MC Event Generator Tutorial

Christian Gütschow

RAL Advanced Graduate Lectures

17 June 2026

Setup: Docker images and containers

- We will be running event generators via Docker
  - Like a virtual Linux machine that you run inside your own PC
  - VM image files are $\mathcal{O}$(1 GB): download these in advance!
- Containers and volume binding
  - You can run an image multiple times: each copy is a container
  - By default the data from each container stays on your machine. . . this eats a lot of disk space!
  - Use `-rm` to make it auto-delete, or periodically `docker system prune`
  - To make it easy to get data in and out of your container, make a "portal" directory:
    `-v /some/host/dir:/some/container/dir`
- Let's start with the MadGraph+Pythia+Rivet image:

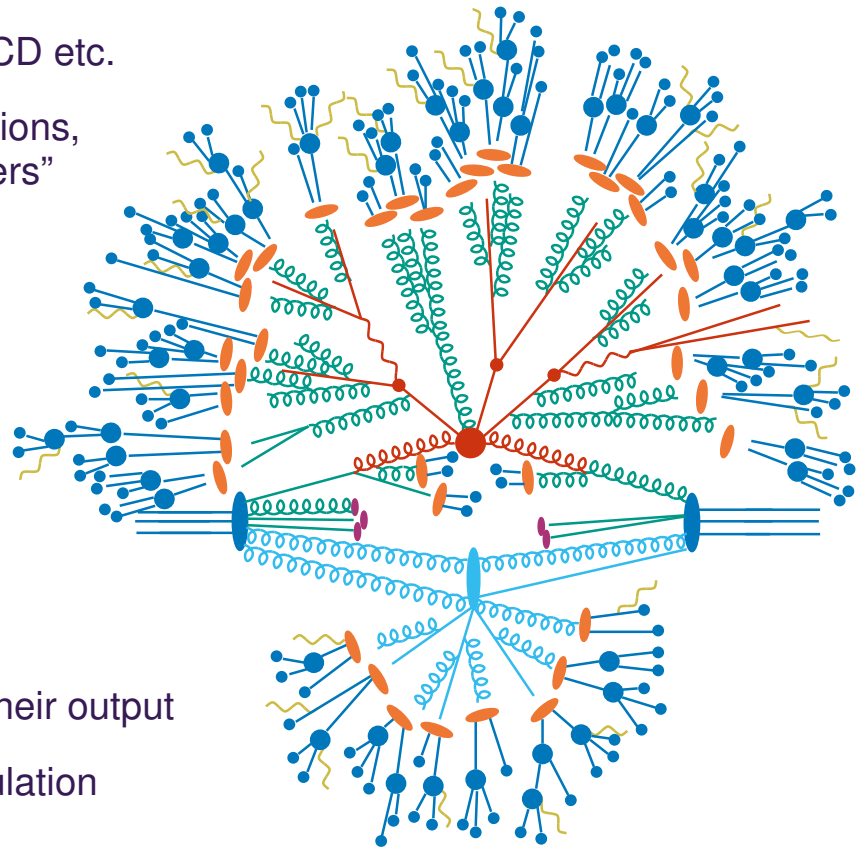
```
> docker pull hepstore/rivet-tutorial
> docker run -it --rm -v $PWD:/host hepstore/rivet-tutorial
```

- Test the tools:

```
> rivet -h
> pythia8-main144 -h
```

MC generation

- MC generation: where theory meets experiment
  - The fundamental *pp* collision, without a surrounding detector
- Components of a fully exclusive MCEG chain
  - QFT matrix element sampling at fixed order in QCD etc.
  - Dressed with approximate collinear splitting functions, iterated in factorised Markov-chain "parton showers"
  - FS parton evolution terminated at $Q \sim 1$ GeV: phenomenological hadronisation modelling. Mixed with MPI modelling.
  - Finally particle decays, and other niceties.
- Today:
  - hands-on tutorial with Pythia8 and MadGraph5
  - for background principles see the lecture slides
  - introduction to running generators and studying their output
  - generation biasing for efficient phase-space population
  - ME/PS merged generation with extra ME jets
  - ~~Writing a Rivet MC-analysis code~~
  - ~~BSM model configuration and generation~~



Generator basics

- First, get your Pythia Docker container started:

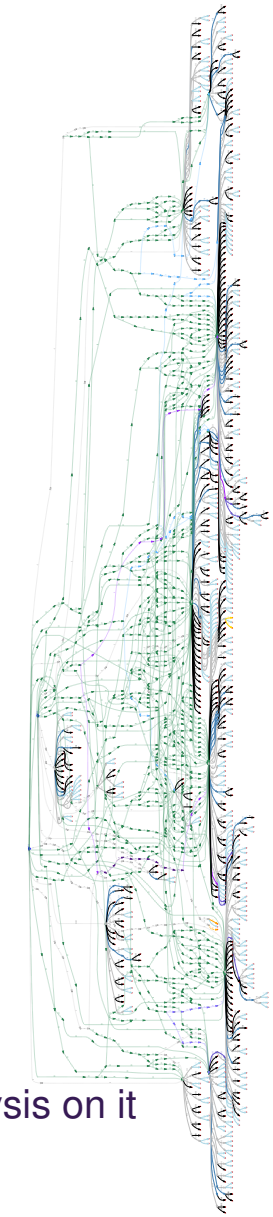
```
> docker pull hepstore/rivet-tutorial
> docker run -it --rm -v $PWD:/host hepstore/rivet-tutorial
```

- Pythia8: shower-hadronisation generator with many LO processes built-in
  - Pythia 8.3 docs: [manual]
  - We'll use the "main93" example interface. Open a blank command file:
    `> nano py8-top.cmnd`
  - Add the lines:

```
Beams:eCM = 13000
Top:all = on
Main:writeHepMC = on
```

  - And run: `> pythia8-main144 -c py8-top.cmnd -o TOP -n 100`

(Save and) exit `nano` with `CTRL+X`, then `Y/N`.
Exit `vi` with `:q!`, save and exit with `ZZ`.



- Examine the output:

```
> less TOP.hepmc
> rivet -a MC_FSPARTICLES TOP.hepmc -H TOP.yoda ← Run a basic physics analysis on it
> less TOP.yoda # yodals -v TOP.yoda ← View the histogram data
> rivet-mkhtml TOP.yoda -o /host/rivet-plots-top ← plot the histogram data
> firefox rivet-plots-top/index.html ← point your (host-system) web browser at it
```

More statistics = no more event files

- The HepMC ASCII files are very large!
  - They waste disk space, and CPU due to the writing/re-reading time
  - Useful for debugging though!
- Better that we pass the events to Rivet in memory instead
  - `> nano py8-top.cmnd`
  - And change to:

```
Beams:eCM = 13000
Top:all = on
Init:plugins = {libpythia8rivet.so::RivetHooks}
Rivet:fileName = TOP.yoda
Rivet:analyses = {MC_TTBAR,MC_JETS,MC_FSPARTICLES,MC_ELECTRONS,MC_MUONS}
```

  - `> pythia8-main144 -c py8-top.cmnd -n 5000`
  - `> rivet-mkhtml TOP.yoda -o /host/rivet-plots-top`
- Inspect the output
  - Do the lepton distributions make sense?
  - How about the jets?
  - What happens to the statistics at high p_T?

Jet-event generation

- Let's make some inclusive-jet events
  - In Pythia, this just means a $pp \to jj$ ME. Everything else comes from the PS, especially ISR.
  - It does remarkably well for that (thanks to a few tricks).
  - But mostly we use higher-order generators for the ME nowadays. Pythia8 is quick though!
- We start with the obvious configuration
  - `> nano py8-jets.cmnd`

```
Beams:eCM = 13000
HardQCD:all = on
PhaseSpace:pThatMin = 10
Init:plugins = {libpythia8rivet.so::RivetHooks}
Rivet:fileName = JETS.yoda
Rivet:analyses = MC_JETS
```

  - `> pythia8-main144 -c py8-jets.cmnd -n 6000`
    (There's a reason for this number of events!)
- View the output
  - `> rivet-mkhtml JETS.yoda -o /host/rivet-plots-jets`
  - And view: what's happened to the p_T tails and 3rd, 4th jet distributions?
  - We can improve this with ME phase-space slicing and/or enhancement

Jet-event slicing

- The statistics died off at high p_T
  - The unweighted events are asymptotically distributed like the physical $d\sigma/dp_T$
  - far too many low-p_T events for our needs! Rapidly drop below systematics threshold
  - Simple solution: stick together several runs in orthogonal slices of ME phase-space
- Three slices, the top-one open-ended
  - Add a max p_T^{hat} to `py8-jets.cmnd`:

```
PhaseSpace:pThatMin = 10
PhaseSpace:pThatMax = 50
> pythia8-main144 -c py8-jets.cmnd -n 2000
```

  - Then a min/max pair above that:

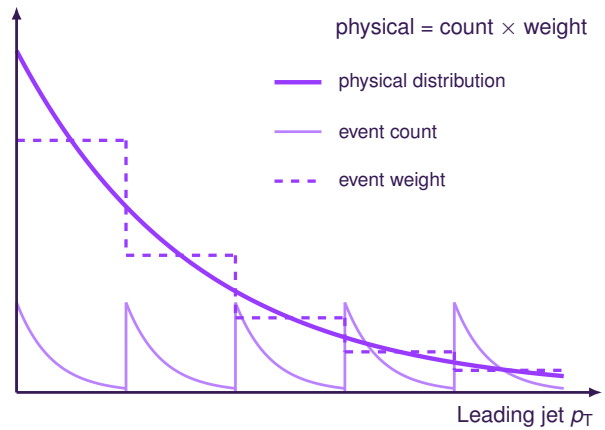
```
PhaseSpace:pThatMin = 50
PhaseSpace:pThatMax = 100
> pythia8-main144 -c py8-jets.cmnd -n 2000
```

  - And a final min-only:

```
PhaseSpace:pThatMin = 100
> pythia8-main144 -c py8-jets.cmnd -n 2000
```

  - Plot and study:

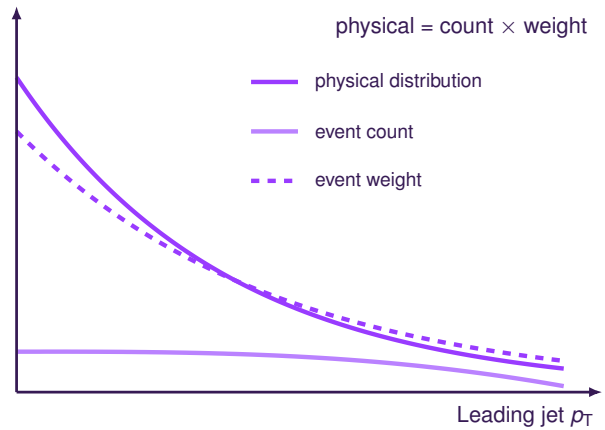
```
> rivet-merge JETS?.yoda -o JETS_SLICE.yoda
> rivet-mkhtml JETS{0,1,2}.yoda:LineStyle=dotted \
               JETS_SLICE.yoda:Sliced \
               -o /host/rivet-plots-jets
```



Don't forget to change the output file name in the config file!

Jet-event enhancement

- The statistics work better now, and the correctly cross-normalised sum is smooth
  - We still have falling stats in each slice, though: "sawtooth" statistical error
  - Can we "continuously slice"? Yes! Sample from $(p_{\mathrm{T}}^{\mathrm{hat}})^n \mathrm{d}\sigma/\mathrm{d}p_{\mathrm{T}}^{\mathrm{hat}}$, with weights $1/(p_{\mathrm{T}}^{\mathrm{hat}})^n$
  - Since LO $2 \to 2$ process, $p_{\mathrm{T}}^{\mathrm{hat}}$ is unambiguous



- Enhanced dijet generation
  - Enable biasing in `py8-jets.cmnd`:

```
PhaseSpace:pThatMin = 10
PhaseSpace:bias2Selection = on
> pythia8-main144 -c py8-jets.cmnd -n 3000
```

  - Pretty-printing of all methods:

```
> rivet-mkhtml JETS.yoda:Raw:LineColor=red \
               JETS{0,1,2}.yoda:LineColor=purple:LineStyle=dotted \
               JETS_SLICE.yoda:Slice:LineColor=green \
               JETS_ENH.yoda:Enh:LineColor=orange \
               -o /host/rivet-plots-jets
```

  - Study the output. Which is better at phase-space coverage? Compare the numbers of events generated

Weak boson plus jets production: Pythia

- W/Z+jets are the biggest and most CPU-consuming MC samples at the LHC
  - Followed by ttbar, diboson, ...
- The "classic" development lab for beyond-LO methods
  - Born process at $2 \to 1$ tree level; jets (and hence all Z p_T) is beyond LO
  - colour-singlet boson is unproblematic for QCD
  - vector boson: symmetry protection (i.e. small NLO corrections with respect to Higgs)
  - massive boson = naturally "anchored" scale choices: more stable than massless jets or photons
- First, let's make a Pythia8 version, then go to MadGraph5
  - `> nano py8-zmm.cmnd`

```
Beams:eCM = 13000
WeakBosonAndParton:qqbar2gmZg = on
WeakBosonAndParton:qg2gmZq = on
PhaseSpace:pThatMin = 20
23:onMode = off
23:onIfAny = 13
Init:plugins = {libpythia8rivet.so::RivetHooks}
Rivet:fileName = Py-ZJ.yoda
Rivet:analyses = {MC_ZJETS:LMODE=MU}
```

  - `> pythia8-main144 -c py8-zmm.cmnd -n 5000`

Weak boson plus jets production: MadGraph

- Use MadGraph via the same image
  - `> ./MG5_aMC/bin/mg5_aMC`
  - MG5 is a fixed-order ME generator that interfaces with Pythia's showers, decays, etc.
- Generate the lowest-order jet-multiplicity sample
  - ```
    > generate p p > mu+ mu- j
    > output PROC-ZJ
    > launch
    ```
    ...**do not** enable Pythia showering
    ...**do** edit run card, run with `<return>`
    ```
    > exit
    ```
  - `> cp -r PROC-ZJ /host/`
  - Look at diagrams and LHE event files:
    `> zless PROC-ZJ/Events/run_01/unweighted_events.lhe.gz`
  - ```
    > ln -s PROC-ZJ/Events/run_01/unweighted_events.lhe.gz .
    > nano py8-lhe.cmnd
    Beams:frameType = 4
    Beams:LHEF = unweighted_events.lhe.gz
    Init:plugins = {libpythia8rivet.so::RivetHooks}
    Rivet:fileName = MG-ZJ.yoda
    Rivet:analyses = {MC_ZJETS:LMODE=MU}
    ```
  - `> pythia8-main144 -c py8-lhe.cmnd -n 5000`

Feynman diagrams are generated automatically in the `SubProcesses` folders. The `> display diagrams` command also works, but not very effectively without graphics.

Weak boson plus jets production: LO multi-jet merging

- We can also make higher-order MEs (here just tree-level)
  - Make different-multiplicity processes, configure to merge and remove ME/PS overlaps

```
> ./MG5_aMC/bin/mg5_aMC
> generate p p > mu+ mu- j
> add process p p > mu+ mu- j j
> output PROC-ZJJ
> launch
```

... edit run card to disable `ickkw` and set `xqcut = 0.0`
... set `ptlund` cut to 30 GeV, run with `<return>`

```
> exit
```

Add a `[QCD]` suffix to generate a process at QCD NLO. Slow!!
➜ PS matching with MC@NLO; merging with FxFx.

- The PS makes the different multiplicities overlap in phase-space: have to avoid double-counting
- CKKW(L) and MLM procedures do this by phase-space weights or cuts

Weak boson plus jets production: MEPS & analysis

- Run Pythia on the MG5 LHE events
  - First, let's do it *wrong*, letting Pythia shower in the phase-space reserved for the ME:
    `> pythia8-main144 -c py8-lhe.cmnd -o MG-ZJJ-sum`
  - And now with the shower merging enabled:
    `> cp /usr/local/share/Pythia8/examples/main164ckkwl.cmnd py8-ckkwl.cmnd`
    Edit file: enable Rivet MC_JETS, set correct LHE and Rivet output filenames
    `> pythia8-main164 -c py8-ckkwl.cmnd` ← note different Pythia command! 164 knows merging
- Analyse the Rivet histograms
  - To plot the central values:
    `> rivet-mkhtml Py-ZJ.yoda MG-ZJJ-merge.yoda MG-ZJJ-sum.yoda`
  - MadGraph put lots of weights in the LHE file, incorporating e.g. scale and PDF variations
  - To plot them all, individually (hard to read, inflates the plot files):
    `> rivet-mkhtml -with-variations MG-ZJJ-{merge,sum}.yoda`
  - To plot with simple envelopes:
    `> rivet-mkhtml MG-ZJJ-{merge,sum}.yoda:"BandComponentEnv=.*"`
  - For more detailed band construction, see the [tutorial]

That's it!

- **Thanks for your time!**
- You now know how to run two popular LHC event generators at Born and merged/matched levels
- ~~And how to set up and run any UFO new-physics model~~
- ~~And write a new Rivet analysis~~
- These are superpowers – use them wisely!
- And the devil is in the details: black-box mode will only get you so far
- Sometimes it goes wrong, sometimes . . . it's complicated
- **Good luck!**

MCnet short-term studentships

- 3-month PhD placements with MC developer teams, in collaboration with the LHC Physics Centre at CERN
- What is it?
  - A chance to work closely with MC generator developers on a focused project.
  - Similar to past MCnet-funded short-term placements — but now open more broadly.
  - Not limited to MCnet member nodes — external projects and students welcome.
- Who can apply?
  - PhD students interested in event generators, theory/phenomenology, or experimental MC use.
  - Especially suited for students looking to deepen their expertise in LHC simulation tools.
- More info & available projects [click me]
  - New rounds of placements will be announced 2–3 times per year.
  - Feel free to reach out with ideas for new projects – external proposals encouraged!

Backup

Writing a custom MC analysis

- Just running pre-made Rivet analyses like MC_JETS would be very limiting
  - Now we will very briefly write our own analysis code
- Inside your container, create a new C++ source file
  - Rather than start from an empty file,
    we use `rivet-mkanalysis` to make a template code:
    ```
    > rivet-mkanalysis MY_TEST
    > nano MY_TEST.cc
    ```
  - Book a new histogram:
    ```
    book(_h["jjmass"], "jjmass", logspace(20, 1.0, 1000.0));
    ```
  - Require and get the two leading jets, add 4-vectors, histogram the mass:
    ```
    if (jets.size() < 2) vetoEvent;
    FourMomentum pjj = jets[0].mom() + jets[1].mom();
    _h["jjmass"]->fill(pjj.mass()/GeV);
    ```
  - Build, run and plot:
    ```
    > rivet-build MY_TEST.cc
    > rivet --pwd -a MY_TEST PROC-ZJJMERGED/.../*.hepmc.gz -H mytest.yoda
    > rivet-mkhtml mytest.yoda -o /host/rivet-plots-mytest
    ```

Documentation on the code & physics objects from [here] and [here].

BSM physics generation

- Pythia8 has several built-in models, e.g. Z', SUSY, XD resonances ...
  - Many are steered just via Pythia8 parameters – see the manual
  - SUSY in particular requires an SLHA file: use `hepstore/rivet-tutorial`
  - Set up a command file with
    ```
    SUSY:all = on
    SLHA:file = gg_g1500_chi100_g-ttchi.slha
    ```
  - Run and analyse

`hepstore/rivet-tutorial` is just the `rivet-pythia` Docker image with a few extra tutorial files in the work directory

- MG5 is really a generator generator: more flexible
  - can build new MEs for ∼any UFO physics model (as can Sherpa, Herwig)
  - For instance, a dark matter model:
    ```
    > import model DMsimp_s_spin1 --modelname
    > generate p p > xd xd~ j
    ```
    etc.
  - DM mass, coupling can be set in the "param card" = SLHA
  - Generate and analyse
  - More control can be imposed by fixing new-physics couplings at amplitude level e.g. `NP==1` or ME-squared level `NP^2==1`