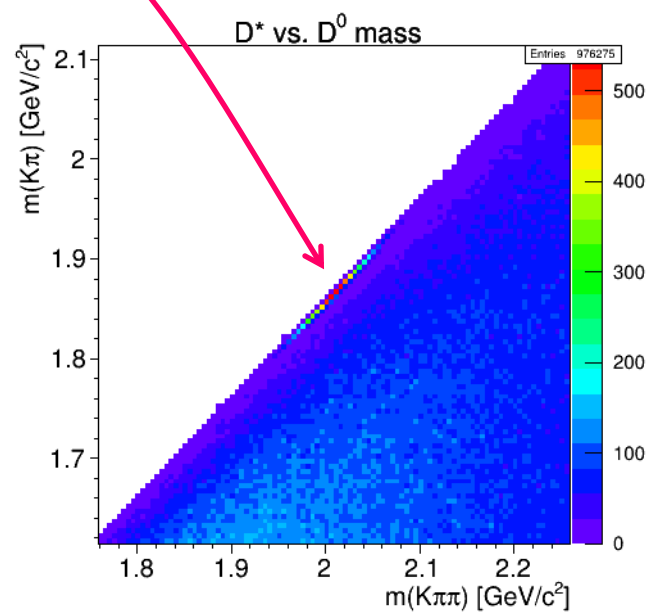


Cascaded Decays with Slow Particles

- Consider decay $D^{*+} \rightarrow D^0\pi^+$
  - Experimental resolution of D^{*+} dominated by that of D^0 due to combinatoric reconstruction
  - Strong correlation between mass distributions
    - very narrow correlation ellipsis

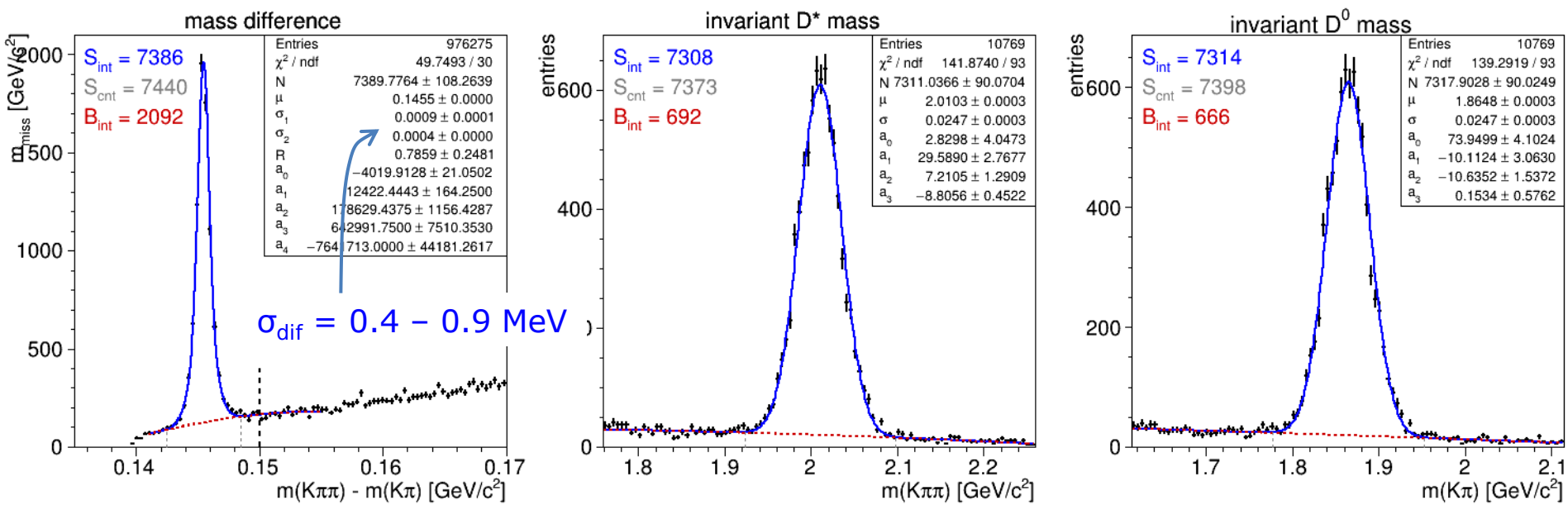






Cascaded Decays with Slow Particles

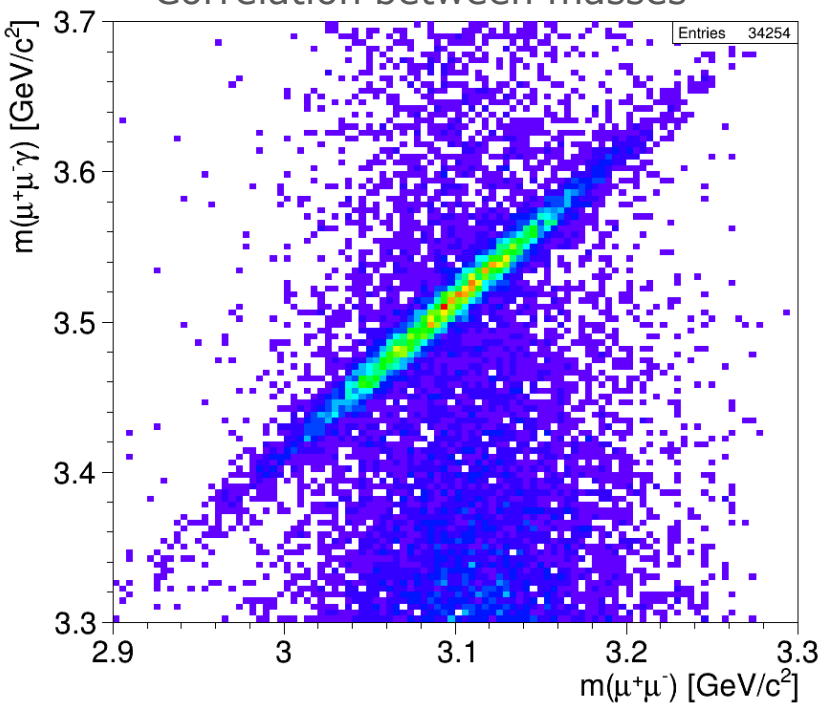
- Good quantity is the mass difference m(D*) - m(D)
- Close to threshold (small nominal difference)
- Very narrow structure to be selected on
- Much better S:N and rel. error
  - $\left(\frac{\Delta N}{N}\right)_{\text{before}} = \frac{679}{8980} = 7.5\% \longrightarrow \left(\frac{\Delta N}{N}\right)_{\text{after}} = \frac{90}{7311} = 1.2\%$

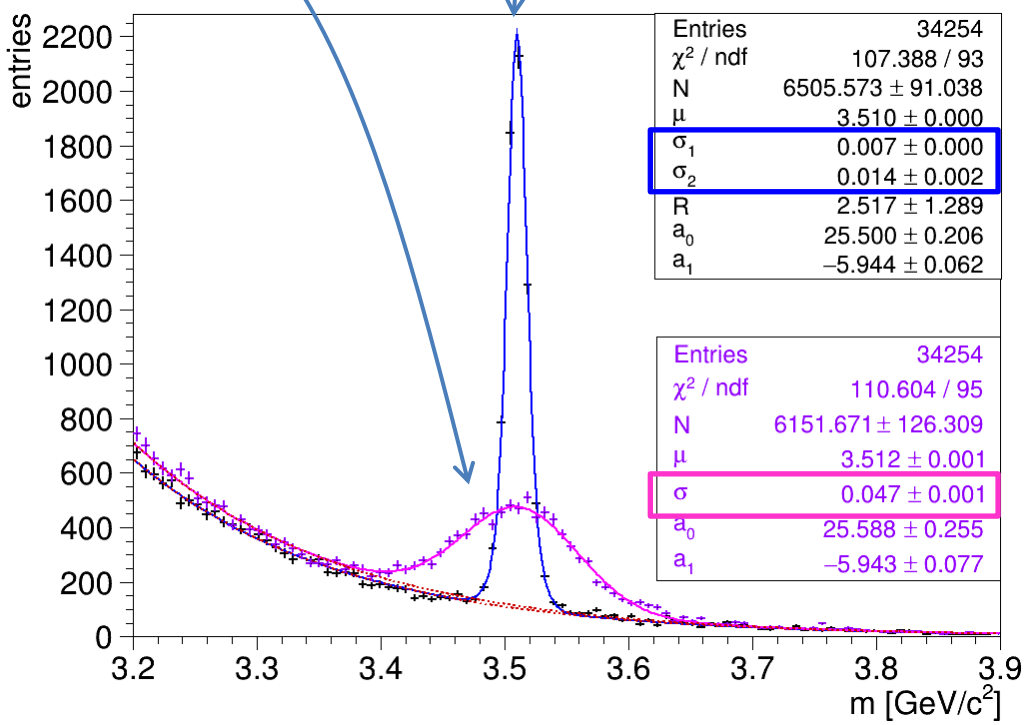


Cascaded Decays with Slow Particles

- Another example: $\chi_{c1} \rightarrow J/\psi(\rightarrow \mu^+\mu^-)\ \gamma$
- Compare mass $m(\mu^+\mu^-\gamma)$ with $m(\mu^+\mu^-\gamma) - m(\mu^+\mu^-) + m_{J/\psi,PDG}$
- Resolution better by factor 4

Correlation between masses





CREATING OUTPUT

TTrees with RhoTuple

- Creating TTree output without overhead

```
RhoTuple *ntp  = new RhoTuple("ntp","J/psi analysis");
 ...
   // ... in event loop ...
   for (j=0;j<jpsi.GetLength();++j)
   {
      // *** store event number and candidate number in current event
      ntp->Column("ev",    (Float_t) evnumber           );
      ntp->Column("cand",  (Float_t) j                  );

      // *** basic information about J/psi
      ntp->Column("jpsim", (Float_t) jpsi[j]->M()       );
      ntp->Column("jpsip", (Float_t) jpsi[j]->P()       );
      ntp->Column("jpsipt",(Float_t) jpsi[j]->P4().Pt() );
      ntp->Column("jpsiE", (Float_t) jpsi[j]->E()       );
      ....
      ntp->DumpData();
   }
 ...
 // ... end of macro ...
 TFile *f=new TFile("ntp.root","RECREATE");
 ntp->GetInternalTree()->Write();
 f->Close();
```

Just create new branches on the fly!

Actually write out what is currently set

PndRhoTupleQA - TTree made simple

- Provides QA functions for persisting values in TTree

```
// *** QA for candidates
void qaCand(TString pre, RhoCandidate *cc, RhoTuple *n, bool skip=false);
void qaP4(TString pre, TLorentzVector c, RhoTuple *n, bool skip=false);
void qaP4Cms(TString pre, TLorentzVector c, RhoTuple *n, bool skip=false);
void qaP4Cov(TString pre, RhoCandidate *c, RhoTuple *n, bool skip=false);

// *** QA for composites
void qaComp(TString pre, RhoCandidate *c, RhoTuple *n);
void qaKs0(TString pre, RhoCandidate *c, RhoTuple *n);
void qaPi0(TString pre, RhoCandidate *c, RhoTuple *n);

// *** QA of event shape
void qaEventShape(TString pre, PndEventShape *evsh, RhoTuple *n);
void qaEventShapeShort(TString pre, PndEventShape *evsh, RhoTuple *n);

// *** QA for parts of eventshape
void qaESPidMult(TString pre, PndEventShape *evsh, double prob, double pmin, RhoTuple *n);
void qaESMult(TString pre, PndEventShape *evsh, RhoTuple *n);
void qaESSum(TString pre, PndEventShape *evsh, RhoTuple *n);
void qaESMinMax(TString pre, PndEventShape *evsh, RhoTuple *n);
void qaESEventVars(TString pre, PndEventShape *evsh, RhoTuple *n);

// *** QA track, vtx, PID, decay
void qaVtx(TString pre, RhoCandidate *c, RhoTuple *n);
void qaPoca(TString pre, RhoCandidate *c, RhoTuple *n);
....
```

Very handy function to store composite info!

PndRhoTupleQA - Application

- In situ it might look like this:

```
double pbarmom = 15.0;
PndAnalysis *ana = new PndAnalysis();

PndRhoTupleQA qa(ana, pbarmom);

RhoTuple *ntp  = new RhoTuple("ntp","J/psi analysis");
...
  // ... in event loop ...
  for (j=0;j<jpsi.GetLength();++j)
  {
     // *** store event number and candidate number in current event
     ntp->Column("ev",    (Float_t) evnumber );
     ntp->Column("cand",  (Float_t) j        );

     // *** all information about composite J/psi,
     // **** including info about daughters, 2-body quantities and MC truth
     qa.qaComp("x", jpsi[j], ntp);

     // *** and fill ntuple
     ntp->DumpData();
  }
```

PndRhoTupleQA - Application

- In situ

This creates branches:

	xpdg	xd0pz	xd0pzcm	xd1py	xd1pycm	xpocvx
xpx	xpxcm	xd0e	xd0ecm	xd1pz	xd1pzcm	xpocvy
xpy	xpycm	xd0p	xd0pcm	xd1e	xd1ecm	xpocvz
xpz	xpzcm	xd0tht	xd0thtcm	xd1p	xd1pcm	xpocmag
xe	xecm	xd0phi	xd0phicm	xd1tht	xd1thtcm	xpocqa
xp	xpcm	xd0pt	xd0mct	xd1phi	xd1phicm	xpocdist
xtht	xthtcm	xd0m	xd0pide	xd1pt	xd1mct	xpocctau
xphi	xphicm	xd0chg	xd0pidmu	xd1m	xd1pide	xvx
xpt	xmct	xd0x	xd0pidpi	xd1chg	xd1pidmu	xvy
xm	xmdif	xd0y	xd0pidk	xd1x	xd1pidpi	xvz
xchg	xoang	xd0z	xd0pidp	xd1y	xd1pidk	xlen
xx	xdecang	xd0l	xd0pidmax	xd1z	xd1pidp	xctau
xy	xcdecang	xd0pdg	xd0pidbest	xd1l	xd1pidmax	xctaud
xz	xd0px	xd0pxcm	xd0trpdg	xd1pdg	xd1pidbest	
xl	xd0py	xd0pycm	xd1px	xd1pxcm	xd1trpdg	

```
double
PndAnal

PndRhoT

RhoTupl
...
  // ..
  for (
  {
     //                                                    ent
     nt
     nt

     // *** all          ion about composite J/psi,
     // **** including info about daughters, 2-body quantities and MC truth
     qa.qaComp("x", jpsi[j], ntp);

     // *** and fill ntuple
     ntp->DumpData();
  }
```

PndRhoTupleQA - Application

- In situ

```
double
PndAnal

PndRhoT

RhoTupl
...
  // ..
  for (
  {
     //                                                    ent
     nt
     nt

     // *** all          ion about composite J/psi,
     // **** including info about daughters, 2-body quantities and MC truth
     qa.qaComp("x", jpsi[j], ntp);

     // *** and fill ntuple
     ntp->DumpData();
  }
```

This creates branches:

infos about daugther 0

	xpdg	xd0pz	xd0pzcm	xd1py	xd1pycm	xpocvx
xpx	xpxcm	xd0e	xd0ecm	xd1pz	xd1pzcm	xpocvy
xpy	xpycm	xd0p	xd0pcm	xd1e	xd1ecm	xpocvz
xpz	xpzcm	xd0tht	xd0thtcm	xd1p	xd1pcm	xpocmag
xe	xecm	xd0phi	xd0phicm	xd1tht	xd1thtcm	xpocqa
xp	xpcm	xd0pt	xd0mct	xd1phi	xd1phicm	xpocdist
xtht	xthtcm	xd0m	xd0pide	xd1pt	xd1mct	xpocctau
xphi	xphicm	xd0chg	xd0pidmu	xd1m	xd1pide	xvx
xpt	xmct	xd0x	xd0pidpi	xd1chg	xd1pidmu	xvy
xm	xmdif	xd0y	xd0pidk	xd1x	xd1pidpi	xvz
xchg	xoang	xd0z	xd0pidp	xd1y	xd1pidk	xlen
xx	xdecang	xd0l	xd0pidmax	xd1z	xd1pidp	xctau
xy	xcdecang	xd0pdg	xd0pidbest	xd1l	xd1pidmax	xctaud
xz	xd0px	xd0pxcm	xd0trpdg	xd1pdg	xd1pidbest	
xl	xd0py	xd0pycm	xd1px	xd1pxcm	xd1trpdg	

QUICK ANALYSIS TOOLS

Quick Analysis

- Based on PndSimpleCombiner & PndSimpleCombinerTask
  - very simple and compact analysis approach
  - runs analysis in FairTask (more reliable)
- Reco e.g. of mode $\psi' \rightarrow J/\psi\ (\rightarrow \mu^+\mu^-)\ \pi^+\pi^-$
  incl. vertex and 4C fit can be done by (tutorials/analysis):

```
root -l -b -q 'quickana.C(
     „signal_pid.root",                                    // input file
      6.232,                                               // p [GeV/c]
     "J/psi->mu+ mu-; pbarpSystem->J/psi pi+ pi-",         // decay tree reco
      0,                                                   // #events (0=all)
     "fit4c<100:fitvtx:mwin(J/psi)=0.8:pidmu=Tight")'      // some parameters
```

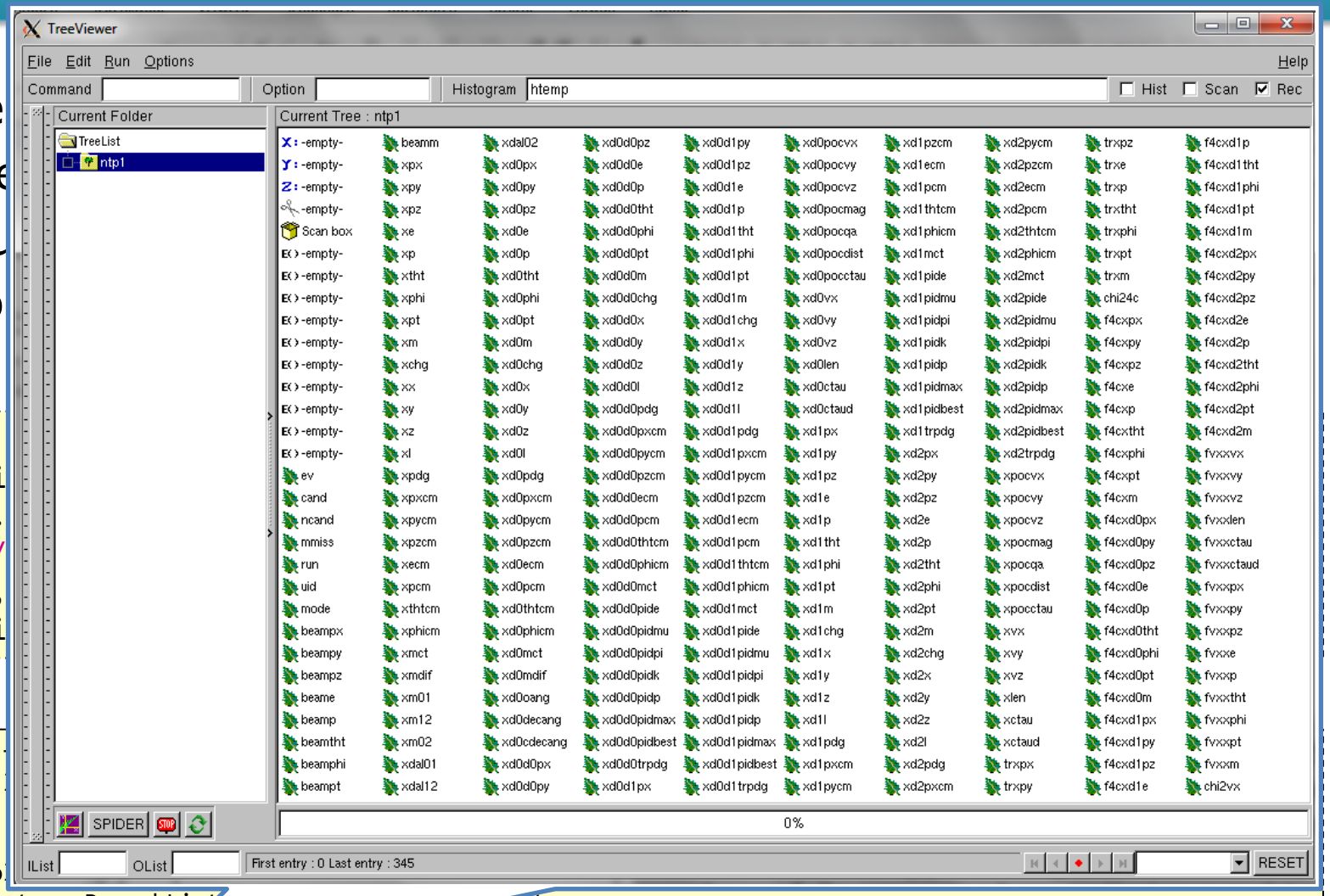
```
Attaching file signal_pid_ana.root as _file0...
root [1] .ls
TFile**         signal_pid_ana.root
 TFile*         signal_pid_ana.root
  KEY: TFolder  cbmout;1        Main Output Folder
  KEY: TList    BranchList;1    Doubly linked list
  KEY: FairFileHeader   FileHeader;1
  KEY: TTree    ntp0;1  J/psi->mu+ mu-
  KEY: TTree    ntp1;1  pbarpSystem->J/psi pi+ pi-
```

Quick Analysis

- Base
  - ve
  - ru
- Reco
  incl.

```
root -l
      „si
       6.
      "J/
       0,
      "fi
```

```
Attaching
root [1] .
TFile**
 TFile*
  KEY: TFo
  KEY: TList     BranchList                                  list
  KEY: FairFileHeader
  KEY: TTree     ntp0;                  /psi->mu+ mu-
  KEY: TTree     ntp1;1  pbarpSystem->J/psi pi+ pi-
```



Quick Analysis Parameters

https://panda-wiki.gsi.de/foswiki/bin/view/Computing/PandaRootRhoTutorial#A_5._Quick_analysis

Parameter	Meaning	
`fit4c[<x]`	Triggers 4-constraint-fitting of the last resonance in the decay list, applying an optional cut `<x` on the chi2.	fitting
`fitvtx[<x]`	Triggers vertex fit for all resonances with at least two tracks, applying an optional cut `<x` on the chi2.	fitting
`mwin = x`	Applies mass window (m_{rec} - m_{PDG}) < x/2 GeV/c² (window width `x`) for **all** resonances. Is overrided by following particle specific definitions.	mass cuts
`mwin(R) = x`	Applies mass window (m_{rec} - m_{PDG}) < x/2 GeV/c² for resonance type R, e.g. `mwin(pi0) = 0.05`. Overrides any previous settings, e.g. by `mwin`.	mass cuts
`mwin(R) = x|y`	Applies mass window x < m_{rec} < y GeV/c² for resonance R. Overrides any previous setting, e.g. by `mwin`.	mass cuts
`emin = x`	Minimum energy cut E>x GeV for photon candidates.	energy/momentum cut
`pmin = x`	Minimum momentum cut p>x GeV/c for tracks.	energy/momentum cut
`pid = x`	Common PID criterion for all particle species. x ∈ `[All, VeryLoose, Loose, Tight, VeryTight, Best]`. Default is `All`.	PID control
`pidY = x`	PID criterion for species Y. x ∈ `[All, VeryLoose, Loose, Tight, VeryTight, Best]`, Y ∈ `[e, mu, pi, k, p]`. Examples: `pidmu = VeryTight`, `pidk = Loose`.	PID control
`algo = x`	Common PID algorithm configuration x, e.g `algo = PidAlgoEmcBayes;PidAlgoDrc`. Default for all species is `PidAlgoEmcBayes;PidAlgoDrc;PidAlgoDisc;PidAlgoStt;PidAlgoMdtHardCuts;PidAlgoSciT;PidAlgoRich`	PID control
`algoY = x`	PID algorithm setting for species Y, Y ∈ `[e, mu, pi, k, p]`. Example: `algoe = PidAlgoEmcBayes: algok = PidAlgoDrc;PidAlgoDisc;PidAlgoStt`.	PID control
`ebrem`	Use the electron lists with bremsstrahlung correction (`ElectronBrem`).	
`qamc`	Stores the MC truth list in the n-tuple `nmc`.	
`qaevtshape`	Stores event shape variables in **each** n-tuple.	
`qapart`	Runs the task **PndParticleQATask**, useful for single particle studies. Creates n-tuples **ntp** (charged), **ntpn** (neutrals), **nmc** (MC truth list)	
`!ntpx`	Prevents storing the n-tuple from the x-th decay definition.	output control
`!mc`	Prevents **PndParticleQATask** to store the MC information in n-tuple `nmc`.	output control
`!neut`	Prevents **PndParticleQATask** to store the single neutral information in n-tuple `ntpn`.	output control
`!chrg`	Prevents **PndParticleQATask** to store the charged track information in n-tuple `ntp`.	output control

Quick Fast Simulation & Analysis

- Same channel with Fast Sim → no simulation stage needed!
  - ⇒ Events are generated on-the-fly

```
root -l -b -q 'quickfsimana.C(
      "jpsi",                                          // output prefix
      "pp_Jpsi2pi_Jpsi_mumu.dec",                      // generator
      6.232,                                           // p [GeV/c]
      "J/psi -> mu+ mu-; pbarpSystem -> J/psi pi+ pi-", // decay tree reco
      1000,                                            // # events generated
      "fit4c<100:fitvtx:mwin=0.8:pidmu=Tight",         // parameters
      0, 1, 23 )'                          // no software trigger, #run=1, #mode=23
```

```
TFile**         jpsi_0_ana.root
 TFile*         jpsi_0_ana.root
  KEY: TFolder  cbmroot;1       Main Folder
  KEY: TList    BranchList;1    Doubly linked list
  KEY: FairFileHeader   FileHeader;1
  KEY: TTree    ntp0;1  J/psi -> mu+ mu-
  KEY: TTree    ntp1;1  pbarpSystem -> J/psi pi+ pi-
  KEY: TTree    cbmsim;1        /cbmroot
```

Quick Fast Simulation & Analysis

- Generator configuration
  - `'xyz.dec[:iniRes]'` : decay file w/ optional initial resonance
  - `'DPM[1/2]'` : DPM; option 1 = inel. + el., 2 = el. only
  - `'FTF[1]'` : FTF; option 1 = inel. + elastic
  - `'BOX:type(pdg,mult):p(min,max):[c]tht(min,max):phi(min,max)'`
    - Configure species/multiplicity and one or multiple kin. ranges
- Decay pattern (names as in `TDatabasePDG`)
  - Bottom up definition → last decay step first
    - Wrong : `"D_s+ -> phi pi+; phi -> K+ K-"`
    - Correct: `"phi -> K+ K-; D_s+ -> phi pi+"`
  - Suppress automatic charged conjugate with `'nocc'`
    - `"D_s+ -> phi pi+ nocc"` (only reconstructs D_s^+)
- Parameters
  - as shown for Quick Analysis

Quick Fast Simulation & Analysis - Examples

Generator	Decay pattern	Parameters
mysignal.dec	J/psi -> e+ e-; pbp -> J/psi pi+ pi-	fit4c:mwin=0.3:qamc:qaevtshape
signal events; do 4C fit; apply mass window; store MC and event shape info in addition		
DPM	J/psi -> e+ e-; pbp -> J/psi pi+ pi-	fit4c:mwin=0.3:qamc:qaevtshape
create background events applying the same analysis		
mysignal.dec		persist
only run/stores FastSim output (PndPidCandidates) for signal events; no analysis reco		
FTF		nevt
stores only TTree with event variables for FTF background		
box:type[13,1]:p[2]:tht[0,60]		qapart:!neut:!mc
single muons; p = 2 GeV/c, 0° < tht < 60°; QA info for charged particles only		
mysignal.dec	K_S0 -> pi+ pi-; D0 -> K_S0 K+ K-	fitvtx:mwin(K_S0)=0.1:!ntp0
vertex fit; 0.1 GeV mass wind. K_S; no mass wind. D0; suppress K_S0 ntuple output		

Multi Variate Analysis with TMVA

- Once you have created ROOT output
  - Need to discriminate signal from background
- Either do by inspecting variables by hand or use some automatic tool after indentification of potential good variables:

  ⇒ **TMVA** = ROOT Multi Variate Analysis Toolset
  [https://root.cern.ch/tmva]

- Simple demo available:

  `TMVATrainer.C, TMVATrainer_608.C, TMVATester.C`

  Interface change in ROOT Ver. > 6.08

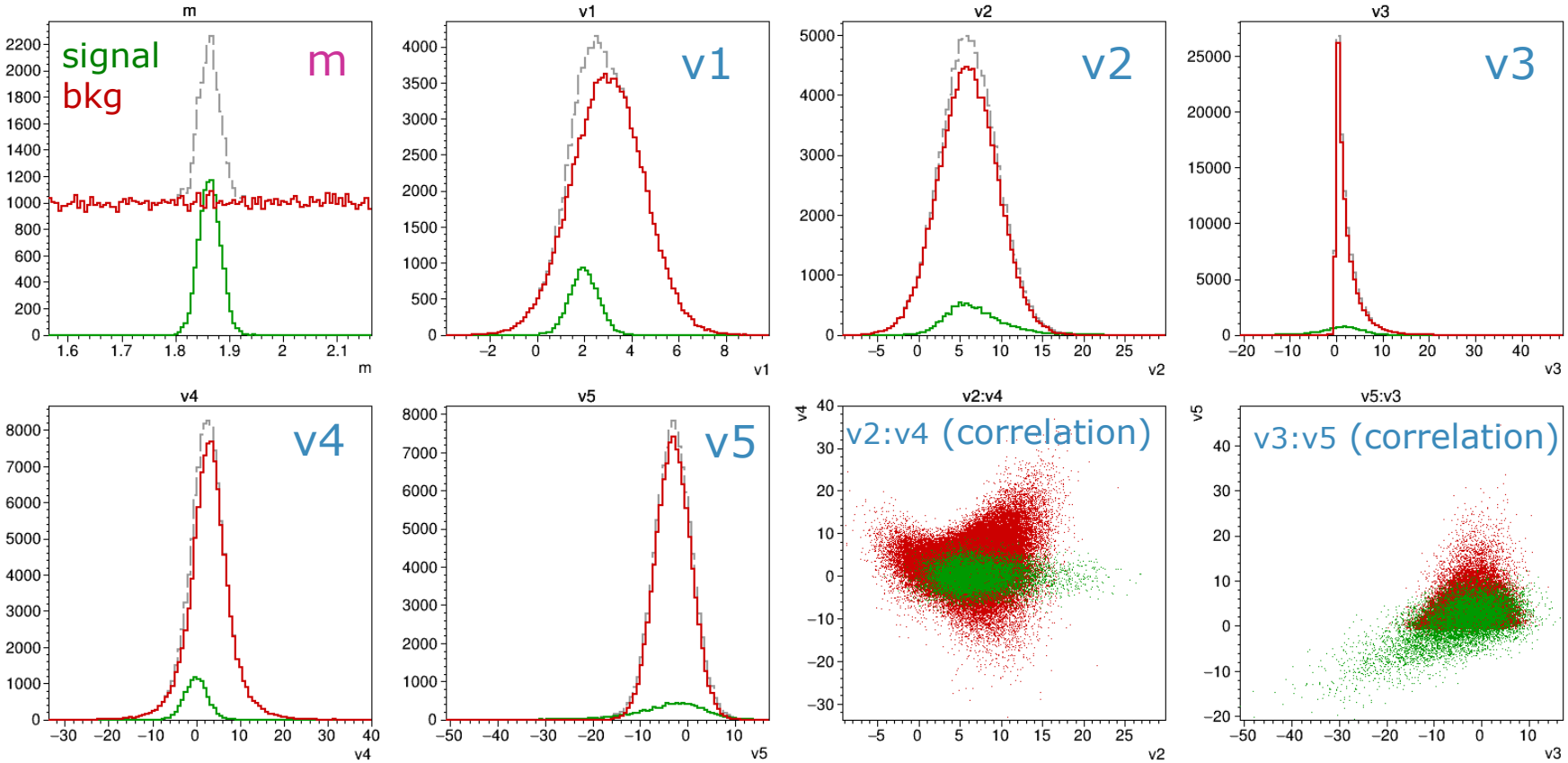
  allow to test
  - Boosted Decision Trees (BDT)
  - Multi Layer Perceptron (MLP)
  - Conventional Likelihood Methode (Likelihood)

Multi Variate Analysis with TMVA

- For demonstration, need some test data; create with

```
tutorials/thailand2017> root -l -b -q makeTMVADemoData.C  // -> demodata.root
```

→ "mass" m + 5 add. variables (and flag 'signal' to distinguish classes)



Multi Variate Analysis with TMVA

- Training of the classifier (using 80% of data for training)

```
> root -l -b -q TMVATrainer.C("demodata.root",        // the input ROOT file
                               "ntp",                 // name of the TTree
                               "signal>0",            // cut selection signal
                               "v1 v2 v3 v4 v5",      // list of variables
                               "BDT",                 // BDT, MLP, LH
                               "")                    // optional precut
```

- Running the classifier

```
> root -l       TMVATester.C("demodata.root",         // the input ROOT file
                             "ntp",                  // name of the TTree
                             "signal>0",             // cut selection signal
                             "weights/..BDT...xml",  // weights file
                             "")                     // optional precut
```

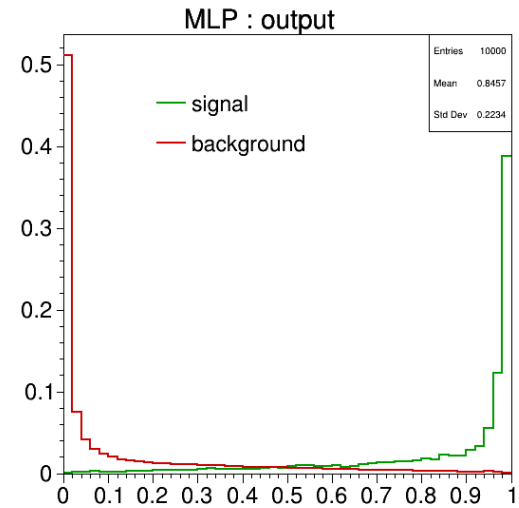
- Plots classifier output after training, ROC curve* and quality

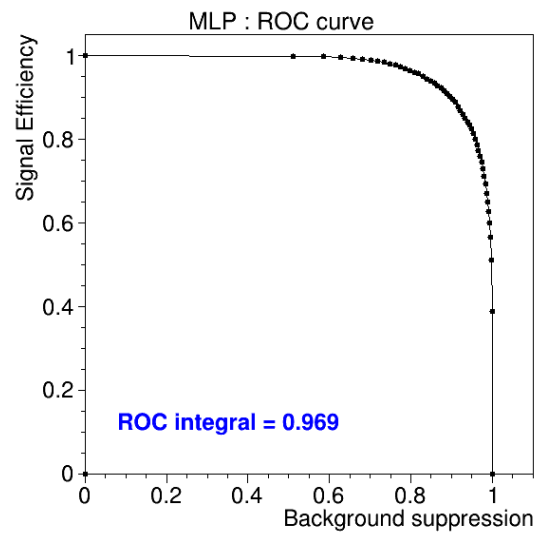
(*Receiver output characteristics = quality figure for binary classifiers)

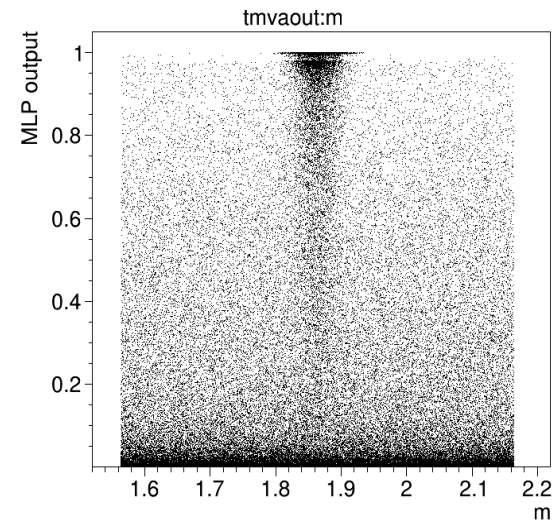
Multi Variate Analysis with TMVA

- MLP and BDT classifier performance:

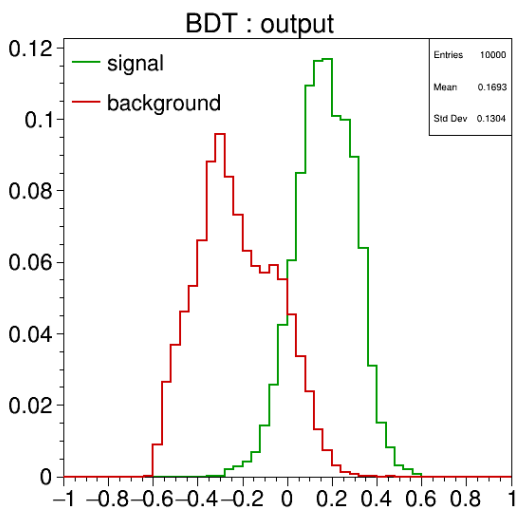
MLP

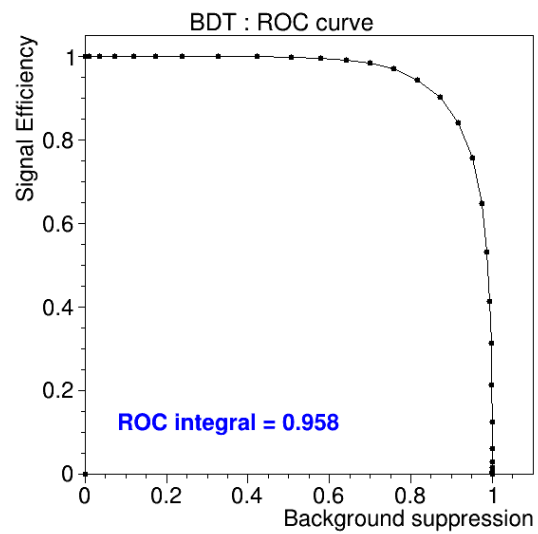


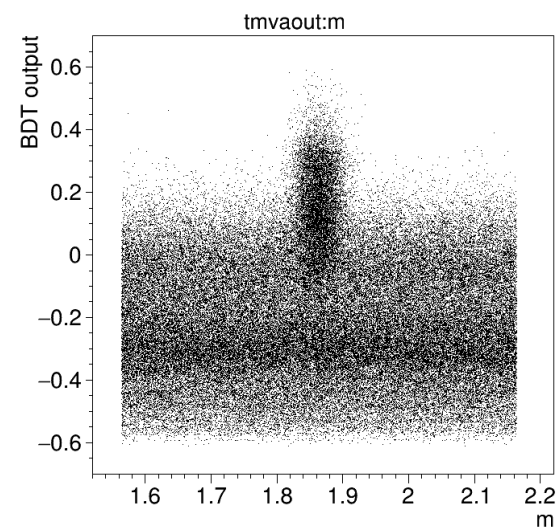




BDT







EXERCISES

Exercises Suggestions

- Rho Tutorial website:
  http://panda-wiki.gsi.de/cgi-bin/view/Computing/PandaRootRhoTutorial
- Take a look to `tutorials/thailand2017/README`
  `Exercises: #10 - #12`

1. Try event-filter in fast/full sim. macros and study impact
2. Simulate some events with quickfsimana.C
3. Create signal and background for some channel
4. Merge both analysis ROOT files with hadd
5. Study different variables
6. Train different TMVA classifiers on merged ROOT file

Exercises Preparation

Preparation/hints in tutorials/thailand2017/README

- **FIRST setup environment: source ~/workshop/build/PandaRoot_trunk.../config.sh**

- To have some data for tutorial macros, do one of the following

```
a) . ./tut_runall.sh 1000   # sim/reco 1000 events pbarp -> J/psi pi+ pi-
b) cp data/signal_p*root .  # preproduced default data
```

- Macros named 'tut_ana...C' are stubs and should be completed by you.

- At places marked with ' #### EXERCISE: ...' some code needs to be added;

```
Run macros (default or different input data) with
> root -l tut_ana...C                 # signal_pid.root, signal_par.root
> root -l 'tut_ana...C(0,"mydata")'   # mydata_pid.root, mydata_par.root
```

- If getting stuck, sample solutions are in the subfolder 'solution'

```
Run solution macros directly (default or different input data) with
> root -l solution/tut_ana...C
> root -l 'solution/tut_ana...C(0,"mydata")'
```