Data Analysis Tools

Matt Graham
HPS Software Workshop
June 3, 2013

Outline

- Tools for java: aida & JAS
  - good for low- & mid-level analysis...algorithm development, debugging
  - can do higher level analysis with these tools, but ROOT people will feel unsatisfied
- Tools in ROOT: TMVA & RooFit
  - MVA training/testing
  - high-level analysis & cuts
  - maximum likelihood fits

Analysis classes in hps-java

- Most of the analysis (such as it is) has been done directly in hps-java
  - There are a ton of examples living in the package
- Extremely useful to have a tool to plug directly into java code ... makes debugging much easier
  - AIDA + IPlotter works very well for this
  -

Example: using aida & IPlotter

org.lcsim.hps.users.mgraham.ExamplePlotter.java

```
/**...*/
public class ExamplePlotter extends Driver implements Resettable {

    private AIDAFrame plotterFrame;
    private AIDA aida = AIDA.defaultInstance();
    IPlotter plotter;
    IAnalysisFactory fac = aida.analysisFactory();
    private String trackCollectionName = "MatchedTracks";
    private double zAtConverter = -674.062;//mm
    private String outputPlots = null;
```

Make the AIDA/IPlotter objects

Initialization (in “detectorChanged”)

```java
protected void detectorChanged(Detector detector) {
    aida.tree().cd("/");
    plotterFrame = new AIDAFrame();
    plotterFrame.setTitle("HPS Tracking Plots");

    plotter = fac.createPlotterFactory().create("HPS Tracking Plots");
    plotter.setTitle("Momentum");
    IPlotterStyle style = plotter.style();
    style.dataStyle().fillStyle().setColor("yellow");
    style.dataStyle().errorBarStyle().setVisible(false);
    plotter.createRegions(2, 3);
    plotterFrame.addPlotter(plotter);

    IHistogram1D trkPx = aida.histogram1D("Track Momentum (Px)", 25, -0.25, 0.25);
    IHistogram1D trkPy = aida.histogram1D("Track Momentum (Py)", 25, -0.1, 0.1);
    IHistogram1D trkPz = aida.histogram1D("Track Momentum (Pz)", 25, 0, 3.5);
    IHistogram1D trkChi2 = aida.histogram1D("Track Chi2", 25, 0, 25.0);
    IHistogram1D xAtConvert = aida.histogram1D("X (mm) @ Converter", 50, -50, 50);
    IHistogram1D yAtConvert = aida.histogram1D("Y (mm) @ Converter", 50, -20, 20);
    plotter.region(0).plot(trkPx);
    plotter.region(1).plot(trkPy);
    plotter.region(2).plot(trkPz);
    plotter.region(3).plot(trkChi2);
    plotter.region(4).plot(xAtConvert);
    plotter.region(5).plot(yAtConvert);

    plotterFrame.pack();
    plotterFrame.setVisible(true);
}
```

Make your plots pretty

Initialize plots

Assign plots to places in the plotter

Fill some histograms & end of data

```
public void process(EventHeader event) {
    aida.tree().cd("/");
    List<Track> tracks = event.get(Track.class, trackCollectionName);
    for (Track trk : tracks) {
        aida.histogram1D("Track Momentum (Px)").fill(trk.getPY());
        aida.histogram1D("Track Momentum (Py)").fill(trk.getPZ());
        aida.histogram1D("Track Momentum (Pz)").fill(trk.getPX());
        aida.histogram1D("Track Chi2").fill(trk.getChi2());

        SeedTrack stEle = (SeedTrack) trk;
        SeedCandidate seedEle = stEle.getSeedCandidate();
        HelicalTrackFit ht = seedEle.getHelix();
        HelixConverter converter = new HelixConverter(0);
        StraightLineTrack slt = converter.Convert(ht);
        HPSTrack hpstrack = new HPSTrack(ht);
        Hep3Vector[] trkatconver = hpstrack.getPositionAtZMap(100, zAtConverter, 1);
        aida.histogram1D("X (mm) @ Converter").fill(trkatconver[0].x()); // y tracker frame?
        aida.histogram1D("Y (mm) @ Converter").fill(trkatconver[0].y()); // z tracker frame?

    }
}
```

Get collection from event

Fill histos with momentum ...use same name as defined in declarations (could also use object names)

Do some stuff to the tracks... ...then fill histograms with these quantities...

```
public void setOutputPlots(String output) {
    this.outputPlots = output;
}
```

...settable in .lcsim file...

```
public void endOfData() {
    System.out.println("Output");
    if (outputPlots != null) {
        try {
            aida.saveAs(outputPlots);
        } catch (IOException ex) {
            Logger.getLogger(ElwinsTrackingRecon.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

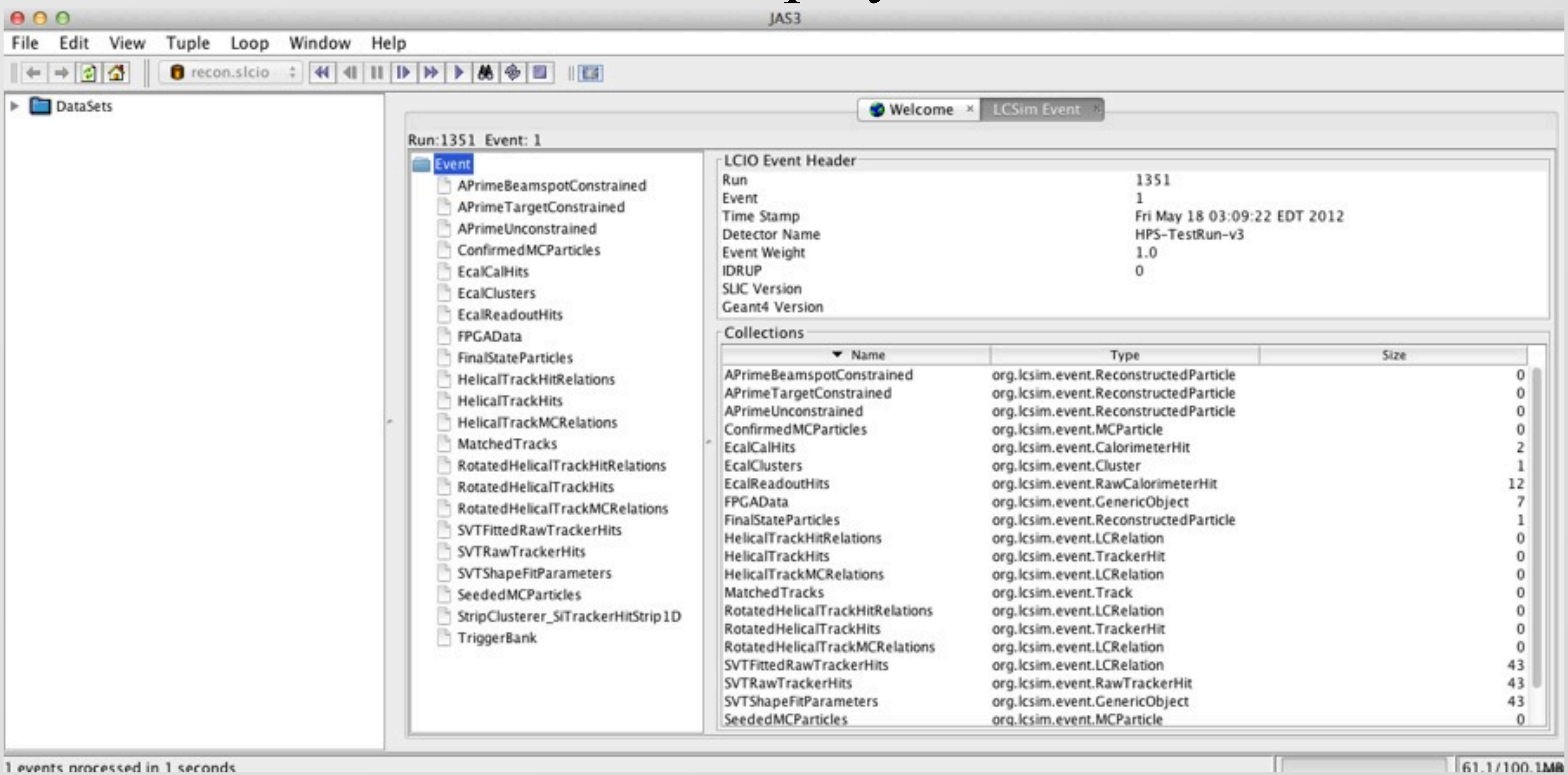
Do some stuff (like save .aida file) at endOfData

aida data containers

- 1d, 2d, profile histograms
- 1d, 2d clouds:
  - warning...in JAS (at least) there is a limit to the number of data points before cloud→histogram (with binning you probably don't want)
- "ituples"; aida's version of an ntuple

aida & JAS & lcio

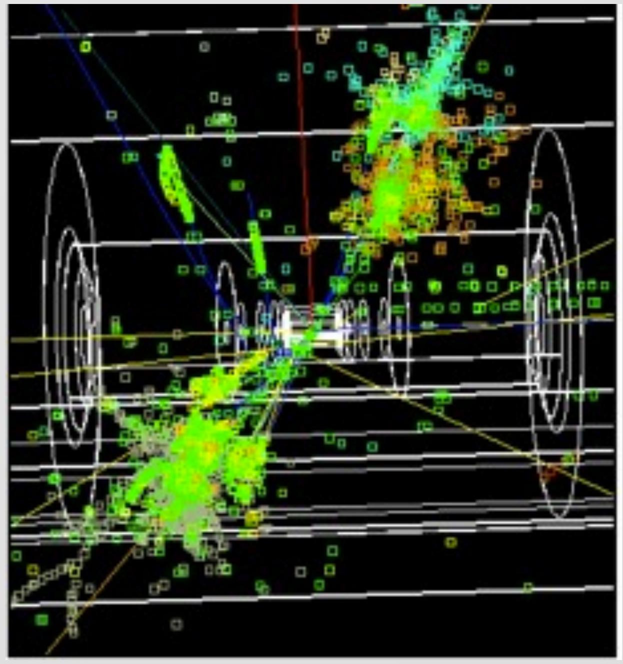
- JAS is a nice tool for looking at lcio files, running hps-java classes, and (potentially) looking at events via the WIRED event display



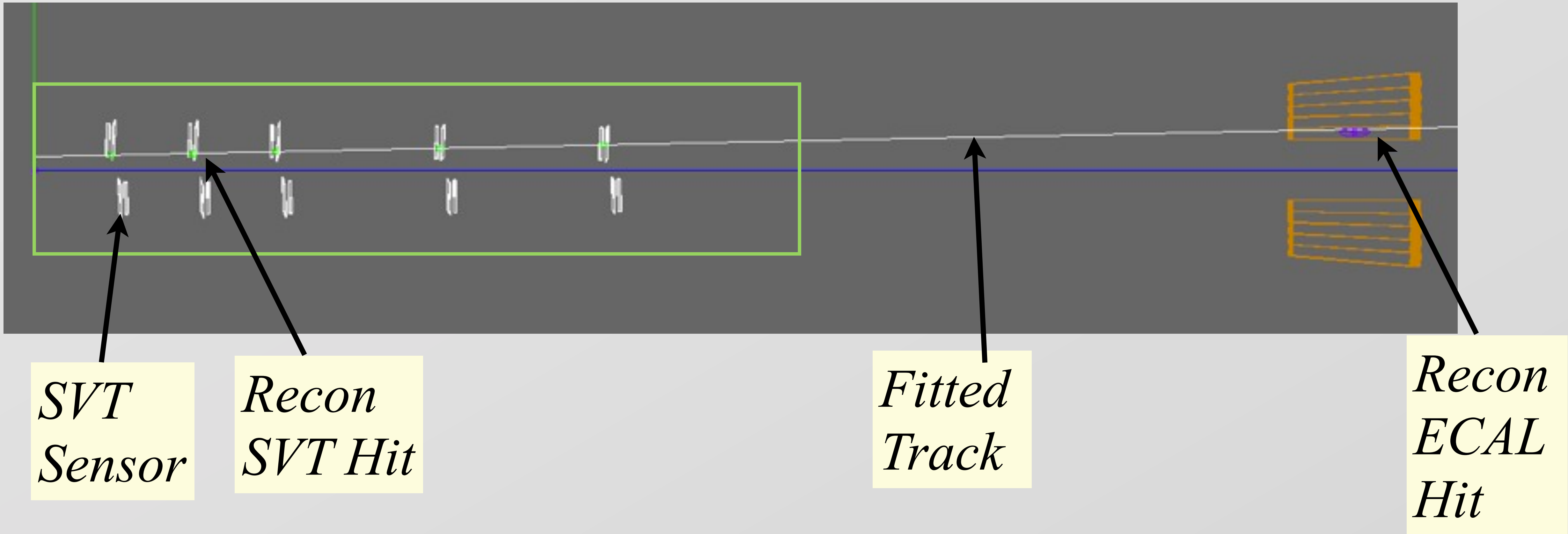
JAS & Wired Event Display

http://wired.freehep.org/

WIRED 4 supports viewing of events using either conventional 3D projections as well as specialized projections such as a fish-eye or a rho-Z projection. Projections allow the user to scale, rotate, position or change parameters on the plot as he wishes. All interactions are handled as separate edits which can be undone and/or redone, so the user can try things out and easily return to a previous state.



Not our data (thanks Zeus)



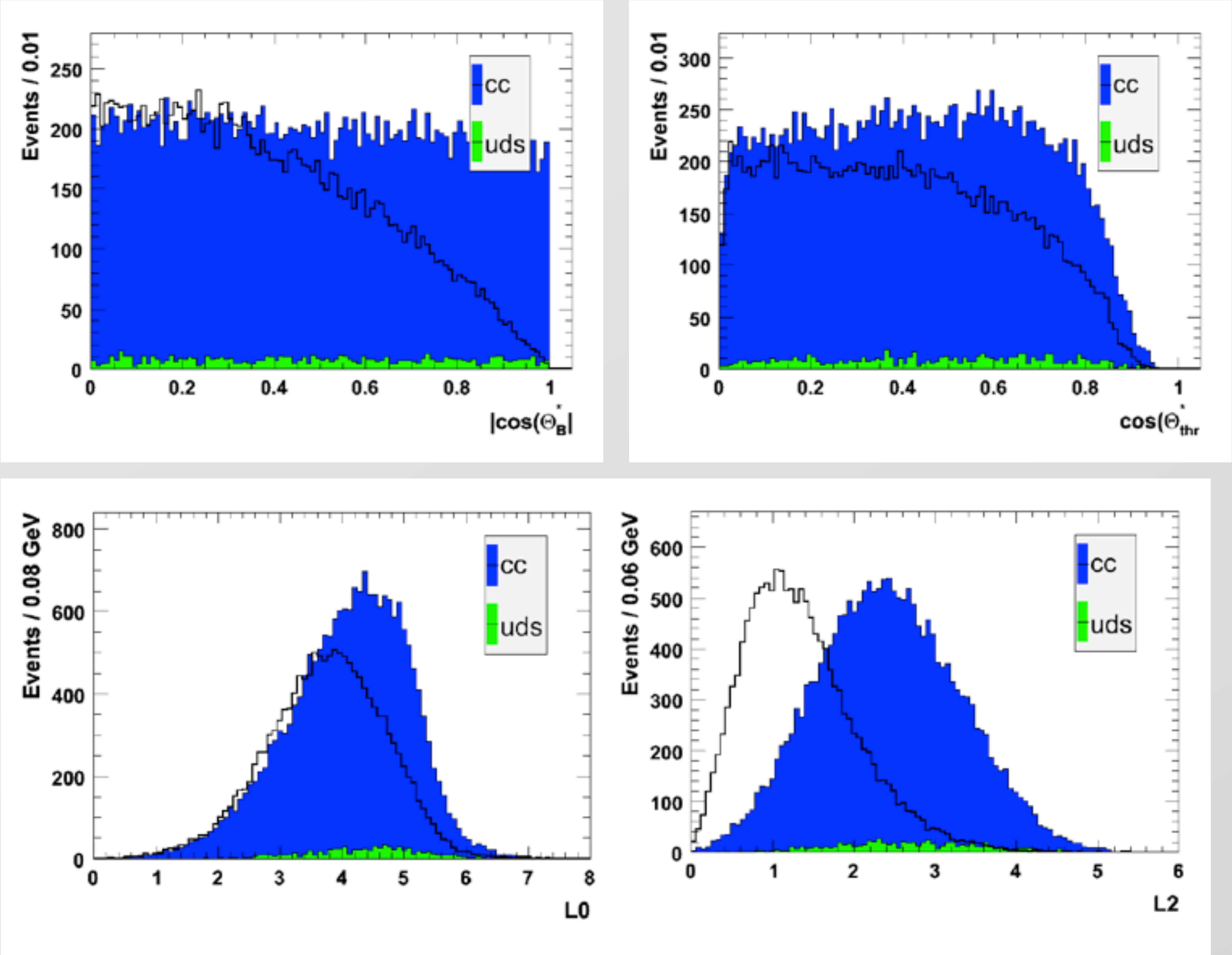
lcio or DSTs & ROOT

- Omar has a very nice DST builder...two things come out from this:
  - Really nice example of how to access lcio data in c++ ... this is how you can access the data directly and interface with ROOT
  - Products of running the default DST code are ROOT TTrees, so you can just look at these directly
- Everyone knows ROOT...I'll just introduce two neat packages: TMVA & RooFit

Signal, Background & Cuts

- One of the first steps of analysis is determining what variables to cut on and where to place the cut
- Some cuts are easy...if it removes 90% of bkg and keeps 90% of signal, do it. If it's 50/50%, probably don't.
  - mass-dependent & vertex-dependent variables are bad to cut on...don't want to distort these distributions (they will be used eventually)
- But, lot of variables have some discriminating power...or there may be some correlations between variables that can be useful for discrimination
  - plain rectangular cuts may not be optimal...optimizing can be painful.

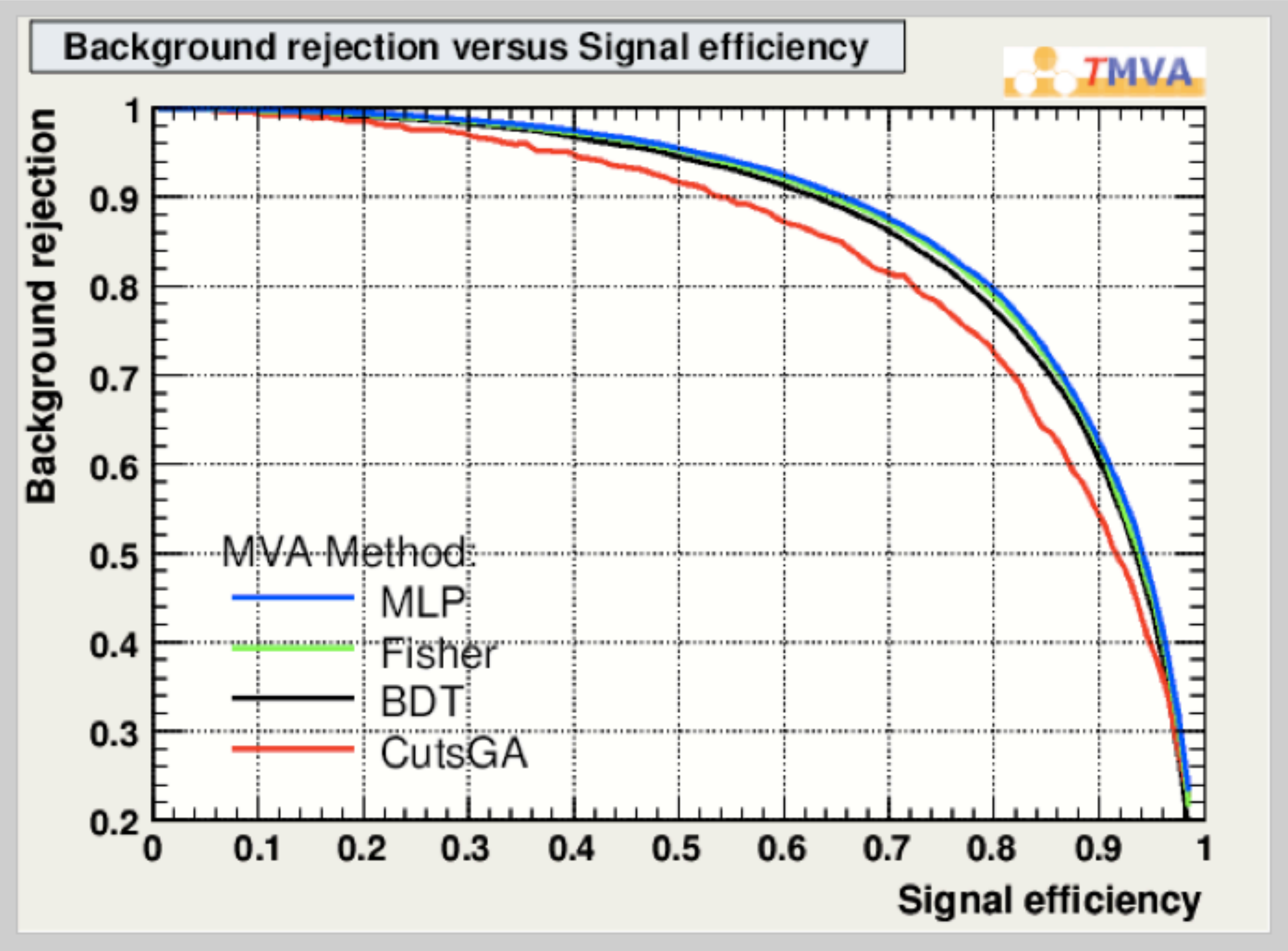
Example from BaBar: Event Shape



TMVA: T MultiVariateAnalyzer

- http://tmva.sourceforge.net/
- Comes standard with ROOT (>5.11)
- Very easy to use; website has good tutorial & user guide...fairy good description of methods
    - Uses standard ROOT objects (e.g. TTrees)
- includes: rectangular cut optimization, NN, BDT, linear dicriminants, SVM (support vector machine...which I just heard of yesterday from Pelle), and other MVAs
- Three steps to making any MVA: Training, testing, evaluation
    - Training: you give a dataset(s) with specified signal/bkg/etc; the MVA optimizes based on these classifications
    - Testing: on a different data set, apply the trained MVA; the discrimination should be roughly the same as in the training sample (otherwise you may have "overtrained")
    - Evaluation: apply the MVA to regular data (where you don't know what signal and background)

Rejection vs Efficiency



Maximum Likelihoods

- Performing fits to data in ROOT is awkward
  - beyond fitting 1d binned histograms with fairly simple functions, it's just really complicated
- Use RooFit instead!
  - comes with ROOT (but you need to add a flag at build time)
- http://roofit.sourceforge.net/
  - developed by BaBarians!
  - Fit binned or unbinned data sets
  - 1→lots number of dependent variables
  - 1→lots number of parameters
  - many common PDFs come included (gaussian, exponential, histogram PDFs, lots more)
    - RooAddPdf, RooProdPdf, RooFFTConvPdf, RooSimultaneous
  - "easy" to build your own very complicated pdfs
  - ML (extended or not) or chi^2 fits supported (uses Minuit)
- $B^0 \to \pi^+ \pi^- \pi^0$: 7-dimensional (time-, tag-dependent dalitz plot, with mES, deltaE, event shape) PDF (with weird correlations) with 7 tagging categories, 25 event species, and 64 free parameters...

simpleRooFitFit

```
void simpleRooFitFit(){

  RooRealVar myVar("myVar","myVar", -2000,2000);

  RooRealVar mean("mean","mean", 666,0,1000);
  RooRealVar sigma("sigma","sigma", 999,500,1500);
  RooGaussian myGaussian("myGuass","This is my Gaussian", myVar, mean, sigma);

  RooDataSet* gaussianSet=myGaussian.generate(RooArgSet(myVar),666);
  gaussianSet->Print("V");

  myGaussian->fitTo(*gaussianSet);

}
```

Make a PDF,
generate toy events,
fit toy data set

...done

```
root [3] .x simpleRooFitFit.cc
DataStore myGuassData (Generated From This is my Gaussian)
  Contains 666 entries
  Observables:
    1)  myVar = 229.671  L(-2000 - 2000)  "myVar"
Warning: wrong member access operator '->' simpleRooFitFit.cc:12:
[#1] INFO:Minization -- RooMinuit::optimizeConst: activating const optimization
 **********
 **   13 **MIGRAD        1000           1
 **********
 FIRST CALL TO USER FUNCTION AT NEW START POINT, WITH IFLAG=4.
 START MIGRAD MINIMIZATION.  STRATEGY  1.  CONVERGENCE WHEN EDM .LT. 1.00e-03
 FCN=5382.43 FROM MIGRAD    STATUS=INITIATE        6 CALLS           7 TOTAL
                     EDM= unknown      STRATEGY= 1      NO ERROR MATRIX
  EXT PARAMETER               CURRENT GUESS       STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  mean         6.66000e+02   1.00000e+02   2.14287e-01  -6.72918e+00
   2  sigma        9.99000e+02   1.00000e+02   2.01358e-01   4.75289e+00
                                 ERR DEF= 0.5
 MIGRAD MINIMIZATION HAS CONVERGED.
 MIGRAD WILL VERIFY CONVERGENCE AND ERROR MATRIX.
 COVARIANCE MATRIX CALCULATED SUCCESSFULLY
 FCN=5382.21 FROM MIGRAD    STATUS=CONVERGED      37 CALLS          38 TOTAL
                     EDM=1.04221e-07    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  mean         6.95938e+02   5.59226e+01   5.11496e-03  -2.88818e-03
   2  sigma        9.97712e+02   4.65923e+01   3.92250e-03   6.30275e-04
                                 ERR DEF= 0.5
 EXTERNAL ERROR MATRIX.    NDIM=  25    NPAR=  2    ERR DEF=0.5
  3.143e+03  1.461e+03
  1.461e+03  2.177e+03
 PARAMETER  CORRELATION COEFFICIENTS
       NO.  GLOBAL      1      2
        1  0.55861   1.000  0.559
        2  0.55861   0.559  1.000
```

Doing a ML Analysis

- We'll use a ML fit to extract our signal (if I have my way)...here are the basic blocks.
- Define PDF...
  - define variables...mass, vertex position, (event prob?), etc
  - how many species...signal, BH bkg, rad bkg, ...???
  - what shapes for each species in each variable...where do you take the parameters from
- Validate PDF
  - generate & fit toy MC→normalization, parameter sensitivity, etc
  - fit "embedded" samples→mixture of real MC, toy MC, appropriate data subsample etc...tests if the PDFs used are appropriate, variable correlation treatment.
- Blind fits to data→anything catastrophic? Iterate from beginning if needed
- Final fit to data→write PRL; accept Nobel Prize.