ROOT

Chris Marker,
Royal Holloway, University of London

The World of ROOT

- **ROOT is an OO c++ analysis package for producing graphics and histograms.**

- **BaBaR has taken ROOT and written RooFit packages to enable the modeling of the distribution of observables such as Δt,M_{es} in terms of physics parameters of interest such as $sin2\beta$, m_b.**

- **RooFit relies heavily upon Variables, PDFs and datasets.**

- **There are two main RooFit packages, RooFitTools and RooFitCore.**

- **RooFitCore is the newest package and everyone is moving to it. It is quicker at doing analysis than RooFitTools but has less documentation.**

Documentation

RooFitTools BAD 18.

RooFitCore webpage at babar BFROOT/www/Organization

/CollabMtgs/2001/detDec2001/Wed1e/Wed1e.html.

Beware although some of the things described in BAD 18 still work in RooFitCore many of the constructors for the various classes have changed. Also many of the most useful macros (a macro being a file with all the root commands to produce a certain result in) ar no longer implemented.

Getting RoFitCore Up and Running .

1. Im assuming you work on the farms bfa bfb here. But the procedure is the same on a slac machine except for step 13. You will need to klog -cell rl.ac.uk.

2. In your .cshrc file will need

setenv ROOTSYS $BFROOT/package/root/$BFARCH
setenv PATH $ROOTSYS/bin:$PATH

3. Build a release, I use analysis-12a

newrel -s /u01/scratch/username -t analysis1-12a root

4. cd root

5. setenv ROOTVER 3.00-06

6. srtpath (very important)

7. addpkg RooFitCore V00-00-11
addpkg RooFitModels V00-00-09
addpkg RooUtils dk-22aug01
addpkg workdir V00-04-08

8. gmake clean

9. gmake RooFitCore.rootlib ROPT=-Shared

10. gmake RooUtils.binscripts

11. gmake RooFitModels.rootlib ROPT=-Shared

12 In your workdir/RooLogon.C Replace the line:

loadSrtLib("RooFitTools");

with:

loadSrtLib("RooFitCore");
loadSrtLib("RooFitModels");

13. Change the LD_LIBRAY_PATH line in root/bin/Linux2/bbrroot to:

LD_LIBRARY_PATH=/bfactory/package/root//3.00-06/Linux2-gcc-2.95.2/lib:/bfactory/package/gcc-2.95.2/lib/

14. In your wordir do bbrroot -l

Example Macro

This macro is designed to read in an charmonium groups ASCII file and to fit $sin2\beta$. Note that this is a cut down version. The full macro is in marker/root/workdir/trythis.cc . A very useful feature of root is tab completion which not only saves time but allows you to look at the constructor of a class if you do CLASSNAME::CLASSSNAME(hit tab).

First of all declare some variables
RooRealVar deltat("deltat","Delta t",-10,10,"ps");Real variable with limits between -10,10 and units pico seconds.

RooRealVar onoffUps("onoffUps","on/off Ups peak",0,2); Real variable with limits but no units.

RooRealVar sigmadt("sigmadt","Sigma of delta t",-10,10,"ps");
RooRealVar zrectrue("zrectrue","mc true z position of reco B ",-100000,100000,"um"); ... etc

Define a category which defines a discrete variable. Note that the constructor for this is (const char* label, Int_t index) so if what you are trying to read in is not an integer but real and you do not put the label in as the real number you reading in you will get an error.

RooCategory cat1("cat1","cat1");
cat1.defineType("1.000",1);
cat1.defineType("0.000",0);

RooCategory BtagCategory("BtagCategory","BtagCategory");Another category in this case there is no problem with integers and reals as the thing being read is an integer.

```
//BtagCategory.defineType("none",0);No tag so dont want to read in.
BtagCategory.defineType("Elekaon",11);
BtagCategory.defineType("Mukaon",12);
BtagCategory.defineType("kaon",13);
BtagCategory.defineType("Electron",14);
BtagCategory.defineType("Muon",15);
BtagCategory.defineType("NT1",22);
BtagCategory.defineType("NT2",23);
```

Build a list of the variables. Note the variables must be added in the order they appear in the ASCII file.

```
RooArgList datalist;
datalist.add(deltat);
```

datalist.add(sigmadt);
... etc
datalist.Print(); Prints what is in datalist

Can now read in the ASCII file

RooDataSet *data= RooDataSet::read("../../mc_B0_JpsiKs-anal8.fit.orig", datalist);

Now define a gaussian resolution model.
RooRealVar mean("mean","Mean",0,"ps");
RooRealVar sigma("sigma","sigma",0.5,"ps");
RooGaussModel resolution("resolution","Resolution",deltat,mean,sigma);
There are other constructors for the Gauss model. There are also many other models that can be defined.

RooRealVar lambda2("lambda2","lambda2",0.000625);Real variable set to one number, this is actually a RooAbsReal which you will see is needed in some classes.

truth model for resolution

RooTruthModel truth1("truth1","Truth Resolution Model",deltat);
RooTruthModel truth2("truth2","Truth Resolution Model",deltat);

Define some CP PDFs.

Constructor for RooBCPEffDecay (const char* name, const char* title, RooRealVar& t, RooAbsCategory& tag, RooAbsReal& tau, RooAbsReal& dm, RooAbsReal& avgMistag, RooAbsReal& CPeigenval, RooAbsReal& a, RooRealVar& b, RooAbsReal& effRatio, RooRealVar& delMistag, const RooResolutionModel& model, RooBCPEffDecay::DecayType type = DoubleSided)

RooBCPEffDecay BDLepton("BDlepton","BDLepton",deltat,cat1,tau,dm, omegal,eta,lambda1,bplusg1,effr,deta,resolution); Note here the omega is the mistag fraction and so is different for each category, bplusg1 is actually sin2beta.

RooBCPEffDecay BDKaon("BDKaon","BDKaon",deltat,cat1,tau,dm,omegak, eta,lambda1,bplusg1,effr,deta,resolution);
RooBCPEffDecay BDnt1("BDnt1","BDnt1",deltat,cat1,tau,dm,omegant1 ,eta,lambda1,bplusg1,effr,deta,resolution);
RooBCPEffDecay BDnt2("BDnt2","BDnt2",deltat,cat1,tau,dm,omegant2, eta,lambda1,bplusg1,effr,deta,resolution);

Define a list of PDFs, There must be the same number of PDFs as tagging categories. We have defined 7 BtagCategory however we will want to combine some of these together into a lepton category so we will end up with 4 categories. Leptons, kaons, NT1, NT2.

```
RooArgList listpdf;
listpdf.add(BDLepton);
listpdf.add(BDLepton);
listpdf.add(BDKaon);
listpdf.add(BDLepton);
listpdf.add(BDLepton);
listpdf.add(BDnt1);
listpdf.add(BDnt2);
```

Simultaneously model and fit to all the PDFs.

```
RooSimultaneous model("model","un+mi",listpdf,BtagCategory);
model.fitTo(*data);
```
fitting to the data read in previously.

Plot data and fit on same plot

RooPlot* deltatframe = deltat.frame(-10,10,50); creating plot parameters

data→plotOn(deltatframe);
BDLepton.plotOn(deltatframe);

Can also plot one type of tagging category
data-→plotOn(deltatframe,"cat1==cat1::B0");
RooDataSet* projdata = data→reduce(BtagCategory);
model.plotOn(deltatframe,"L",1.0,RooAbsReal::Relative,projdata);
deltatframe→Draw();