Table of Contents

How to add UserData to the output D3PD: adding new branches and filling them with user-defined custom formulas

Introduction

We provide below two examples of how to include UsedData, i.e. user-defined quantities, in the output D3PD.

Adding UserData to the D3PD means defining two new branches and filling them with user-defined formulas.

⚠ It does not mean dumping information from AOD, for example. For that you should refere to ATLASWatchManHowToDumpInfoFromContainers

How to add UserData: examples

In our example we define two new branches, to store `Meff` (Effective Mass) and `Sphericity`.

The first one is filled with a user-defined custom formula, while the latter is filled with a buil-in formula. (You can find the formula in `ATLASWatchMan/python/ATLASWatchMan_CutsLib.py`).

```
userD3PDBranchesToFill = {
                        'meffCSC4j0lep' : {'label': 'meff4j', 'type': 'float', 'formula': 'meff
                        'sphericityCSC' : {'label': 'sphCSC', 'type': 'float', 'formula': 'sphe
}
```

In `userD3PDBranchesToFill` we set our definitions for the new branches. In `'formula'` we specify the name of the formula. That name will be searched first in the `userFormula` dictionary (details below) and then, if not found, in the built-in library of pre-defined furmulas (you can find the formulas in `ATLASWatchMan/python/ATLASWatchMan_CutsLib.py`).

So the branch `'sphericityCSC'` will be filled with the formula `'sphericity'` from the built-in library, while `'meffCSC4j0lep'` in this example is a user-defined formula and it has been defined like here below, in the `userFormula` dictionary:

```
userFormula = {
'meff':
"""
meff = 0.
for i,jet in enumerate(candidates['jet']):
    meff += candidates['jet'][i].pt()
meff += candidates['met'].et()
return meff
""",
```

The triple quotes `"""`   `"""` are the Python character to declare an indented multi-lines string.

So you can actually write your own formula in a very simple way, and you can use the `candidates` dict to access particles, or everything you put in `collections` dictionary setting them as `'select': True` (for details see).

For example to access the `jet.Pt()` of the 3rd jet, in your formula you can use `candidates['jet'][2].pt()` , and the same for `met.Phi()`: `candidates['met'].phi()`.

Every branch you specify here in `userD3PDBranchesToFill` will be written in the output D3PD as two branches, in our example:

- meffCSC4j0lepValues
- meffCSC4j0lepChannels

The first one storing the actual values from your computation, and the latter storing a vector of strings which say to you, for each value, wich sets of candidates have been used to compute that value.

This because each channel and/or analysis you define in your *steering file* can have a custom **object selection** and/or **overlap removal**, and so each channel with custom selection cuts "owns" a different set of candidates, depending of those having passed the custom cuts.

And so the quantity you define as UserData is computed with all these sets of candidates, the different values are stored in `Values` branch, and the actual channels name whose set of candidates was used to compute that value is stored in `Channels` branch.

So in the end you have only to plot `meffCSC4jolepValues` branch, asking `if 'myChannel' is in meffCSC4j0lepChannels[i]` as we explain here: ATLASWatchManReadD3PD

-- RiccardoMariaBianchi - 11 Mar 2009

This topic: Main > ATLASWatchManHowToAddUserDataBranches
Topic revision: r2 - 2009-03-23 - RiccardoMariaBianchi