Rivet for BSM search analyses

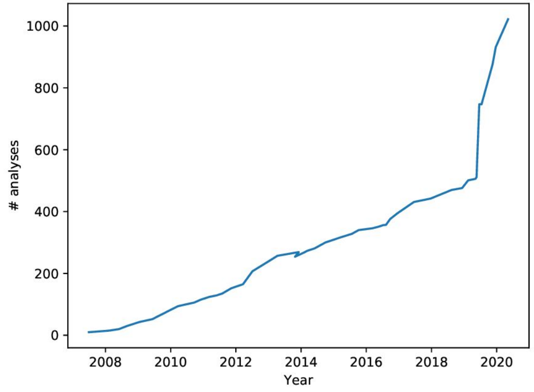
Preserving logic & detector performance

Andy Buckley, University of Glasgow
for the Rivet team

CMS MC and Physics Tool tutorials
30 October 2020

Rivet and BSM

- **Rivet v3 from June 2019 to current 3.1.2, July 2020**
  - automatic MC systematics multiweight handling
  - heavy ion machinery, analysis parameters, …
  - Docker images for rivet and rivet+$generator
  - **and: BSM search-logic tools and detector emulation**
- **Why BSM analysis preservation?**
  - 10 years of null searches: statistically in a time of diminishing returns = time to "save our progress", engage with pheno
  - likely that impact won't be purely through your experimental paper, but data and code preserved for community re-use
- **And why Rivet?**
  - need to consider more complex models = fast equivalent code
  - expertise/support established via long SM measurement experience



Rivet analysis coverage (searches only)

Rivet analyses exist for 54/1068 papers = 5%. 12 priority analyses required.

Total number of Inspire papers scanned = 2633, at 2020-07-02

Breakdown by identified experiment (in development):

Key	ALICE	ATLAS	CMS	LHCb	Forward	HERA	$e^+e^-(\geq 12$ GeV)
Rivet wanted (total):	4	152	207	32	0	58	35
Rivet REALLY wanted:	0	2	10	0	0	0	0
Rivet provided:	0/4 = 0%	28/180 = 16%	9/216 = 4%	0/32 = 0%	0	0/58 = 0%	10/45 = 22%

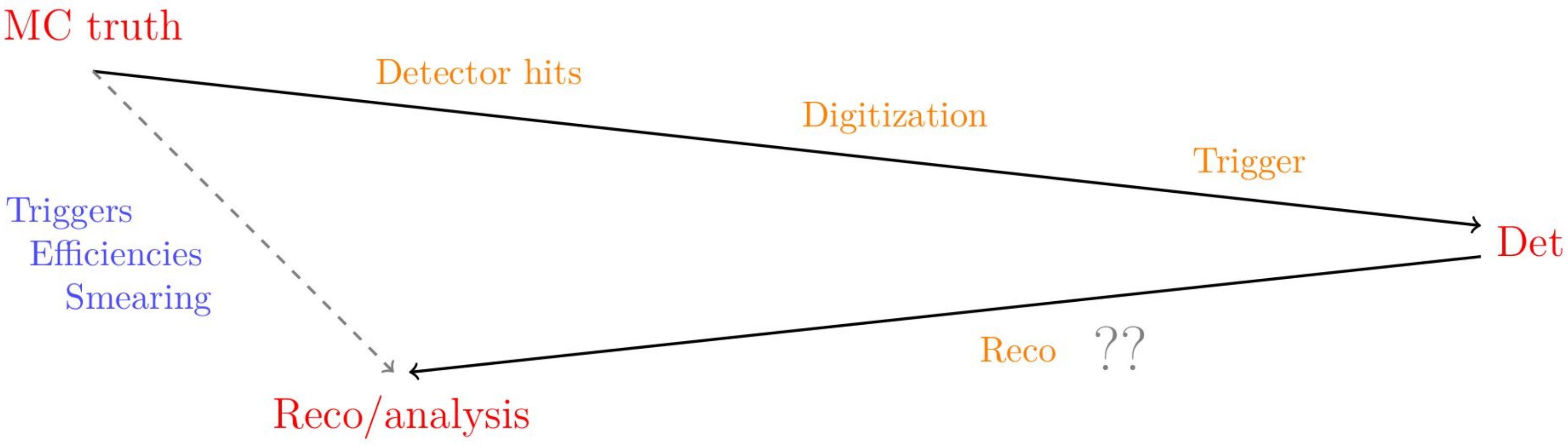
Search-recasting: general approach

- **Follow the experimental procedure as closely as possible**
  - as for measurements, avoid digging in the event record to get a more faithful representation
- **But you can avoid some details since truth MC and signal-only**
  - Definitely things like vertexing (unless recasting LLP searches)
  - Pile-up corrections are usually skippable — but jet grooming may be required
  - Lepton and photon isolation can often be replaced by a "promptness" requirement
  - Various details in isolation/OR process may be replaceable
    (by efficiency numbers/functions or other shortcuts like directness/promptness)
- **Output format?**
  - for now we mostly report via YODA histograms or lists of counters
    — we're extending these to be more suitable
  - really needs to match HepData content

Search-recasting: detector emulation

Nearly all search analyses are at reco level: detector-specific. Time-investment in unfolding not worthwhile: dilutes sensitivity unless full correlations given, etc.

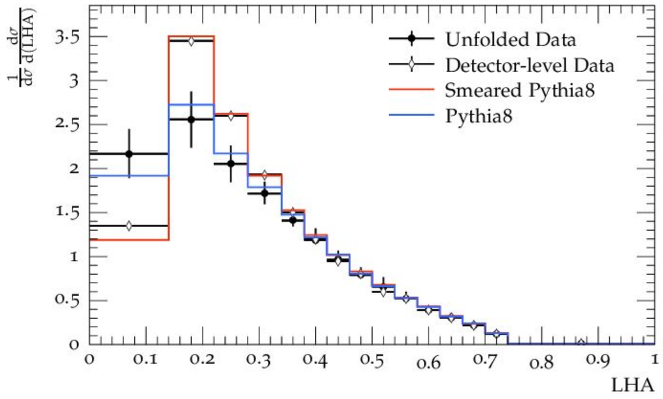
Re-interpretation is limited, unless an accurate detector model is given. *How* accurate?



Not as much as you might think: "explicit" fast sims don't necessarily help, smearing approaches go a long way. Especially if specific to the analysis phase-space

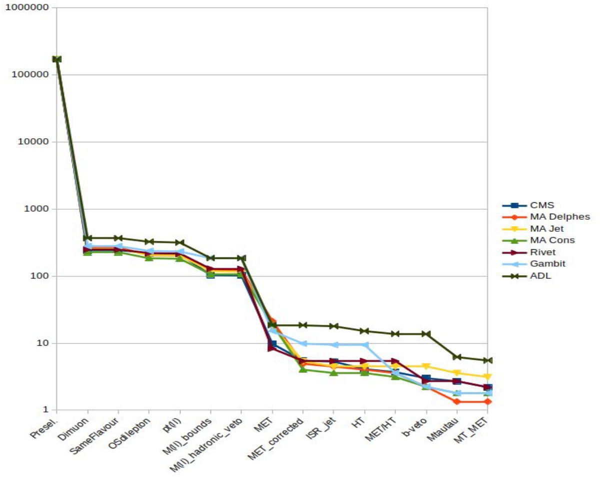
Search-recasting tools: detector emulation

- **Detector smearing system:**
  - developed based on Gambit experience
  - key features cf. Delphes, but more flexible & more analysis-specific
  - Paper: https://arxiv.org/abs/1910.01637 (including "tuned" *jet-substructure smearing*)



- Same speed as Delphes via HepMC
- Coded into analysis logic: unified treatment

- Included in Les Houches 2019 (soft-lepton) cutflow comparisons and global-fit tests:
  Performance very good!



Search-recasting: more tools

- **Container and isolation utilities**
  - large suite of tools for “functional” transformations, enumeration, and slicing of physics-object lists
  - physics-object filtering tools and isolation/OR helpers
- **Cut-flow monitoring**
  - cut-flows are an essential aspect of validating reinterpretation-analysis faithfulness
  - but a serious pain to have to maintain in parallel
  - Rivet’s version integrates cut-flows with analysis flow-control statements
- **Not finished yet…**
  - still open areas: integrated jet grooming, automatic jet substructure smearing, plottable cut-flows, …
  - use it, and we’ll prioritise requests!

Hands-on exercise setup

- Everything based on Rivet+Pythia8 Docker;

  more general models via MG5 were too slow for live use (and I ran out of prep time!)
  so we'll just do some generic search logic rather than a "real" analysis today

- Get the Rivet tutorial Docker image:

  ```
  docker pull hepstore/rivet-tutorial:3.1.2
  ```

- Enter the container, with a path to your laptop filesystem at /host:

  ```
  docker run -it --rm -v $PWD:/host hepstore/rivet-tutorial:3.1.2
  $ rivet -h
  ```

- Create a dummy analysis code to work on:

  ```
  $ rivet-mkanalysis MYSEARCH
  ```

Filtering and overlap-removal tools

- **Writing loops (in loops in loops) is tedious. We're here to help!**
- First, filtering a C++ vector (e.g. to apply a new cut) is not easy: calling erase in a loop invalidates iterators! Filter functions do it efficiently:

```
ifilter_select(myparticles, Cuts::pT > 100*GeV)
```

- C++ allows passing functions as arguments, so we can make more complex, *stateful* filtering decisions via standard or custom functors (including lambdas):

```
ifilter_select(myjets, hasBTag(Cuts::pT > 5*GeV));    or
filter_discard(electrons, deltaRLess(myjet, 0.2));
filter_select(myjets, [](const Jet& j){ return j.particles(Cuts::pT > 5*GeV).size() > 3;});
```

- And even higher-level: cuts via comparisons to whole sets of objects:

```
idiscardIfAnyDeltaRLess(myjets, isoleptons, 0.4);
```

- More helper functions for manipulating physics-object lists:

```
ht = sum(jets, Kin::pT, 0.0);    or    if (all(leptons, pTGtr(50*GeV)))    or …
```

Exercise 1: object selection

- In your MYSEARCH.cc file, get particle-level truth jets, electrons, and muons
  - Choose |eta| < 4, p_T > 30 GeV for jets; |eta| < 2.5, p_T > 20 GeV for leptons
  - What particles do you forbid from being jet constituents?
    Do analysis papers always make this clear?!?
- The jet collection will also include at least the electrons (and their photon halo):
  - Remove any jets within 0.2 of an electron, discard any electrons < 0.4 from a remaining jet
  - Remove any muon < 0.4 from a jet with > 4 tracks
- Filter out the b-tagged jets within |eta| < 2.5
  - Should there be a kinematic cut on the tagging b-hadron? Is this reported in papers?
- What could you shortcut using PromptFinalState and NonPromptFinalState?
  How accurate is it?

Cut-flow monitoring

- Rivet provides the Cutflow type for a single weighted cut-flow, Cutflows for many.

```
#include "Rivet/Tools/Cutflow.hh"
Cutflow flow{"Sel", strings{"> 2 jets", "> 1 lep", "> 1 b-jet", "MET", “HT”}};
Cutflows _flows.addCutflow(flow);
```

- Cuts are defined by integer or string index. Fill many at a time if desired:

```
_flows.fillinit();  //< fill before any cuts
_flows.fill(1); _flows.fillnext(pT1 > 300*GeV);
_flows.fillnext({pT2 > 0.5*pT1, HT > 1*TeV, meff > 1.2*TeV});
```

- Flow fills return the final cut result, so can be embedded in control statements:

```
if (_flows["Sel"].filltail({nbjet == 3, aplanarity < 0.3})) _srcounter->fill();
```

- Print out a nice string repr at the end: `MSG_INFO(_flows);`
- Plotting and full (multi)weight integration… a nice project!

Exercise 2: event selection

- Create a set of 3 cut-flows, for 1, 2 and >2 lepton events
- Require as a common selection that your events have:
  - At least 3 QCD jets
  - At least 2 b-jets with pT > 60 GeV
  - At least 1 isolated lepton
  - HT > 800 GeV
  - MET > 200 GeV

  Fill these selection requirements into your cut flows
- Finally apply separate lepton-multiplicity cuts for each signal region, and fill an event-yield Counter in each
- Generate gluino → t t χ events with Pythia and process with your analysis:

```
$ pythia8-main93 -f gg_g1500_chi100_g-ttchi.cmnd -n 1000
$ rivet --pwd -a MYSEARCH pythia.hepmc
```

Using detector emulation

- Detector smearing & efficiencies are implemented via wrapper projections:

```
#include "Rivet/Projections/Smearing.hh"
SmearedParticles(electronfs, ELECTRON_EFF_CMS_RUN2);
SmearedJets(fastjets, JET_SMEAR_CMS_RUN2, JET_BTAG_EFFS(0.77, 1/6., 1/134.));
SmearedMET(met, MET_SMEAR_CMS_RUN2);
```

- These “standard” functions are taken from Delphes and reco performance papers: see Rivet/Tools/SmearingFunctions.hh. They are generic and incomplete! Much better is to implement the critical ones specific to your analysis, as named functions or lambdas

- Smearing and efficiency functions can be chained, to get specific effects or to apply multiple kinds of distortion. Generic smearing/eff-function helpers are found in Rivet/Tools/{ParticleBase,Particle,Jet}SmearingFunctions.hh

- There’s always room to improve… let us know!

Exercise 3: smearing functions

- Now we're going to apply some smearing & efficiency functions to emulate the reco-level nature of the analysis. The main effect here will be on lepton and b-tag efficiencies (and probably some p_T-cut migration)
- Use the "standard" CMS Run 2 jet smearing, and a b-tag efficiency tuple b=0.7, c=0.1, l=1/120
- For electrons, use standard smearing and a custom efficiency = 0.85 $(1-(eta/5)^2)$ (1 - 0.1 exp(10 - pT/2 GeV)). For muons use standard smearing and fixed 80% eff
- For MET, use the standard smearing
- Note that you will need to change the apply<T>(...) template types to more generic ones: FinalState ⟶ ParticleFinder, FastJets ⟶ JetFinder, MissingMomentum ⟶ METFinder
- What are the effects on yields & cut-flows?
  Try adding -lProjection.SmearedParticles=DEBUG . Maybe useful: yodals -v Rivet.yoda

Exercise 4: what needs to be published?

- As a final exercise, let's see what it's like to implement an analysis "from outside", by looking in a couple of recent papers
- ATLAS RPV b-jets: https://inspirehep.net/literature/1821239
  - Can you find reference cut-flows and similar information?
  - Are the tight leptons and lepton overlap-removal needed?
  - What signal regions are usable?
  - How exactly can we make the relevant MC signal?
- CMS bottom-type VLQs: https://inspirehep.net/literature/1812970
  - Where are the cut-flows, yield data, and MC model info?
  - does Njet mean before or after overlap removal between the AKT4 and AKT8 jets?
  - if 2 AKT4 jets overlap with one AKT8, are those specific AKT4s "forced" to be Z/H candidates?
  - what are the target mean and sigma values in the chi2_{mod}?
  - what are the event overlaps & syst correlations between Njet and decay-assumption bins?

Summary

- Rivet is a well-established toolkit for measurement preservation, and has a strong feature set for BSM direct searches
- Emphasis on clarity without sacrificing accuracy: detailed control of isolation/OR, analysis-specific smearing, etc.
- Preserving these searches in a fast, clear, and accurate form is more important than ever, as stat gains dwindle and simplified models are no longer sufficient
- So use it, submit feature requests (and merge requests, thanks!), and we'll support & develop accordingly!
- New contributors are very welcome! BSM development could be a 3-4 month (remote) MCnet studentship...