HSG5 Analysis Framework for Run 2

Kristian Gregersen

Outline

- How the code is organised
- Calibration/Analysis tools in the framework
- Contents of the output file
- Technical details on I/O structure
- Configuration and running
- Performance numbers
- Mailing list and links

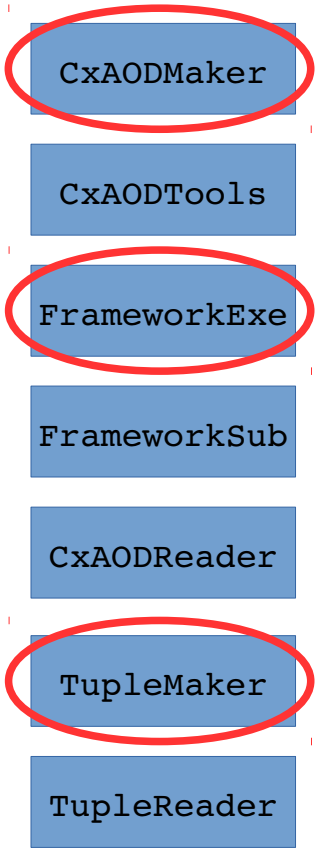
Framework layout

- Based on RootCore (a C++ compilation framework which depends on ROOT).
- Uses EventLoop (EL) and SampleHandler.
- Currently there are 7 "RootCore" packages :

CxAODMaker	Contains the main code for processing the input xAOD files, applying the calibration tools (CP tools), and writing the output CxAOD (C for Calibrated).
CxAODTools	Contains tools that are shared outside of the main CxAODMaker work-flow, such as object and event selection.
FrameworkExe	Contains the code for the executables for running CxAODMaker, along with any associated configuration files.
FrameworkSub	Contains files related to defining datasets and information for tracking processing.
CxAODReader	Contains code to read CxAOD using the xAOD EDM. Implements plotting style used in HSG5.
TupleMaker	Contains example code for producing a compact flat TTree based tuple.
TupleReader	Contains the common TTree definition and example code to read a compact flat TTree.

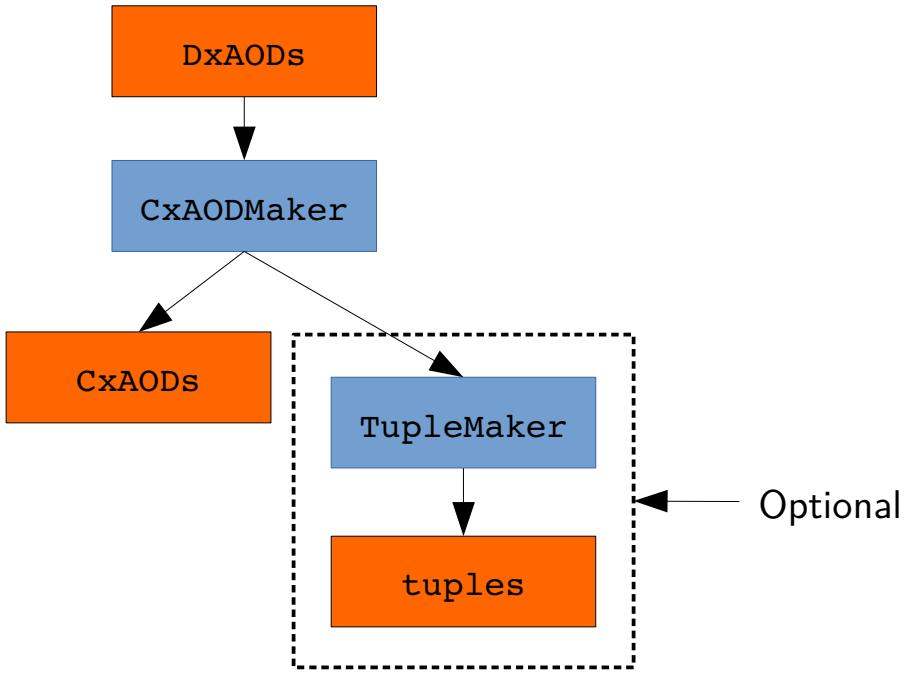
Framework layout

- Based on RootCore (a C++ compilation framework which depends on ROOT).
- Uses EventLoop (EL) and SampleHandler.
- Currently there are 7 "RootCore" packages :



Focus will be on these today

Work flow for processing (D)xAODs:



CxAODMaker

- **List of Calibration/Analysis Tools in Framework**

  - Jets :
    - JetCalibrationTool
    - JetCleaningTool
    - JERTool
    - JERSmearingTool
    - BTaggingEfficiencyTool
  - Taus :
    - TauSmearingTool
    - TauSelectionTool
    - TauEfficiencyCorrectionsTool
  - Photons :
    - EgammaCalibrationAndSmearingTool
    - AsgPhotonIsEMSelector
  - Electrons :
    - EgammaCalibrationAndSmearingTool
    - AsgElectronLikelihoodTool
    - AsgElectronIsEMSelector
  - Muons :
    - MuonCalibrationAndSmearingTool
    - MuonSelectionTool
    - MuonEfficiencyScaleFactors
  - MET :
    - METRebuilder
  - Overlap removal :
    - OverlapRemovalTool

CxAODMaker

- **Contents of the output xAOD**

- We store as little as possible – which in practical terms implies a trade-off :
  - The output file is small and (fairly) well organised, without too many variables cluttering/creating confusion.
  - But, while developing the code, this will also result in important variables missing, and thus re-running of the code.

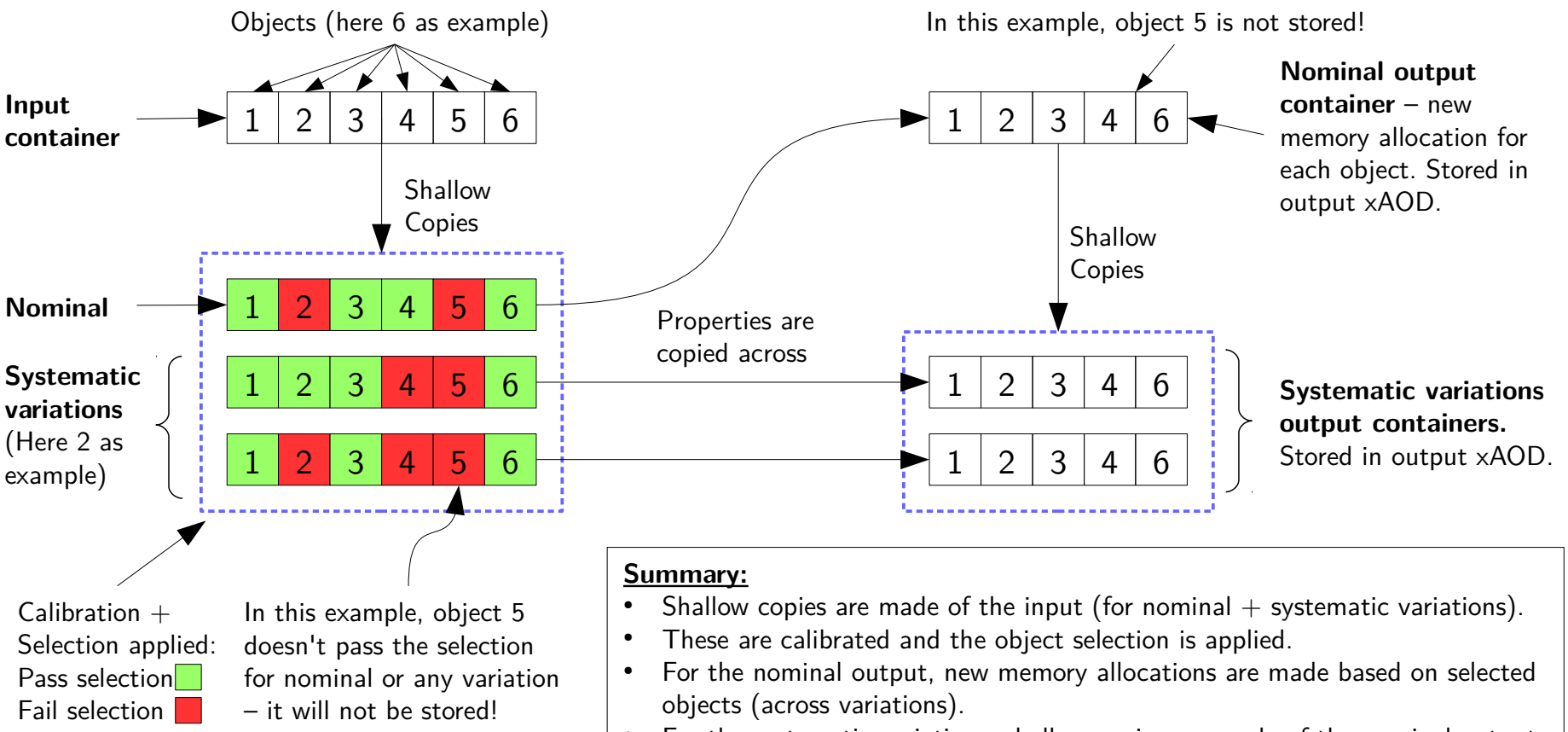
- For particle-type objects:
  - The 4-vector is always stored.
  - Other chosen quantities are explicitly added by hand (with 'decorators') and stored.
    (e.g. object scale factors, or flags which specify if an object passed some selection criteria).

- Event information:
  - Run number, event number, NVtx3Trks, MCEventWeight, ...
  - Eventually, also trigger information.

- Systematic variations are stored with the original collection name as "base" name and then the name of the variation appended to it (after 3 "_"), so e.g "Muons____Nominal" and "Muons____someVariation".

CxAODMaker

- **Technical details on I/O structure**



Summary:

- Shallow copies are made of the input (for nominal + systematic variations).
- These are calibrated and the object selection is applied.
- For the nominal output, new memory allocations are made based on selected objects (across variations).
- For the systematic variations, shallow copies are made of the nominal output.
- Properties are copied from calibrated shallow copies to output containers.

CxAODMaker

- **Configuration**
- The configuration file is located here:
  `FrameworkExe/data/CxAODMaker-job.cfg`

```
# This is a ConfigStore file, please see ConfigStore.cxx for info.

bool debug                    = false

bool printObjectCounts        = false
bool printKnownVariations     = false

bool applyObjectSelection     = true
bool applyOverlapRemoval      = true

bool applyEventSelection      = true
string selectionName          = 1lep
bool autoDetermineSelection   = false

# note: container names can be commented out to disable their processing

string muonContainer          = Muons
string electronContainer      = ElectronCollection
# no taus in HIGG5 derivations
#string tauContainer           = TauRecContainer
string photonContainer        = PhotonCollection
string jetContainer           = AntiKt4LCTopoJets
string jetAlgoName            = AntiKt4LCTopo
# JetUncertaintiesTool does not like AntiKt4TopoEM
#string jetSpectatorContainer  = AntiKt4EMTopoJets
#string jetSpectatorAlgoName   = AntiKt4TopoEM
string fatJetContainer        = CamKt12LCTopoJets
#string trackJetContainer      = AntiKt4ZTrackJets
string METContainer           = MET_RefFinal
string truthParticleContainer = TruthParticle
string truthVertexContainer   = TruthVertex
string grl = $ROOTCOREBIN/data/CxAODMaker/data12_8TeV.periodAllYear_DetStatus-v61-pro14-02_DQDefects-00-01-00_PHYS_StandardGRL_All_Good.xml

# "Nominal" is always written
vector<string> variations     = EG_SCALE_ALL MUONS_SCALE MUONS_ID MUONS_MS
vector<string> weightVariations = MUONSFSYS
vector<string> variationsSymm = JetEResol
bool storeOriginal            = true
```

Note: By default the VHbb "1 lepton" selection is applied

Names of collections, used by MuonHandler, JetHandler,...

List of sys. variations

TupleMaker

- **Produces flat ntuples**
  - int, float and arrays
  - An EventLoop algorithm, like AnalysisBase in CxAODMaker.
  - Configured independent of CxAODMaker:
    `FrameworkExe/data/TupleMaker-job.cfg`

```
Line
 1  # This is a ConfigStore file, please see ConfigStore.cxx for info.
 2
 3  string TupleMaker.PhotonsIn       = PhotonCollection
 4  string TupleMaker.MuonsIn         = Muons
 5  string TupleMaker.ElectronsIn     = ElectronCollection
 6  string TupleMaker.JetsIn          = AntiKt4LCTopoJets
 7  string TupleMaker.FatJetsIn       = CamKt12LCTopoJets
 8  string TupleMaker.MetIn           = MET_RefFinal
 9  string TupleMaker.Label           = tuple
10
11  vector<string> TupleMaker.Variations = Original Nominal
12  string         TupleMaker.VariationMode = block
13  bool           TupleMaker.UseEventInfo = true
```

Names of collections (sys. variations are appended automatically)

List of sys. variations

- Run with : `hsg5frameworkTuple`
  - will run CxAODMaker followed by TupleMaker such that the output collections in TEvent from the CxAODMaker are read in by TupleMaker.

- The systematic variations can be written out in 3 different ways:
  - "file" : A separate file with one TTree for each variation
  - "tree" : One file with one TTree for each variation
  - "block" : One file with one tree where the variables have the variation appended to their name

FrameworkExe

- **The framework executables**

  - Contains the files that defines the "main()" methods which are compiled into executables
    - `hsg5framework` (runs CxAODMaker)
    - `Hsg5frameworkTuple` (runs CxAODMaker and TupleMaker in sequence)
    - `Hsg5frameworkReader` (runs CxAODMakerReader, which processes CxAODs)

  - Configured in `FrameworkExe/data/framework-run.cfg`

```
1  # This is a ConfigStore file, please see ConfigStore.cxx for info.
2
3  int maxEvents                = -1
4  string CxAODConfig           = data/FrameworkExe/CxAODMaker-job.cfg
5  string TupleConfig           = data/FrameworkExe/TupleMaker-job.cfg
6  string COMEnergy             = 13TeV
7
8  ###################
9  # for grid only -> it is not affecting interactiv run
10 ###################
11 # tag version
12 string vtag                   = p1_141216
13 # test job
14 bool submit                   = false
15 # exlcude site
16 string excludedSite           = none
17 #string excludedSite           = ANALY_ARNES,ANALY_LUNARC,ANALY_SiGNET,ANALY_RAL_SL6,ANALY_BNL_SHORT,ANALY_DCSC,ANALY_NSC,ANALY_UIO
18 # lot of files per job
19 double nGBPerJob              = 8
20 double nFilesPerJob           = -1
21 # 1 file only per job
22 #double nGBPerJob             = -1
23 #double nFilesPerJob           = 1
24
```

Reads in the config files for CxAODMaker and TupleMaker from the previous 2 slides by default.

  - Final note on runtime configuration:
    - The steering file is distributed to all the Handlers in CxAODMaker, so modifying/adding configuration variables to a Handler is easy.

Running CxAODMaker / TupleMaker

Quick Start

For a quick setup of the framework please log into a clean lxplus session. Support for local machines is planned. Then you can copy&paste the following script to check out and compile the code:

```
# setup ATLAS environment
setupATLAS

# make some working direcory
mkdir CxAODFramework
cd CxAODFramework/

# setup RootCore
rcSetup Base,2.0.23
# Note, from Base,2.0.16 and later there are a lot of compile time warnings concerning the boost lib
# which are due to a new compile flag in ROOT, -Wunused-local-typedefs

# check out CxAODFramework
svn co svn+ssh://svn.cern.ch/reps/atlasphys/Physics/Higgs/HSG5/software/VHAnalysis/LHCRun2/CxAODFramework/CxAODMaker/trunk CxAODMaker
svn co svn+ssh://svn.cern.ch/reps/atlasphys/Physics/Higgs/HSG5/software/VHAnalysis/LHCRun2/CxAODFramework/CxAODReader/trunk CxAODReader
svn co svn+ssh://svn.cern.ch/reps/atlasphys/Physics/Higgs/HSG5/software/VHAnalysis/LHCRun2/CxAODFramework/CxAODTools/trunk CxAODTools
svn co svn+ssh://svn.cern.ch/reps/atlasphys/Physics/Higgs/HSG5/software/VHAnalysis/LHCRun2/CxAODFramework/FrameworkExe/trunk FrameworkExe
svn co svn+ssh://svn.cern.ch/reps/atlasphys/Physics/Higgs/HSG5/software/VHAnalysis/LHCRun2/CxAODFramework/FrameworkSub/trunk FrameworkSub
svn co svn+ssh://svn.cern.ch/reps/atlasphys/Physics/Higgs/HSG5/software/VHAnalysis/LHCRun2/CxAODFramework/TupleMaker/trunk TupleMaker
svn co svn+ssh://svn.cern.ch/reps/atlasphys/Physics/Higgs/HSG5/software/VHAnalysis/LHCRun2/CxAODFramework/TupleReader/trunk TupleReader

# scan packages and compile
rc find_packages
rc compile
```

A test job can be run with:

```
hsg5framework
```

Performance

- **Sample specifics**
  - HIGG2D4 DxAOD (2 lepton filter),
  - 10000 ttbar events (2.3 Gb)
    (mc14_8TeV.117050.PowhegPythia_P2011C_ttbar.merge.DAOD_HIGG2D4.e1727_s1933_s1911_r5591_r5625_p1784)

- **Reading/writing:**
  - ElectronCollection
  - Muons
  - AntiKt4LCTopoJets
  - CamKt12LCTopoJets
  - AntiKt4ZTrackJets
  - MET_RefFinal

- **Configuration**
  - 10 systematic variations
  - VHbb "2 lepton" event selection

- **Processing time (CPU time):**
  - $\sim$ 120 evt/sec

- **Disk space:**
  - Output CxAOD file size 3.5 Mb (reduction factor $\sim$ 650)

Mailing list and links

- HSG5 Framework mailing list:
  atlas-phys-higgs-hsg5Framework
- Framework TWiki:
  CxAODFramework
- Framework repository:
  Physics/Higgs/HSG5/software/VHAnalysis/LHCRun2/CxAODFramework

Backup slides

FrameworkExe

- **Grid running**
  - This part of the framework is not covered here, please see the documentation provided on our TWiki:
    https://twiki.cern.ch/twiki/bin/viewauth/AtlasProtected/CxAODFramework

Grid running

For setting up the grid log into a clean lxplus session and run in your working direcory:

```
setupATLAS
localSetupDQ2Client --skipConfirm # just for dq2-get -> no need for job submission
voms-proxy-init -voms atlas
localSetupPandaClient --noAthenaCheck
rcSetup
```

Find the list of samples in FrameworkSub /In, comment non-desired samples to run on and execute (for example) :

```
hsg5frameworkTuple FrameworkSub/In/list_sample_grid.*TeV.AOD.txt
```

Modifying the code

- **Adding a variable to the output**
  - Output variables are set in the method "setVariables()" which each handler has, e.g. ElectronHandler:

```
285 EL::StatusCode ElectronHandler::setVariables(xAOD::Electron * inElectron, xAOD::Electron * outElectron, bool isSysVar)
286 {
287
288   //TODO clean up
289   //TODO add check if variables changed due to calibrations?
290
291   // set four momentum
292   setP4( inElectron , outElectron );
293
294   //set isolation
295   outElectron->setIsolationValue(inElectron->isolationValue(xAOD::Iso::ptcone20),xAOD::Iso::ptcone20);
296   //outElectron->setIsolationValue(inElectron->isolationValue(xAOD::Iso::topoetcone30_corrected),xAOD::Iso::topoetcone30_corrected);
297   outElectron->setIsolationValue(inElectron->isolationValue(xAOD::Iso::topoetcone30),xAOD::Iso::topoetcone30);
298
299   // set something without having a pre-defined method
300   m_decorator.copy(inElectron, outElectron, ElecIntProps::isVeryTightLH);
301   m_decorator.copy(inElectron, outElectron, ElecIntProps::isVHLooseElectron);
302   m_decorator.copy(inElectron, outElectron, ElecIntProps::isZHSignalElectron);
303   m_decorator.copy(inElectron, outElectron, ElecIntProps::isWHSignalElectron);
```

- The I/O structure in the code is not entirely transparent – the "inElectron" is an object from a socalled "shallow copy" of the collection in the input file. This is the calibrated object which has additional "decorations" attached to it. The "outElectron" is the object which is written out. For the nominal calibration, this is a new memory allocation, while for sys. variations, it is a shallow copy of the nominal output object.

Modifying the code

- **Adding a variable to the output**
  - Output variables are set in the method "setVariables()" which each handler has, e.g. ElectronHandler:

```
285 EL::StatusCode ElectronHandler::setVariables(xAOD::Electron * inElectron, xAOD::Electron * outElectron, bool isSysVar)
286 {
287
288   //TODO clean up
289   //TODO add check if variables changed due to calibrations?
290
291   // set four momentum
292   setP4( inElectron , outElectron );
293
294   //set isolation
295   outElectron->setIsolationValue(inElectron->isolationValue(xAOD::Iso::ptcone20),xAOD::Iso::ptcone20);
296   //outElectron->setIsolationValue(inElectron->isolationValue(xAOD::Iso::topoetcone30_corrected),xAOD::Iso::topoetcone30_corrected);
297   outElectron->setIsolationValue(inElectron->isolationValue(xAOD::Iso::topoetcone30),xAOD::Iso::topoetcone30);
298
299   // set something without having a pre-defined method
300   m_decorator.copy(inElectron, outElectron, ElecIntProps::isVeryTightLH);
301   m_decorator.copy(inElectron, outElectron, ElecIntProps::isVHLooseElectron);
302   m_decorator.copy(inElectron, outElectron, ElecIntProps::isZHSignalElectron);
303   m_decorator.copy(inElectron, outElectron, ElecIntProps::isWHSignalElectron);
```

- So, what this method does is to copy properties from the calibrated object ("inElectron") to the output object ("outElectron").
- **Thus, to add a variable to "outElectron", a line must be put here copying the property**.

Modifying the code

- **Adding a variable to the output**
  - In case **the variable is *not* a known property** to "inElectron", it must first be calculated and added to this object, e.g. in the method decorate():

```
107 EL::StatusCode ElectronHandler::decorate(xAOD::Electron * electron)
108 {
109
110   //selection tools
111   //---------------
112   //retrieve decision from tools and decorate electron with it
113   //perform actual selection later
114   int isVeryLooseLH = static_cast<int>(m_checkVeryLooseLH->accept(electron));
115   int isVeryTightLH = static_cast<int>(m_checkVeryTightLH->accept(electron));
116   int isTightPP     = static_cast<int>(m_ElectronIsEMSelector->accept(electron));
117   m_decorator.set(electron, ElecIntProps::isVeryLooseLH, isVeryLooseLH);
118   m_decorator.set(electron, ElecIntProps::isVeryTightLH, isVeryTightLH);
119   m_decorator.set(electron, ElecIntProps::isTightPP, isTightPP);
120
121   return EL::StatusCode::SUCCESS;
122
123 }
```

This is the same object as "inElectron" on the previous slide

  - There is more...

Modifying the code

- **Adding a variable to the output**
  - … the property also needs to be added the object's decorator, which is defined in CxAODTools, e.g. for electrons: CxAODTools/CxAODTools/ElectronDecorator.h

```
1  #ifndef CxAODTools_ElectronDecorator_H
2  #define CxAODTools_ElectronDecorator_H
3
4  #include "ObjectDecorator.h"
5
6  enum class ElecBoolProps {
7    // OR tool
8    overlaps,
9    passPreSel,
10 };
11
12 enum class ElecIntProps {
13   // common stuff
14   partIndex,
15     passOR,
16     passORGlob,
17     // e-gamma ID
18     isVeryLooseLH,
19     isVeryTightLH,
20     isTightPP,
21     isMediumPP,
22     isLoosePP,
23     // analysis quality assignment
24     isVHLooseElectron,
25     isVHSignalElectron,
26     isZHSignalElectron,
27     isWHSignalElectron,
28     isTTLHOLRElectron,
29     isTTLHDiLepVetoElectron,
30     isTTLHPreSelElectron,
31     isTTLHIsolElectron,
32     isTTLHSoftElectron,
33
34 };
35
36 enum class ElecFloatProps { IDEff, IDEffSys };
37
```

Boolean properties

Integer properties

Float properties

  - That should do it