The (Technical) Challenges of Finding a <Particle>

Stephan Hageböck
29.01.2019

Intro

- PhD in ATLAS
  - VH → Vbb search
  - Worked for University of Bonn
    - Group-local analysis framework
    - Local computing cluster (~500 logical CPUs)
  - Main topics:
    - Object + event selection
    - Machine learning
    - Statistical Models
- Now:
  - ROOT team
  - Work on improving RooFit

Analysis Workflow

1. Monte Carlo simulation, reconstruction: centrally on grid
2. Skim & slim samples, apply some (centrally provided) calibrations

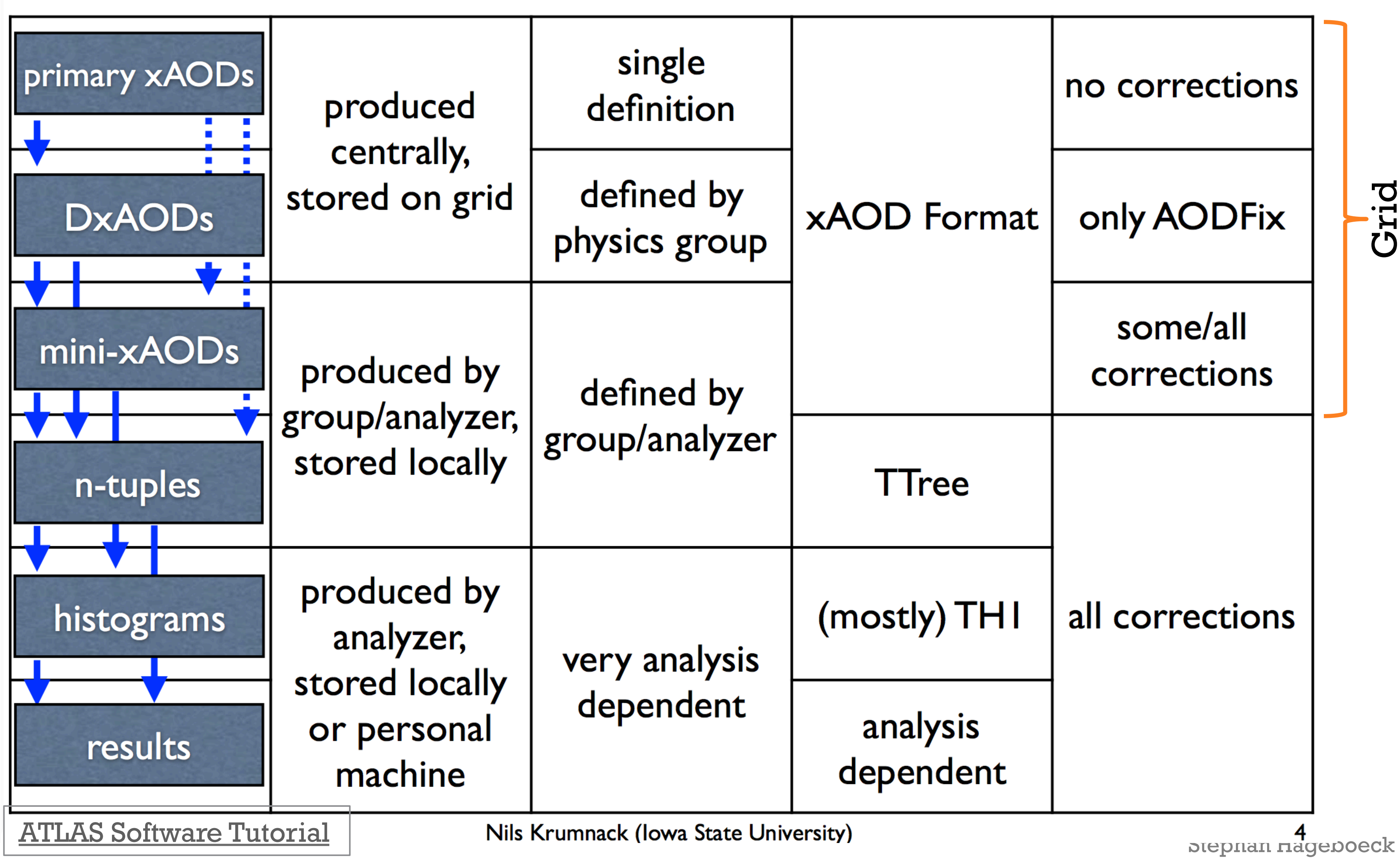
Grid

3. Modelling checks
    1. Apply ATLAS corrections
    2. + analysis-specific correct.
    3. Compare Data & MC
    4. Compare MC & MC
    5. Derive corrections
    6. Cross-checks
    7. Refine selection

5. Train MVA
6. Evaluate uncertainties
7. Statistical Model (RooFit model created from thousands of analysis histograms)
8. Cross checks
9. Results

Note to self: How did steps evolve?

Analysis Workflow



primary xAODs	produced centrally, stored on grid	single definition	xAOD Format	no corrections	Grid
DxAODs		defined by physics group		only AODFix	
mini-xAODs	produced by group/analyzer, stored locally	defined by group/analyzer		some/all corrections	
n-tuples			TTree	all corrections	
histograms	produced by analyzer, stored locally or personal machine	very analysis dependent	(mostly) TH1		
results			analysis dependent		

Stephan Hageboeck

Personal Workflow

- Grid:
  - Centrally provided ATLAS software stack, small own modules
- Afterwards:
  - ssh → university cluster, office computer
  - Develop & test on office computer
    `Happy ? CheckIn() : Repeat()`
  - Check out on cluster, submit jobs
  - Software: Always from **CVMFS** + own git
- NB: Never analysis on laptop
  - Reason: Heavy lifting happens on university cluster
    → Don't want to maintain 2 setups
  - 1 job on laptop is ok, but need ~600 jobs for full analysis
  - Looking at root file on laptop is ok

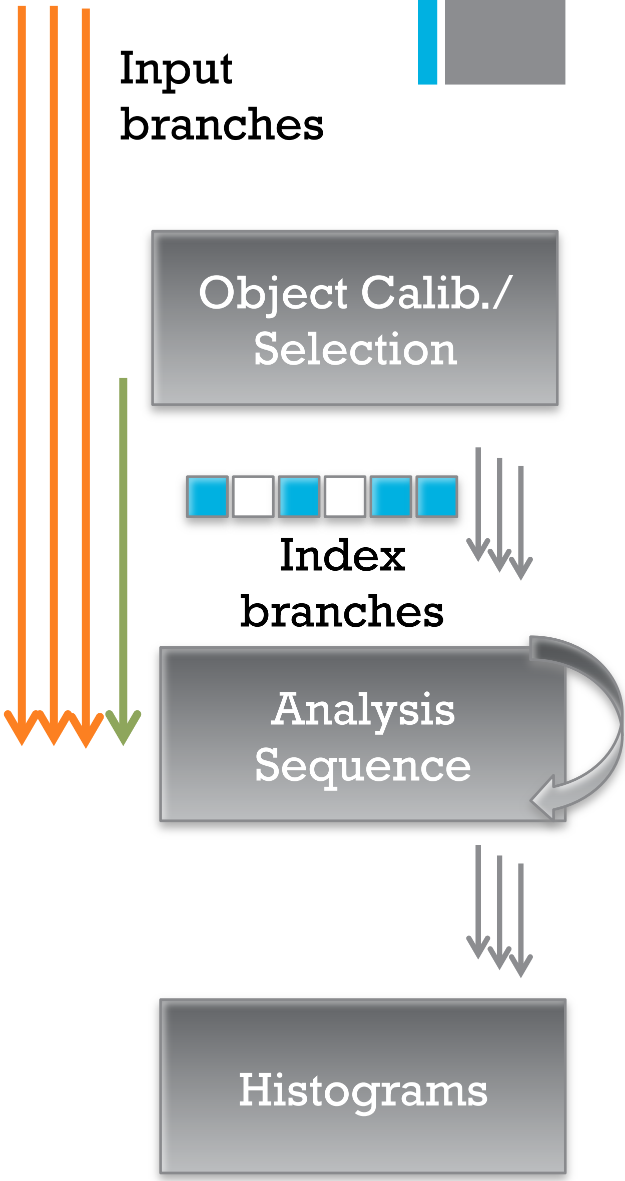
Analysis Interface

- Early steps ATLAS-central: e.g. Tracking, jet clustering
- Next steps (My opinion):
  - For a sufficiently complex (=normal) analysis, a non-trivial framework is necessary (not provided by ATLAS / ROOT)
- Bonn approach: The "Overkill" C++ framework
  - C++, compiled into libraries
    - Does everything from calibrating, selecting, categorising to filling histograms
  - Either load into interpreter or compile into simple executable
  - Job submission:
    - Python script to collect & manage input file and configs
    - Automatic splitting and scheduling: submit.py --events 300000
    - Submit analysis jobs to PBS cluster (optionally: grid)
      - + merge jobs
      - + output collector job
  - **Check with GUI!**

The Heavy Lifting: C++ Framework

- ~ 240 MC samples scattered over ~ 1000 – 2000 root files. More or less flat, but rebuild objects from branches. ~ 5 Tb
- Each job has data flow between modules by reading / writing branches
  - Only things that actually change get written to mem (disc):
    - Index branches
    - Calibrated energies
- My experience: It’s nice to have configurable modules
  - **Module with standard ATLAS calibrations**
  - Swap in/out different selections using config file
  - Snapshot subset of “active” branches at any point (e.g. to train MVA)
  - Run 30 sec test job and immediately look at histograms

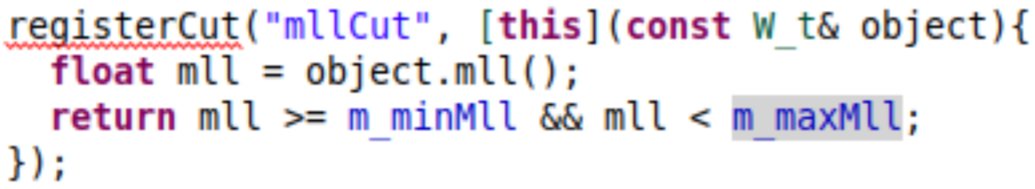
  → Interactivity & exploratory analysis



The Heavy Lifting: C++ Framework

- Want: Interactive & automated processing of many files with changing configs

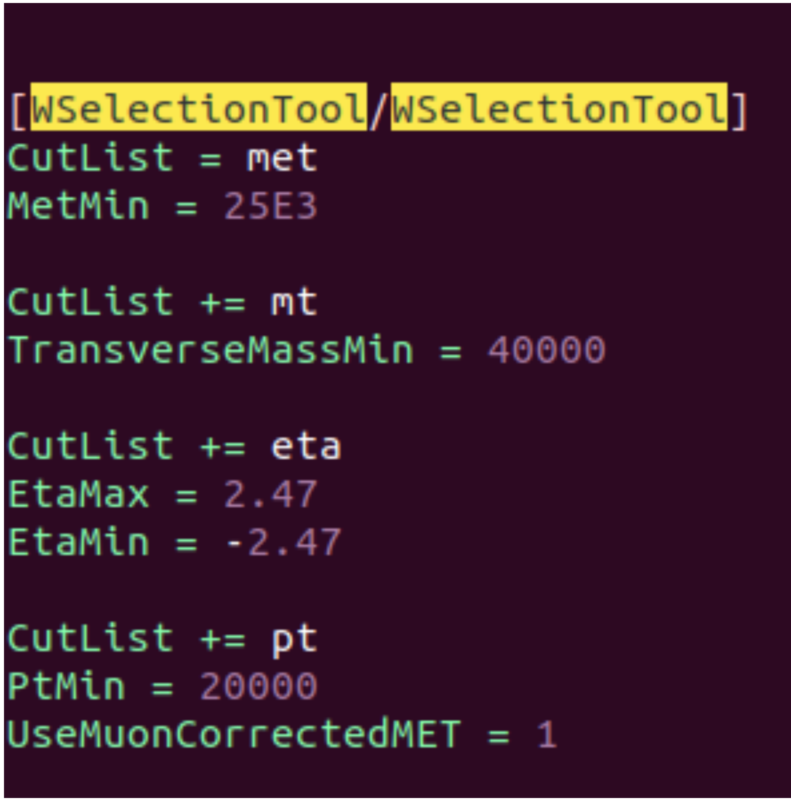
- Config needs to handle both cases easily: Configurable modules & cuts



```
registerCut("mllCut", [this](const W_t& object){
  float mll = object.mll();
  return mll >= m_minMll && mll < m_maxMll;
});
```

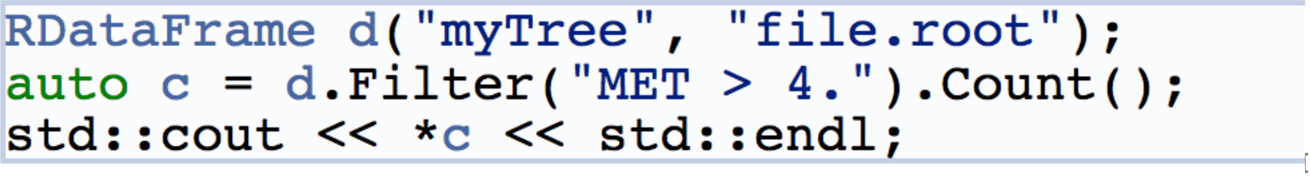
- Want to be able to book different selection / calibration tools
  → Create different analysis branches

- Cut flow histograms should drop out **automatically**



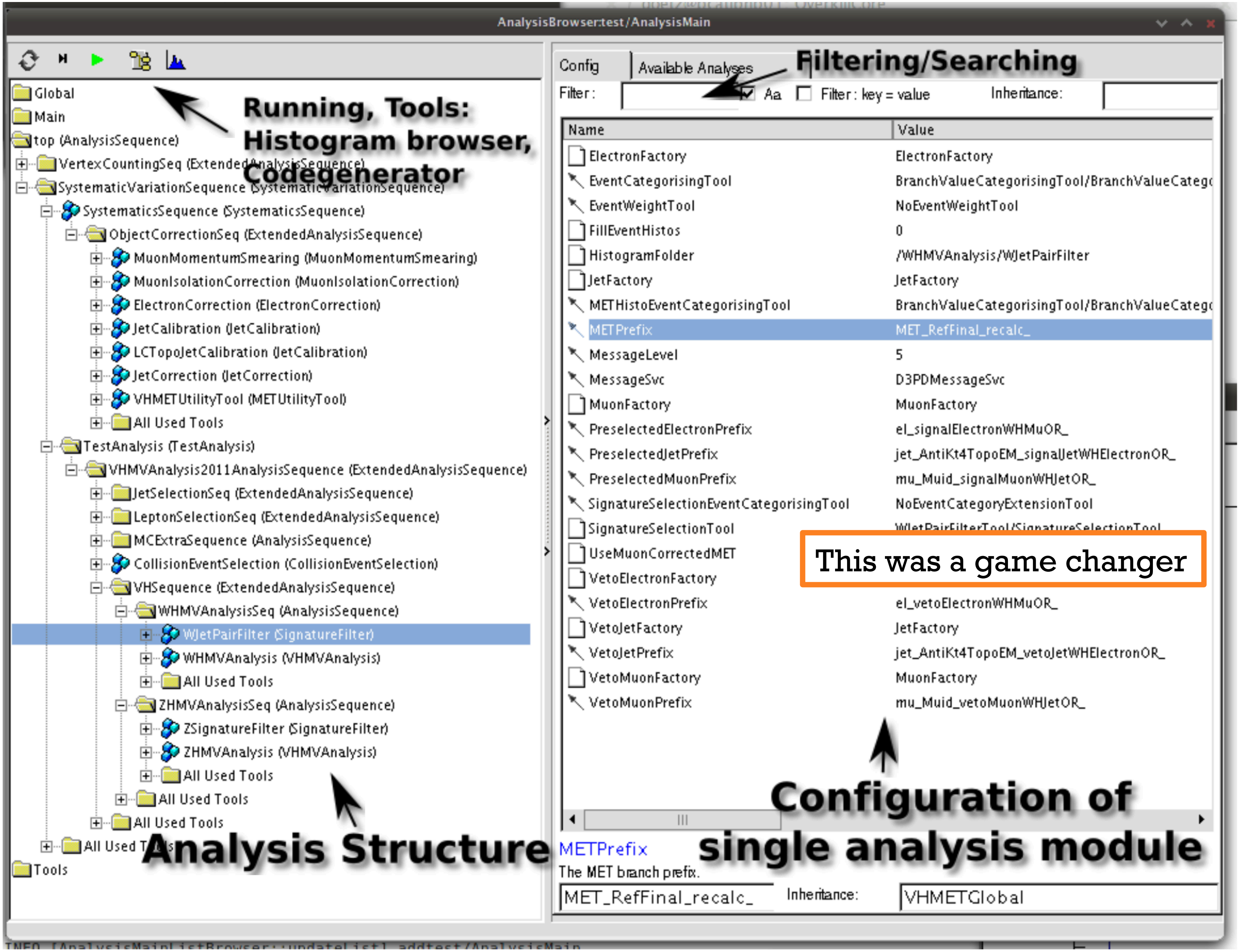
```
[WSelectionTool/WSelectionTool]
CutList = met
MetMin = 25E3

CutList += mt
TransverseMassMin = 40000

CutList += eta
EtaMax = 2.47
EtaMin = -2.47

CutList += pt
PtMin = 20000
UseMuonCorrectedMET = 1
```

- **Does this collide with**



```
RDataFrame d("myTree", "file.root");
auto c = d.Filter("MET > 4.").Count();
std::cout << *c << std::endl;
```

?



AnalysisBrowser:test/AnalysisMain

Running, Tools: Histogram browser, Codegenerator

Filtering/Searching

```
Global
Main
top (AnalysisSequence)
  VertexCountingSeq (ExtendedAnalysisSequence)
  SystematicVariationSequence (SystematicVariationSequence)
    SystematicsSequence (SystematicsSequence)
      ObjectCorrectionSeq (ExtendedAnalysisSequence)
        MuonMomentumSmearing (MuonMomentumSmearing)
        MuonIsolationCorrection (MuonIsolationCorrection)
        ElectronCorrection (ElectronCorrection)
        JetCalibration (JetCalibration)
        LCTopoJetCalibration (JetCalibration)
        JetCorrection (JetCorrection)
        VHMETUtilityTool (METUtilityTool)
        All Used Tools
    TestAnalysis (TestAnalysis)
      VHMVAnalysis2011AnalysisSequence (ExtendedAnalysisSequence)
        JetSelectionSeq (ExtendedAnalysisSequence)
        LeptonSelectionSeq (ExtendedAnalysisSequence)
        MCExtraSequence (AnalysisSequence)
        CollisionEventSelection (CollisionEventSelection)
        VHSequence (ExtendedAnalysisSequence)
          WHMVAnalysisSeq (AnalysisSequence)
            WJetPairFilter (SignatureFilter)
            WHMVAnalysis (VHMVAnalysis)
            All Used Tools
          ZHMVAnalysisSeq (AnalysisSequence)
            ZSignatureFilter (SignatureFilter)
            ZHMVAnalysis (VHMVAnalysis)
            All Used Tools
          All Used Tools
        All Used Tools
    All Used Tools
Tools
```

Config | Available Analyses

Filter: ☑ Aa ☐ Filter: key = value Inheritance:

Name	Value
ElectronFactory	ElectronFactory
EventCategorisingTool	BranchValueCategorisingTool/BranchValueCatego
EventWeightTool	NoEventWeightTool
FillEventHistos	0
HistogramFolder	/WHMVAnalysis/WJetPairFilter
JetFactory	JetFactory
METHistoEventCategorisingTool	BranchValueCategorisingTool/BranchValueCatego
METPrefix	MET_RefFinal_recalc_
MessageLevel	5
MessageSvc	D3PDMessageSvc
MuonFactory	MuonFactory
PreselectedElectronPrefix	el_signalElectronWHMuOR_
PreselectedJetPrefix	jet_AntiKt4TopoEM_signalJetWHElectronOR_
PreselectedMuonPrefix	mu_Muid_signalMuonWHJetOR_
SignatureSelectionEventCategorisingTool	NoEventCategoryExtensionTool
SignatureSelectionTool	WJetPairFilterTool/SignatureSelectionTool
UseMuonCorrectedMET	
VetoElectronFactory	
VetoElectronPrefix	el_vetoElectronWHMuOR_
VetoJetFactory	JetFactory
VetoJetPrefix	jet_AntiKt4TopoEM_vetoJetWHElectronOR_
VetoMuonFactory	MuonFactory
VetoMuonPrefix	mu_Muid_vetoMuonWHJetOR_

This was a game changer

METPrefix

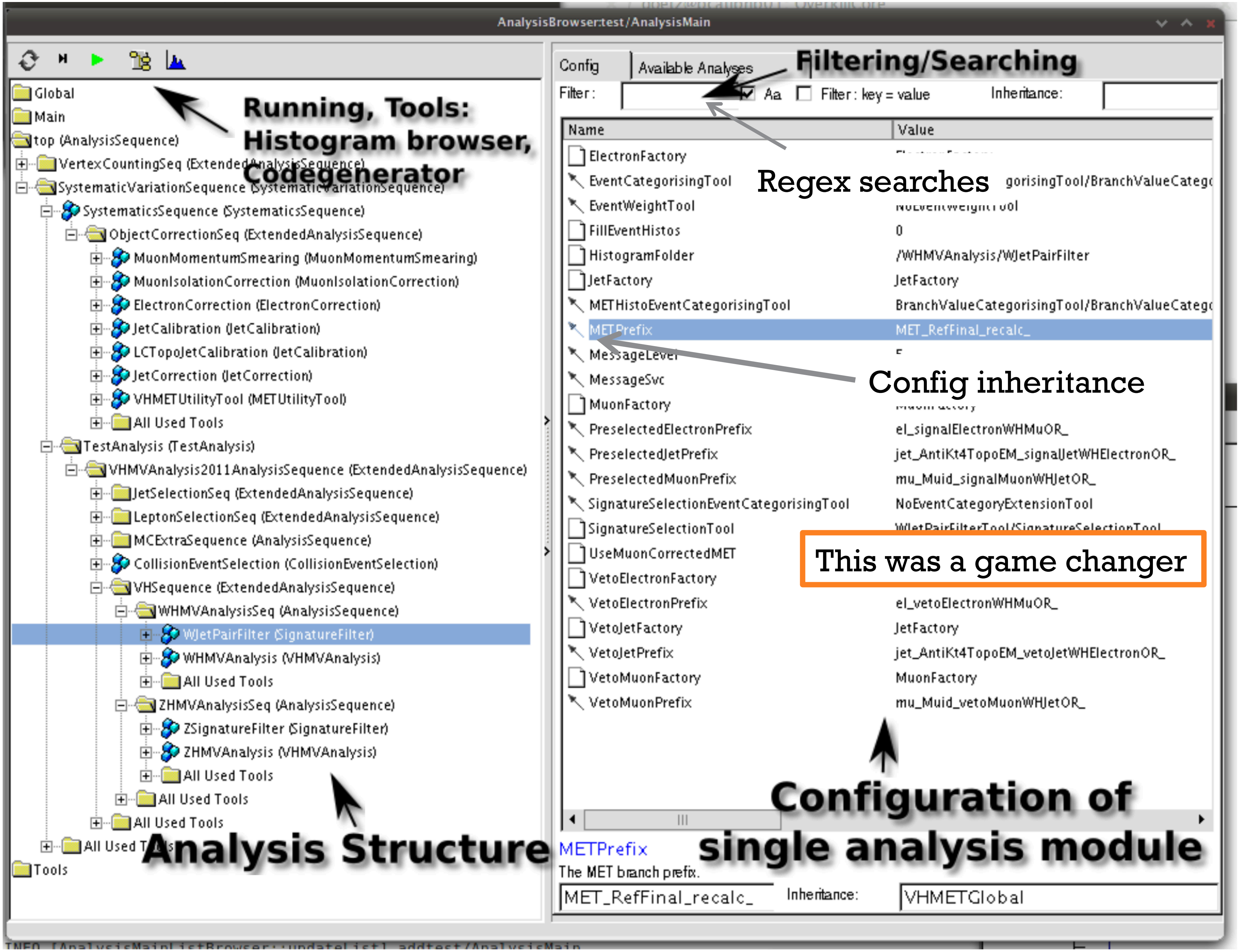
The MET branch prefix.

MET_RefFinal_recalc_ Inheritance: VHMETGlobal

Analysis Structure

Configuration of single analysis module

```
INFO [AnalysisMainListBrowser::updateList] addtest/AnalysisMain
```



Running, Tools: Histogram browser, Codegenerator

Filtering/Searching

Regex searches

Config inheritance

This was a game changer

Configuration of single analysis module

Analysis Structure

- Global
- Main
- top (AnalysisSequence)
  - VertexCountingSeq (ExtendedAnalysisSequence)
  - SystematicVariationSequence (SystematicVariationSequence)
    - SystematicsSequence (SystematicsSequence)
      - ObjectCorrectionSeq (ExtendedAnalysisSequence)
        - MuonMomentumSmearing (MuonMomentumSmearing)
        - MuonIsolationCorrection (MuonIsolationCorrection)
        - ElectronCorrection (ElectronCorrection)
        - JetCalibration (JetCalibration)
        - LCTopoJetCalibration (JetCalibration)
        - JetCorrection (JetCorrection)
        - VHMETUtilityTool (METUtilityTool)
        - All Used Tools
    - TestAnalysis (TestAnalysis)
      - VHMVAnalysis2011AnalysisSequence (ExtendedAnalysisSequence)
        - JetSelectionSeq (ExtendedAnalysisSequence)
        - LeptonSelectionSeq (ExtendedAnalysisSequence)
        - MCExtraSequence (AnalysisSequence)
        - CollisionEventSelection (CollisionEventSelection)
        - VHSequence (ExtendedAnalysisSequence)
          - WHMVAnalysisSeq (AnalysisSequence)
            - WJetPairFilter (SignatureFilter)
            - WHMVAnalysis (VHMVAnalysis)
            - All Used Tools
          - ZHMVAnalysisSeq (AnalysisSequence)
            - ZSignatureFilter (SignatureFilter)
            - ZHMVAnalysis (VHMVAnalysis)
            - All Used Tools
          - All Used Tools
        - All Used Tools
    - All Used T
- Tools

Name	Value
ElectronFactory	[illegible]
EventCategorisingTool	gorisingTool/BranchValueCatego
EventWeightTool	NoEventWeightTool
FillEventHistos	0
HistogramFolder	/WHMVAnalysis/WJetPairFilter
JetFactory	JetFactory
METHistoEventCategorisingTool	BranchValueCategorisingTool/BranchValueCatego
METPrefix	MET_RefFinal_recalc_
MessageLevel	5
MessageSvc	
MuonFactory	MuonFactory
PreselectedElectronPrefix	el_signalElectronWHMuOR_
PreselectedJetPrefix	jet_AntiKt4TopoEM_signalJetWHElectronOR_
PreselectedMuonPrefix	mu_Muid_signalMuonWHJetOR_
SignatureSelectionEventCategorisingTool	NoEventCategoryExtensionTool
SignatureSelectionTool	WJetPairFilterTool/SignatureSelectionTool
UseMuonCorrectedMET	
VetoElectronFactory	
VetoElectronPrefix	el_vetoElectronWHMuOR_
VetoJetFactory	JetFactory
VetoJetPrefix	jet_AntiKt4TopoEM_vetoJetWHElectronOR_
VetoMuonFactory	MuonFactory
VetoMuonPrefix	mu_Muid_vetoMuonWHJetOR_

METPrefix

The MET branch prefix.

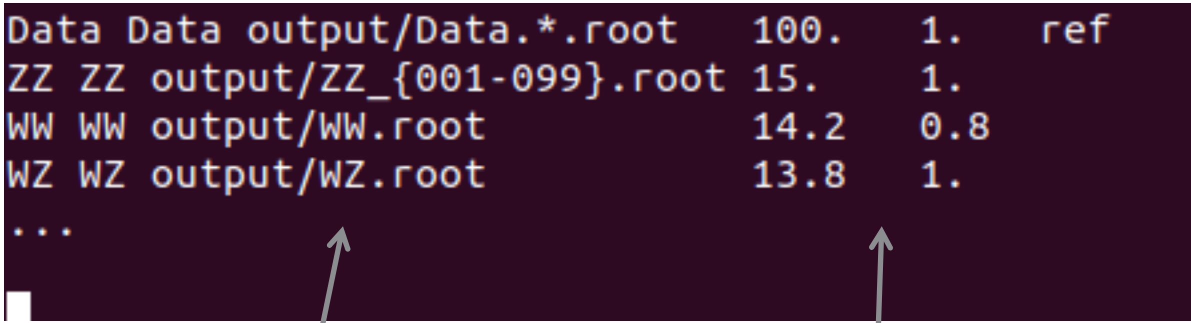
MET_RefFinal_recalc_ Inheritance: VHMETGlobal

```
INFO [AnalysisMainListBrowser::updateList] add test/AnalysisMain
```

Missing: Efficiently Checking Histograms

- Workflow reminder:
  - Test & develop on single sample, iterate
  - When ok, send jobs for all samples
  - Merge & retrieve histograms from hundreds of jobs
- Many people: Plotting macro
- Overkill: ShowMulti(ple types)
  - Rapidly show stacked, scaled & coloured histograms

ShowMulti inputs:



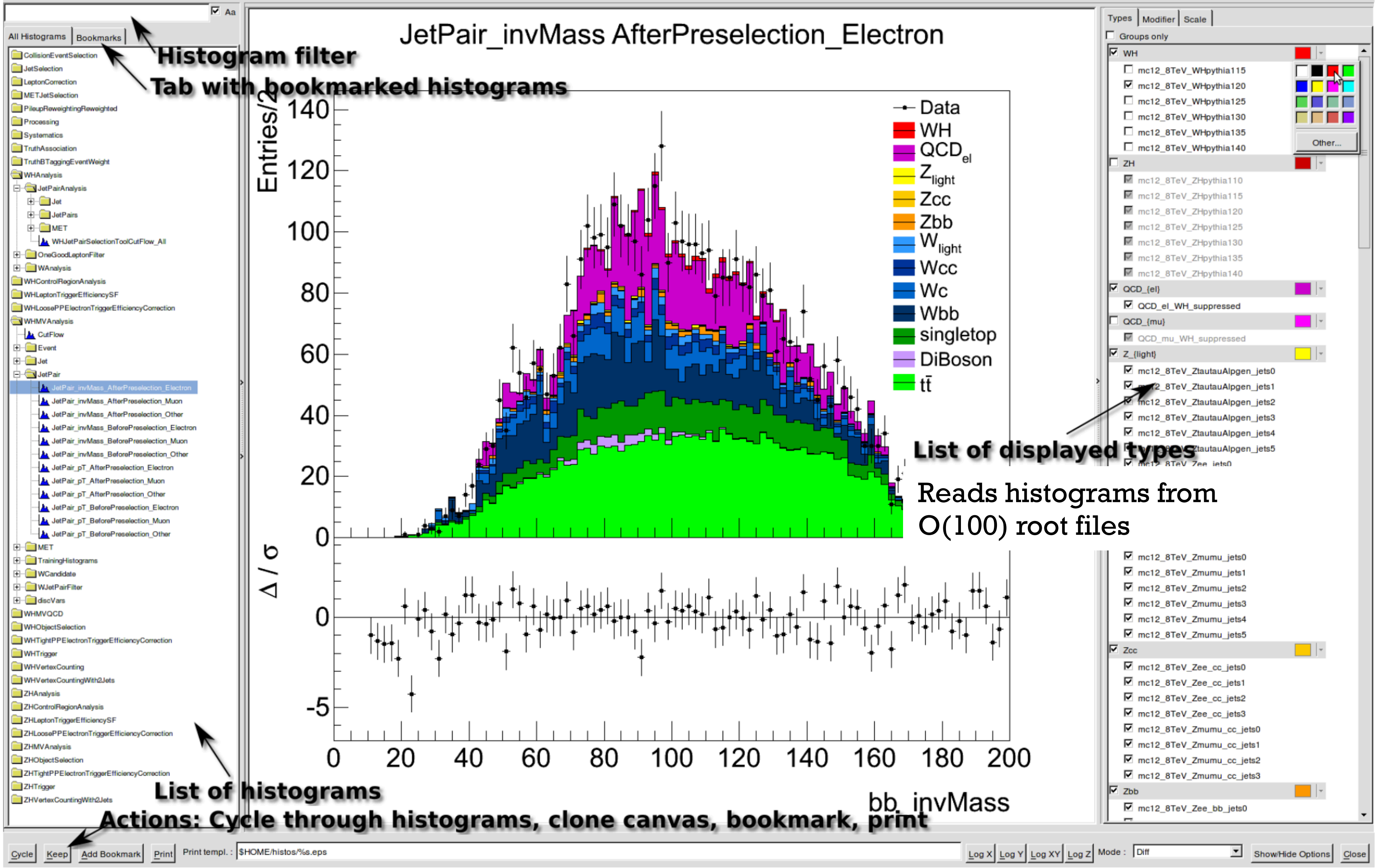
```
Data Data output/Data.*.root   100.   1.   ref
ZZ ZZ output/ZZ_{001-099}.root 15.    1.
WW WW output/WW.root           14.2   0.8
WZ WZ output/WZ.root           13.8   1.
...
```

Path with regex & globbing

Lumi + Filter Eff for auto-scaling

Missing: Efficiently Checking Histograms

This was **the** game changer


☑ Aa
All Histograms
Bookmarks
CollisionEventSelection
JetSelection
LeptonCorrection
METJetSelection
PileupReweightingReweighted
Processing
Systematics
TruthAssociation
TruthBTaggingEventWeight
WHAnalysis
JetPairAnalysis
Jet
JetPairs
MET
WHJetPairSelectionToolCutFlow_All
OneGoodLeptonFilter
WAnalysis
WHControlRegionAnalysis
WHLeptonTriggerEfficiencySF
WHLoosePPElectronTriggerEfficiencyCorrection
WHMVAnalysis
CutFlow
Event
Jet
JetPair
JetPair_invMass_AfterPreselection_Electron
JetPair_invMass_AfterPreselection_Muon
JetPair_invMass_AfterPreselection_Other
JetPair_invMass_BeforePreselection_Electron
JetPair_invMass_BeforePreselection_Muon
JetPair_invMass_BeforePreselection_Other
JetPair_pT_AfterPreselection_Electron
JetPair_pT_AfterPreselection_Muon
JetPair_pT_AfterPreselection_Other
JetPair_pT_BeforePreselection_Electron
JetPair_pT_BeforePreselection_Muon
JetPair_pT_BeforePreselection_Other
MET
TrainingHistograms
WCandidate
WJetPairFilter
discVars
WHMVQCD
WHObjectSelection
WHTightPPElectronTriggerEfficiencyCorrection
WHTrigger
WHVertexCounting
WHVertexCountingWith2Jets
ZHAnalysis
ZHControlRegionAnalysis
ZHLeptonTriggerEfficiencySF
ZHLoosePPElectronTriggerEfficiencyCorrection
ZHMVAnalysis
ZHObjectSelection
ZHTightPPElectronTriggerEfficiencyCorrection
ZHTrigger
ZHVertexCountingWith2Jets
JetPair_invMass AfterPreselection_Electron
Entries/2
140
120
100
80
60
40
20
0
Data
WH
QCD_{el}
Z_{light}
Zcc
Zbb
W_{light}
Wcc
Wc
Wbb
singletop
DiBoson
$t\bar{t}$
Δ / σ
0
-5
0
20
40
60
80
100
120
140
160
180
200
bb_invMass
Types
Modifier
Scale
Other...
Cycle
Keep
Add Bookmark
Print
Log X
Log Y
Log XY
Log Z
Show/Hide Options
Close

☐ Groups only

☑ WH
- ☐ mc12_8TeV_WHpythia115
- ☑ mc12_8TeV_WHpythia120
- ☐ mc12_8TeV_WHpythia125
- ☐ mc12_8TeV_WHpythia130
- ☐ mc12_8TeV_WHpythia135
- ☐ mc12_8TeV_WHpythia140

☐ ZH
- ☑ mc12_8TeV_ZHpythia110
- ☑ mc12_8TeV_ZHpythia115
- ☑ mc12_8TeV_ZHpythia120
- ☑ mc12_8TeV_ZHpythia125
- ☑ mc12_8TeV_ZHpythia130
- ☑ mc12_8TeV_ZHpythia135
- ☑ mc12_8TeV_ZHpythia140

☑ QCD_{el}
- ☑ QCD_el_WH_suppressed

☐ QCD_{mu}
- ☑ QCD_mu_WH_suppressed

☑ Z_{light}
- ☑ mc12_8TeV_ZtautauAlpgen_jets0
- ☑ mc12_8TeV_ZtautauAlpgen_jets1
- ☑ mc12_8TeV_ZtautauAlpgen_jets2
- ☑ mc12_8TeV_ZtautauAlpgen_jets3
- ☑ mc12_8TeV_ZtautauAlpgen_jets4
- ☑ mc12_8TeV_ZtautauAlpgen_jets5
- ☑ mc12_8TeV_Zee_jets0
- ☑ mc12_8TeV_Zmumu_jets0
- ☑ mc12_8TeV_Zmumu_jets1
- ☑ mc12_8TeV_Zmumu_jets2
- ☑ mc12_8TeV_Zmumu_jets3
- ☑ mc12_8TeV_Zmumu_jets4
- ☑ mc12_8TeV_Zmumu_jets5

☑ Zcc
- ☑ mc12_8TeV_Zee_cc_jets0
- ☑ mc12_8TeV_Zee_cc_jets1
- ☑ mc12_8TeV_Zee_cc_jets2
- ☑ mc12_8TeV_Zee_cc_jets3
- ☑ mc12_8TeV_Zmumu_cc_jets0
- ☑ mc12_8TeV_Zmumu_cc_jets1
- ☑ mc12_8TeV_Zmumu_cc_jets2
- ☑ mc12_8TeV_Zmumu_cc_jets3

☑ Zbb
- ☑ mc12_8TeV_Zee_bb_jets0

Print templ. : $HOME/histos/%s.eps

Mode : Diff

Histogram filter

Tab with bookmarked histograms

List of displayed types

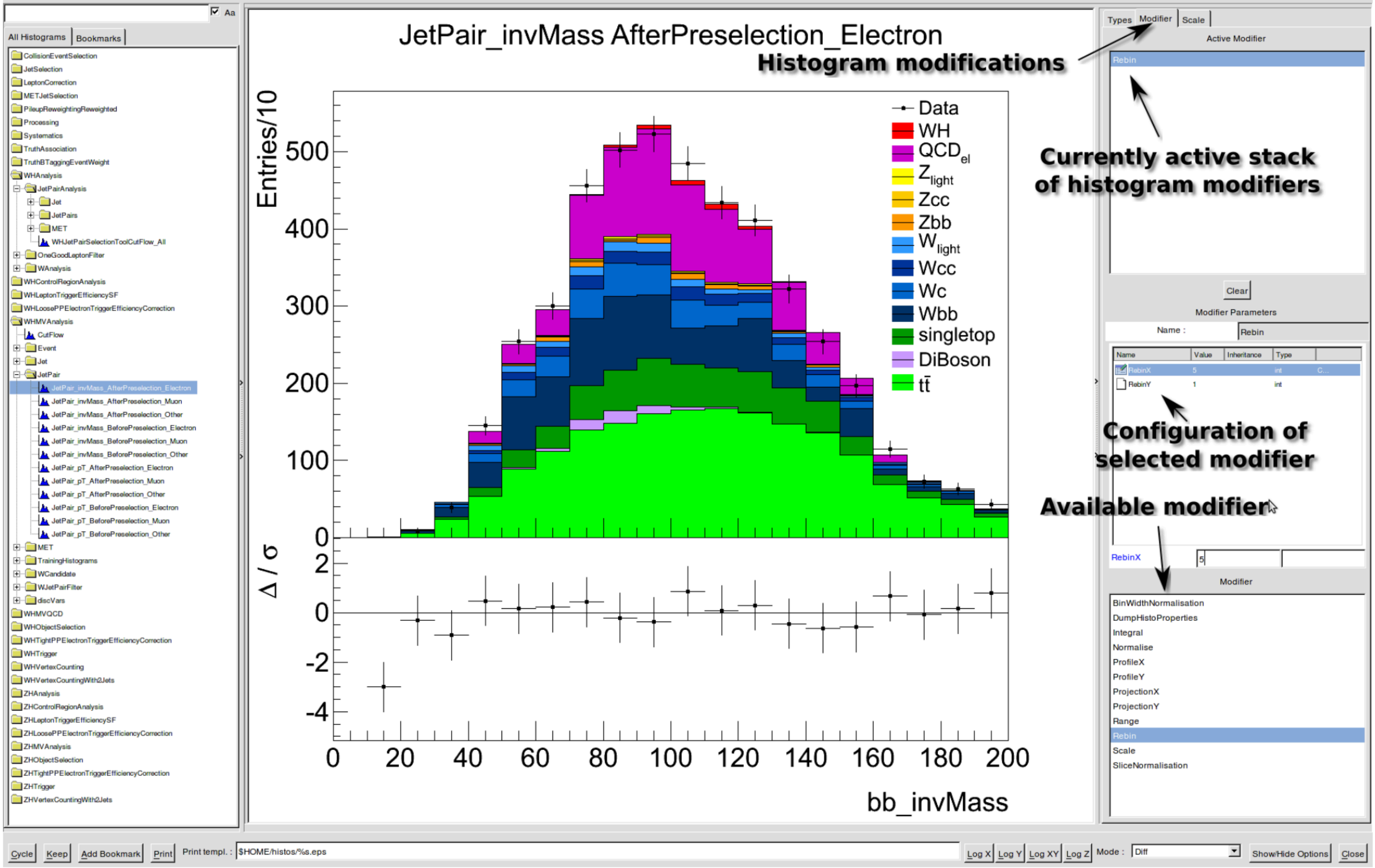
Reads histograms from O(100) root files

List of histograms

Actions: Cycle through histograms, clone canvas, bookmark, print

Missing: Efficiently Checking Histograms

This was **the** game changer



- CollisionEventSelection
- JetSelection
- LeptonCorrection
- METJetSelection
- PileupReweightingReweighted
- Processing
- Systematics
- TruthAssociation
- TruthBTaggingEventWeight
- WHAnalysis
  - JetPairAnalysis
    - Jet
    - JetPairs
    - MET
    - WHJetPairSelectionToolCutFlow_All
  - OneGoodLeptonFilter
  - WAnalysis
- WHControlRegionAnalysis
- WHLeptonTriggerEfficiencySF
- WHLoosePPElectronTriggerEfficiencyCorrection
- WHMVAnalysis
  - CutFlow
  - Event
  - Jet
  - JetPair
    - JetPair_invMass_AfterPreselection_Electron
    - JetPair_invMass_AfterPreselection_Muon
    - JetPair_invMass_AfterPreselection_Other
    - JetPair_invMass_BeforePreselection_Electron
    - JetPair_invMass_BeforePreselection_Muon
    - JetPair_invMass_BeforePreselection_Other
    - JetPair_pT_AfterPreselection_Electron
    - JetPair_pT_AfterPreselection_Muon
    - JetPair_pT_AfterPreselection_Other
    - JetPair_pT_BeforePreselection_Electron
    - JetPair_pT_BeforePreselection_Muon
    - JetPair_pT_BeforePreselection_Other
  - MET
  - TrainingHistograms
  - WCandidate
  - WJetPairFilter
  - discVars
- WHMVQCD
- WHObjectSelection
- WHTightPPElectronTriggerEfficiencyCorrection
- WHTrigger
- WHVertexCounting
- WHVertexCountingWith2Jets
- ZHAnalysis
- ZHControlRegionAnalysis
- ZHLeptonTriggerEfficiencySF
- ZHLoosePPElectronTriggerEfficiencyCorrection
- ZHMVAnalysis
- ZHObjectSelection
- ZHTightPPElectronTriggerEfficiencyCorrection
- ZHTrigger
- ZHVertexCountingWith2Jets

Print templ. : $HOME/histos/%s.eps

Mode : Diff

Histogram modifications

Active Modifier

Rebin

Currently active stack of histogram modifiers

Modifier Parameters

Name : Rebin

Name	Value	Inheritance	Type	
RebinX	5		int	C...
RebinY	1		int	

Configuration of selected modifier

Available modifier

RebinX 5

Modifier

- BinWidthNormalisation
- DumpHistoProperties
- Integral
- Normalise
- ProfileX
- ProfileY
- ProjectionX
- ProjectionY
- Range
- Rebin
- Scale
- SliceNormalisation

eck

Analysis Workflow: Final Steps

1. O(10k) Bonn histograms scattered over one or more files per sample
2. Collect, rename, merge into ATLAS H→bb format using Overkill's ShowMulti. Obtain single output file with histograms.
3. ATLAS-H→bb-specific renaming & splitting tool puts histograms into different file structure
4. ATLAS-H→bb-specific tool (WSMaker) creates RooFit workspaces
5. Standard ATLAS macros run to extract/cross-check results (Batch cluster)

- Could have been simplified, but histogram naming was up to each group when analysis was set up
- RooStats::HistFactory interface seemingly too complicated (at least I know someone who is responsible now)

Scaling

- **Bonn**: Create objects from less-derived inputs always run ATLAS calibrations, object / event selection, overlap removal
  - Turn around:
    - 30 sec for 40k events on signal sample (test & develop, office computer)
    - ~ 1 h for data and all nominal MC (cluster)
    - 6 – 8 h for all systematics (cluster)
  - Could easily (~ 2h) check new calibrations, selection strategies, cutflow, new systematics, overlap removal procedure

I value this approach, but, does not scale indefinitely

- **Others**:
  - Centrally-produced ntuples with most calibrations, object selection, overlap removal applied
  - Turn around:
    - Nominal histograms: ~ mins for histograms from nominal ntuple
    - Nominal ntuple: ~ days (Grid) + ~ days for testing
    - 2 – 3 weeks for all ntuples with all systematics

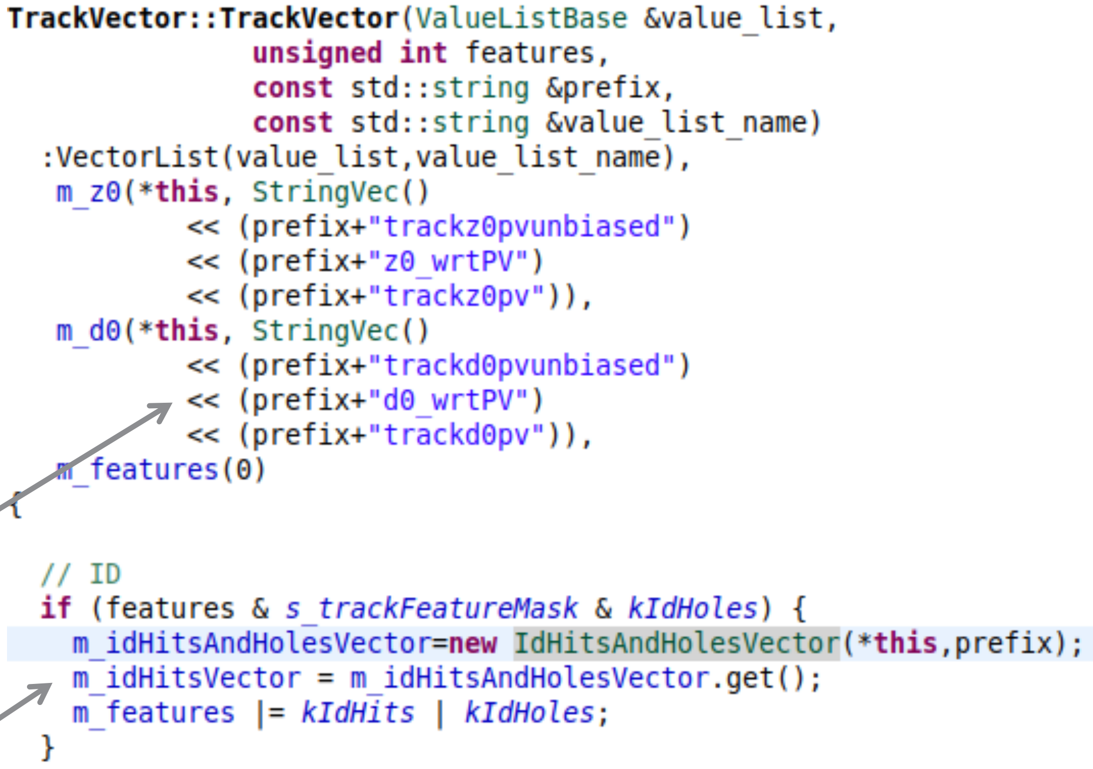
Group NTuples and Data Duplication

- From talking to people:
  I have the feeling that people produce 1 ntuple per syst. uncertainty
- > 90% of branches are copied
- Friend trees or decorators are the solution
- Possible reasons:
  - Frameworks don't provide easy-enough interface to do better
  - People want a simple ntuple to "just make histograms"
  - Everything must be super flat, preferably no objects
- Should we try to help here? This problem has been solved many times …

Reusability & Moving Targets

- Branch names changed often
  - Renamed, new software release
  - Read ntuples from different groups
  - Overkill in use for H→bb, Z → μμ, Z → bb analysis

- Overkill solution:
  - Try different names until success
  - Switch on/off branches based on features required by tools (faster processing)
  - Automatic conversions (to higher precision types) in case branch type changes



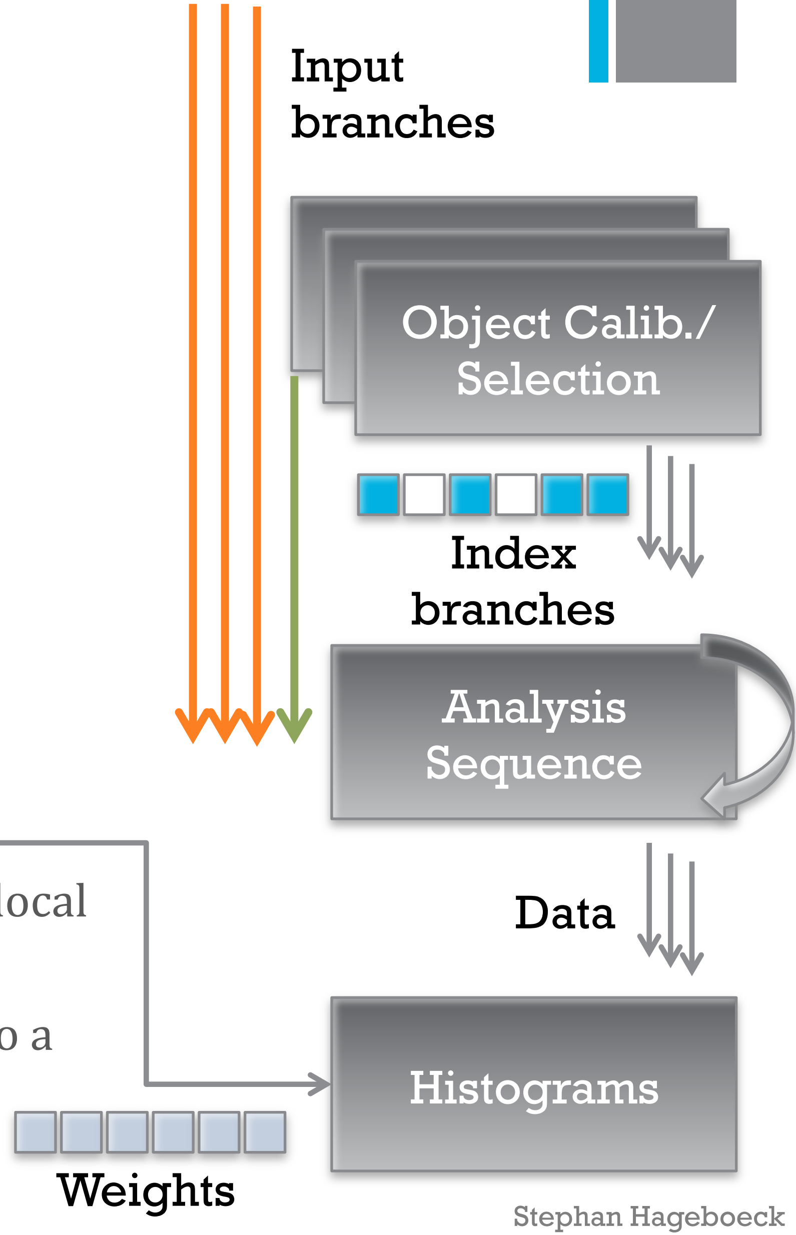
```
TrackVector::TrackVector(ValueListBase &value_list,
                  unsigned int features,
                  const std::string &prefix,
                  const std::string &value_list_name)
  :VectorList(value_list,value_list_name),
   m_z0(*this, StringVec()
            << (prefix+"trackz0pvunbiased")
            << (prefix+"z0_wrtPV")
            << (prefix+"trackz0pv")),
   m_d0(*this, StringVec()
            << (prefix+"trackd0pvunbiased")
            << (prefix+"d0_wrtPV")
            << (prefix+"trackd0pv")),
   m_features(0)
{

  // ID
  if (features & s_trackFeatureMask & kIdHoles) {
    m_idHitsAndHolesVector=new IdHitsAndHolesVector(*this,prefix);
    m_idHitsVector = m_idHitsAndHolesVector.get();
    m_features |= kIdHits | kIdHoles;
  }
```

- Switching to new inputs took ~ hours / 1 – 2 days (only once)

- Backward compatible / flexible

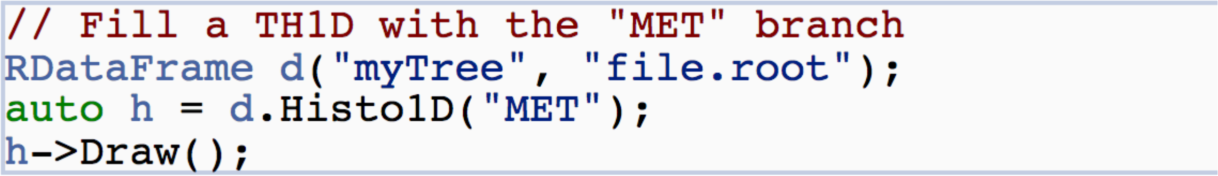
Evaluating Systematic Uncertainties

- Object uncertainties:
  - Book different object calibration/selection sequences
  - Write index branches + decorators for each uncertainty
  - Run the analysis sequence
    - Uncertainty-agnostic
    - Easy to configure/program
- Weight (=probability) uncertainties:
  - Run weight calculation sequence, i.e. retrieve probabilities from ATLAS + analysis-local tools
  - Each uncertainty provider adds one element to a vector of weights + a vector of uncertainty names



Missing: A Histogram Categoriser

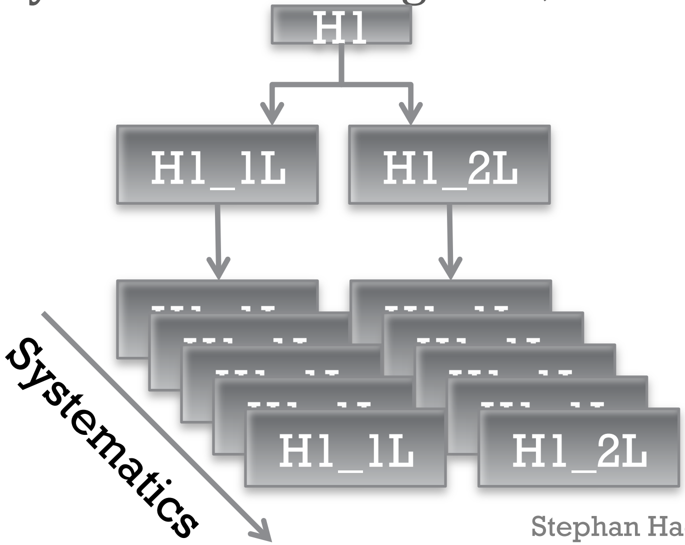
- Or: Why



```
// Fill a TH1D with the "MET" branch
RDataFrame d("myTree", "file.root");
auto h = d.Histo1D("MET");
h->Draw();
```

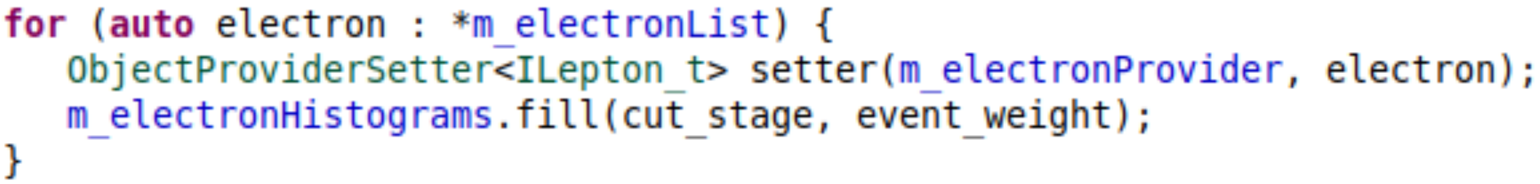
doesn’t cut it

- An analysis module should be able to fill a set of standard histograms
  - Debugging, investigations, cross-checks, understand what’s going on, results
- Don’t want to book (and configure) manually the set of histograms, and manage them
  - Before cut C, after cut D
  - For lepton pT, eta, phi, E
  - For electron collection A, B, C
  - For systematic uncertainty XXX
  - For category YYYY



Missing: A Histogram Categoriser

- Overkill solution: “Histogram List”
- initialise():
  Book histogram list that can take any number of variables from objects provided during execute(), creates histograms for all of them
- execute():



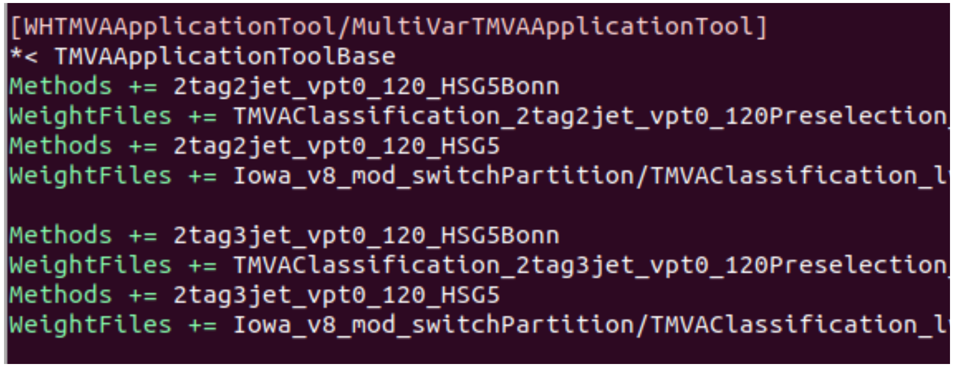
```
for (auto electron : *m_electronList) {
   ObjectProviderSetter<ILepton_t> setter(m_electronProvider, electron);
   m_electronHistograms.fill(cut_stage, event_weight);
}
```

  - One call fills various histograms using the ILepton_t interface (configurable)
  - Automatically categorises into systematics, analysis categories, cut stages ....
  - Automatically creates folder structure + name pre- and suffixes
- See a bit more code in backup

Missing: Multi-MVA Inference Tool

- Often need to test multiple classifiers trained with different configurations
  - Order or variables different
  - Different sets of variables in use
  - Other machine-learning toolkit
  - Model from different group (i.e. different naming)

- Multi-MVA inference tool:
  - Parse (TMVA / xgboost / ...) configs, extract variables needed
  - **Request variables from Overkill** + Regex-Match to category names
  - If not found: Ask user to provide mapping from e.g. `jet0_pT` →`pt_jet0`



```
[WHTMVAApplicationTool/MultiVarTMVAApplicationTool]
*< TMVAApplicationToolBase
Methods += 2tag2jet_vpt0_120_HSG5Bonn
WeightFiles += TMVAClassification_2tag2jet_vpt0_120Preselection
Methods += 2tag2jet_vpt0_120_HSG5
WeightFiles += Iowa_v8_mod_switchPartition/TMVAClassification_l

Methods += 2tag3jet_vpt0_120_HSG5Bonn
WeightFiles += TMVAClassification_2tag3jet_vpt0_120Preselection
Methods += 2tag3jet_vpt0_120_HSG5
WeightFiles += Iowa_v8_mod_switchPartition/TMVAClassification_l
```

The game changer:

- The framework automatically provides / maps variables that tools/MVAs require
- No configuration, coding necessary

Reproduce Analysis?

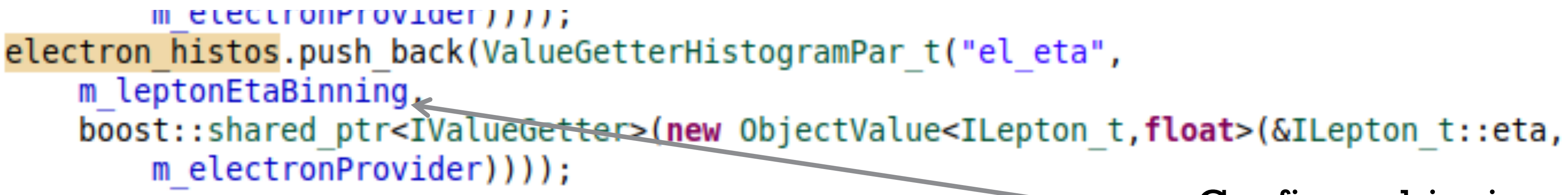
- The short answer: Possible, but not really
- Longer:
  - RooFit workspaces and histograms archived
  - Code archived
  - Most of documentation in TWiki
  - But:
    - Don’t have the machines to run it (Containers being archived in the future)
    - No Monte Carlo / Data available (need to regenerate, takes forever)
    - Who knows how to run these steps?
      → .bash_history
- Would notebooks help?
  - Yes and no (see summary)

Summary

- I never understood the fuzz about dataframes, notebooks, “let’s get flat”
  - Heavy (ATLAS central) & medium lifting (group framework) use 90% (?) of CPU cycles
  - Keep this in mind for future software needs? This might be the bottleneck
- Notebooks, dataframes & Co are nice!
  - Think of Master/Bachelor students: Significantly lower the bar
  - **But**: Work only for the “simple” steps of the analysis
    Someone has to do the heavy lifting before …
- By ignoring “let’s get flat and super simple as fast as possible”, we (Bonn) contributed a lot to solving the difficult problems of new analyses

Missing: A Histogram Categoriser

- The Overkill solution: Book (once when analysis module initialised)

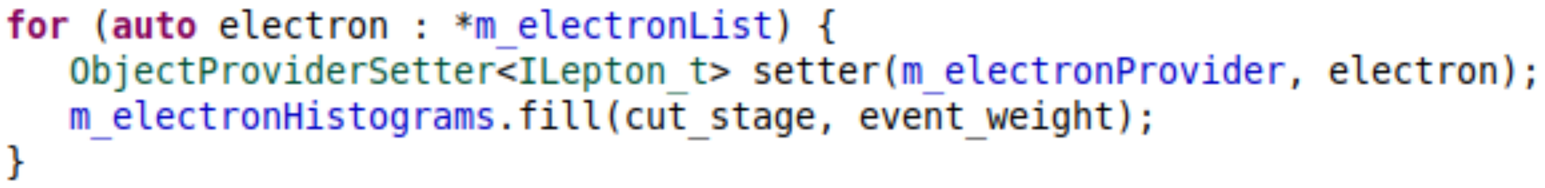


```
        m_electronProvider))));
electron_histos.push_back(ValueGetterHistogramPar_t("el_eta",
    m_leptonEtaBinning,
    boost::shared_ptr<IValueGetter>(new ObjectValue<ILepton_t,float>(&ILepton_t::eta,
        m_electronProvider))));
```

Configure binning from outside
Inherited for all modules

- Register automatically, label axes, book into folder

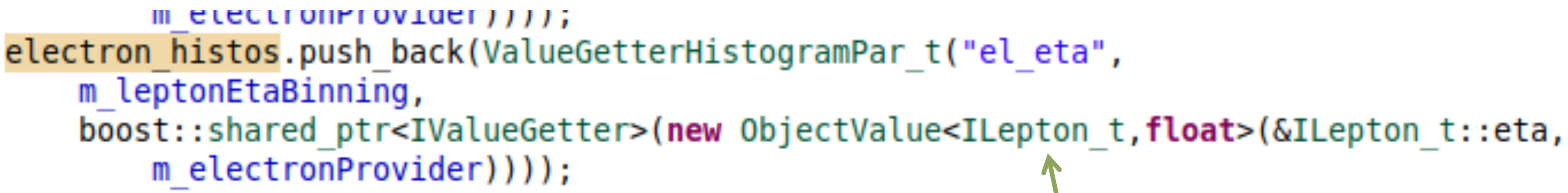
- Later (in various places):



```
for (auto electron : *m_electronList) {
   ObjectProviderSetter<ILepton_t> setter(m_electronProvider, electron);
   m_electronHistograms.fill(cut_stage, event_weight);
}
```

Missing: A Histogram Categoriser

- The Overkill solution: Book (once when analysis module initialised)

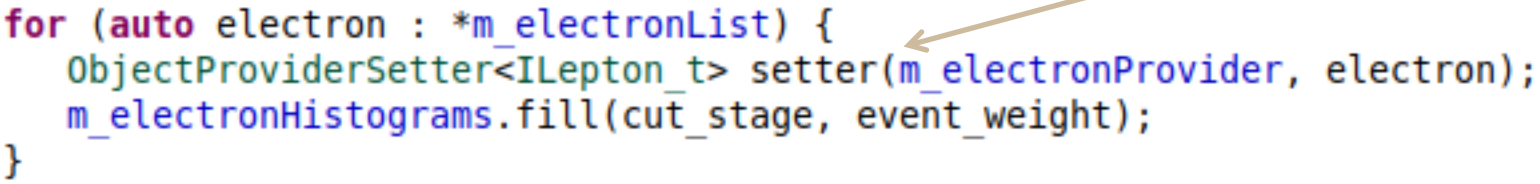


```
        m_electronProvider))));
electron_histos.push_back(ValueGetterHistogramPar_t("el_eta",
    m_leptonEtaBinning,
    boost::shared_ptr<IValueGetter>(new ObjectValue<ILepton_t,float>(&ILepton_t::eta,
        m_electronProvider))));
```

- Register automatically, label axes, book into folder

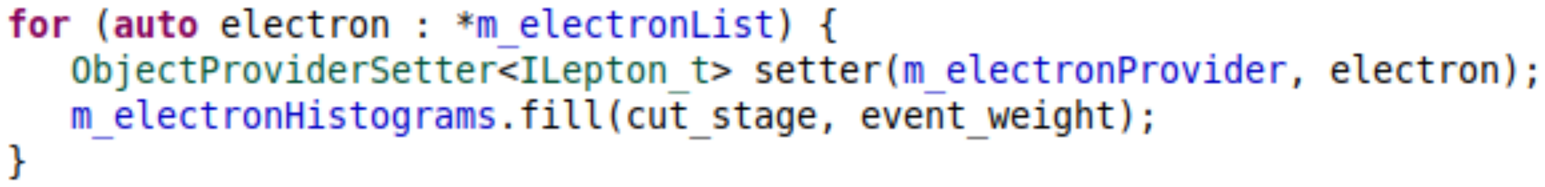
Read a value
from an object
stored in a provider

- Later (in various places):



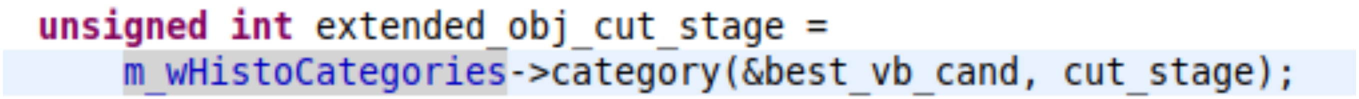
```
for (auto electron : *m_electronList) {
   ObjectProviderSetter<ILepton_t> setter(m_electronProvider, electron);
   m_electronHistograms.fill(cut_stage, event_weight);
}
```

Missing: A Histogram Categoriser



```
for (auto electron : *m_electronList) {
   ObjectProviderSetter<ILepton_t> setter(m_electronProvider, electron);
   m_electronHistograms.fill(cut_stage, event_weight);
}
```

- Benefits:
  - Can histogram any electron multiple times at any stage of the analysis with different weights
  - cut_stage is incremented between selection steps, and switches between histograms before/between/after selection steps
  - cut_stage can be expanded by an object/cut/systematic categorisation tool:



```
unsigned int extended_obj_cut_stage =
    m_wHistoCategories->category(&best_vb_cand, cut_stage);
```

- Result:
  - Book few histograms, get one for every cut stage, systematic, category …
  - Automatic sorting into folders / automatic name pre-/suffixes
  - Module does not have to know systematics / analysis categories