DST/Tuple Maker

DSTs:

How it works:

- generates ROOT file using C++ API that contains rough summary of all final particles/collections from recon file

Advantages:

- Multiple collections stored in single file (access to tracks, clusters separate from vertex collections)
- Robust way to look at full event picture

Things to consider:

- No easy way to look at vertexed particles in different constrained collections (comparing unc and bsc vertex collections)
- Not easy to make plots from the ROOT command line
- Significant development of analysis codes using DST offline

How I've used it:

- 3 prong studies, WAB studies

Experience:

- Cumbersome to get libraries and everything working well

Tuples:

How it works:

- Add variables directly in hps-java, run over recon/skim files, outputs tridents, mollers, and fee tuples.

Advantages:

- Fast running on recon files, runs easily using hps-java framework
- Contains relevant collections
- Easy to plot stuff from the command line

Things to consider:

- Only contains relevant collections (not useful for WAB studies)
- If you need something, may have to go back to hps-java to add it

How I've used it:

- Vertex analysis, timing calibration

Experience:

- Easy if you know how to use hps-java

Key Points:

- We absolutely need both! They serve different purposes
- The DSTs are a robust way to study events. Current bump hunt codes use this.
- The Tuples are very accessible if you need to add variables, study event anomalies, and look at the effects of different vertex constraints
- Both are maintained in GitHub
- If you want to add/change something in the TupleMaker, then just make an issue in GitHub, and follow standard HPS coding (if you know how to use hps-java, then it's pretty easy)