MESONEX ANALYSIS TOOLS

Derek Glazier
University of Glasgow

13/11/18
Context

Software layout

General Analysis Algorithm

Data classes

Use of TMVA

Use of Jupyter ROOTbooks

Further developments

With much support at Glasgow Uni.

Undergraduates
Michael Williams, Adam Mihalyi
Richard Tyson, Justinas Slepavicius

PostGrads
Louise Clark, Chris Mullen, Gavin Murdoch,
Peter Pauli, Adam Thornton

Staff
Dave Ireland, Bryan McKinnon
HSFINALSTATE ALGORITHM

HSdata Event
vector<THSParticle>
EventInfo

Loop over FinalStates

Loop over Topologies

Check particle Ids

Loop over combitorials

Topo_X()
PostTopoAction()
Kinematics()
CheckTruth()
PostWorkAction()
UserProcess()

Done

Done

User
Defined FinalStates
Defined Topologies
Use PIDs?
Use Inclusive events?
Cut definitions
Output Data
Tree
Histogram

User may provide
User must provide

Same code data/sim

Same code for different topologies
You can use default or create your own equivalent and set HSEXP=your_dir

**THSParticle**
- HSLorentzVector fP4;
- HSLorentzVector fTruthP4;
- HSPosition fVertex;
- HSPosition fTruthV;
- TMatrixD fCovMatrix;
- Double32_t fPDGMass=0;
- Double32_t fMeasMass=0;
- Double32_t fTime=0;
- Double32_t fPath=0;
- Double32_t fDoca=0;
- Double32_t fEdep=0;
- Double32_t fDeltaE=0;
- Double32_t fPreE=0;
- Float_t fTrChi2=0;
- Int_t fNPhot=0;
- Short_t fPDGCode=0;
- Short_t fTruthPDG=0;
- Short_t fDetector=0;
- Short_t fStatus=0;
- Short_t fFiducialCut=1;

**EventInfo**
- Long_t fVTPTrigBit=0; //clas12 vtp trigger bit
- Long_t fTrigBit=0; //clas12 trigger bit
- Float_t fCJSTTime=0;//COATJAVA defined starttime
- Float_t fRFTime=0;//RF Time (ns)
- Float_t fBeamPol=0; //helicity polarisation of beam
- Float_t fTarPol=0;
- Int_t fNEvent=0;//Event Number
- Short_t fBeamHel=0; //helcity direction of beam

**RunInfo**
- Float_t fTotCharge=0;
- Float_t fMeanCurrent=0;
- Int_t fNRun=0;//Run Number
- Short_t fType=0;//Event Type (Data 0 or MC 1)

Must define Reader class to fill these
e.g HipoReader
void RunTrainSignalID(){

    auto input = GetClassificationFile();

    auto *signalTree = dynamic_cast<TTree*>(input->Get("TreeS"));
    auto *background = dynamic_cast<TTree*>(input->Get("TreeB"));

    TrainSignalID train("TMVAClassificationTut");

    train.SetOutDir("/work/dump/tmva/");
    train.IgnoreBranches(""); // Any branches in tree not used must be flagged!
    train.AddSignalTree(signalTree);
    train.AddBackgroundTree(background);
    train.PrepareTrees();

    // Can Book methods either via standard TMVA::Factory interface...
    train.BookMethod(TMVA::Types::kBDT, "BDT","!H:!V:Ntrees=850:MinNodeSize=2.5%:MaxDepth=3:
        BoostType=AdaBoost:AdaBoostBeta=0.5:UseBaggedBoost:BaggedSampleFraction=0.5:
        SeparationType=GiniIndex:nCuts=20");
    // Or predefined methods (See HSMVA::MethodConfigure.h)
    train.BookMethod(HS::MVA::Meths.MLP);
    train.DoTraining();
}
void RunTrainSignalID()
{
    auto input = GetClassificationFile();

    auto *signalTree = dynamic_cast<TTree*>(input->Get("TreeS"));
    auto *background = dynamic_cast<TTree*>(input->Get("TreeB"));

    TrainSignalID train("TMVAClassificationTut");

    train.SetOutDir("/work/dump/tmva/");
    train.IgnoreBranches("")./Any branches in tree not used must be flagged!
    train.AddSignalTree(signalTree);
    train.AddBackgroundTree(background);
    train.PrepareTrees();

    //Can Book methods either via standard TMVA::Factory interface...
    train.BookMethod(TMVA::Types::kBDT, "BDT","!H:!V:Ntrees=850:MinNodeSize=2.5%:MaxDepth=3:
    BoostType=AdaBoost:AdaBoostBeta=0.5:UseBaggedBoost:BaggedSampleFraction=0.5:
    SeparationType=GiniIndex:nCuts=20");
    //..or predefined methods (See HSMVA::MethodConfigure.h)
    train.BookMethod(HS::MVA::Meths.MLP);
    train.DoTraining();
}

ResultByTree classif2("TMVAClassificationTut","MLP",signalTree);
Use it → classif2.AddToTree();
HSMVA - TMVA - TMVAClassification plots

TMVA overtraining check for classifier: MLP

Kolmogorov-Smirnov test: signal (background) probability = 0.475 (0.011)

Background rejection versus Signal efficiency

MVA Method:
- MLP
- BDT
Interactive ROOT and build

Currently Build and run performed via ROOT scripts and ACLiC
No need for make, cmake, scons,…
        could be added…

setenv $HSCODE ??/HASPECT/
setenv $HSEXP $HSCODE/hsexperiments/clas12  (or your code)

Load individual components
root --hsdata --hsfinal --hsmva --hsskeleton --hsfit
        --hsin=infile  -hsout-outfile

Clear compiled code
root -cleanall -cleanfit -cleandata -cleanfinal -cleanfit

Run final state algorithm,
root LoadMyFinalState.C+ RunFSRoot.C
The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.

**PyROOT**
- Run root in jupyter using pyROOT bindings

**ROOTBooks**
- Run ROOT kernel in jupyter. Includes cling C++ compilation

Is that any use to us?

Greatly enhance code tutorials
Combine analysis code and analysis note
JupyterHub allows remote access/analysis via browser from anywhere, OpenData
Becoming standard in Universities, i.e. simple for students to start working
Implications for code design...
# ROOTbook Directory – g8 Phi SDME analysis

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Last Modified</th>
<th>File size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1._CreateMyFinalState.ipynb</td>
<td>Running</td>
<td>22 minutes ago</td>
<td>8.64 kB</td>
</tr>
<tr>
<td>2._Develop_Phi_to_KK_Analysis_Code.ipynb</td>
<td></td>
<td>7 days ago</td>
<td>13.5 kB</td>
</tr>
<tr>
<td>3._Run_FinalState_with_my_DeltaTimePCut.ipynb</td>
<td></td>
<td>7 days ago</td>
<td>50.4 kB</td>
</tr>
<tr>
<td>3b_Make_DeltaTimePCut.ipynb</td>
<td></td>
<td>7 days ago</td>
<td>6.8 kB</td>
</tr>
<tr>
<td>4._Train_MLP.ipynb</td>
<td></td>
<td>7 days ago</td>
<td>10.8 kB</td>
</tr>
<tr>
<td>5._Apply_MLP_Classification_to_Data.ipynb</td>
<td></td>
<td>7 days ago</td>
<td>422 kB</td>
</tr>
<tr>
<td>6._sPlot_Fit_to_KKMass.ipynb</td>
<td></td>
<td>15 minutes ago</td>
<td>65.5 kB</td>
</tr>
<tr>
<td>7a._Fit_Phi_SDME_Minuit.ipynb</td>
<td></td>
<td>15 minutes ago</td>
<td>183 kB</td>
</tr>
<tr>
<td>7b._Fit_Phi_SDME_MCMC.ipynb</td>
<td></td>
<td>15 minutes ago</td>
<td>136 kB</td>
</tr>
</tbody>
</table>

MesonEx AnalysisTools
Create final state analysis for CLAS g8 ϕ production

First 2 lines set the environment you should have HSCODE set as a shell environment variable when you start up root -- notebook Standard Load skeleton code maker and create an object for building CLAS analysis.

In []: gROOT->ProcessLine(".x HSCODE/hsskeleton/LoadSkeleton.C");

Create the skeleton code maker. This will be used to configure and produce the final state class.

In []: CLASSkeleton sk;

Call my finalstate analysis class K2

In []: sk.SetFinalState("K2");

Set names of actual final state particles of the reaction that you will use.

First detected particles
I consider final states with the possible detection of tagger beam photon, proton, \( K^+ \) and \( K^- \).

In []: sk.SetFinalStateParts("Beam:Beam, Proton:proton, Kp:K+, Km:K-");
1. Create final state analysis for CLAS g8 φ production

First 2 lines set the environment you should have HSCODE set as a shell environment variable when you start up root -notebook Standard Load skeleton code maker and create an object for building CLAS analysis.

In [1]: gROOT->ProcessLine(".x $HSCODE/hsskeleton/LoadSkeleton.C+");

CLASSkeleton sk;

sk.SetFinalState("K2");

sk.SetFinalStateParts("Beam:Beam, Proton:proton, Kp:K+, Km:K-");

sk.SetFinalStateTopo("Beam:Proton:Kp:Km, Beam:Proton:Kp, Beam:Proton:Km, Beam:Kp:Km");

sk.SetFinalStateParents(" ");

sk.MakeCode();

.gls

gROOT->ProcessLine(".x LoadK2.C+");

K2 finalstate;
Develop the final state class

There are 3 parts of the class the user is primarily responsible for:
1. Calculations specific to each topology e.g. start time, missing mass...
2. Calculations common to all topologies, typically the kinematics of the reaction
3. The data to be saved in an output file

Topology Specific

In the Topology specific functions:
1) Subtract the start time from all the particles
2) Apply energy loss correction previously calculated via eloss
3) Reconstruct the missing particle and save mass and missing mass squared for each topology
4) For topologies where 1 particle is missing, give it the missing 4-vector and fix its mass to PDG
5) Save particle time, momentum and angle

This is defined for the 4 topologies declared previously.

+ similar code for 3 other topologies

Run this cell to save file
ROOTBook – Develop FinalState Analysis Class .2

Common Calculations: Kinematics

For this reaction the kinematical quantities are calculated via the HSKinematics calculator. Here I calculate:

- MesonMass = invariant mass of K⁺ and K⁻
- HyperonMass = invariant mass of proton and K⁻
- Eγ = Photon beam energy
- PolState = Para or Perp photon Linear Polarisation
- Pol = degree of linear polarisation
- t = squared 4- momentum transfer
- MesoCosTheta = CM production angle of $\phi$ (dependant)
- MesoPhi = Azimuthal production angle (polarisation dependent)
- HelCosTh = Phi decay angle in helicity frame
- HelCosPhi = Azimuthal decay angle

+ Define TreeData class => TD.
Define cuts
Configure training data
Default variables:
"P","Th","Phi","Time","Edep","DeltaE"
Set signal and background data
Process events

Do the training

We supply all experimental data which will act as background in the training as it has a very small fraction of actual phi events. We supply simulated phi phase space simulation events to use as signal in training.

In []: //create DataManager
auto dmstd::make_shared<DataManager>();
//And make a chain of data files
TChain chain("N5Particles");
//Add actual data to train as background
chain.Add("/w/work/jlab/hallb/NS_g1/1_5_list/plot_040888.dat.root");
chain.Add("/w/work/jlab/hallb/NS_g1/1_5_list/plot_040899.dat.root");
//Add simulated data to train as signal
chain.Add("/w/work/jlab/hallb/NS_g5/8831_2k_h/RecEdgenn_root.cat.xml");
dCHAIN(chain);
//connect FinalState to Data by moving the pointer
fs.SetDataManager(dm);

Analyse all the events. Not you may prefer just looping over the data manager directly via while(dn->GetEvent() != Process())

In []: cout<<"Number of Events to process " <<chain.GetEntries();
fs.ProcessData(); //No number given, analyse all events in chain
cout<<"Done "<<dn->GetEntry()<<" " events "<<end;
Similar to training code +

Add the MLP response to the data stream. This just needs the directory created in the training routine. Also apply a rough cut at MLP=0.1 to get rid of most of the background.

```cpp
In [1]: MVASignalIDManager sid('/scratch/dplazier/g8/particles/3MWANLP'); //directory from training
    sid.SetParticleVars("Beam","P","Time");
    sid.ConSumeResults("MLP",&fs);
    sid.SetCut(0.1); //Set classifier cut value, this will be applied to all events
    fs.RegisterPostTopoAction(sid);
```

Now we configure the input data. Here we use a ROOT file presaved in HSDATA format. It is possible to analyse other data formats via the DataManager class. Now analyse all the data in the 1.9GeV peak setting ~ 500k events.

```cpp
In [2]: #create datamanager
    auto dm::std::make_shared<DataManager>();
    //Add a chain of data files
    TChain chain("HSParticles");
    chain.Add("/work1/job/hallb/HS06/10_list/flist_*.root");
    chain.Add("/work1/job/hallb/HS06/g8sim_2k_h/rec_edge_*.root");
    dm=IniChain(dm,chain);
    //Connect FinalState to data by moving the pointer
    fs.SetDataManager(dm);
```

Analyse all the events. This took around 8 hours with the call to MLP. BDT 5-10% slower.

```cpp
In [3]: cout<<"Number of Events to process "<<chain.GetEntries()<<endl;
    fs.ProcessData(); //No number give, analyse all events in chain
    cout<<"Done "<<dm->GetEntry()<<" events "<<endl;
```
Apply MLP Classifier

Apply single MVA cut on data

Response of MLP for all events

Missing Mass MLP > 0.1

Missing Mass MLP > X

For pK+ topology

For pK+ topology
**ROOTBook – Perform sPlot background subtraction 1.**

### Load Variables

Load Signal PDF

```
// Load variables
RF->LoadVariable("MesonMass[8.99,8.05]"); // should be same name as variable in tree
RF->LoadAuxVars("Eq[1.7,1.9]"); // Not to be fitted but limits applied to dataset
RF->LoadAuxVars("Topo[0.5,2.5]"); // Not to be fitted but limits applied to dataset
RF->LoadAuxVars("MesonCosTheta[0.5,0.5]"); // Not to be fitted but limits applied to dataset

// Define the signal MesonMass PDF
use custom RooHistoPDF class to base PDF shape on simulated data. Additional parameters will be constrained in fit:

alpha = Gaussian convolution width: >0, MLV = 0, sigma = 0.1/5 GeV
off = MesonMass peak offset: MLV = 0, sigma 0.1/5 GeV
scale = MesonMass scale factor : MLV+ 1, sigma 2.1/5, limits 8.9 and 1.1

// Give the simulated data tree to the PDF

// Register signal PDF to the Extended Maximum Likelihood fit
```
ROOTBook – Perform sPlot background subtraction 2.

A ROOT file with HS::Weights is now saved

In out2/
ROOTBook – Perform LinPol SDME fit to phi 1.

MesonEx AnalysisTools

Fit for $\phi$ Spin Denisty Matrix elements with linear polarisation

This code performs an extended maximum likelihood fit for 9 SDMEs in $\phi$ photoproduction. For this example just one large kinematic bin is selected.

In [ ]:
gROOT->ProcessLine(".x $HPCODE/hsfit/LoadH5Fit.C++");
Load predefined SDME PDF based on Schilling and Wolf. Soon it will be possible directly create this PDF within a notebook.

In [ ]:
gROOT->ProcessLine(".x VectorSDME.cxx++");
Create fit class based on RooFit. Final results will be saved to outSDME

In [ ]:
H5RooFit = RF->newH5RooFit("phiSDME");
RF->SetOutDir("outSDME/");
Load fit observables

MesonPhi : production plane angle relative to linear polarisation
HelCosth : decay angle of phi in helicid frame
HelPhi : azimuthal decay angle of phi in helicity frame
Pol : degree of linear polarisation of photon
PolState: parallel or perpendicular polarisation

Cut on observables, these should be the same as sPlot although they will not contribute to likelihood if there are outwith the sPlot limits.
ROOTBook - Perform LinPol SDME fit to phi 2.

In [ ]:
```python
auto dfile=new TFile("/work/g8dump/hsexamples/convertestmvamlpinc2b.root");
auto dtree=(TTree*)dfile->Get("newtree");
RF->SetIDBranchName("UID");
RF->LoadDataSet(dtree);
```

Load the weights from the sPlot. Choose species Phi (see sPlot notebook)

In [ ]:
```python
RF->LoadWeights("out2/WeightsSF.root ","HssWeights");
RF->SetWeightName("Phi"); //Same as Signal species in FitHSSimpleBins
RF->SetDataWeight();
RF->GetDataSet()>>Print();
```

To correctly calculate the normalisation integral the simulation needs values for polarisation and polarisation state that matches the experimental data. This is handled in AddProData.

In [ ]:
```python
pdf->AddProtoData(RF->GetDataSet());
```

Do a fit

In [ ]:
```python
RF->TotalPDF();
RF->RunAFit(1);
```
Further Developments

HS Code is being restructured
data, finalstate, experiment, mva done(ish)

HSSelector would provide multicore processing for ROOT files
?How to handle hipo data
?ROOT multithreading?

Histogramming tools to be added
Investigating DataFrames or HSSelector (A.Thornton)

Further develop Jupyter ROOTbooks
Use of PyROOT and python kernels (seems to work OK)
Further support for HSFIT
Import skeleton PDF maker, data bin splitting, TOYMC

Merit of MVA approach under investigation
ROOTBook – and RDataFrame histogramming

In [2]:
   auto fileName = "/output.root";
   auto treeName = "FinalTree";
   ROOT::RDataFrame df(treeName, fileName);

In [3]:
   TCanvas *c1 = new TCanvas("c1","multipads",900,700);

In [4]:
   h1 = d.Filter("Topo==0") .Hist0D(["MissingMass2", "e p pi+ pi-; GeV; Events", 200, -1, 4], ["MissingMass2");
   h2 = d.Filter("Topo==1") .Hist0D(["MissingMass2", "e p pi+; GeV; Events", 200, -1, 4], ["MissingMass2");
   h3 = d.Filter("Topo==2") .Hist0D(["MissingMass2", "e p pi-; GeV; Events", 200, -1, 4], ["MissingMass2");
   h4 = d.Filter("Topo==3") .Hist0D(["MissingMass2", "e pi+ pi-; GeV; Events", 200, -1, 15], ["MissingMass2");

In [5]:
   c1->Clear();
   c1->Divide(2,2);
   c1->cd(1); h1->Draw();
   c1->cd(2); h2->Draw();
   c1->cd(3); h3->Draw();
   c1->cd(4); h4->Draw();
   c1->Draw();
Preparing for full chain data analysis of CLAS12 MesonEx

Previously shown hsdata, hsfinalstate, hsfit

Added hsmva and Jupyter ROOTbook applications

HSMVA provides a simplified interface to ROOT TMVA (which can use PyKeras, R)
Many possible applications to our analysis...

Coupled with HSFINASTATE analysis can proceed via machine learning classifiers
Single cut applied to all combitorials
Investigating data driven approaches to training

All code is being adapted to run intuitively on Jupyter Notebooks
Full chain analysis can be peformed and presented as series of notebooks
Notebooks could be directly used to supplement analysis note
Could lead to much faster analysis development and review.
Combitorials

THSCombitorial, (Templated class works on vector of any types)

Select p from n

THSParticleIter, (recursive class allowing any depth of combitorial selections)

Select particles s from n and r from (n-s)...

e.g. select 2\gamma from 6 for \pi^0, 2 from 4 for \pi^0, remaining 2 for \eta

THSTopology, Configure selections based on different track charge or PID or mixed.

For given number of particles or inclusive.

Accounting for particles from short lived (narrow states) parents

e.g. one of the \pi^0 came from an \omega in an \omega\pi^0 final state not \pi^+\pi^-\pi^0\pi^0

If so must permutate \pi^0s

If no parent do not permutate

Always permutate particles of same charge but different species, if selecting based on charge

e.g. K^+ and \pi^+

THSFinalState::AutoIter() hack it all together!
Event generation + analysis, no I/O

KNL, 64 physical cores
Ad-hoc implementation (patched ROOT 5 + POSIX threads) vs RDF

Performance analysis by X. Valls Pla

h1analysis + duplicated input data

Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz
PROOF-lite vs RDF interpreted vs RDF compiled

Performance analysis by D. Piparo

E. Guiraud, “RDataFrame”, ROOT users' workshop 2018
From Common Tools Committee

...particular scheme to implement tools for signal selection and physics analysis, **strict guidelines and checks to ensure validity** and encourage the use of common approaches should be enforced. As such we suggest any algorithm or technique should be **pre-reviewed by an Analysis Review Group** prior to any final Analysis Note to create a speedier final review Process. Any individual analysis could then just be required to check the technique for their reaction through a simulated validation. Anyone implementing a new technique should be encouraged to make common software available to the rest of the collaboration, thereby building up our stock of Innovative and reliable Common Tools...