

LHADA: A PROGRESS REPORT

16-18 Nov 2016

CERN



MAIN GOALS OF THE LHADA FILE AS I UNDERSTAND THEM

- A stand-alone description of an LHC analysis that describes:
 - ✓ The analysis (reco) objects and how to get them from different inputs — internal experimental data objects, output of a detector simulator, truth level MC particles (this is the big debate; more on this later).
 - ✓ Cuts: How signal (and possibly control) regions are determined in terms of properties these objects and variables defined from these.
 - ✓ Results; i.e. expected and observed signal/background events in these regions. Limits based on these.
 - ✓ Wishlist: binned data, correlations, likelihood (we need a more concrete implementation to iron out the kinks)

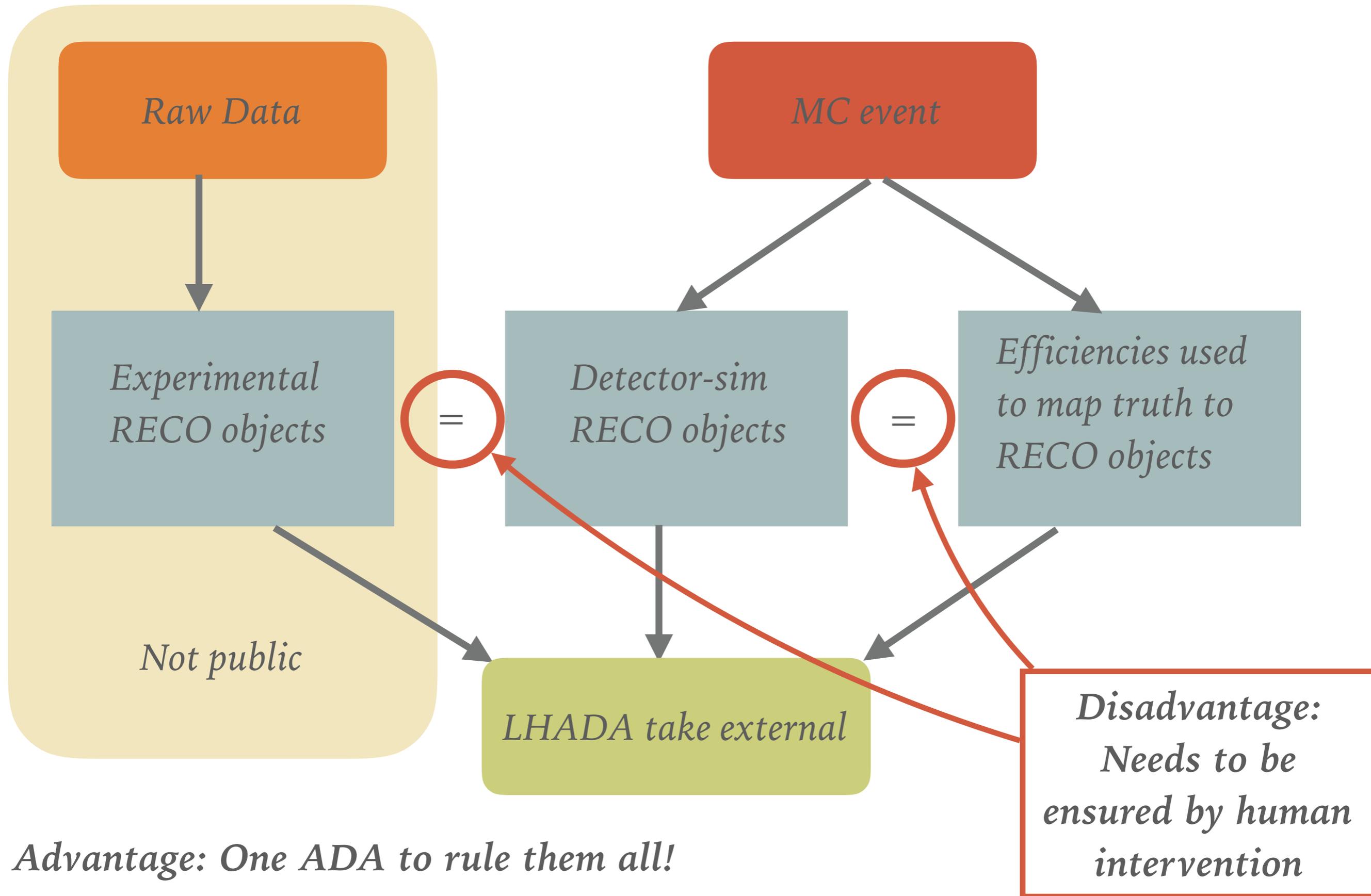
THE MAIN ACHIEVEMENTS IN THIS MEETING:

- We have three implementation ideas: RIVET, CheckMATE and CutLang
 - ★ We find it is easy to translate the LHADA proposal format into XML and JSON (with caveats, e.g. multiple arg tags)
 - ★ We find it is possible to write a realistic analysis in LHADA
 - ★ We find it is also possible to assemble the cuts in order as written in the `cut` block (implemented in both RIVET and CheckMATE)
 - ★ We decided that a “fundamental object” which we call LHAParticle is needed that can be used (and further generalised).
 - ★ The mode of operation of the keywords `take`, `apply` and `select` was clearly clarified.
- We need a “validator” in the parser that checks syntax and performs basic checks

WHAT CAN BE AUTOMATED?

- Writing code expressions for cuts
- Downloading the functions from the link and compiling them
- Getting the final (orthogonal?) signal regions by finding the leaves of the cutflow tree or based on results table. We decided list of signal regions should be one of the outputs during the parser's validation stage. However cutflow for all intermediate cuts also always written.
- Validation of the analysis thus assembled against published expected signal events for the benchmark using the generated cutflow tables.
- Calculation of likelihoods/limits based on observed and expected data in the signal regions. (Of course how close this is to the full likelihood depends on how much data the LHADA file contains. If it's only one number each for expected & observed, it won't be exact.)
- Possibility: statistical correlations between regions could be determined automatically by passing (background) events through each. Not computationally cheap but possible? (For experimentalists to answer in the future: is there a way to also include systematic correlations too?)

BEFORE LHADA BEGINS: CURRENT PICTURE

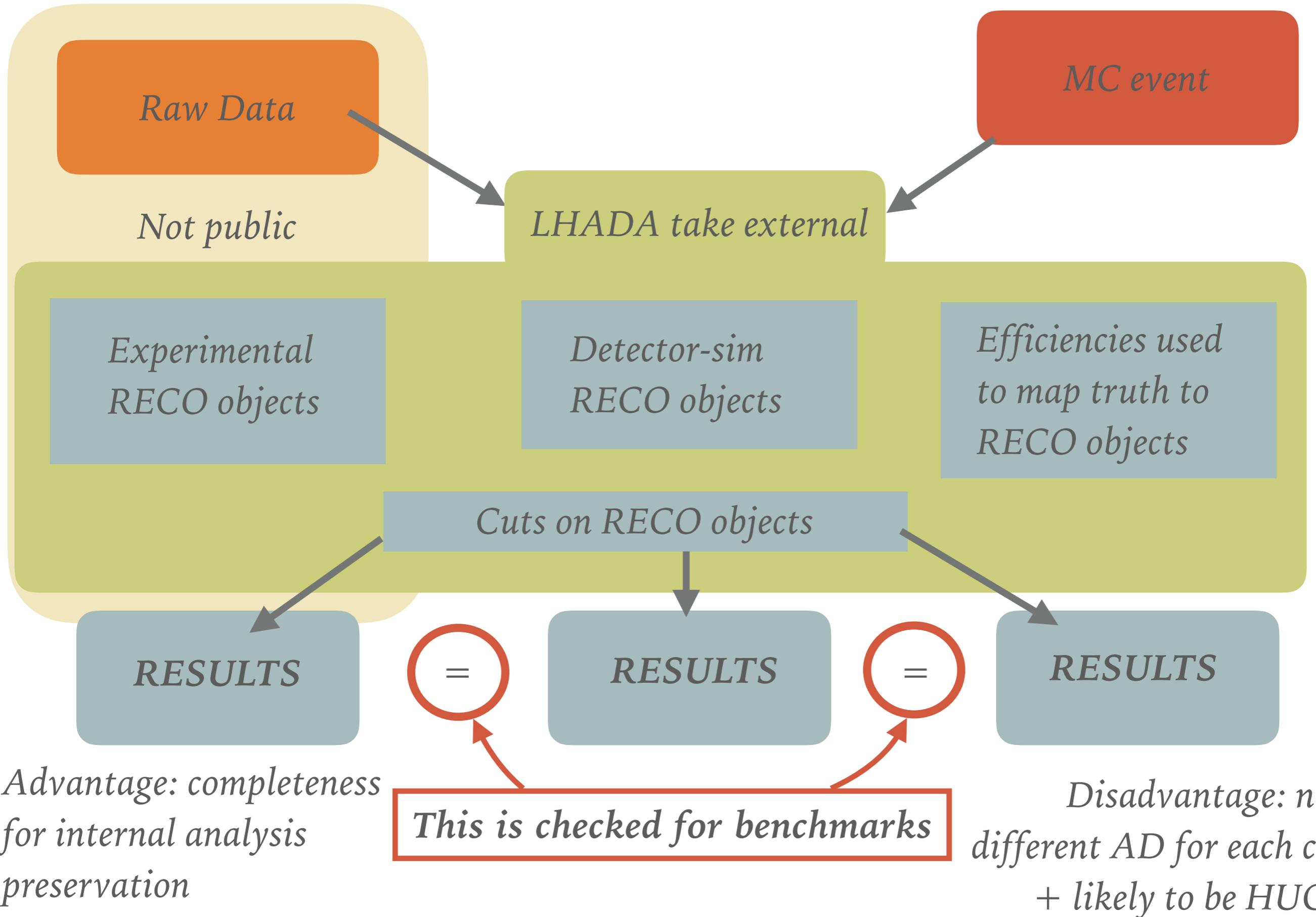


WHAT CAN NOT (NEVER?) BE AUTOMATED

► Defining the analysis objects.

- The **take external** refers to raw data/some semi-reconstructed object for experiments, output of detector simulators for recast tools, MC particles and efficiencies/smearing functions for RIVET).
- *Take inspiration from the accord for jet algorithms: input can be particles, calo towers, “subjets”. What matters is the final result of what we call a “jet” has the same kinematics irrespective of input.*
- The authors of the individual tools have to ensure correspondence between the experimental data object and simulated object. **WE NEED TO AGREE ON SOME OBJECTS TO START** (currently these are electrons, muons, photons, MET, jets)
- We can already think of more complex objects e.g. a hadronic top based on top tagging. There would be an “object library” for each category.
- We may want to add tracks and/or vertices if we want to use displaced vertices. Any others needed?

BEFORE LHADA BEGINS: LINES REDRAWN TO OTHER EXTREME



LHA PARTICLE OBJECT

- The LHAParticle object should contain at a minimum
 - ➔ a momentum 4-vector (actually better a TLorentzVector)
 - ➔ a PDG code (not necessarily used in experimental implementations)
 - ➔ Charge and spin of particle (also can be blank in experimental analysis)
- A certain amount of useful quantities can be made permanently made available based on the above (e.g. pT, eta, phi, anything else? — e.g. impact parameter). Also remember there may be a suitable ROOT object already or something easily incorporated into ROOT.
- On top of this, we want an extendible list of properties (simple in python: use dict, more complicated in C/C++) for things like b-tags, e/h, or any other extra information (essentially a name:value pair). We want two functions (or at least functionality):
 - “hasProperty(name)” which makes sure the property is indeed accessible
 - “getPropertyVal(name)” which gets the value of the property
- Whenever an analysis uses a new property, a version of the LHAParticle (like a child class) would have to be coded. But there is already a lot that can be done with the minimum,

WHAT NEXT?

- Implement a real analysis (WE NEED AN EXPERIMENTALIST TO TRY)
- Implement a non-SUSY analysis
- Decide on LHAParticle, implement the addition of properties
- Decide what level we want to start LHADA from (what objects we need for take external)
- Talk some more about what data should be in and how it can be used.