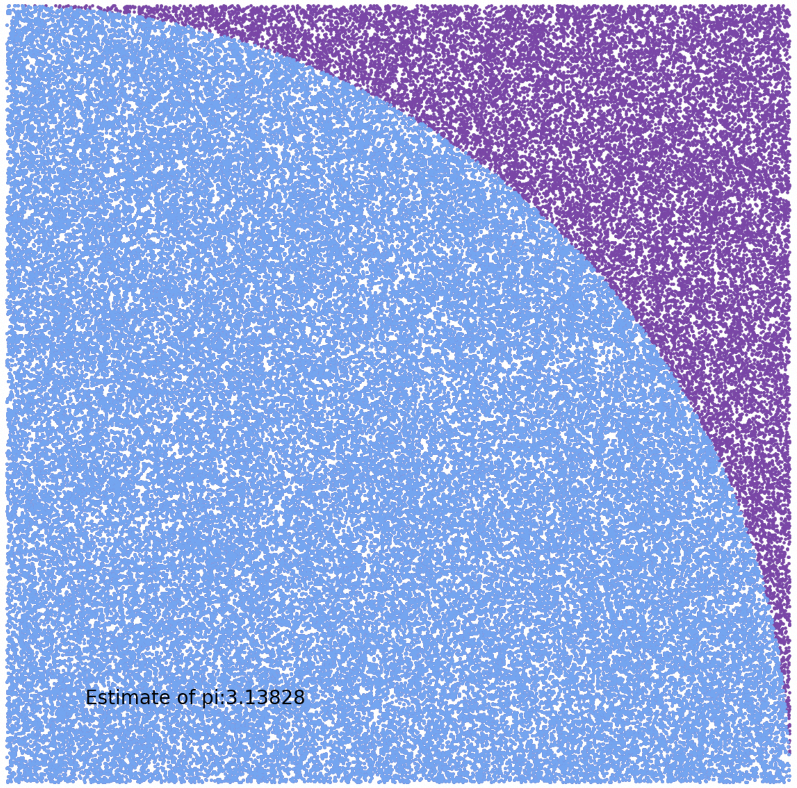


OVERVIEW OF MONTE CARLO IN ATLAS

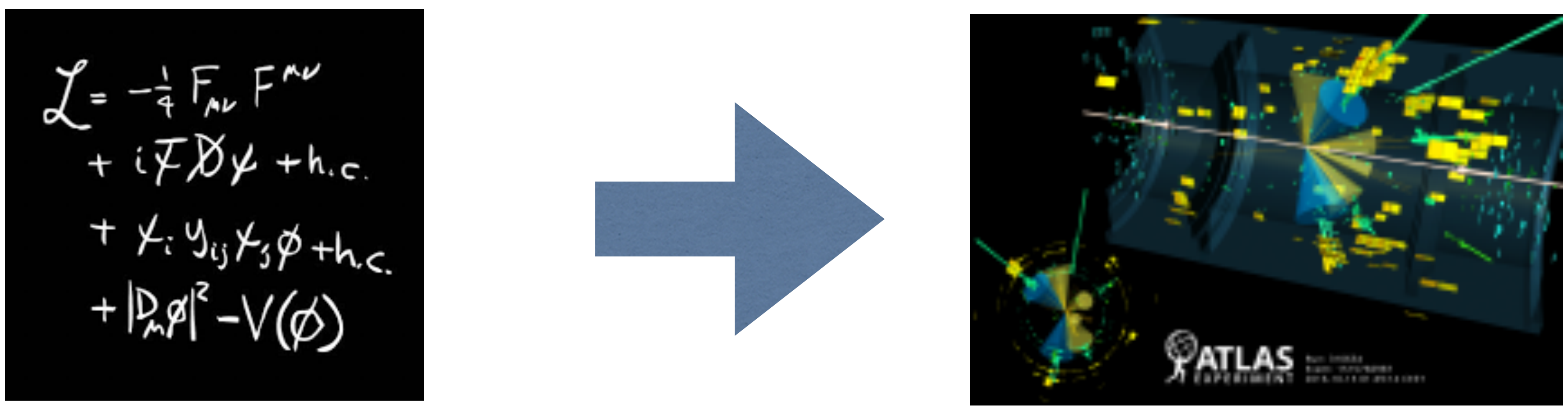
ATLAS SOFTWARE TUTORIAL, 25TH OCTOBER 2019

ANA CUETO

Why should I care about MC?

- Measurements and searches in ATLAS rely on Monte Carlo (e.g. for background estimation or correct for detector effects in the unfolding).

- They are designed to bridge the huge gap between the SM theory and simulated collision events.



- Learning more about what goes inside your MC implies a better understanding of the physics underneath everything we do in LHC experiments.

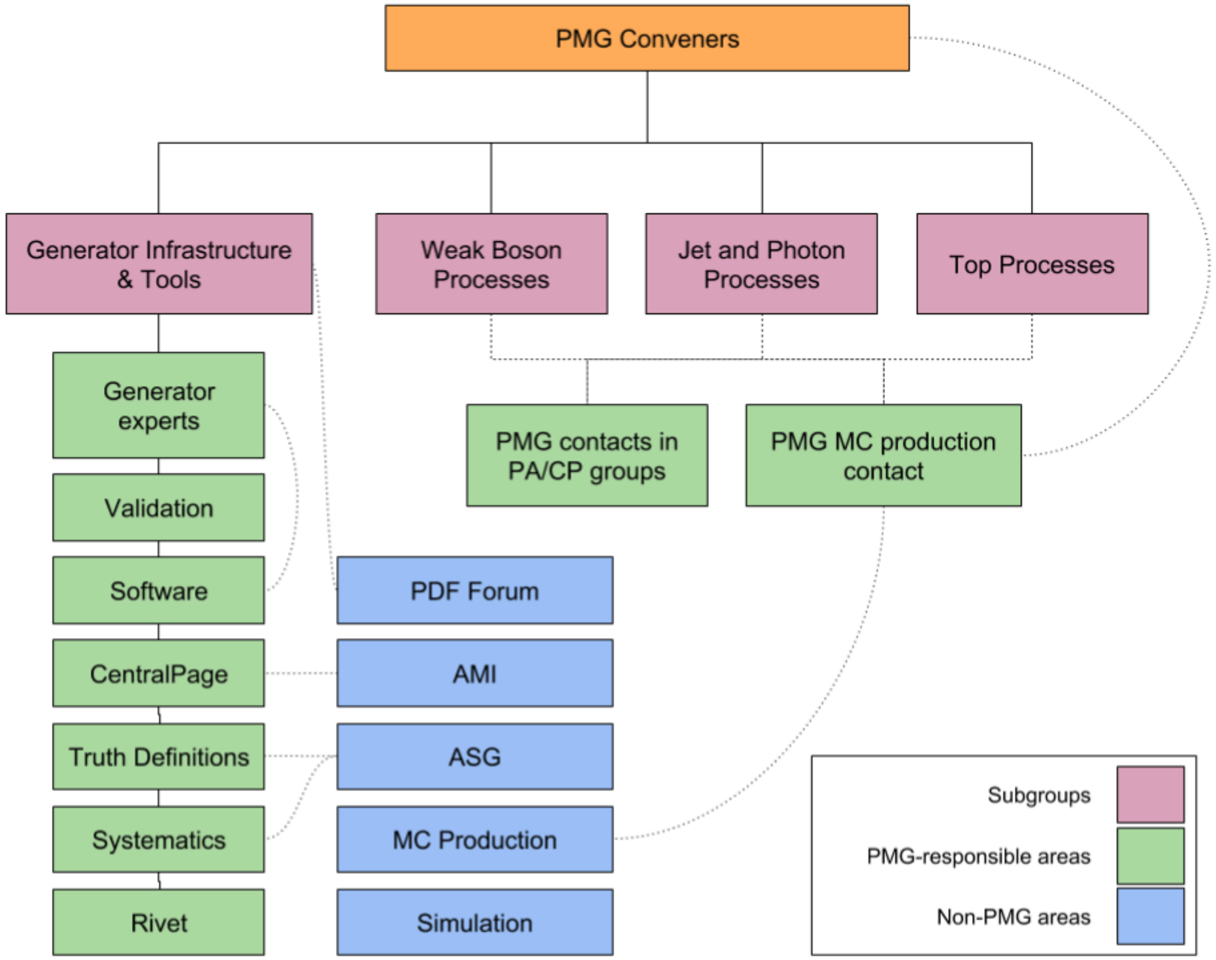
Introduction

- This session is aimed to give an introduction on how MC samples are produced in ATLAS and how you can run your own MC
  - Detector simulation and reconstruction not covered
- **PLEASE ASK QUESTIONS!!**
  - We are all here to learn. Do not be shy!
- More detailed information about the topics covered here can be found in:
  - **Monte Carlo and ATLAS Tutorial** (includes simulation and reconstruction): https://indico.cern.ch/event/440423
  - **Physics Analysis Workbook:** https://twiki.cern.ch/twiki/bin/viewauth/AtlasProtected/PhysicsAnalysisWorkBookRel21MC

The Physics Modelling group

- ATLAS group responsible for:
  - Developing and validating MC generators interfaces and MC samples. Providing common recipes for theory uncertainties.
  - Planning MC production and approval of requests.

- **Twiki:** PhysicsModellingGroup
- **PMG process groups**
  - Define MC configurations for the main SM processes
- **GIT group**
  - Maintaining generator interfaces, and tools for retrieving samples information and deriving systematic uncertainties



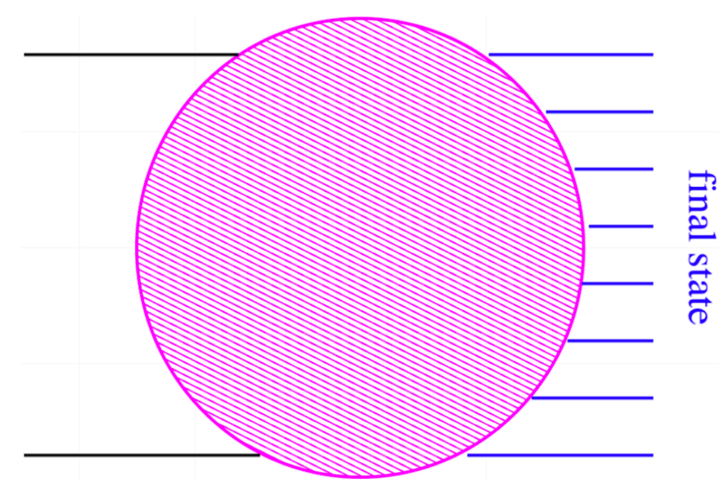
Outline

- (Very) brief introduction to MC event generators
- The Monte Carlo generation chain
  - Typical steps of MC generation in ATLAS
- Overview of generators used in ATLAS
- How to run MC generation within ATLAS
  - jobOptions and the Generate transform.
- Hands-on excercises

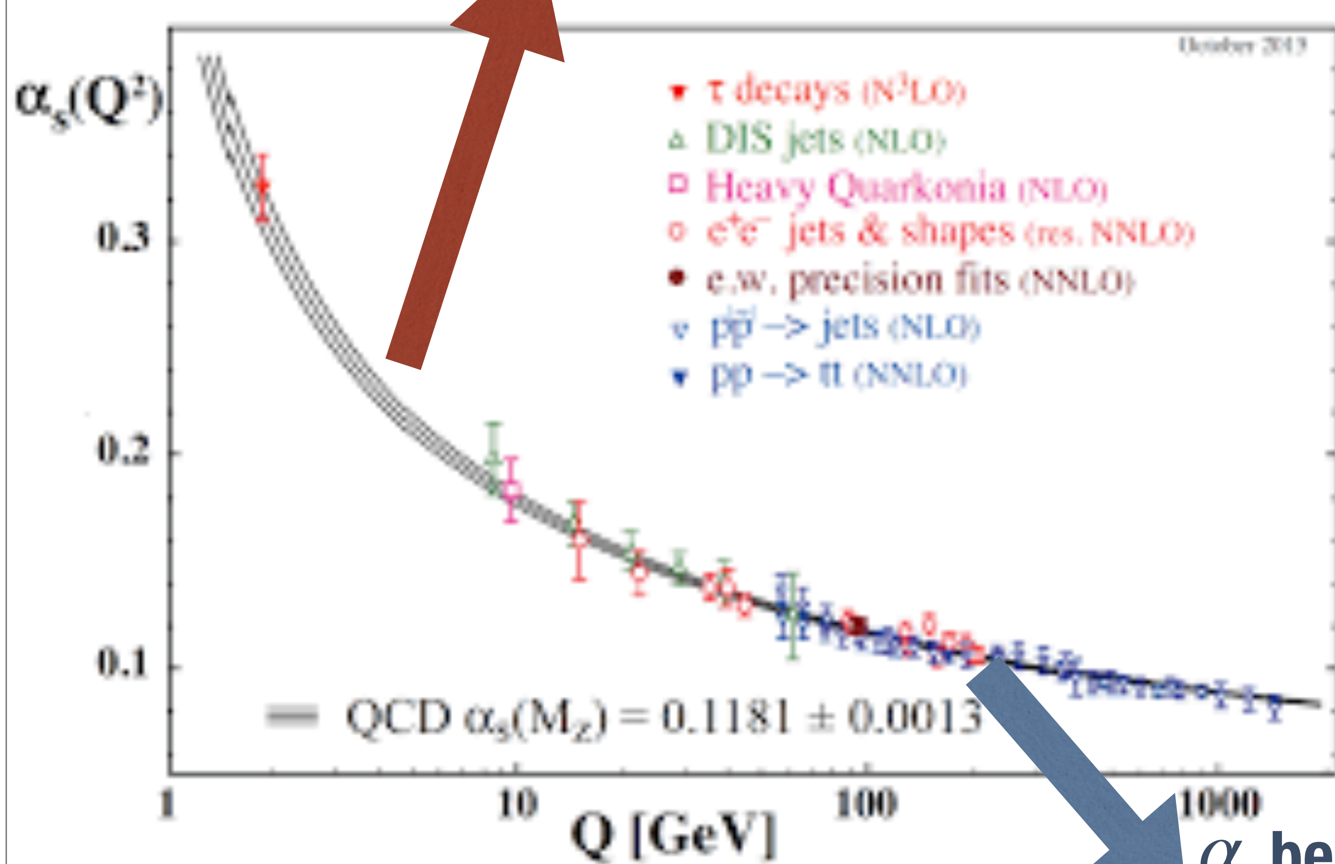
Introduction to MC

- Goal: produce events based on the cross sections for the generated process.

$\mathscr{L}_{\text{int}} \longleftrightarrow$ Final states



Confinement of quarks and gluons into hadrons. Non-perturbative regime.

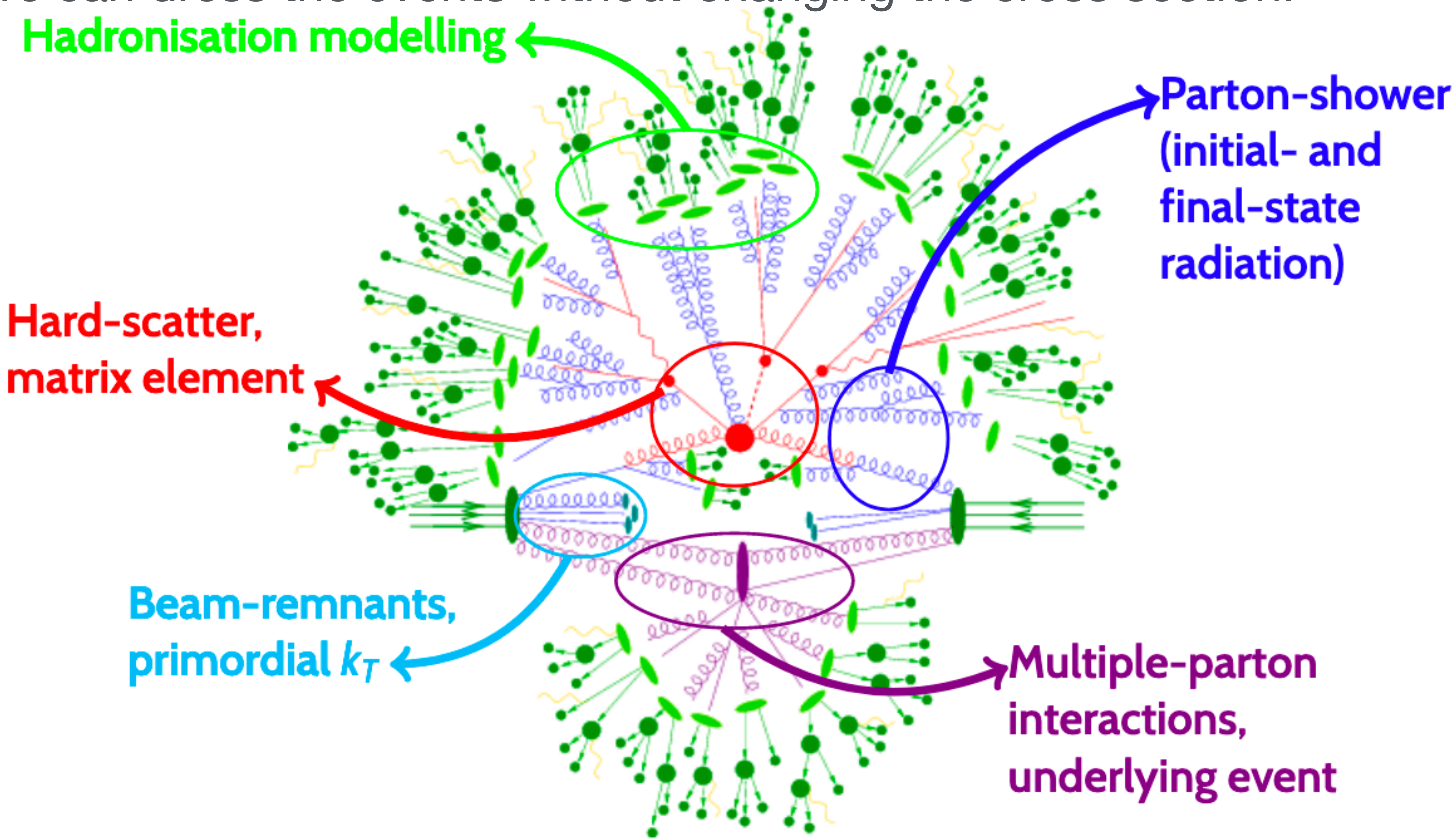


- Many different scales probed from hard interaction to final stable particles.
  - Different behaviour of the underlying theory (QCD)
  - Asymptotic freedom at high scales versus confinement at low scales.

α_s becomes small: Perturbation theory works. Gluons and quarks are “free".

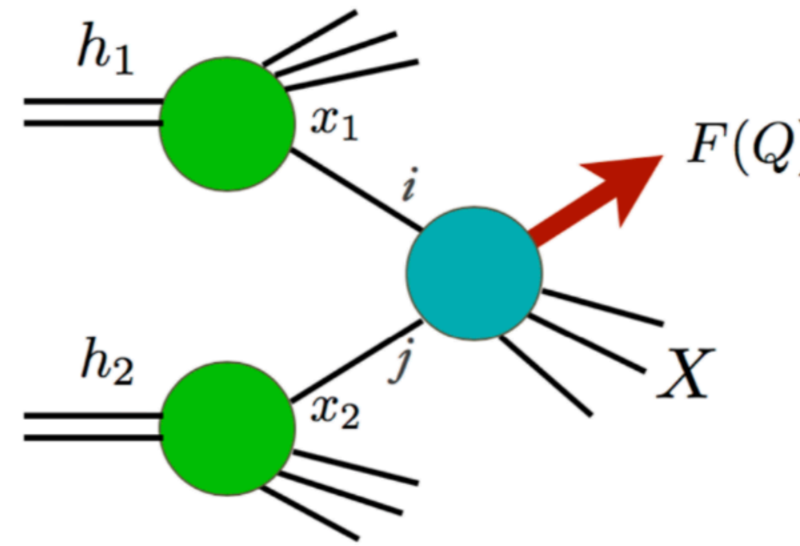
Simplistic MC event structure

Factorization ansatz: Assuming the hard and soft scales are separable, we can dress the events without changing the cross section.



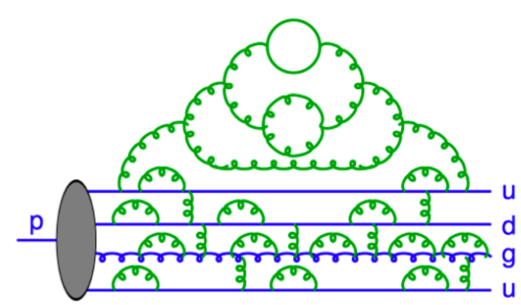
Hard-scattering

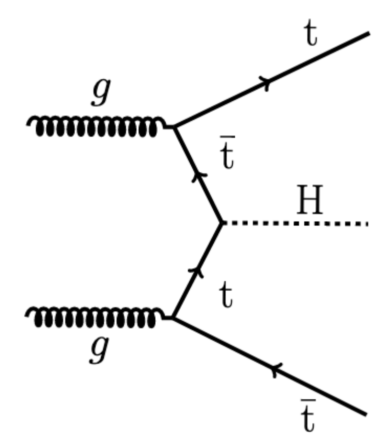
- Cross sections calculations in proton-proton collisions:



  - Protons are composite objects with a non-perturbative dynamics. Can be decoupled from the hard scale interactions that we are interested in -> **Factorization theorem**

- Cross section can be written as:





$$\sum_{a,b} \int dx_1 dx_2 d\Phi_{\mathrm{FS}}\, f_a(x_1, \mu_F) f_b(x_2, \mu_F)\, \hat{\sigma}_{ab\to X}(\hat{s}, \mu_F, \mu_R)$$

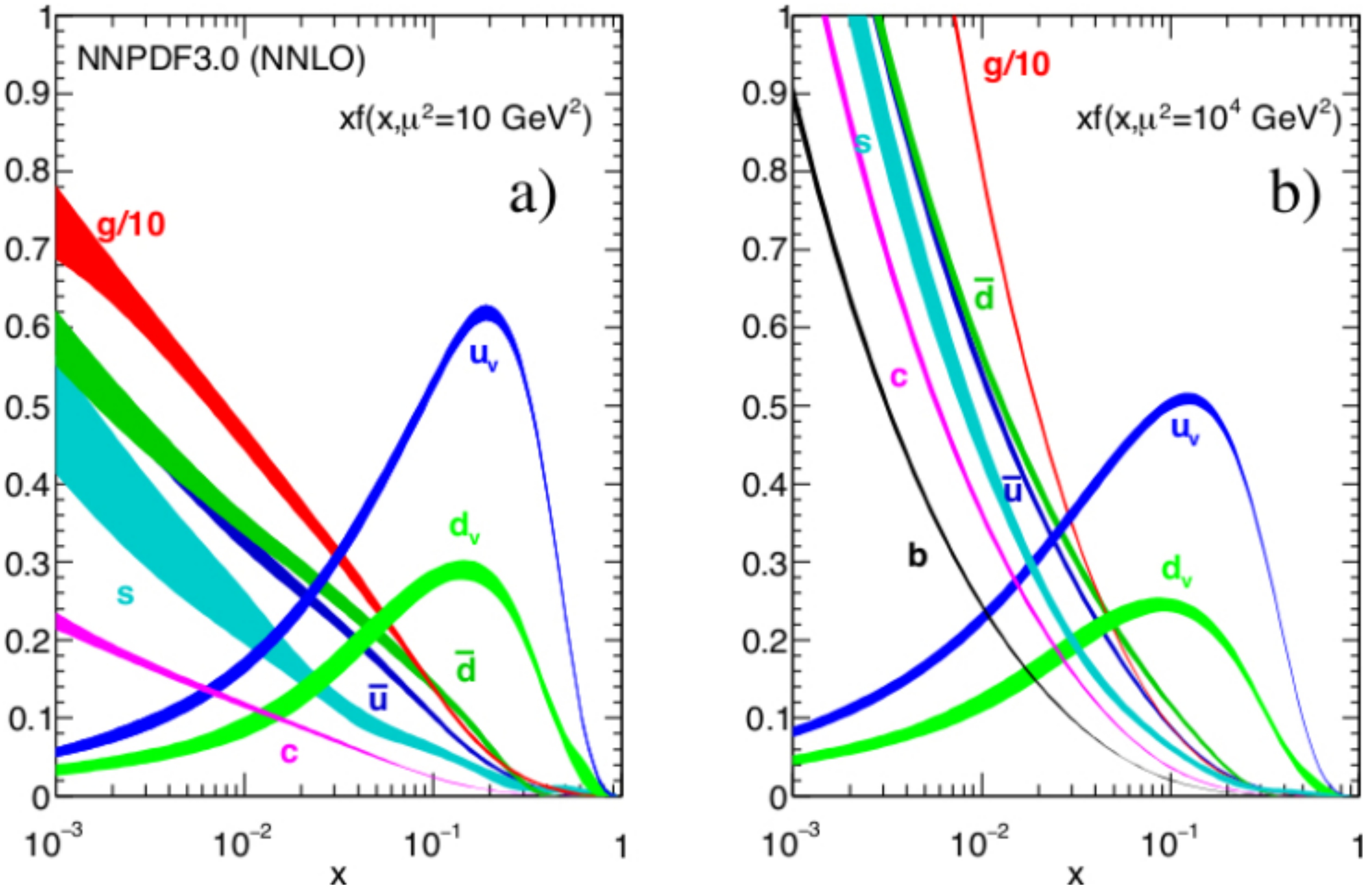
Phase-space integral

Parton density functions

Parton-level cross section

Parton density functions

- Probability of finding a parton (gluon or quark) of a determined flavour carrying a fraction x of the proton momentum
  - Parametrise our ignorance of what happens inside the proton below a given scale, μ_F
  - Non-perturbative objects that cannot be computed analytically. Fits to data are needed.
- Their scale evolution is predicted by the SM (DGLAP equations)
- They are universal:
  - PDFs extracted from a process can be used for any other process.

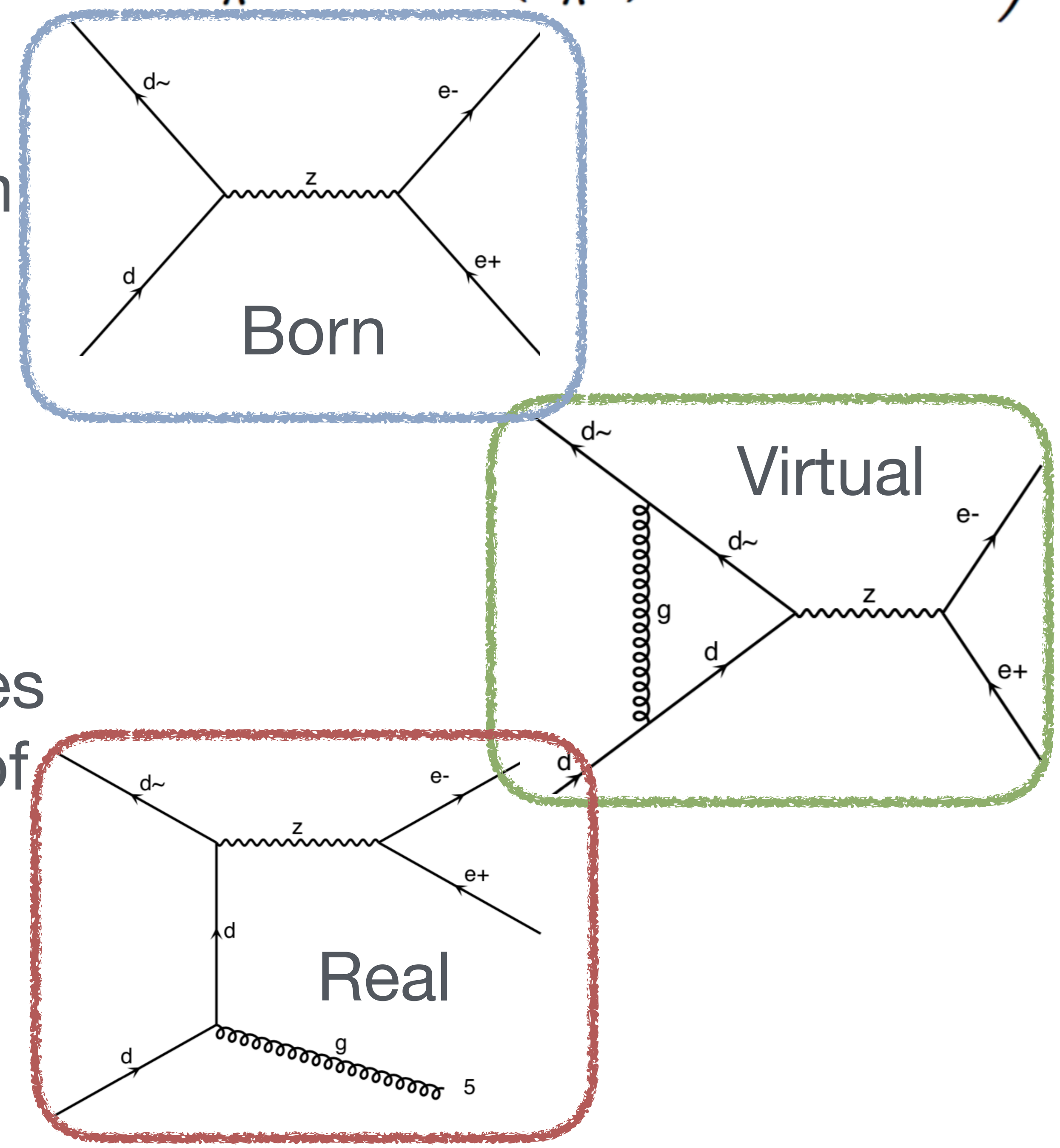


Matrix-elements

- Partonic cross-section can be expressed as a series in powers of the strong coupling.

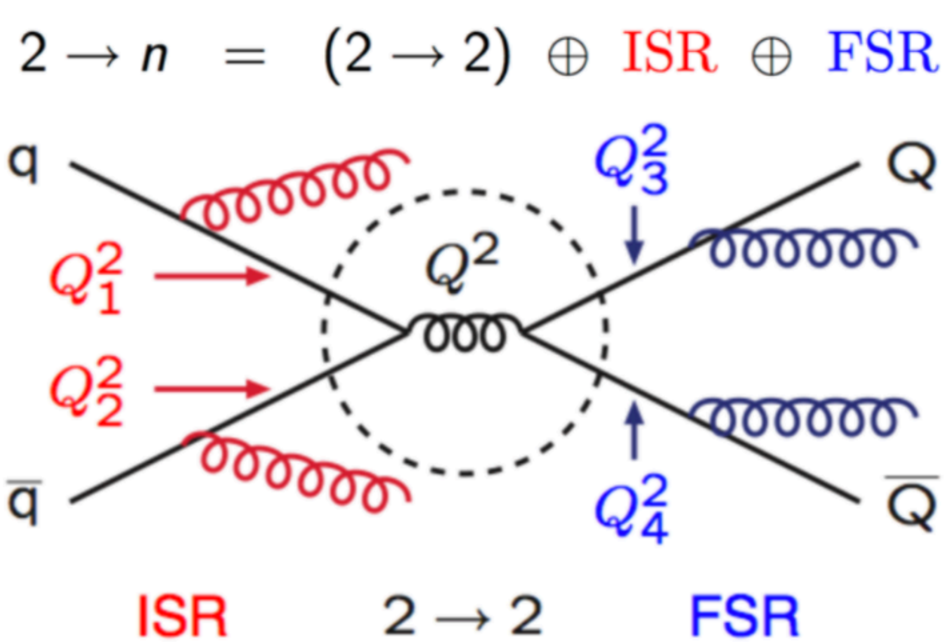
$$\hat{\sigma} = \alpha_S^k \left(\hat{\sigma}^{(0)} + \frac{\alpha_S}{\pi}\,\hat{\sigma}^{(1)} + \left(\frac{\alpha_S}{\pi}\right)^2 \hat{\sigma}^{(2)} + \dots \right)$$

- Leading-order: first non-zero term in the series.
- Going higher orders:
  - Starting from the **born** level, include **virtual** corrections and **real** corrections.
  - Reduced dependence on scales
  - Allows a meaningful estimate of uncertainties.



Parton-showers

- In the soft and collinear regime, QCD shows logarithmic divergences:
  - Resummation would only work for some observables
  - Showers “dress" partons with radiation by iteratively generating 1->2 branchings up to the non-perturbative limit (Q ~1 GeV).
- Exclusive description of the event with typically LL accuracy.
- Emissions have to be ordered (in kT, virtuality, opening angle,…).



FSR = Final-State Radiation

$Q_i^2 \sim m^2 > 0$ decreasing

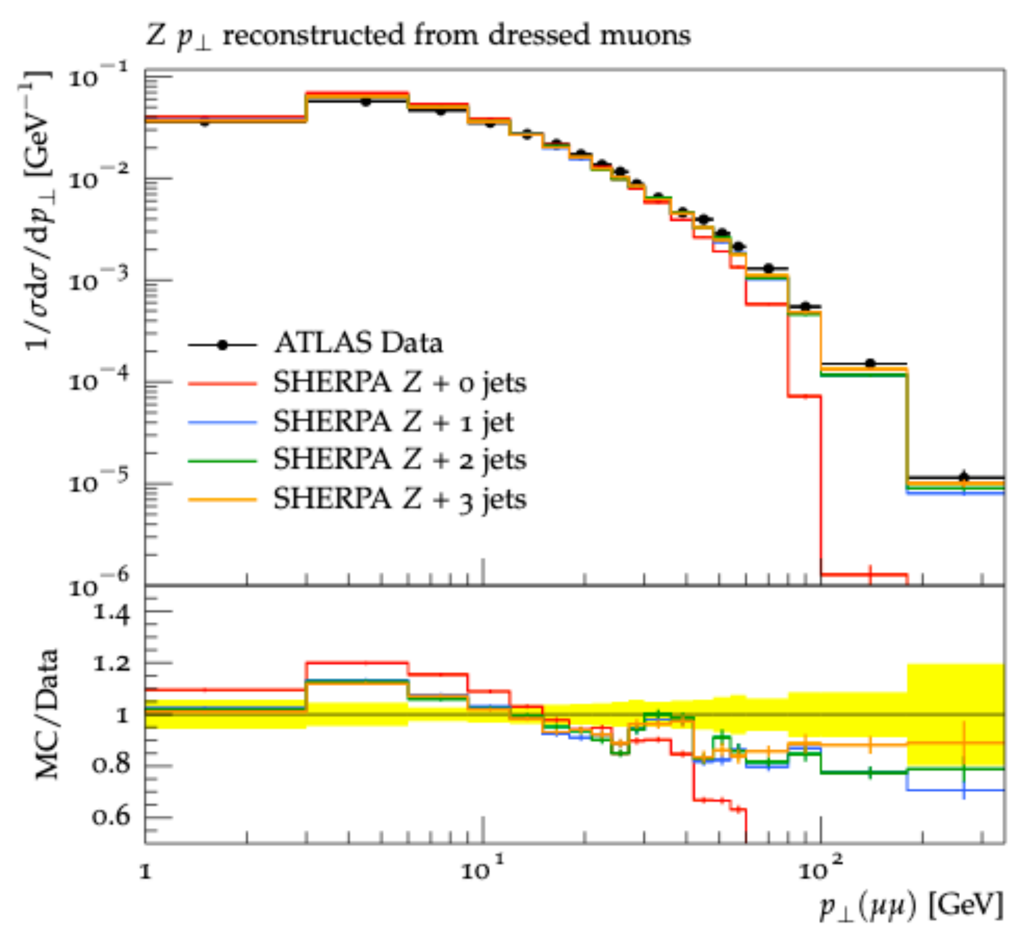
ISR = Initial-State Radiation

$Q_i^2 \sim -m^2 > 0$ increasing

Merging and matching

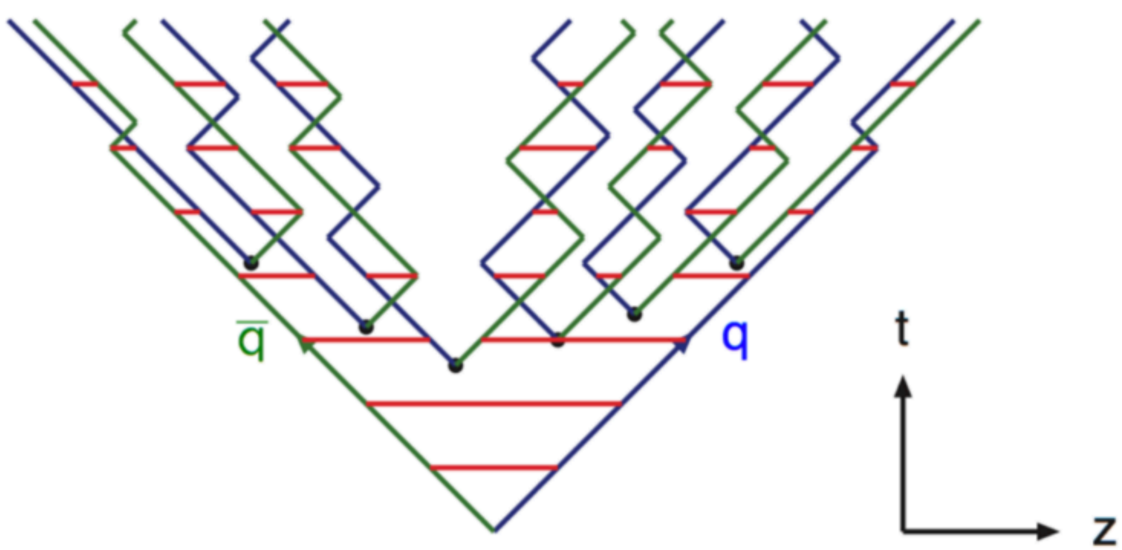
- Matrix elements describe the production of hard and well separated particles (jet production) while PS populate the soft-collinear region (structure inside jets).
  - Different methodologies to avoid double counting and handle the transition between the hard and soft regime.

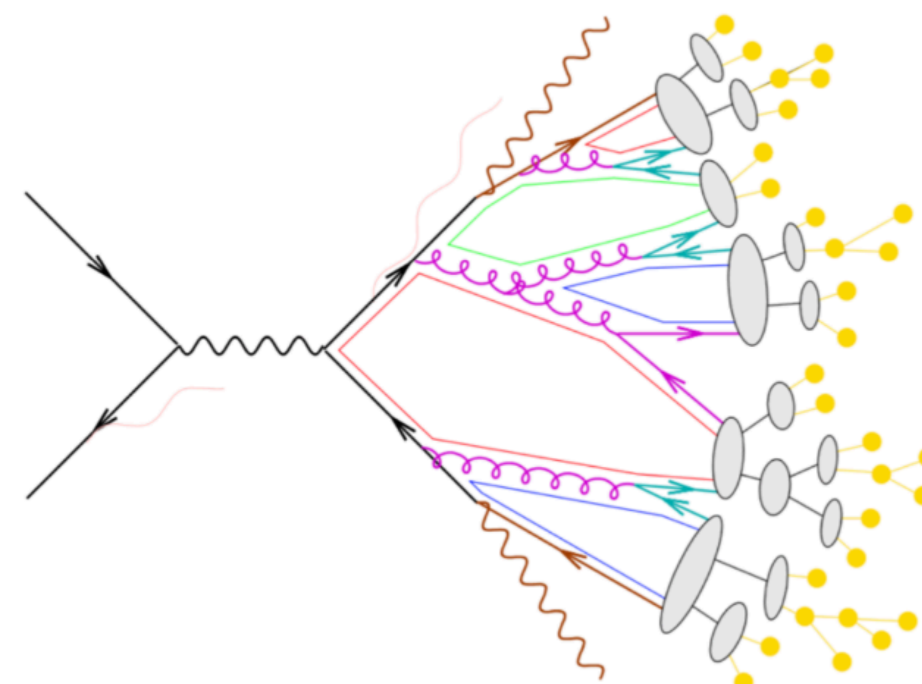
- Different options (increasing in complexity):
  - Match to the lowest non-trivial order (ME corrections)
  - Merge leading-order multi-parton processes (CKKW/MLM)
  - Match to an NLO calculation (POWHEG, aMC@NLO)
  - Combine several NLO orders (UNLOPS/FxFx/…)



Hadronisation

- Need to convert the low energy QCD partons into physically observable particles (hadrons).
- Non-perturbative regime. Relies on phenomenological models that have to be tuned to data.
  - Lund string fragmentation (Pythia)
  - Cluster hadronisation (Herwig, Sherpa)





Generators in ATLAS

PYTHIA (begun 1978)

Originated in hadronisation studies: Lund String model

Still significant emphasis on soft/non-perturbative physics

HERWIG (begun 1984)

Originated in coherence studies: angular-ordered showers

Cluster hadronisation as simple complement

SHERPA (begun ~2000)

Originated in ME/PS matching (CKKW-L)

Own variant of cluster hadronisation

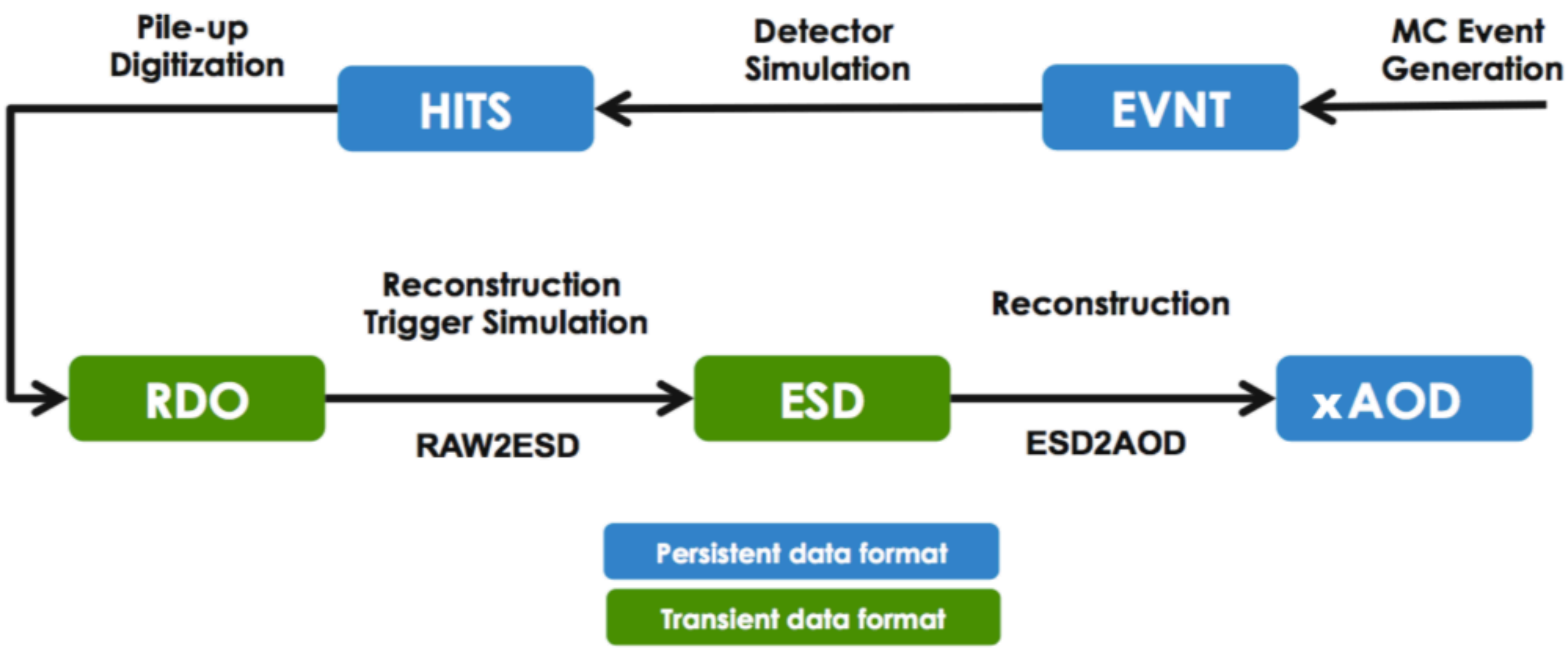
In addition matrix-element generators such as **MadGraph5_aMC@NLO** and **Powheg**

References

- Some more useful material can be found in:
  - Monte Carlo ATLAS Tutorial:
    https://indico.cern.ch/event/440423/timetable/
  - PMG guide to theory predictions:
    https://cds.cern.ch/record/2305435/
  - CERN Academic lectures on MC generators (by Fabio Maltoni)
    https://indico.cern.ch/event/181765/
    https://indico.cern.ch/event/181766/
    https://indico.cern.ch/event/181767/
  - MadGraph5_aMC@NLO tutorial
    https://indico.cern.ch/event/555228/
  - Sherpa tutorial
    https://indico.cern.ch/event/555805
  - Herwig7 tutorial
    https://indico.cern.ch/event/567362/

MC production - overview

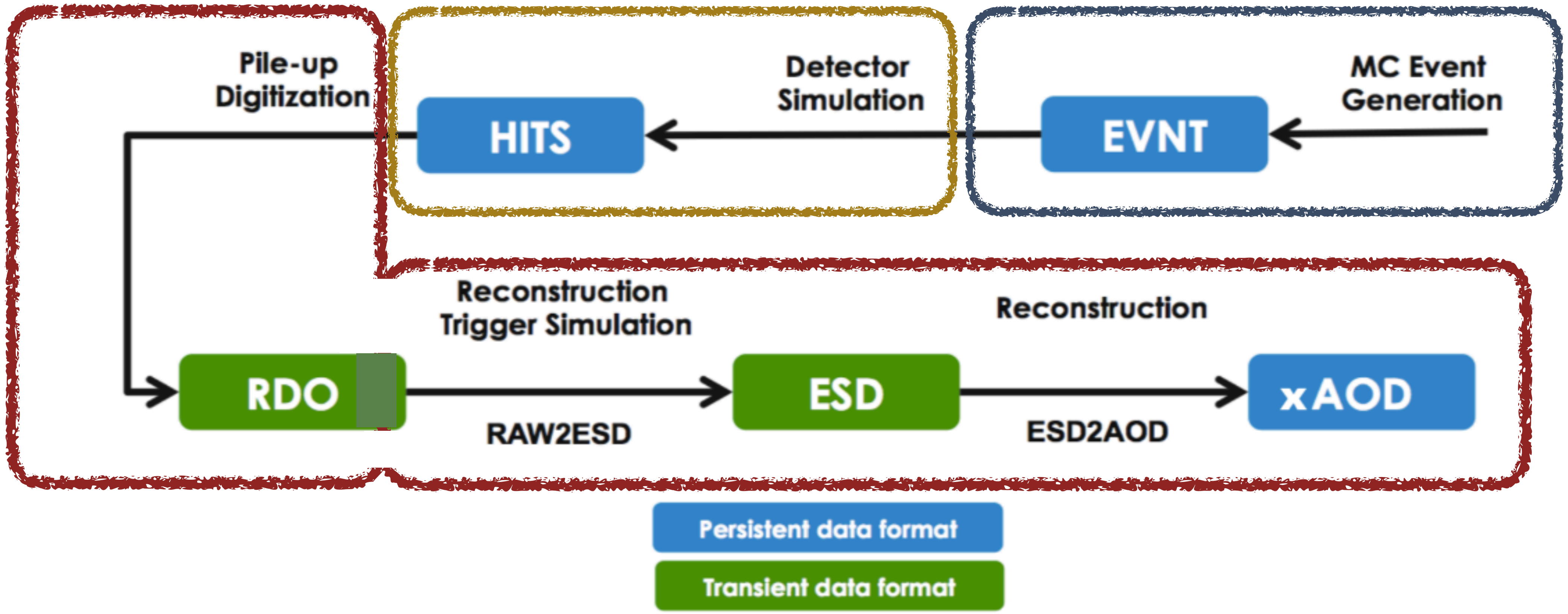
Scheme of Monte Carlo production in ATLAS



MC production - overview

Monte Carlo **generation** of particle level events of the process of interest

Simulation of the transport and interactions of the generated particles with the detector material



Reconstruction of physics objects from detector level information

MC generator chain

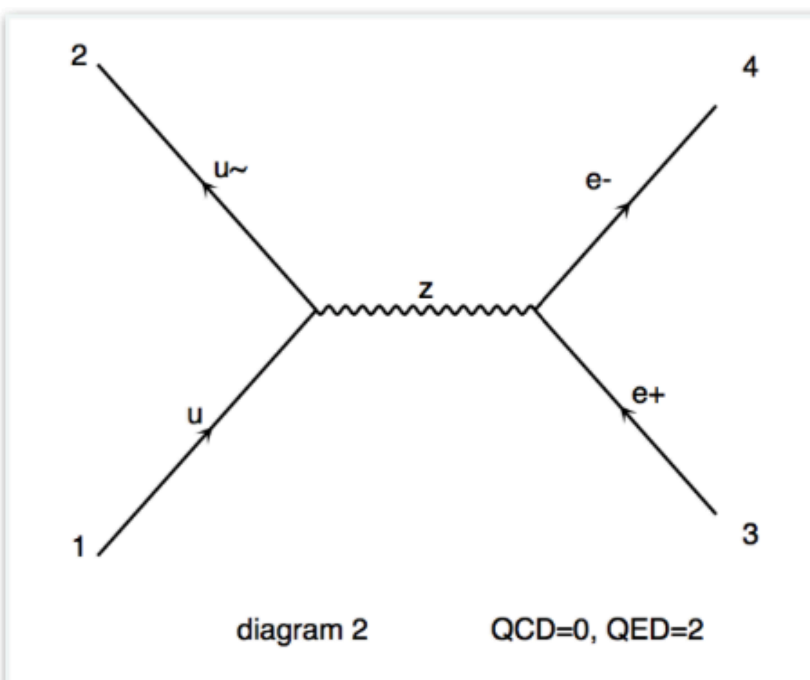
Process definition

↑

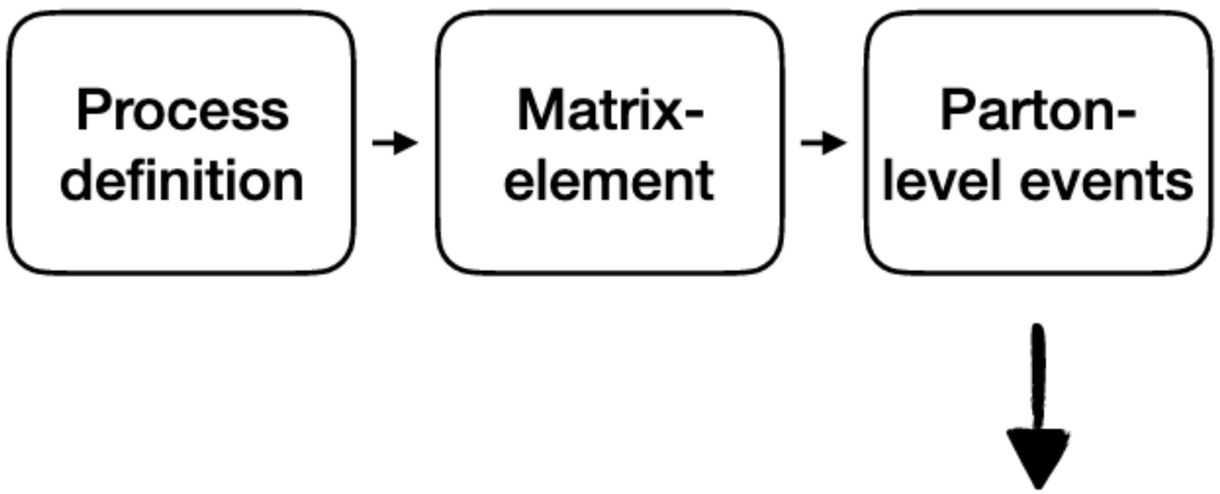
generate p p > e+ e-
CoM Energy = 13 TeV
PDF = NNPDF2.3LO
min_mll = 10 GeV
etc...

MC generator chain

Process definition → **Matrix-element**



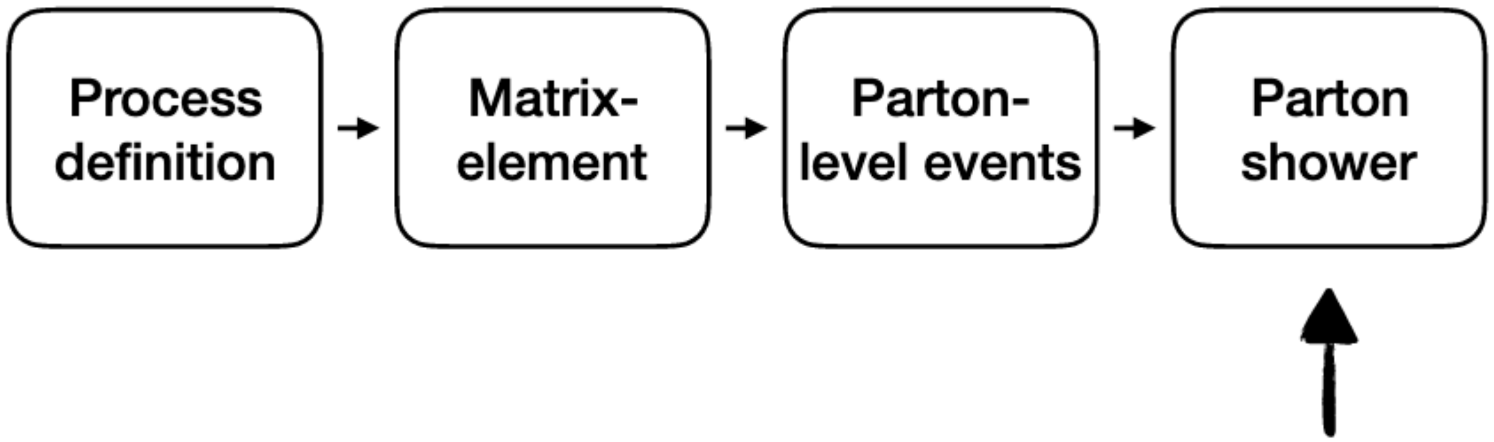
MC generator chain




```
<event>
 5  1 0.3244300E+00 0.9118800E+02 0.7546771E-02 0.1179815E+00
        2  -1    0    0  501    0  0.00000000000E+00  0.00000000000E+00  0.80444845014E+03  0.80444845014E+03  0.00000
       -2  -1    0    0    0  501  0.00000000000E+00  0.00000000000E+00 -0.24994407405E+01  0.24994407405E+01  0.00000
       23   2    1    2    0    0  0.00000000000E+00  0.00000000000E+00  0.80194900940E+03  0.80694789088E+03  0.89681
      -11   1    3    3    0    0 -0.30723622217E+02  0.27827348871E+02  0.55482913354E+03  0.55637547550E+03  0.00000
       11   1    3    3    0    0  0.30723622217E+02 -0.27827348871E+02  0.24711987585E+03  0.25057241538E+03  0.00000
</event>
```


- Parton-level information is stored in Les Houches Event files
  - Contains four-vectors of incoming and outgoing particles

MC generator chain



PDF = NNPDF2.3LO

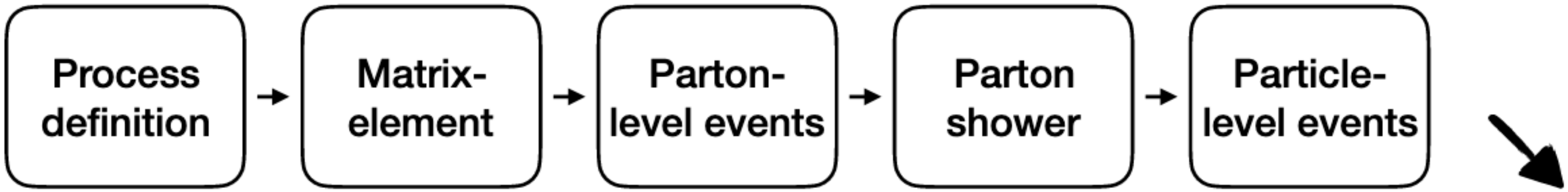
Tune = A14

Merging scheme = CKKW-L

Merging scale = 30 GeV

etc…

MC generator chain

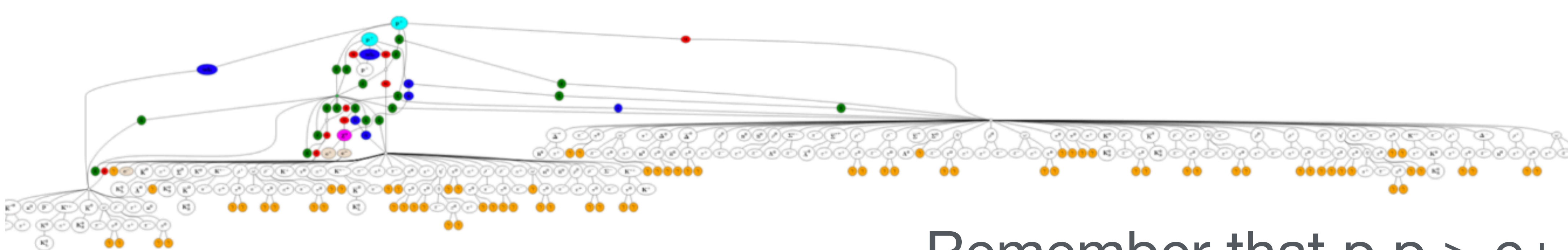


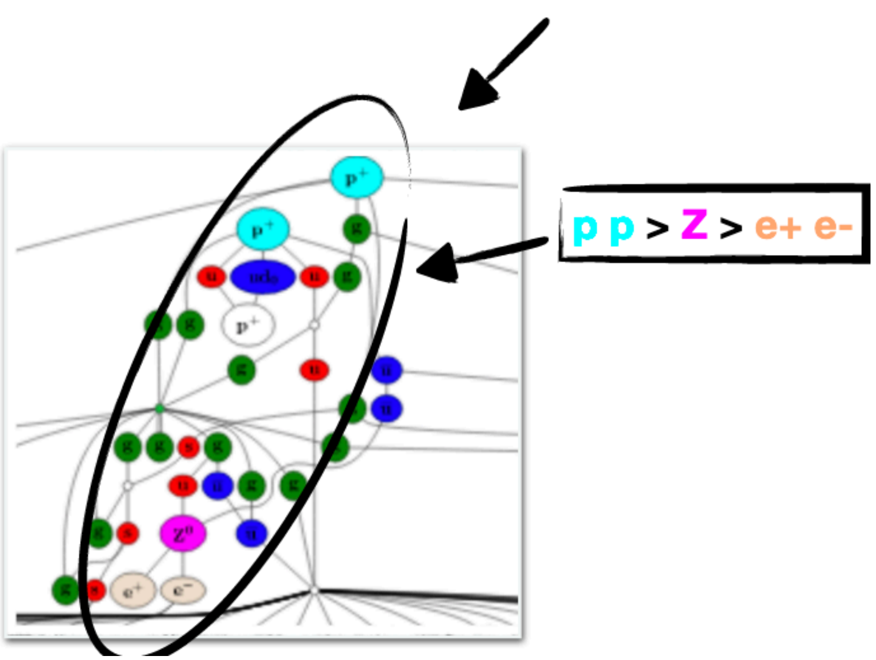

```
E 0 -1 -1.0000000000000000e+00 -1.0000000000000000e+00 -1.0000000000000000e+00 0 0 95 1 2 0 1 4.7292949000000004e-10
N 1 "0"
U MEV MM
C 4.7292949000000006e-01 4.7292948999999999e+03
V -1 0 0 0 0 0 0 2 0
P 3 3 0 0 3.6673387060249966e+04 3.6673387060249966e+04 0 21 0 0 -3 1 1 501
P 9 21 1.0766234477407208e+03 2.4494227262266604e+03 1.5084741024303903e+03 3.0715279353578953e+03 0 43 0 0 -10 2 1 502 2 501
V -2 0 0 0 0 0 0 1 0
P 4 -3 0 0 -5.8025401060249962e+04 5.8025401060249962e+04 0 21 0 0 -3 1 2 501
V -3 0 0 0 0 0 0 1 0
P 5 23 0 0 -2.1352013999999999e+04 9.4698786698633674e+04 9.2260239000000001e+04 22 0 0 -6 0
V -4 0 0 0 0 0 0 2 0
P 6 3 2.2204460492503131e-13 0 3.9495339086932072e+04 3.9495339086932079e+04 0 41 0 0 -1 1 1 502
P 14 21 -2.0542118125169582e+03 -2.0157537529171391e+03 1.0022227598261232e+04 1.0427276508965933e+04 0 43 0 0 -15 2 1 503 2 502
V -5 0 0 0 0 0 0 1 0
P 7 -3 -1.1102230246251565e-13 0 -5.8025401060249955e+04 5.8025401060249962e+04 0 42 0 0 -2 1 2 501
```


- Particle-level information is stored in HepMC format
  - Contains particles and vertices information of all particles produced
  - EVNT format is simply an Athena readable HepMC

MC generator chain

Process definition → Matrix-element → Parton-level events → Parton shower → HepMC visualisation



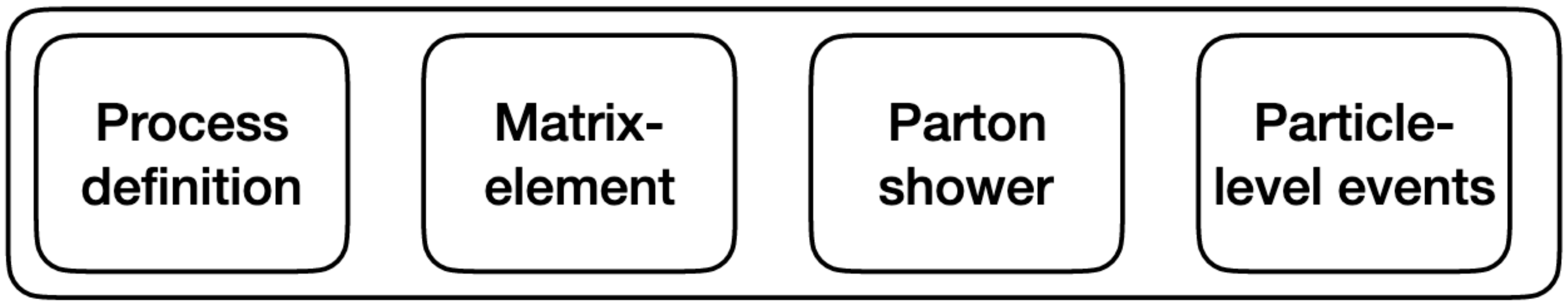


Remember that p p > e+ e- is the process we generated!

generate p p > e+ e-
CoM Energy = 13 TeV
PDF = NNPDF2.3LO
min_mll = 10 GeV
etc...

Production modes

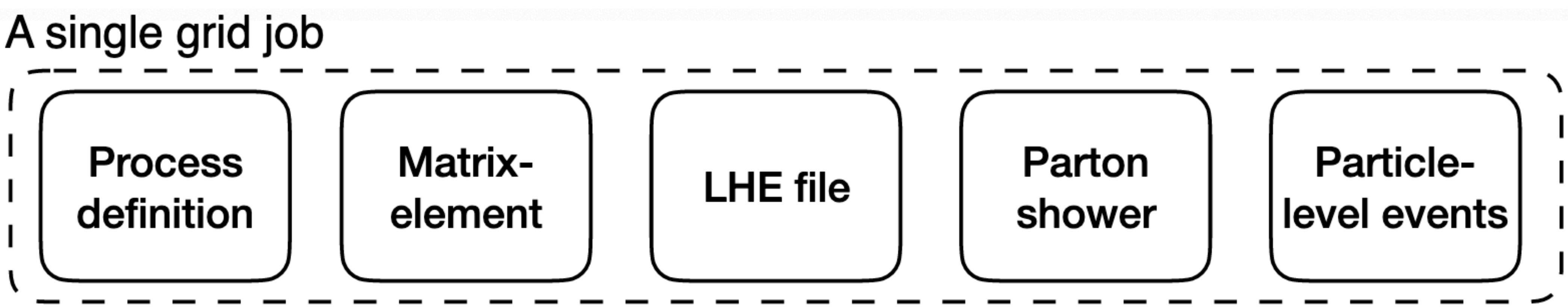
- **Example 1: Pythia8/Herwig7/Sherpa**
  - ME generation and showering in a single job



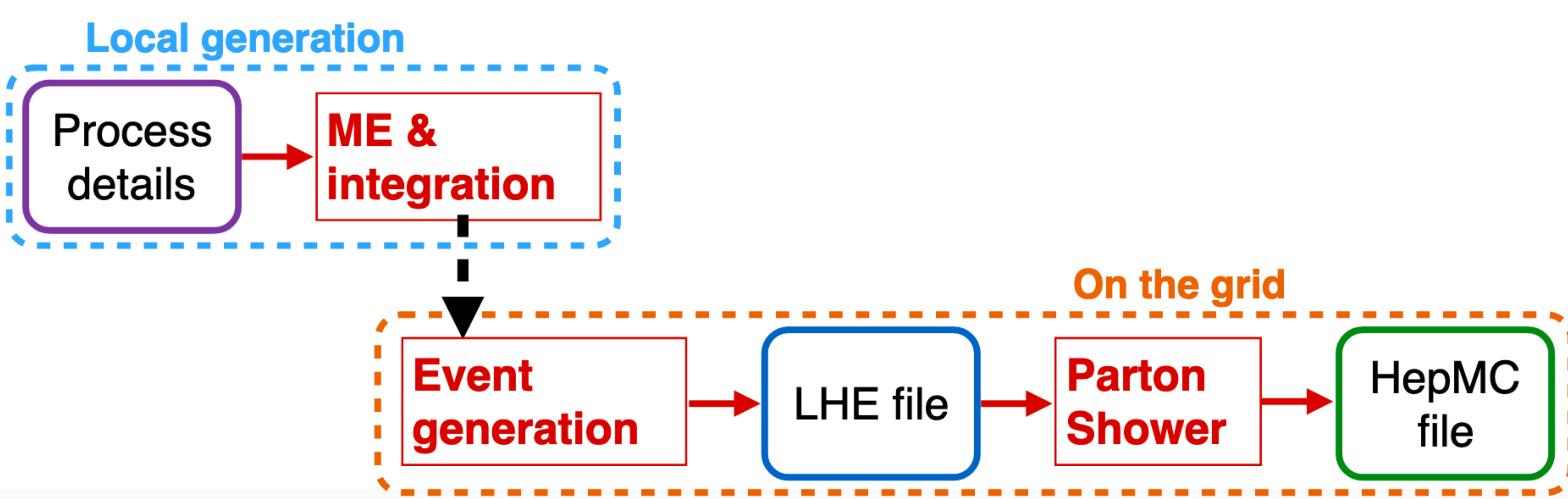
- **Example 2: MadGraph/Powheg + shower**
  - MadGraph and Powheg generates only ME
  - Need to be interfaced with other generator producing the shower

Production modes

- **On-the-fly: LHE generation and showering in a single job**



- **With pre-integration grids:**
  - Monte Carlo integration can be quite CPU and time consuming. The grids just need to be generated once and can be used for all subjobs.



Generators in ATLAS

- Each generator is supported by a group of ATLAS experts
- Each twiki provides useful information about how to generate events with a specific generator.

Twiki link

General List of MC generators (still to be updated for MC16-specific generators)

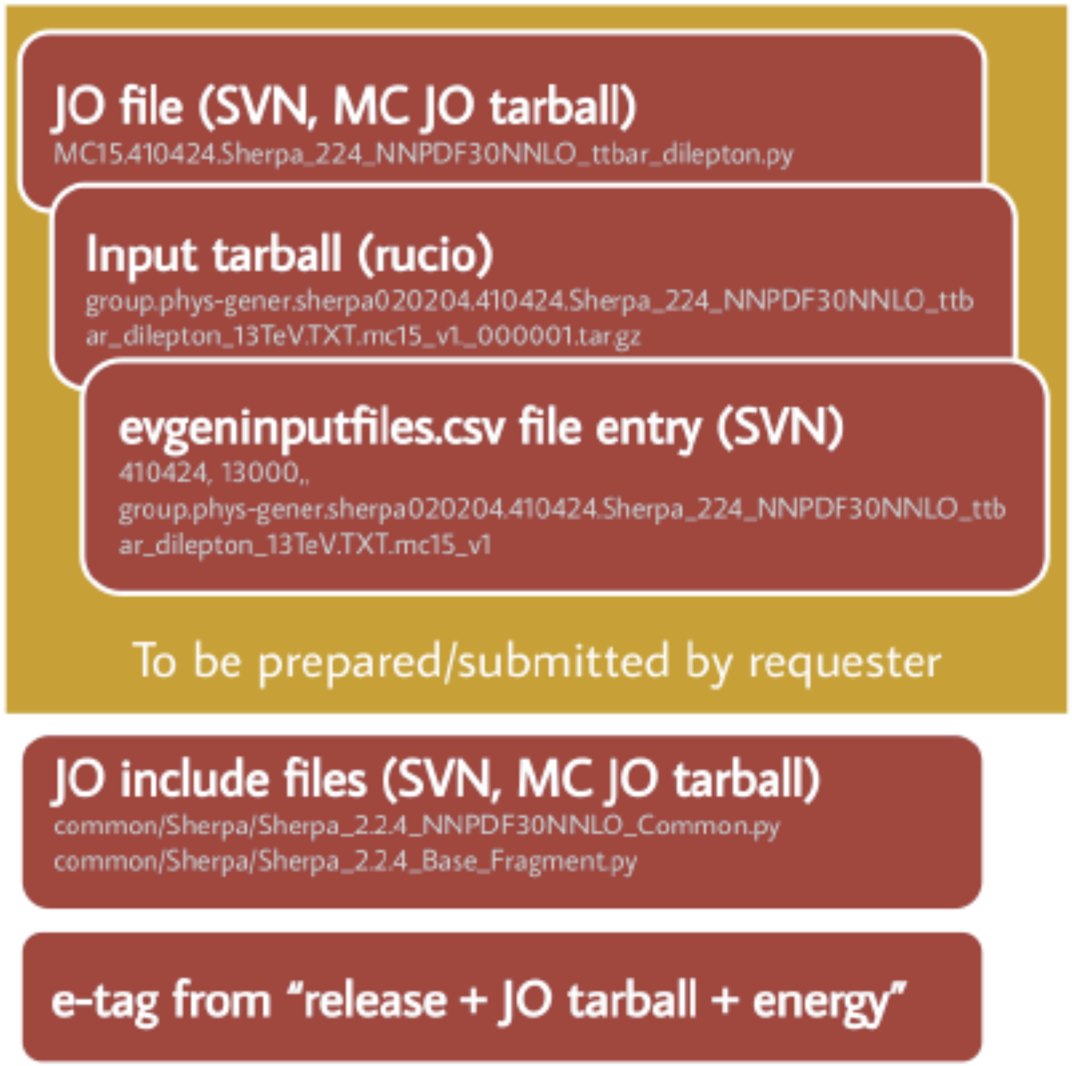
Generator	TWiki	General mailing list	Contact people
Powheg	PowhegForATLAS	atlas-generators-powhegcontrol@cern.ch	atlas-generators-powhegcontrol-experts@cern.ch Daniel Hayden, Stefan Richter, Riccardo di Sipio
MadGraph5_aMC@NLO	MadGraph5aMCatNLOForAtlas	atlas-generators-madgraphcontrol@cern.ch	atlas-generators-madgraphcontrol-experts@cern.ch Zach Marshall, Josh McFayden, Stefan von Buddenbrock, Hannes Mildner
Sherpa	SherpaForAtlas	atlas-generators-sherpa@cern.ch	atlas-generators-sherpa-experts@cern.ch Chris Gutschow, Frank Siegert, Heberth Torres
Pythia8	Pythia8ForAtlas	atlas-generators-pythia@cern.ch	atlas-generators-pythia-experts@cern.ch Andy Buckley, Giancarlo Panizzo, Marjorie Shapiro
Herwig7	Herwig7ForAtlas	atlas-generators-herwig7@cern.ch	atlas-generators-herwig7-experts@cern.ch David Yallup, Ludovic Scyboz, Tetiana Moskalets

Running MC within ATLAS

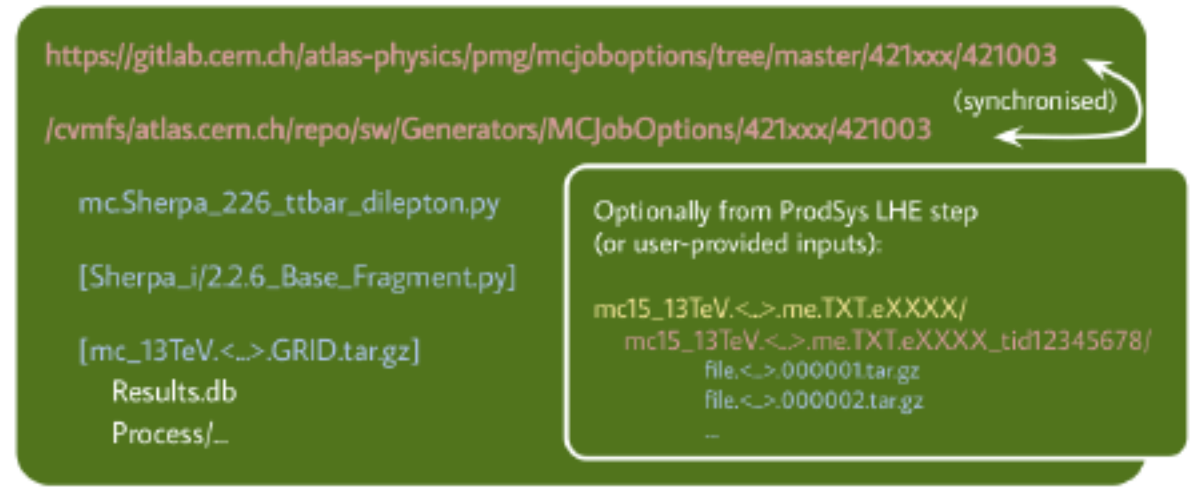
- We use "**jobOptions**" (JO) to run MC generation.
  - They are python steering files that tell Athena what you want to generate.
  - One needs to define in the jobOptions:
    - Which generator
    - Which process
    - Which settings
    - Etc…
- This information is passed to Athena through a "**generation transform**"
  - An executable that takes several arguments as inputs. E.g. the jobOption that we want to run, the number of events to be generated, the pre-integration grids that we want to use (if any),…

Moving to a new workflow

Current structure:



New structure:

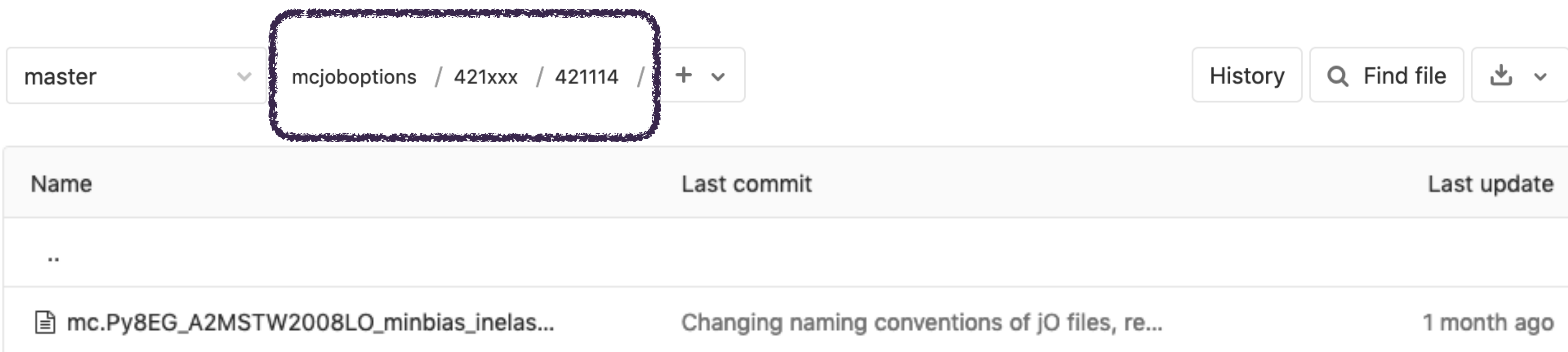


- Main motivation was to improve automated checks.
  - All inputs needed for a JO will go into a same directory.
  - Use MR system in Gitlab
  - Consistency checks in Gitlab CI

- The following slides will provide the needed information for sample production with the new workflow for event generation (started to work in October 2019).

Finding existing JO

- JOs are identified with an unique dataset ID (DSID):
  - It coincides with the parent directory of the JO
- They can be found all be found in gitlab:
  - **https://gitlab.cern.ch/atlas-physics/pmg/mcjoboptions/tree/master**
- The JOs live in the **[YZW]xxx/[DSID]** directory
  - YZW are the first 3 digits of the DSID
  - E.g. for the dataset with DSID 421114, the JO is: mc.Py8EG_A2MSTW2008LO_minbias_inelastic_high.py
  - And it is located in the folder 421xxx/421114



JobOptions- Pythia example

mc.Py8EG_A2MSTW2008LO_minbias_inelastic_high.py 679 Bytes

Edit | Web IDE | Replace | Delete

```python
#based on 361035
evgenConfig.description = "Low-pT inelastic minimum bias events for pile-up, with the A2 MSTW2008LO tune and EvtGen"
evgenConfig.keywords = ["QCD", "minBias", "SM"]

evgenConfig.saveJets = True

include("Pythia8_i/Pythia8_A2_MSTW2008LO_EvtGen_Common.py")

genSeq.Pythia8.Commands += \
    ["SoftQCD:inelastic = on"]

#include("GeneratorFilters/AntiKt4TruthJets_pileup.py")
#include("GeneratorFilters/AntiKt6TruthJets_pileup.py")

include ("GeneratorFilters/FindJets.py")
CreateJets(prefiltSeq, 0.6)
AddJetsFilter(filtSeq,runArgs.ecmEnergy, 0.6)

from AthenaCommon.SystemOfUnits import GeV
filtSeq.QCDTruthJetFilter.MinPt = 35.*GeV

evgenConfig.minevents = 100
```

Metadata

Pythia8 tunes

Process definition

Filter

JobOptions- Pythia example

- A JO will typically include other jobOptions ("Top JOs")
  - Those are used to define common settings needed by multiple other JOs (particle masses, PDFs, tunes, …)

mc.Py8EG_A2MSTW2008LO_minbias_inelastic.py 532 Bytes

Edit | Web IDE | Replace | Delete

```
#based on 361033
evgenConfig.description = "Inelastic minimum bias, with the A2 MSTW2008LO tune and EvtGen"

evgenConfig.keywords = ["QCD", "minBias", "SM"]

# Note: The tune used here (A2 MSTW2008LO) is not the standard one for high pT physics.  It is what we use for pile up at t

include("Pythia8_i/Pythia8_A2_MSTW2008LO_EvtGen_Common.py")

genSeq.Pythia8.Commands += ["SoftQCD:inelastic = on"]

evgenConfig.minevents = 1000
```

They live inside the DSID folder or in a common Generator area

master | mc15joboptions / nonStandard / Pythia8 / **Pythia8_A2_MSTW2008LO_Common.py**

Find file | Blame | History | Permalink

Pythia8_A2_MSTW2008LO_Common.py 1.42 KB

Edit | Web IDE | Replace | Delete

```
## Config for Py8 tune A2 with MSTW2008LO
## This is the version without EvtGen, but the default is to use EvtGen
## The default version is in common/Pythia8/Pythia8_A2_MSTW2008LO_Common.py
## PDF syntax depending on  Pythia8 version

include("MC15JobOptions/Pythia8_Base_Fragment.py")

genSeq.Pythia8.Commands += [
    "Tune:pp = 5",
    "MultipartonInteractions:bProfile = 4",
    "MultipartonInteractions:a1 = 0.03",
    "MultipartonInteractions:pT0Ref = 1.90",
```

They are easy to find using the "Find file" function

JobOptions- Sherpa example

- Let's look at: 421002/mc.Sherpa_Example2.py

- It includes the "top" JO from the common area:

mc.Sherpa_Example2.py 631 Bytes

Edit | Web IDE | Replace | Delete

```
include("Sherpa_i/2.2.7_NNPDF30NNLO.py")

evgenConfig.description = "Sherpa 2.2.x example JO, Z+0,1-jet production."
evgenConfig.keywords = [ "2lepton" ]
evgenConfig.contact  = [ "atlas-generators-sherpa@cern.ch", "frank.siegert@cern.ch"]

genSeq.Sherpa_i.RunCard="""
(processes){
  Process 93 93 -> 11 -11 93{0}
  Order (*,2)
  CKKW sqr(20/E_CMS)
  End process;
}(processes)

(selector){
  Mass 11 -11 40 E_CMS
}(selector)
"""

genSeq.Sherpa_i.Parameters += [ "LOG_FILE=", "MI_HANDLER=None" ]
genSeq.Sherpa_i.OpenLoopsLibs = []
genSeq.Sherpa_i.ExtraFiles = []
genSeq.Sherpa_i.NCores = 1

genSeq.Sherpa_i.CleanupGeneratedFiles = 1
```

Metadata

Process definition+ Matching scale

Generator cuts

Running the generation

- Once we have the complete JO, we can run it using the "**generate transform**":
  - Gen_tf.py

    --ecmEnergy=13000 **Center-of-mass energy**

    --firstEvent=1 **EventNumber of first event.**

    --maxEvents=5000 **Maximum number of events**

    --randomSeed=1234 **Random number seed**

    --jobConfig=421002 **DSID**

    --outputEVNTFile=test_Zee_421002.EVNT.root **Output File**

- Use the common "-h" (**Gen_tf.py** -h) to see all supported arguments. More detailed information here.

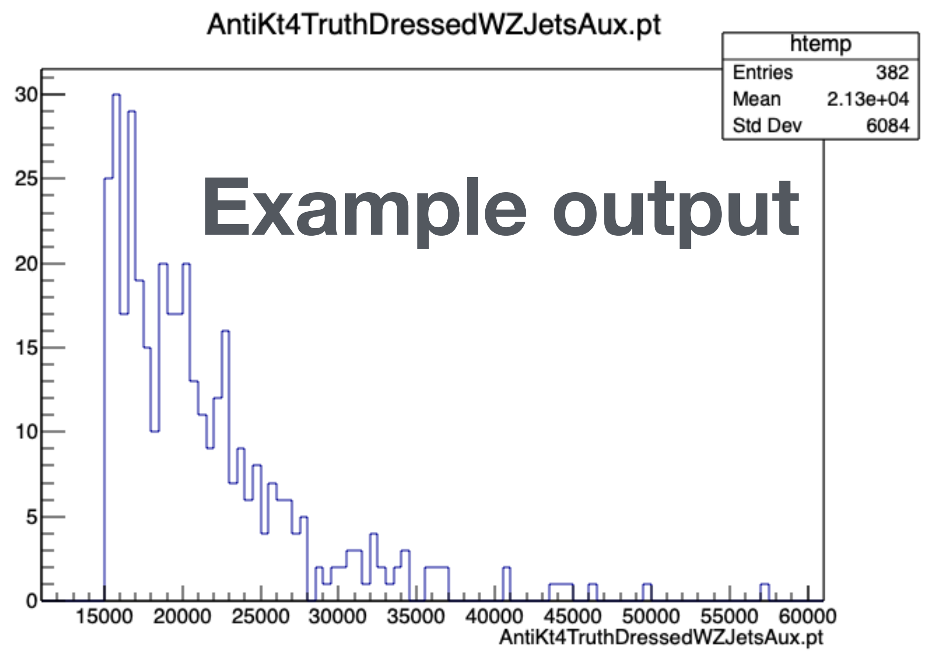
Looking at the output

- The final output will be a EVNT file. The most common ways of inspecting them are:
  - RIVET: A toolkit for validation of MC generator and analysis preservation.
    - https://rivet.hepforge.org/
  - TRUTH derivations: A convenient way of analysing the generator output within Athena or ROOT
    - https://twiki.cern.ch/twiki/bin/viewauth/AtlasProtected/TruthDAOD

Truth Derivations

- The "**Reco transform**" transforms the HepMC data in the EVNT file in a TRUTH_DAOD.
- The syntax is easy
  - Reco_tf.py
    --inputEVNTFile=test_Zee_421002.EVNT.root **Input EVNT file**
    --outputDAODFile=tutorial.pool.root **Output DAOD file**
    --reductionConf=TRUTH3 **Truth type**

- Different reductionConf corresponds to different level of information:
  - From TRUTH0 (just a copy of the truth record, no built objects) to TRUTH3 (usual format, already built physics objects: jets, photons, electrons,..)



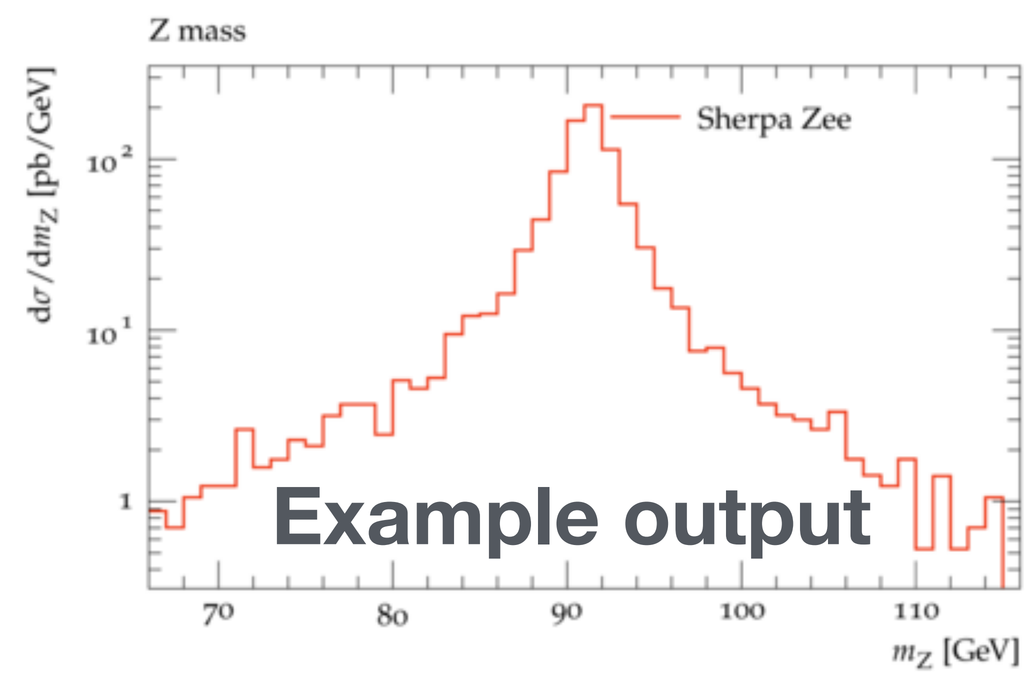
Rivet

- Rivet provides a large set of experimental analyses

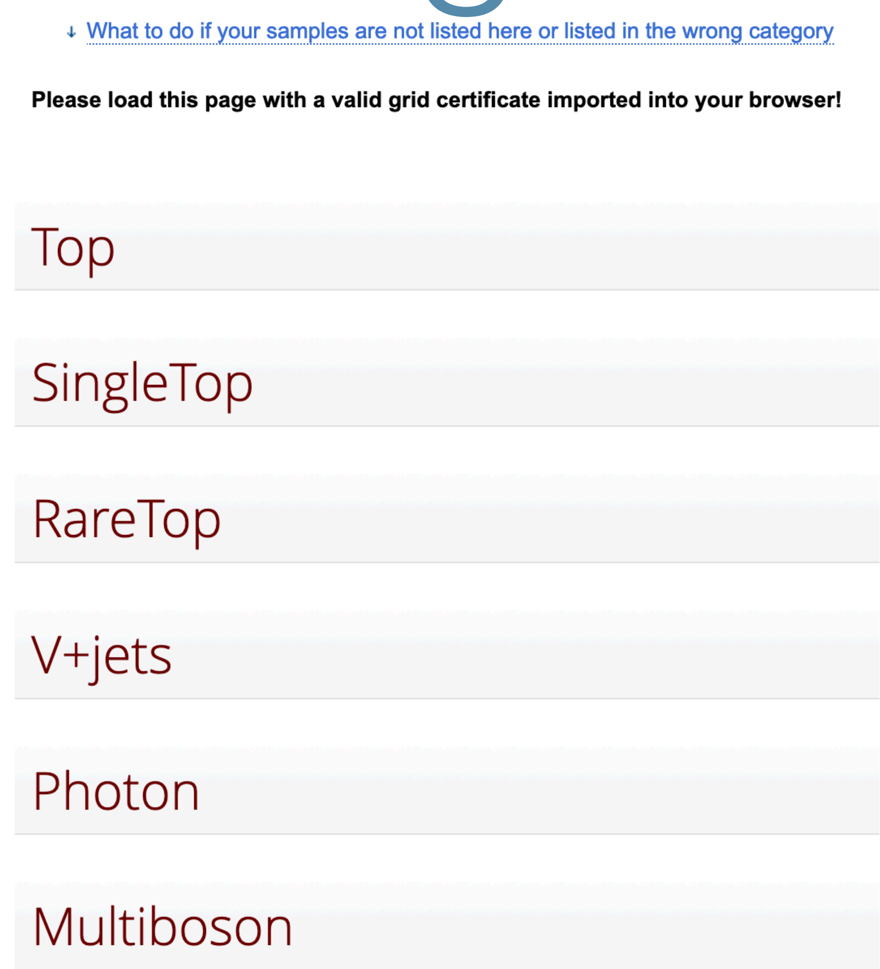
  - ATLAS_2016_I1502620 – W and Z inclusive cross sections at 7 TeV
  - ATLAS_2017_I1495243 – ttbar + jets at 13 TeV
  - ATLAS_2017_I1509919 – Track-based underlying event at 13 TeV in ATLAS
  - ATLAS_2017_I1514251 – Z plus jets at 13 TeV
  - ATLAS_2017_I1517194 – Electroweak Wjj production at 8 TeV
  - ATLAS_2017_I1589844 – k_T splittings in Z events at 8 TeV
  - ATLAS_2017_I1591327 – Inclusive diphoton cross-sections at 8 TeV

- Trivial to run within the generate transform

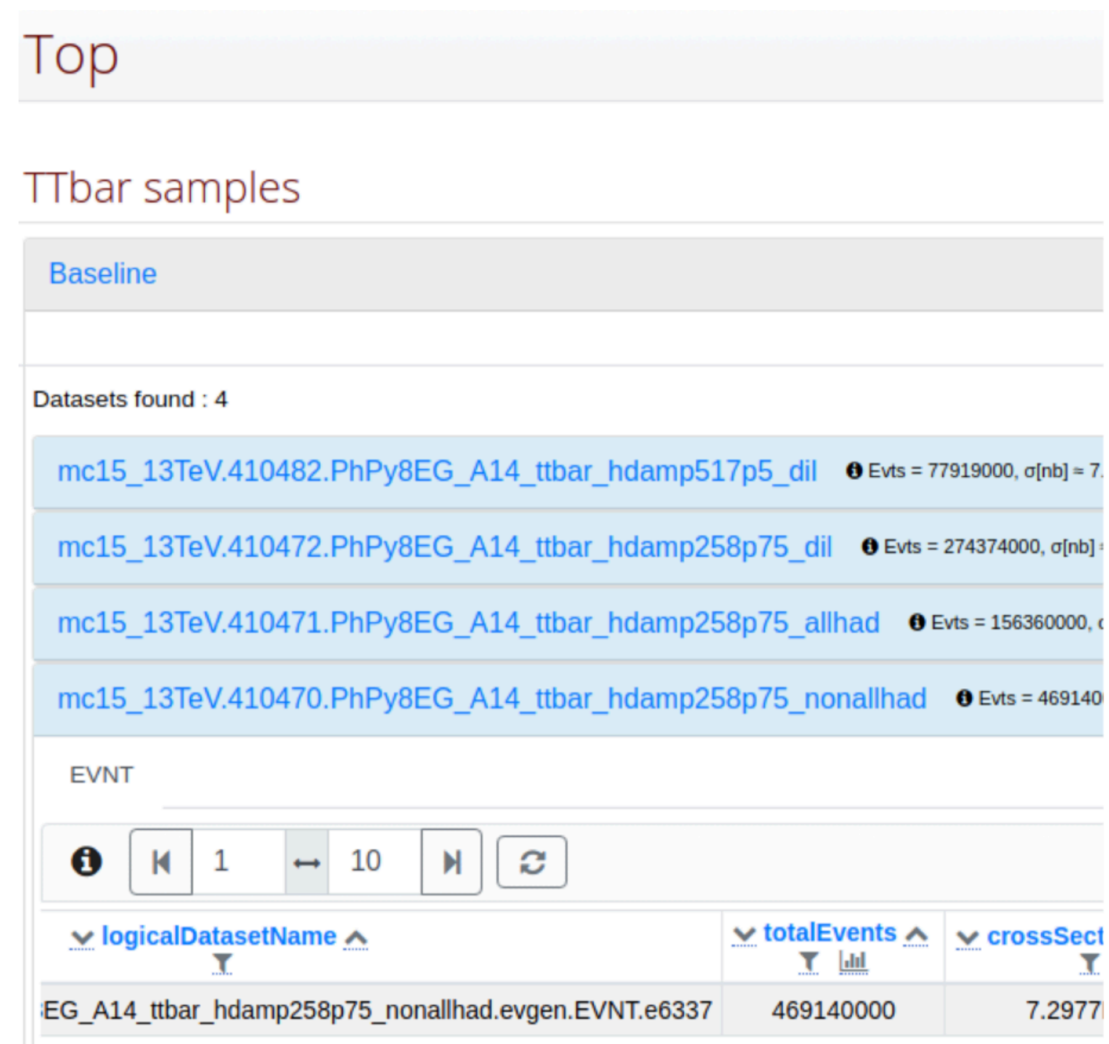
  - Gen_tf.py
    --ecmEnergy=13000
    --firstEvent=1
    --maxEvents=5000
    --randomSeed=1234
    --jobConfig=421002
    --outputEVNTFile=test_Zee_421002.EVNT.root
    --rivetAnas=MC_ZJETS,ATLAS_2017_I1514251



Finding samples



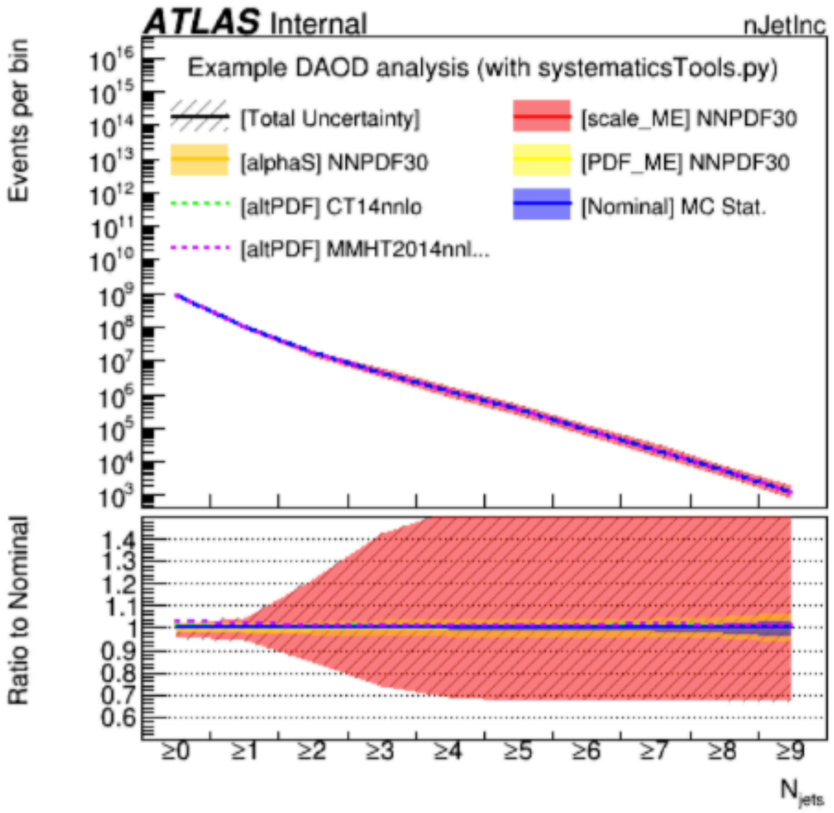
- All our samples are documented in the PMG central page: https://twiki.cern.ch/twiki/bin/view/AtlasProtected/CentralMC16ProductionList



- Information about (from AMI):
  - Sample statistics
  - Cross section
  - Filter efficiencies
  - Higher orders
  - Hashtags

PMG systematics tool

- PMG provides recommendations for estimating modelling uncertainties of bulk MC samples.
  - Typically by comparing scales, PDF, α_S or shower parameters variations made available as weight.
- Propagating this information to the analysis is an important task and a tool was developed to simplify this:
  https://gitlab.cern.ch/atlas-physics/pmg/tools/systematics-tools
  - Automatically process generator weights or variations into systematics.
  - Helper function to submit Rivet/AthAnalysis jobs for each different weight/variation
  - Combine yoda/Root files applying proper recipe to compute each systematic variation



Time for questions!

BACK-UP

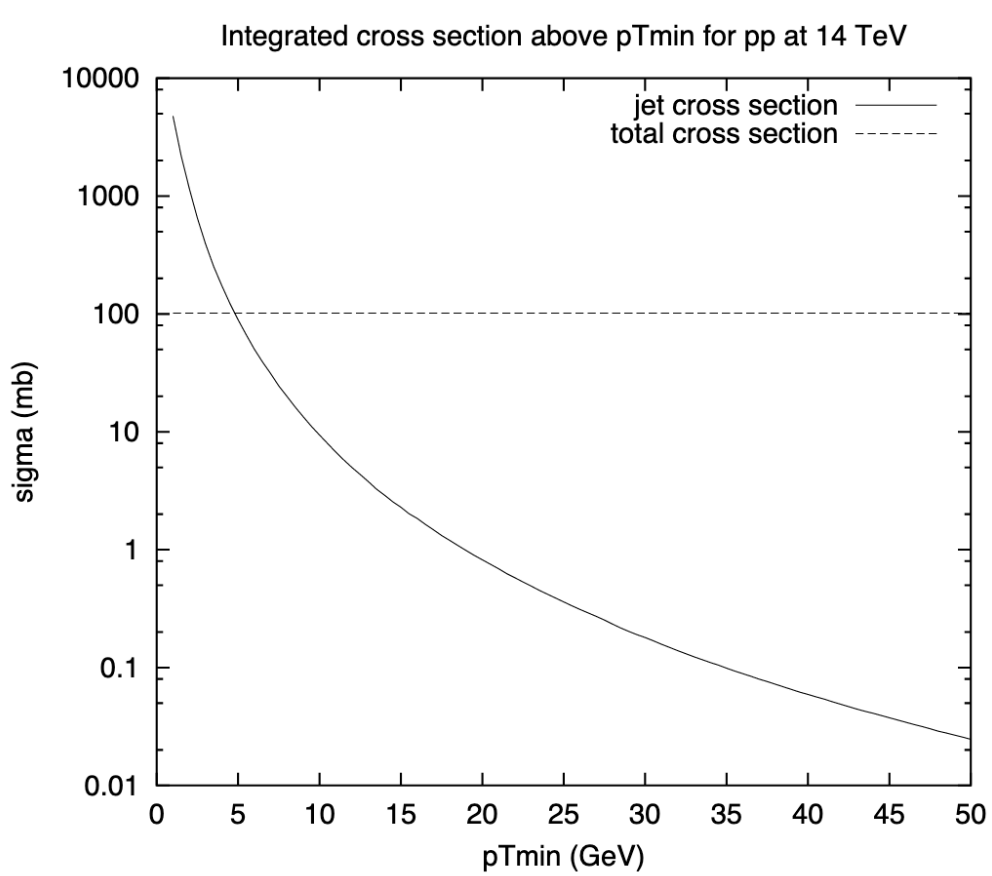
Multiple parton interactions

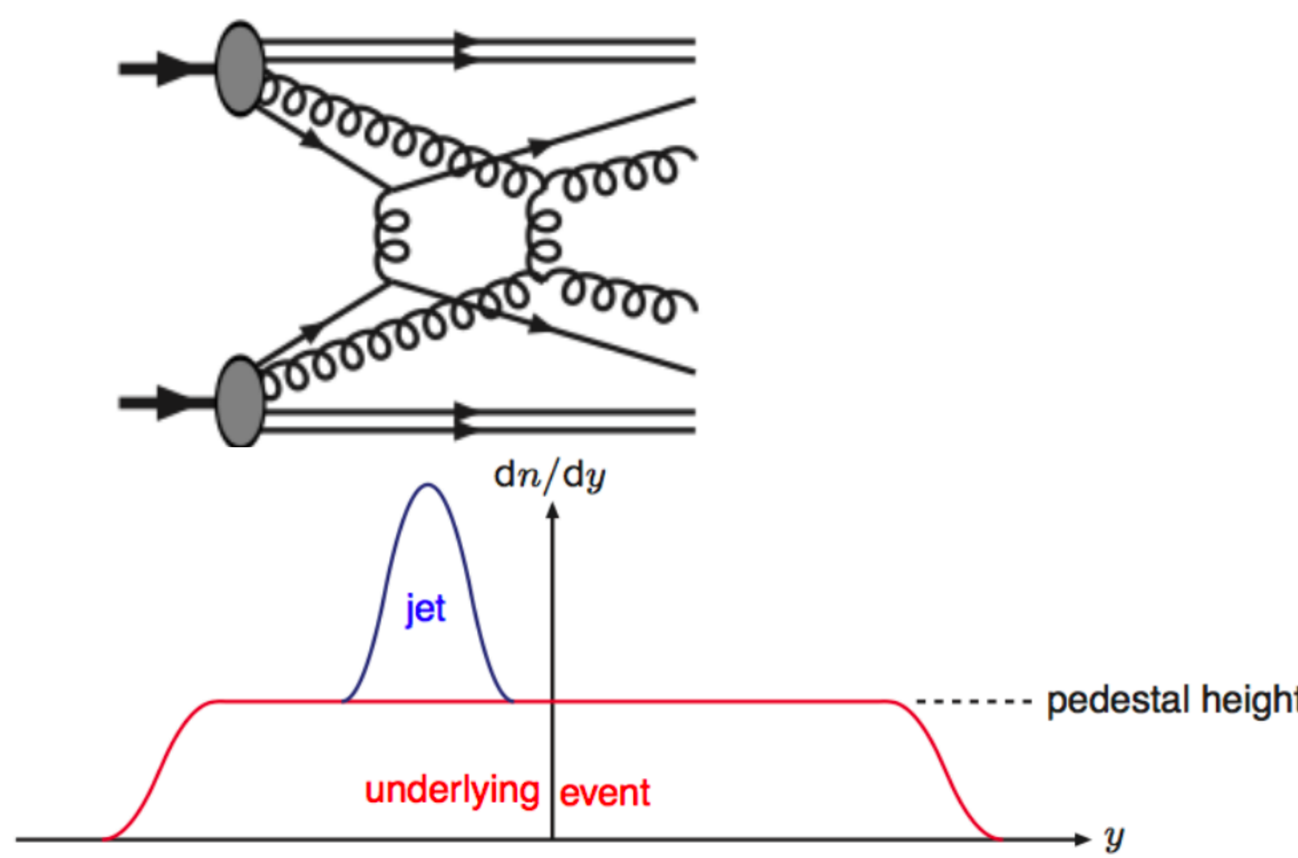
- QCD diverges at low p_T as $\sim 1/p_T^4$ with $\sigma_{int}(p_{T,min}) > \sigma_{tot}$
  - More than one parton-parton interaction per event

$$\frac{d\hat{\sigma}}{dp_\perp^2} \propto \frac{\alpha_s^2(p_\perp^2)}{p_\perp^4} \to \frac{\alpha_s^2(p_{\perp 0}^2 + p_\perp^2)}{(p_{\perp 0}^2 + p_\perp^2)^2}$$

p_{T0}: free parameter ~2 GeV

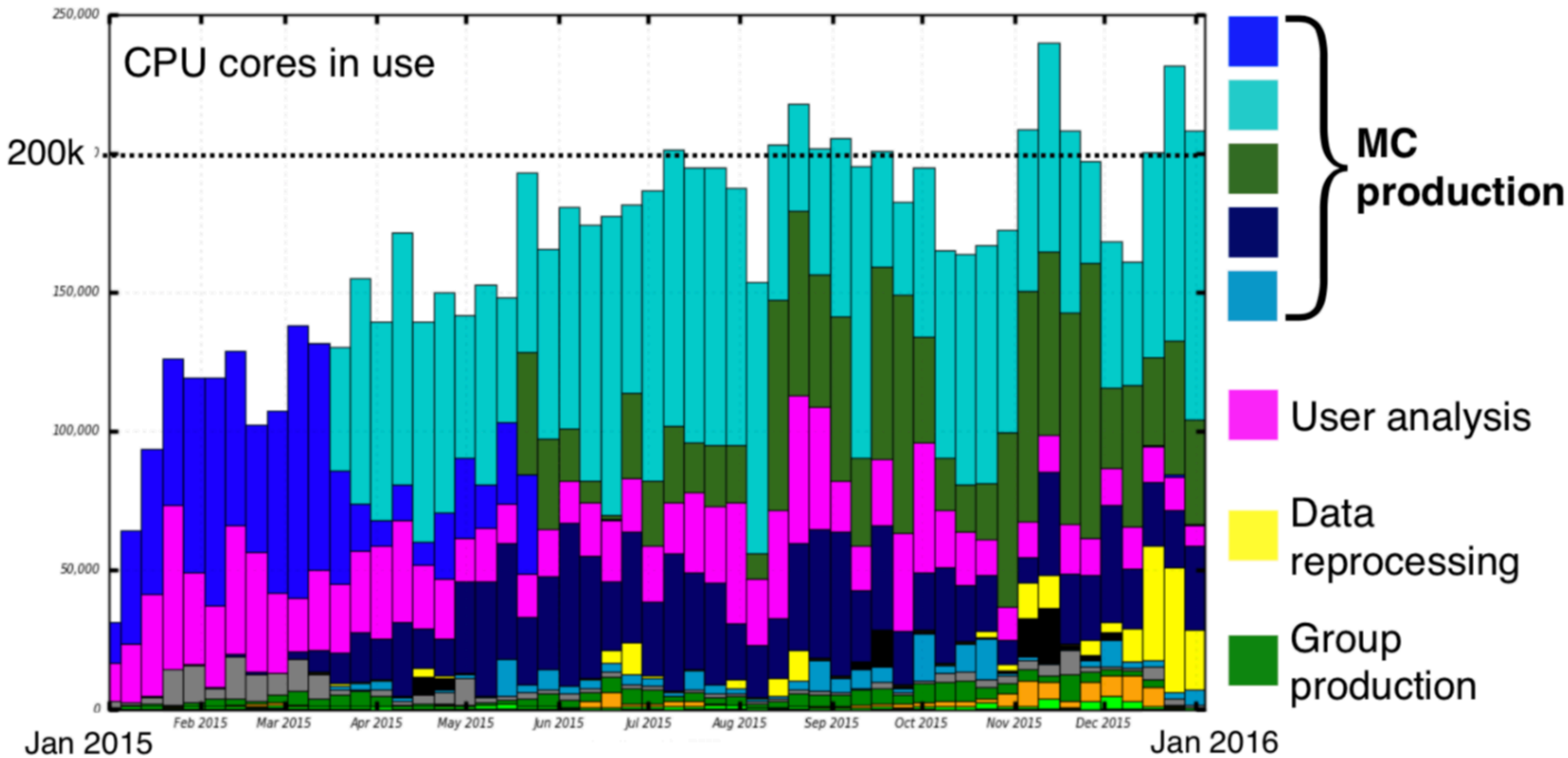
  - Gives rise to additional activity (underlying event, together with the beam remnants).



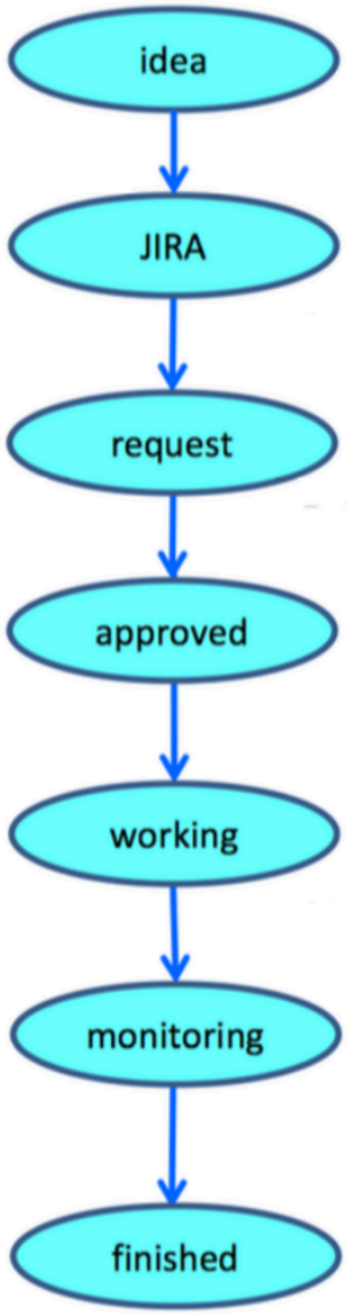


ATLAS MC production system

- ATLAS Production System overview:
  - Most of ATLAS's CPU consumption dedicated to MC production



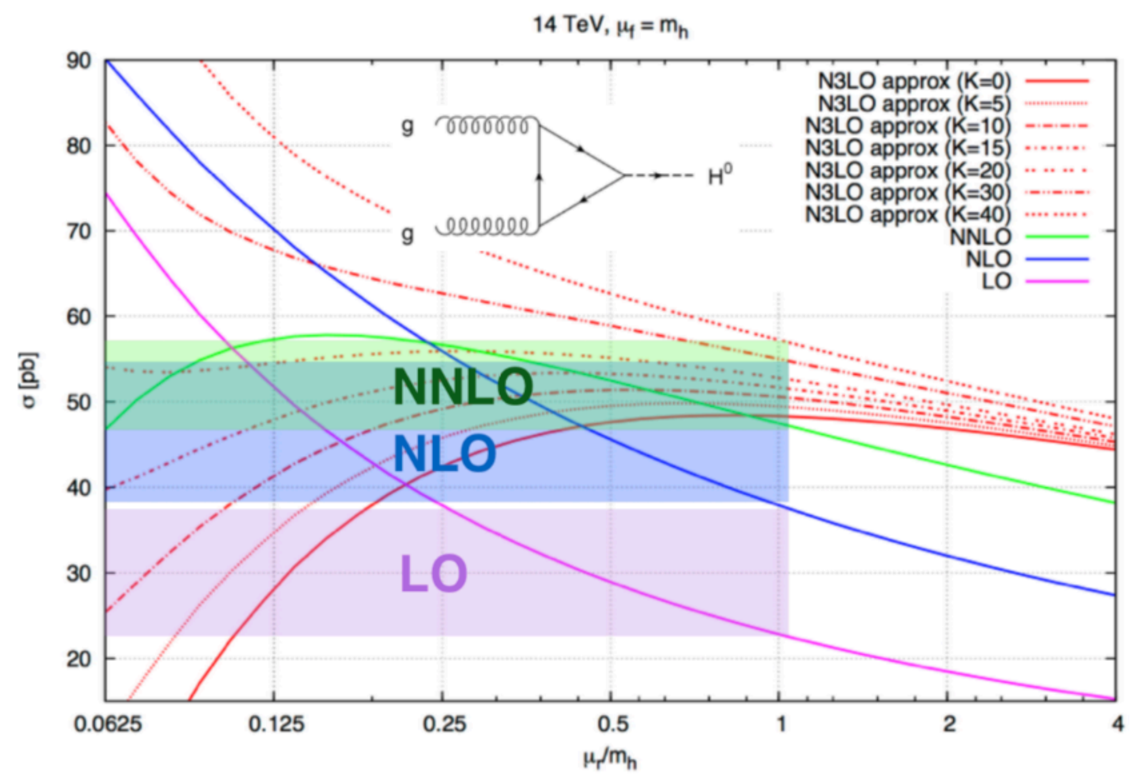
Monte Carlo requests



- Requester talks to their sub-group convenor about MC needs and to their MC contact person about samples
- ATLMCPROD JIRA ticket is created, usually by MC contact
- Production request is created in ProdSys by MC contact, via input spread-sheet
- PMG convenor approves request
- MC production team member takes over the request (becomes manager), modifying it if necessary, before submitting it for production
- MC production manager monitors the request until finished
- MC production manager puts notification in JIRA when done

Scale uncertainties

- QCD has UV divergencies that are cured by renormalization.
  - When truncating the perturbative series we introduce a dependency on an unphysical **renomalization scale**, μ_R.
  - The dependency gets smaller as we include higher-orders
- A similar scale, the **factorisation scale** μ_F, enters when we factorise the collinear divergences absorbed in the PDFs.
- Varying these scales one can get an estimate on the uncertainty from the missing higher-orders.
  - Commonly factors of 0.5 and 2 are used
  - No guarantee that the e.g. NNLO prediction will be contained within the NLO uncertainty band. Can be verified a posteriori (e.g. gg->H)



Generator uncertainties

- The prediction from a MC event generator has a large number of uncertainties stemming from various sources
  - The value of fundamental parameters
  - The order of the prediction
  - Validity of theoretical assumptions
  - The phenomenological models used in hadronization, MPI,…
  - Uncertainty in the tuned parameters

- Hard if not impossible to disentangle each source of uncertainty
- Often one is forced to simply compare the output of two different programs
  - This is however just an order of magnitude estimate, and should not be taken as a meaningful number!

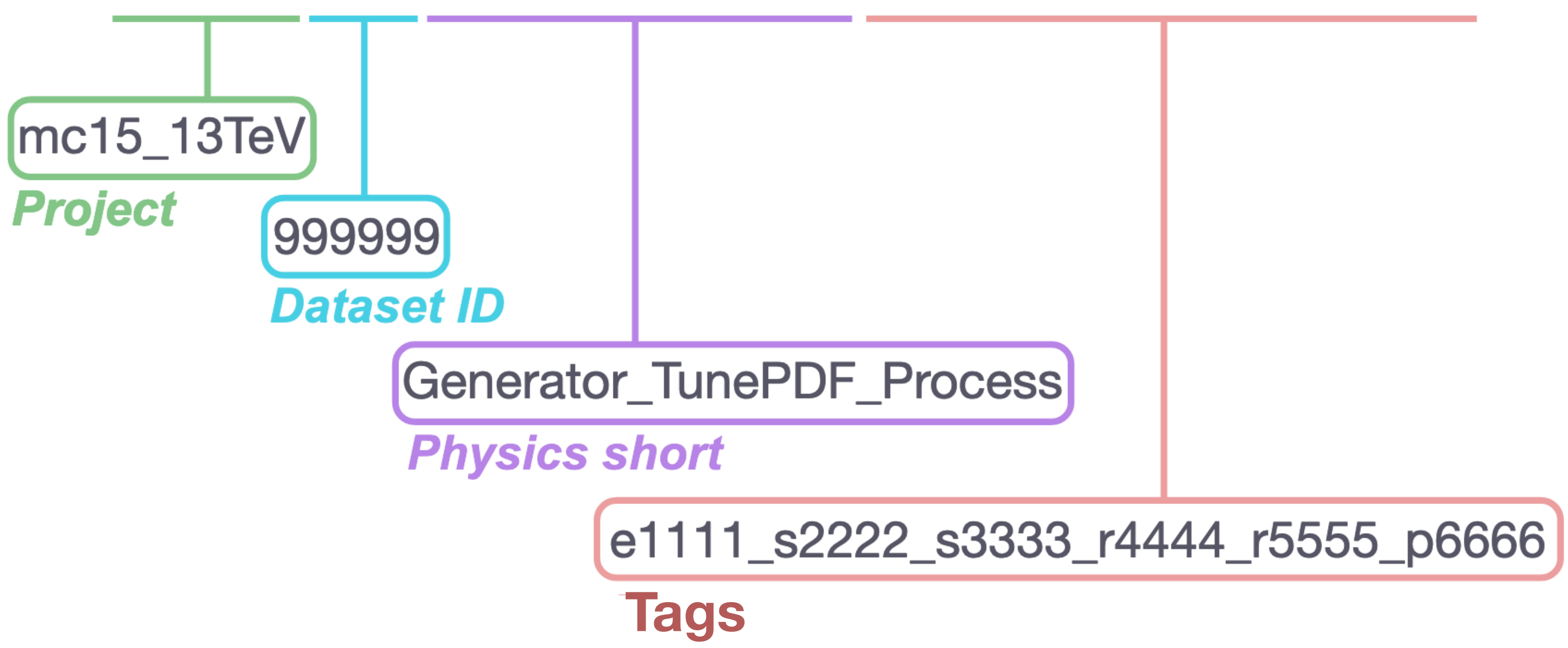
- If significant variations are found one should be very wary of the results, try to identify the source of the disagreement (ME, PS, hadronization, …) and discuss with the authors how to best address it

Dataset name

- Monte Carlo dataset names follow the structure:

`project.datasetNumber.physicsShort.prodStep.dataType.AMITag`

  - mc15_13TeV.999999.Generator_TunePDF_Process.e1111_s2222_s3333_r4444_r5555_p6666



Dataset name

- physicsShort provides some detail of what has been produced in this sample

  - *E.g.*
    - *MadGraphPythia8EvtGen_A14NNPDF23LO_Zee_Np4*

    - **Generator**: MadGraph
    - **Shower**: Pythia8
    - **Afterburner**: EvtGen
    - **Tune**: A14
    - **PDF**: NNPDF2.3LO
    - **Process**: $Z \rightarrow ee$
    - **Filter/Detail**: Np4 (i.e. this is the 4-additional parton sample)
      - Could also have other details here, e.g. filter name, mass of particles, etc.

Finding existing JO-old WF

- JOs are identified with an unique dataset ID (DSID):
  - mc15_13TeV.999999.*   999999 would be the DSID
- They can be found all be found in gitlab:
  - https://gitlab.cern.ch/atlas-physics/pmg/infrastructure/mc15joboptions
- The JOs live in the share/DSID[YZW]xxx folder
  - YZW are the first 3 digits of the DSID
  - E.g. for the dataset: mc15_13TeV.361036.Pythia8_A2MSTW2008LO_minbias_inelastic.evgen.EVNT.e3580
  - The JO is in the share/DSID361xxx folder

master | mc15joboptions / share / DSID361xxx / **MC15.361036.Pythia8_A2MSTW2008LO_minb...** | Find file | Blame | History | Permalink

MC15.361033.Pythia8EvtGen_A2MSTW200...	reduced minevents on MB and jetjet	4 years ago
MC15.361034.Pythia8EvtGen_A2MSTW200...	reduced minevents on MB and jetjet	4 years ago
MC15.361035.Pythia8EvtGen_A2MSTW200...	reduced minevents on MB and jetjet	4 years ago
MC15.361036.Pythia8_A2MSTW2008LO_mi...	add 361214-24	4 years ago
MC15.361037.Pythia8_A2MSTW2008LO_mi...	add MB JO without EvtGen	4 years ago

JobOptions- a more complex example

- Let's look at

MC15.361500.MadGraphPythia8EvtGen_A14NNPDF23LO_Zee_Np0.py 62 Bytes

```
include('MC15JobOptions/MadGraphControl_Zjets_LO_Pythia8.py')
```

- It includes the "top" JO from the common area:

MadGraphControl_Zjets_LO_Pythia8.py 6.49 KB

Edit Web IDE Replace Delete

```
from MadGraphControl.MadGraphUtils import *

# General settings
nevents=150000
mode=0
nJobs=1
gridpack_dir=None
gridpack_mode=False
cluster_type=None
cluster_queue=None

# MG Particle cuts
mllcut=40

# Merging settings
maxjetflavor=5
ickkw=0
nJetMax=4
ktdurham=30
dparameter=0.4


### DSID lists (extensions can include filters, systematics samples, etc.)
Zee_5fl_Np0=[361500]
Zee_5fl_Np1=[361501]
Zee_5fl_Np2=[361502]
Zee_5fl_Np3=[361503]
Zee_5fl_Np4=[361504]

```

New WF: The DSID will not be specified in the JO name but the runNumber will be given as an argument to the generator transform.

JobOptions- a more complex example

- Process definition:

```
### Electrons
if runArgs.runNumber in Zee_5fl_Np0:
    mgproc="generate p p > e+ e- @0"
    name='Zee_Np0'
    process="pp>e+e-"
elif runArgs.runNumber in Zee_5fl_Np1:
    mgproc="generate p p > e+ e- j @1"
    name='Zee_Np1'
    process="pp>e+e-"
elif runArgs.runNumber in Zee_5fl_Np2:
    mgproc="generate p p > e+ e- j j @2"
    name='Zee_Np2'
    process="pp>e+e-"
elif runArgs.runNumber in Zee_5fl_Np3:
    mgproc="generate p p > e+ e- j j j @3"
    name='Zee_Np3'
    process="pp>e+e-"
elif runArgs.runNumber in Zee_5fl_Np4:
    mgproc="generate p p > e+ e- j j j j @4"
    name='Zee_Np4'
    process="pp>e+e-"

    nevents=5000
    gridpack_mode=True
    gridpack_dir='madevent/'

    mode=1
    cluster_type='pbs'
    cluster_queue='medium'
    nJobs=20
```


```
if runArgs.runNumber in Zee_5fl_Np0:
    mgproc="generate p p > e+ e- @0"
    name='Zee_Np0'
    process="pp>e+e-"
```

JobOptions- a more complex example

Select your model, and add the definition of the process

```
fcard = open('proc_card_mg5.dat','w')
fcard.write("""
import model sm-no_b_mass
define p = g u c d s b u~ c~ d~ s~ b~
define j = g u c d s b u~ c~ d~ s~ b~
"""+mgproc+"""
output -f
""")
fcard.close()


beamEnergy=-999
if hasattr(runArgs,'ecmEnergy'):
    beamEnergy = runArgs.ecmEnergy / 2.
else:
    raise RuntimeError("No center of mass energy found.")


process_dir = new_process(grid_pack=gridpack_dir)
```

Set generation parameters

```
#Fetch default LO run_card.dat and set parameters
extras = { 'lhe_version'   : '2.0',
           'cut_decays'    : 'F',
           'pdlabel'       : "'nn23lo1'",
#           'pdlabel'       : "'lhapdf'",
#           'lhaid'         : 247000,
           'maxjetflavor'  : maxjetflavor,
           'asrwgtflavor'  : maxjetflavor,
           'ickkw'         : 0,
           'ptj'           : 20,
           'ptb'           : 20,
           'mmll'          : mllcut,
           'mmjj'          : 0,
           'drjj'          : 0,
           'drll'          : 0,
           'drjl'          : 0.4,
           'ptl'           : 0,
           'etal'          : 10,
           'etab'          : 6,
           'etaj'          : 6,
           'ktdurham'      : ktdurham,
           'dparameter'    : dparameter,
           'use_syst'      : 'F' }
```

PDF

Cuts between particles

Matching parameters

JobOptions- a more complex example

- Generation, Pythia8 shower settings and matching options

Parton level generation

```
build_run_card(run_card_old=get_default_runcard(proc_dir=process_dir),run_card_new='run_card.dat',
               nevts=nevents,rand_seed=runArgs.randomSeed,beamEnergy=beamEnergy,xqcut=0.,
               extras=extras)
print_cards()

generate(run_card_loc='run_card.dat',param_card_loc=None,mode=mode,njobs=nJobs,proc_dir=process_dir,
         grid_pack=gridpack_mode,gridpack_dir=gridpack_dir,cluster_type=cluster_type,cluster_queue=cluster_queue,
         nevents=nevents,random_seed=runArgs.randomSeed)

arrange_output(proc_dir=process_dir,outputDS=stringy+'._00001.events.tar.gz')

```

Pythia settings

```
#### Shower
evgenConfig.description = 'MadGraph_'+str(name)
evgenConfig.keywords+=['Z','electron','jets','drellYan']
evgenConfig.inputfilecheck = stringy
runArgs.inputGeneratorFile=stringy+'._00001.events.tar.gz'

include("MC15JobOptions/Pythia8_A14_NNPDF23LO_EvtGen_Common.py")
include("MC15JobOptions/Pythia8_MadGraph.py")


PYTHIA8_nJetMax=nJetMax
PYTHIA8_TMS=ktdurham
PYTHIA8_Dparameter=dparameter
PYTHIA8_Process=process
PYTHIA8_nQuarksMerge=maxjetflavor
include("MC15JobOptions/Pythia8_CKKWL_kTMerge.py")
```

Running the generation

- Once we have the complete JO, we can run it using the "**generate transform**":

```
Generate_tf.py
--ecmEnergy=13000
—firstEvent=1
--runNumber=361500
--maxEvents=-1
--randomSeed=1234
--jobConfig=
MC15.361500.MadGraphPythia8EvtGen_A14NNPDF23LO_Zee_Np0.py
--outputEVNTFile=test_Zee_361500.EVNT.root
```

Center-of-mass energy

EventNumber of first event

DSID

Maximum number of events

Random number seed

jobOption

Output EVNT file

- Use the common "-h" (Generate_tf.py -h) to see all supported arguments