Institut
"Jožef Stefan"
Ljubljana, Slovenija

ATLAS EXPERIMENT

Experience With Using ROOT for Final ATLAS Data Analysis Steps

ROOT Users' Workshop,
September 12, 2018

Tadej Novak

Jožef Stefan Institute

Introduction

- ROOT is a very integral part of the majority of ATLAS analyses.
- ATLAS stores its data using xAOD format which can be read with pure ROOT.
- Most of analyses use `TTree`-based intermediate files:
  - using simplified object instances (mini-xAODs)
  - using flat ntuples (using only C++ STL types)
- Processing of those files is very frequent and should be fast and efficient.
- Most of the time is spent at this analysis stage where also final histograms are produced.

Disclaimers:

- I will present my personal views on ROOT.
- I haven't played with new ROOT 7 features yet.

Large Amount of Events

- Preselection applied at ntuple level but still keeping a lot of events.
- Multiple samples with different sizes that have to be processed individually.
- Parallel processing of the events is mandatory.
- Events are processed multiple times so it should be fast — less than 1 h.

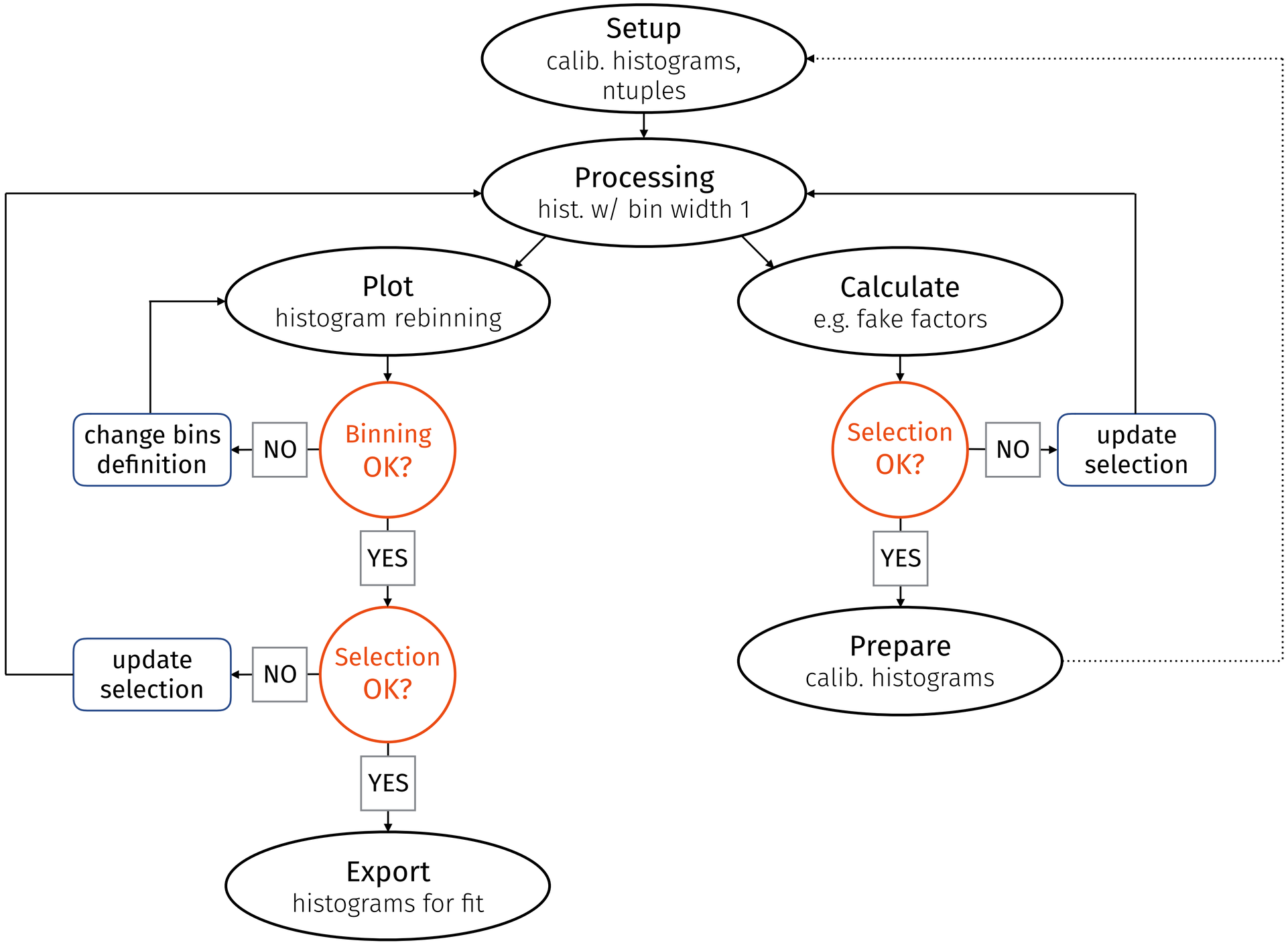
Approximate number of events when starting an analysis

	Data	MC
Background estimations	700M	150M
Signal region	100M	100M

My Analysis Framework Structure

- ROOT as a base (6.12), compiled with CMake:
  - Using ATLAS provided version from CVMFS for portability (compile once, run on different locations).
  - Needs to be compiled: better warnings for code, C++ interpreter uses random numbers for reference counting (slow on clusters).
- Python for simple configuration:
  - No need to compile each time you change a property.
  - Simple to code (dynamically typed).
- Shell scripts for running:
  - Simple creation of jobs and working environment.
- Parallel execution:
  - Runs locally, on LSF (lxplus) and Nordugrid ARC (my institute) based clusters.
  - Quickly extendable — defining few shell functions how to submit and monitor jobs.

Workflow

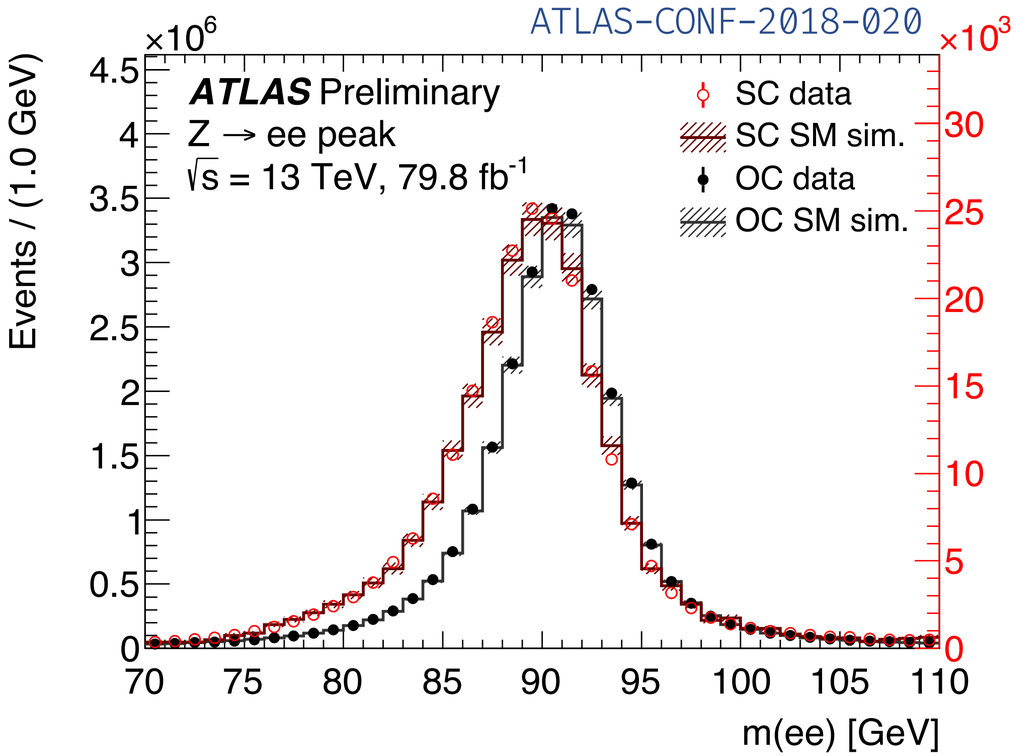

Setup
calib. histograms, ntuples
Processing
hist. w/ bin width 1
Plot
histogram rebinning
Calculate
e.g. fake factors
change bins definition
NO
Binning OK?
Selection OK?
NO
update selection
YES
YES
update selection
NO
Selection OK?
Prepare
calib. histograms
Export
histograms for fit

Requirements of an Analysis Framework (in context of ROOT)

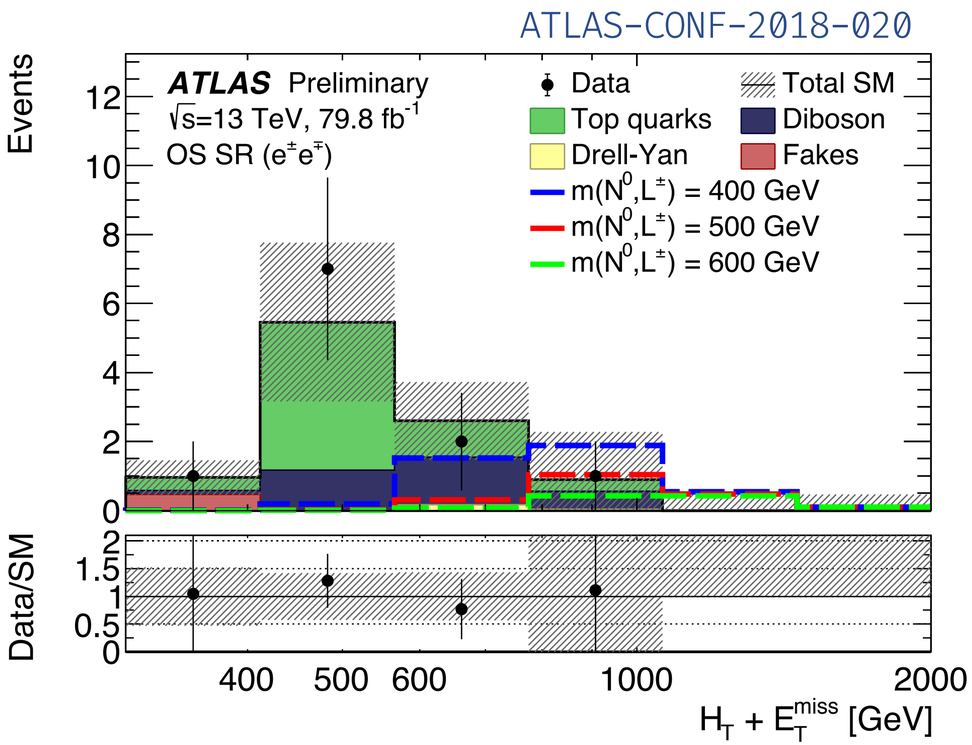
- Should not be limited on one specific platform — ROOT works on Linux, macOS and Windows.
- Needs to be parallelisable and flexible — any future ROOT parallelism options (e.g. `RDataFrame`) should be simply adaptable to different cluster options.
- Should be simple to configure — Python is a nice language for configuration and physicists are familiar with it. PyROOT is a nice bridge between C++ and Python but it takes a noticeable time to initialise for simple scripts.
- Running on full datasets should be fast, flexible rebinning should be possible:
  - Trees can be read efficiently but reading branches can be tricky.
  - Rebinning with non-equidistant bins is needed for TH2D (maybe even TH3D).
- Simple to code — e.g. RooFit is not directly compatible with core ROOT object (see the talk by Attila and Will on TRooFit).

Making Plots in ROOT

- It is not simple to make a nice looking plot in ROOT.
- Margins, positions and sizes should not depend on the pad/canvas size:
  - Impossible to create a global TStyle.
  - It is hard to make nice ratio plots (overlapping pads, different size values).
- The width of a legend affects the width of the plot representation.
- It is not possible to get the width of a label (TLatex) to avoid overlap.



Conclusions



- ROOT is a nice tool that can be used for particle physics analysis.
- Some aspects look a bit dated and make some parts over complicated (e.g. plotting).
- Documentation is good, contains examples (but hard to link classes with components/ tutorials directly — ROOT-8431).
- My ROOT-based code successfully used for published results and plots.
- Looking forward to planned modernisation and new features in ROOT 7!