CERN

SCUOLA NORMALE SUPERIORE

CMS

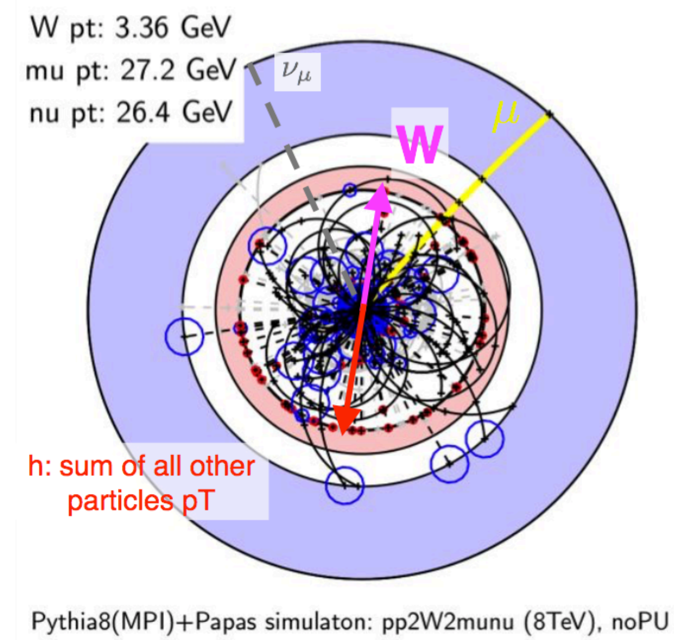
in collaboration with

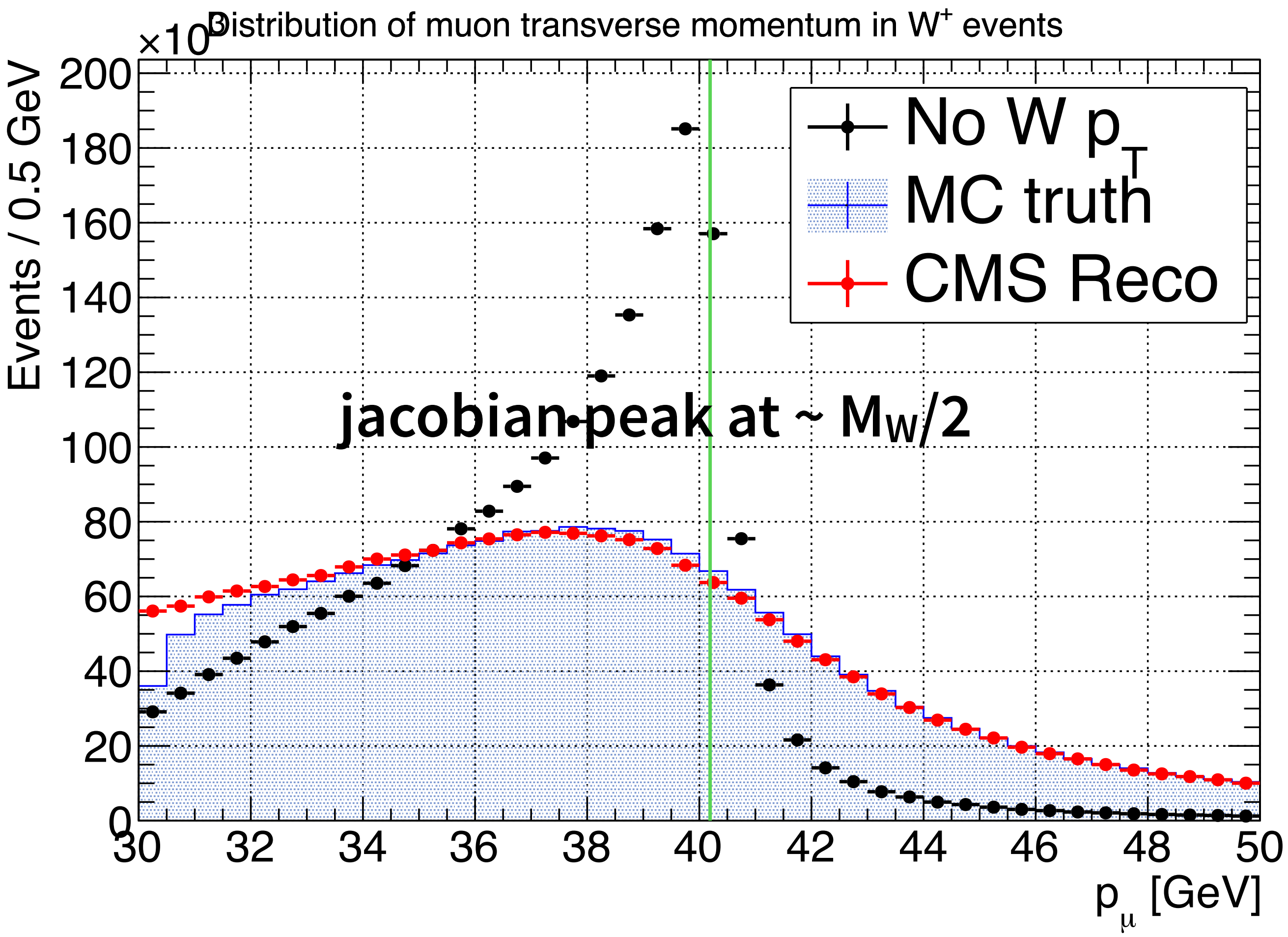
ROOT Data Analysis Framework

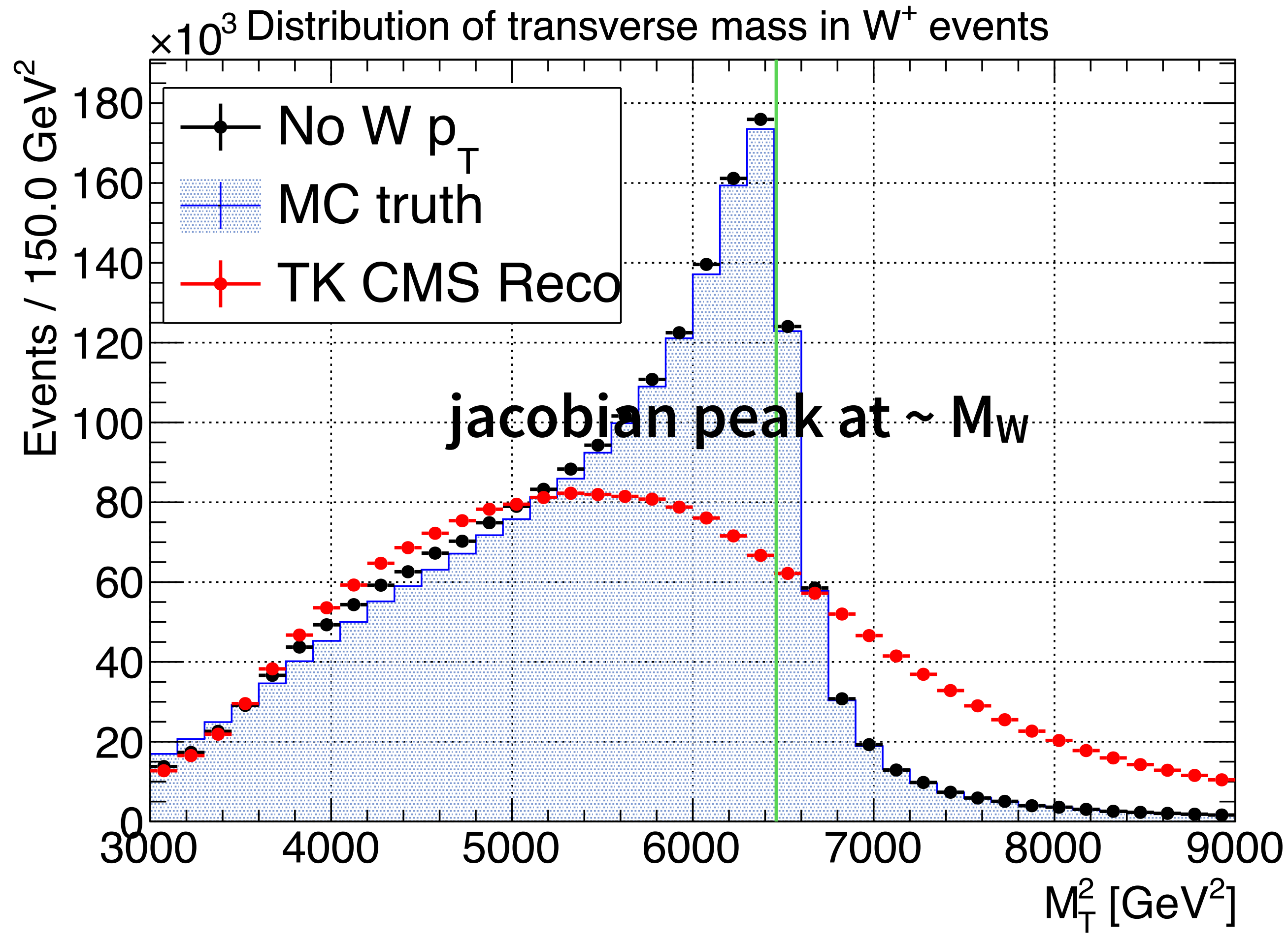
Elisabetta Manca - Scuola Normale Superiore, INFN Pisa & CERN

Misura della massa del W con tecniche di Big Data

need to look at transverse quantities due to the presence of the neutrino in the final state

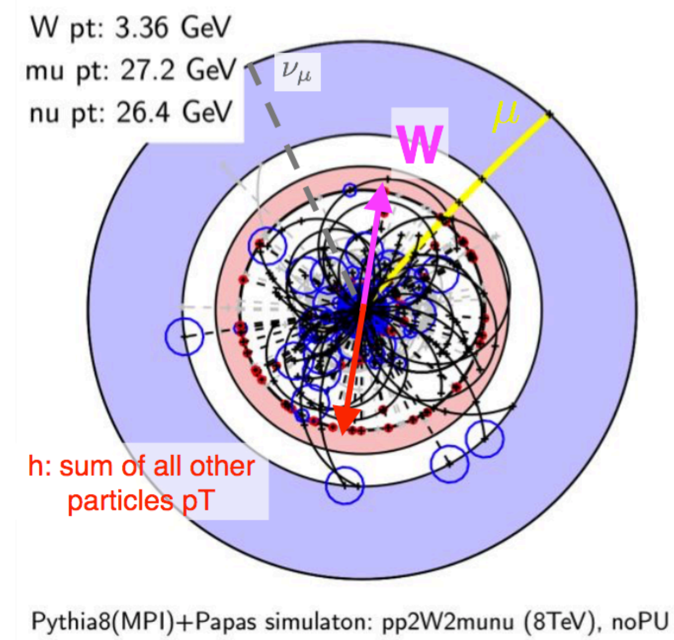






W mass in a "traditional" way

however transverse quantities are **not Lorentz invariant**
measurement is strongly dependent on the **production model**



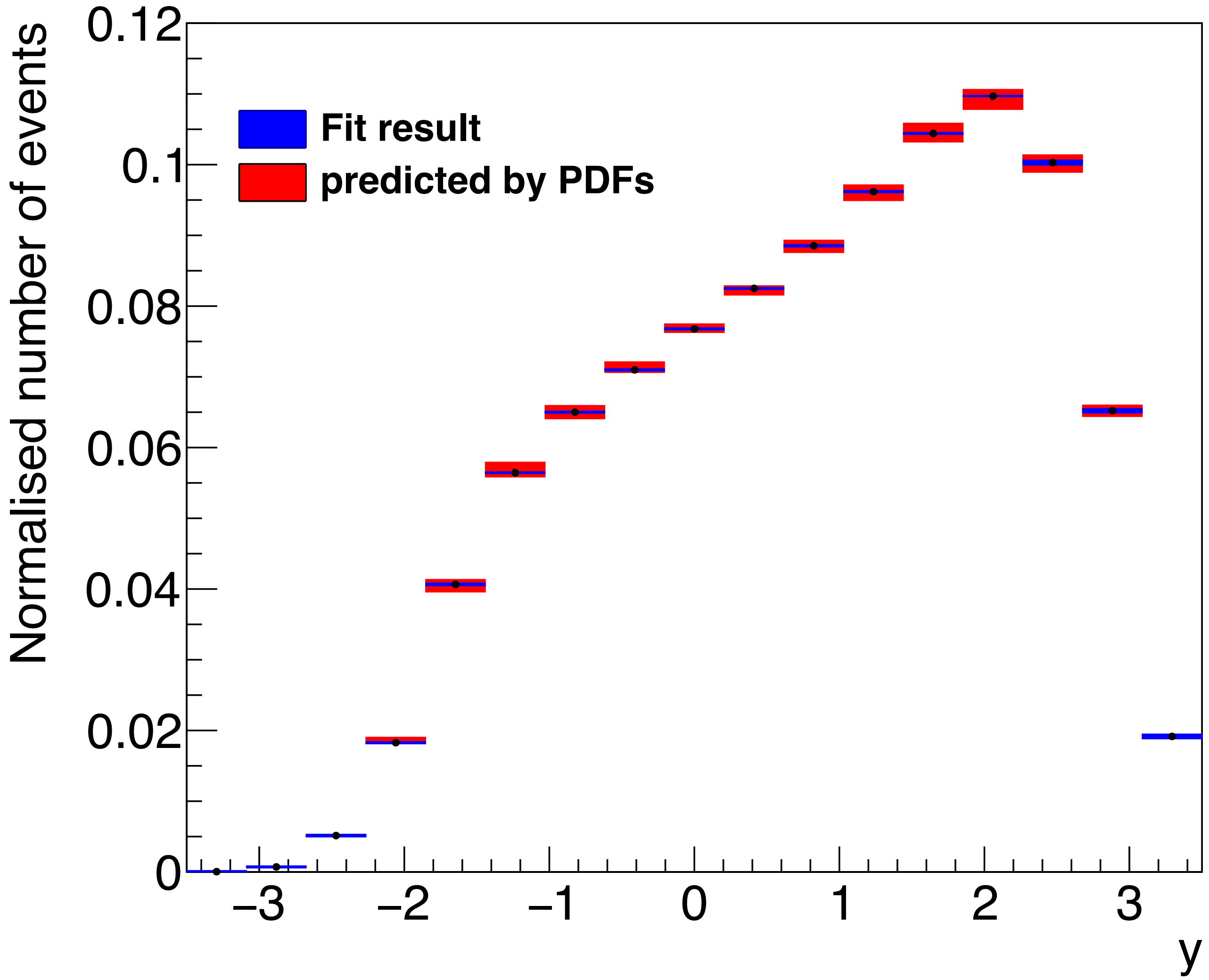
Combined categories	Value [MeV]	Stat. Unc.	Muon Unc.	Elec. Unc.	Recoil Unc.	Bckg. Unc.	QCD Unc.	EW Unc.	PDF Unc.	Total Unc.	χ^2/dof of Comb.
m_{T}-p_{T}^{ℓ}, $W^{\pm}$, e-μ	80369.5	6.8	6.6	6.4	2.9	4.5	8.3	5.5	9.2	18.5	29/27

uncertainty on ATLAS measurement dominated by **theory modelling** in our paper we have shown that it is possible to unfold the rapidity distribution of the W directly from data

About the rapidity and helicity distributions of the W bosons produced at LHC

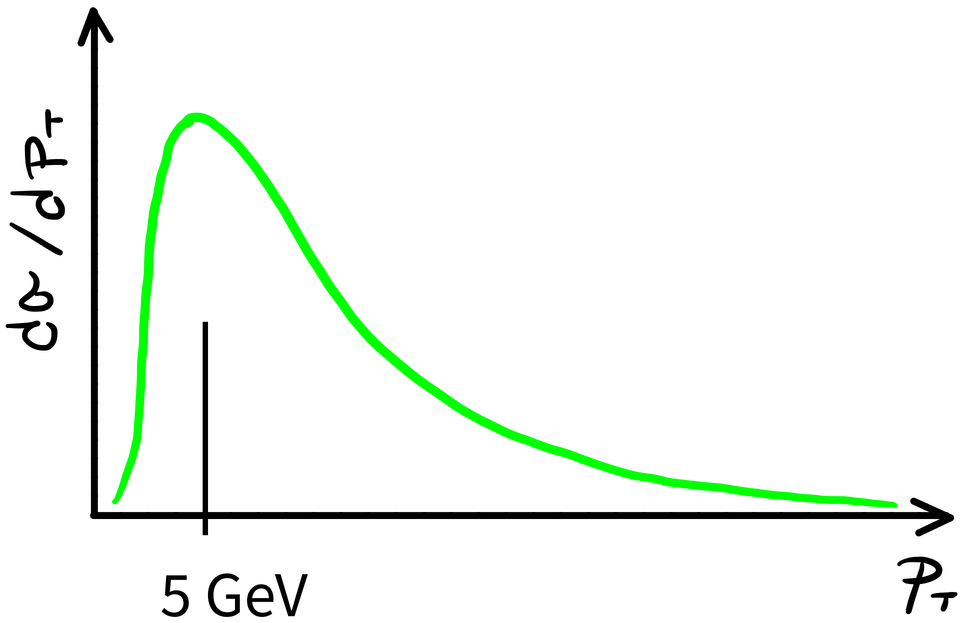
E. Manca,[a,b,1] O. Cerri,[a,b] N. Foppiani[a,b] and G. Rolandi[a,b,c]

[a] *Scuola Normale Superiore, Piazza dei Cavalieri 7, Pisa, 56126 Italy*



error bars in the fit smaller than PDF prediction!

what about transverse momentum?



things start to look much more complicated…

A_i coefficients are functions of W production and multiply spherical harmonics of 2nd order

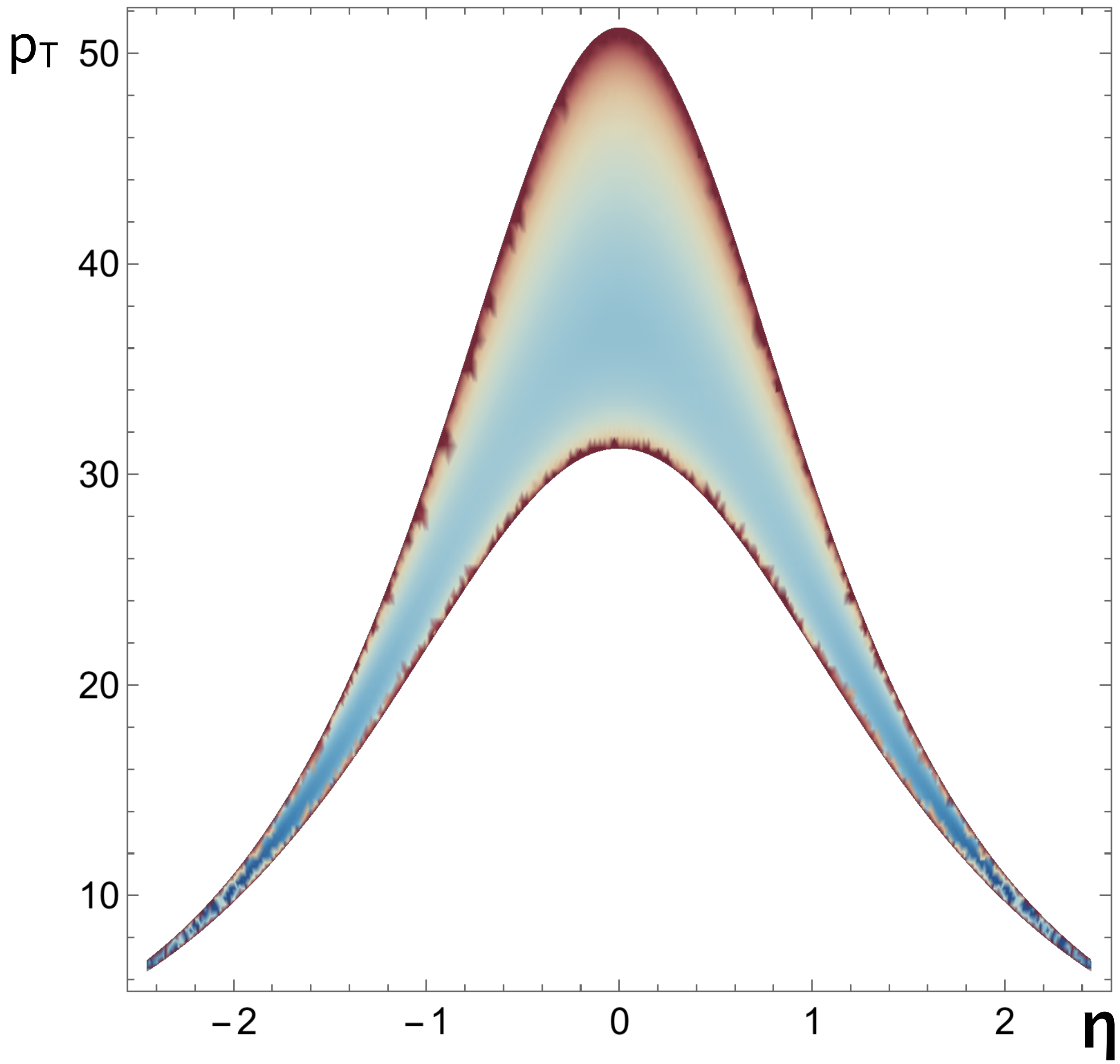
θ and φ of lepton in W rest frame

W p_T and rapidity in lab frame

$$\frac{d\sigma}{dp_T^2\,dY\,d\cos\theta\,d\phi} = \frac{3}{16\pi}\frac{d\sigma^{U+L}}{dp_T^2\,dY}[(1+\cos^2\theta)$$
$$+1/2\;A_0(1-3\cos^2\theta) + A_1\sin(2\theta)\cos\phi$$
$$+1/2\;A_2\sin^2\theta\cos(2\phi) + A_3\sin\theta\cos\phi + A_4\cos\theta$$
$$+A_5\sin^2\theta\sin(2\phi) + A_6\sin(2\theta)\sin\phi + A_7\sin\theta\;\;\sin\phi]$$

given **y** and W $\mathbf{p_T}$,
a lepton in the lab frame has a $\mathbf{p_T}$, **η** and a **φ** (with respect to the **φ** of the W)

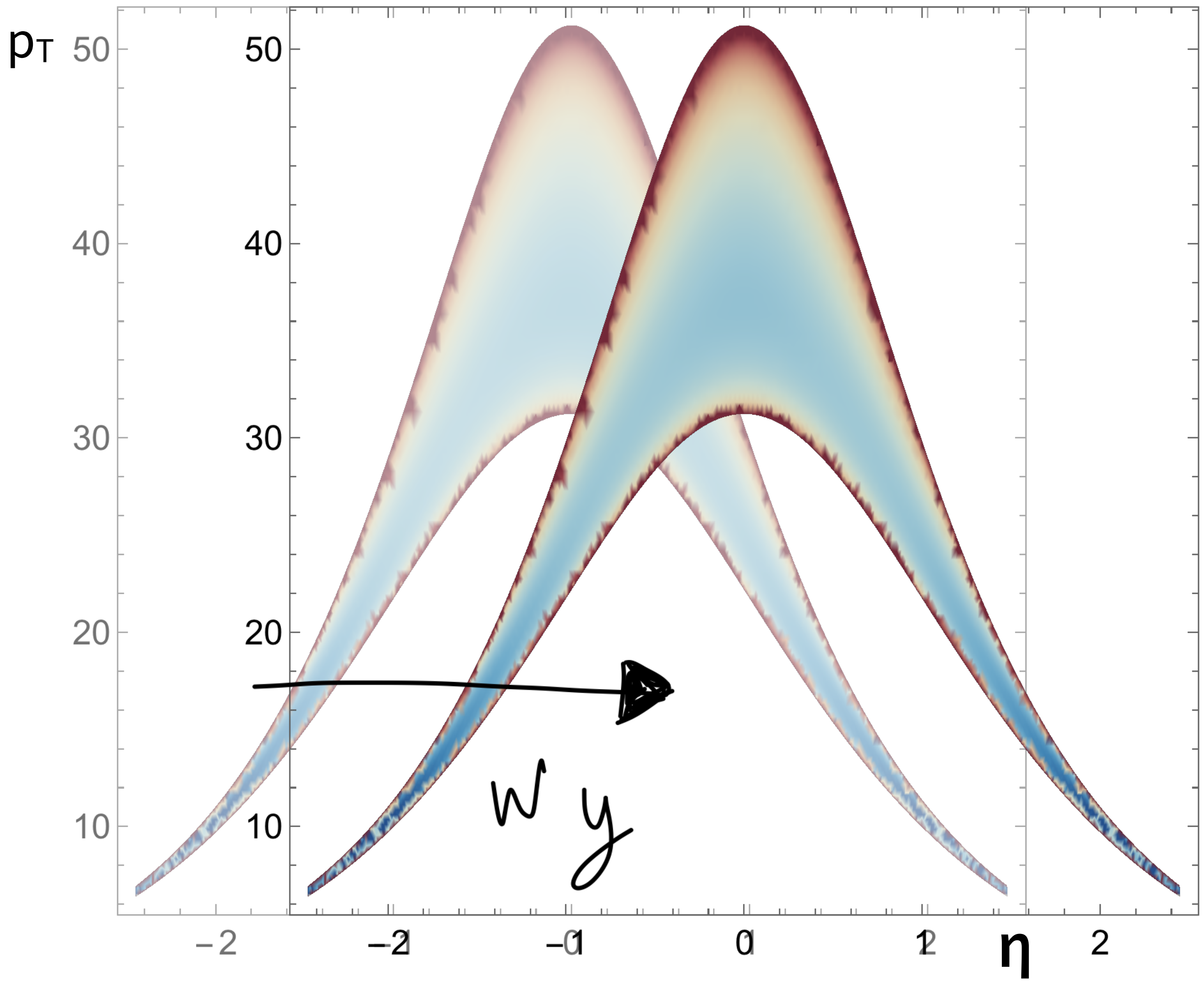
however the three variables have a constraint (in rest frame E*= M/2)



$y = 0$, W $p_T = 20$ GeV

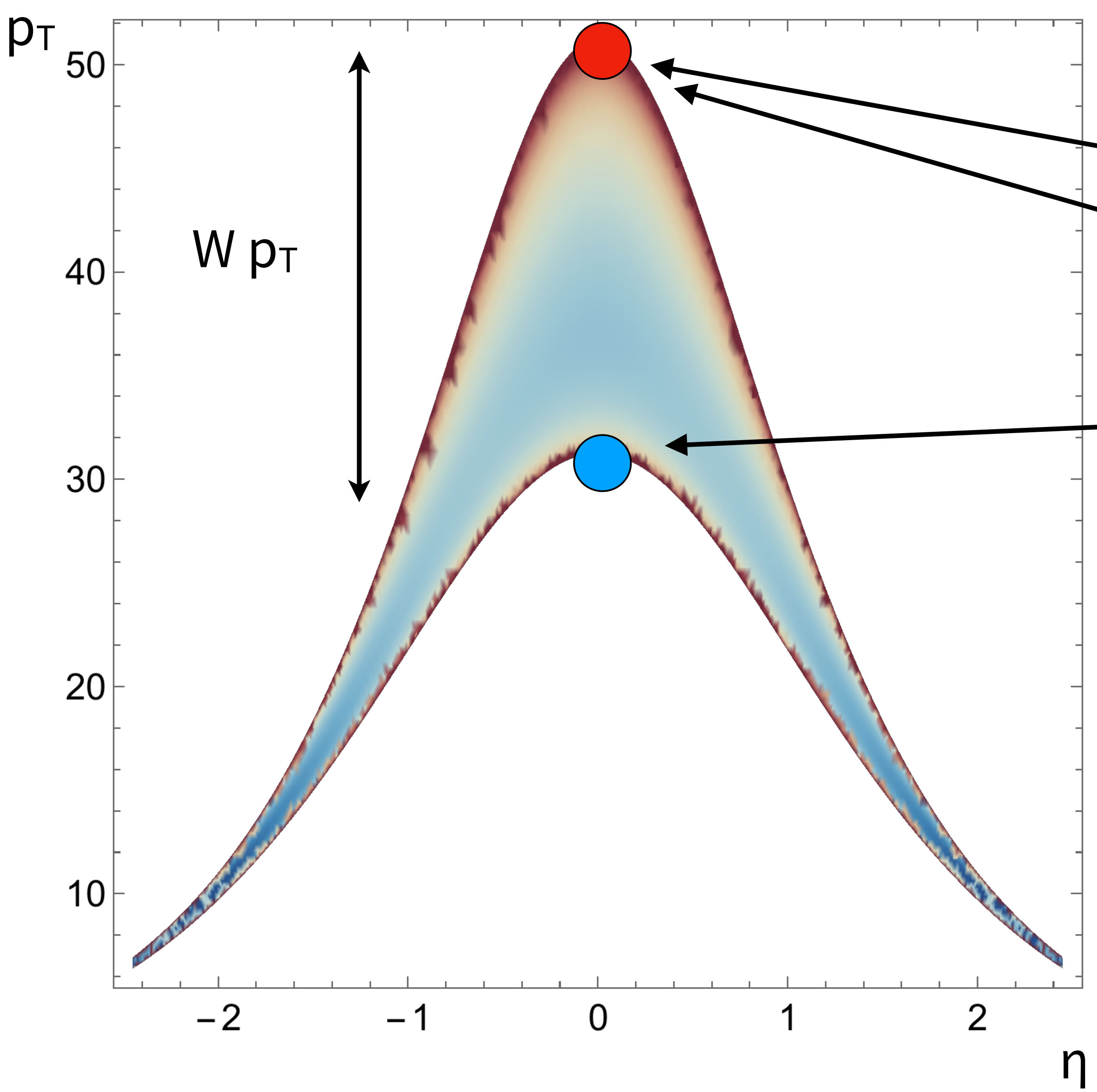
given **y** and W $\mathbf{p_T}$,
a lepton in the lab frame has a $\mathbf{p_T}$, **η** and a **φ** (with respect to the **φ** of the W)

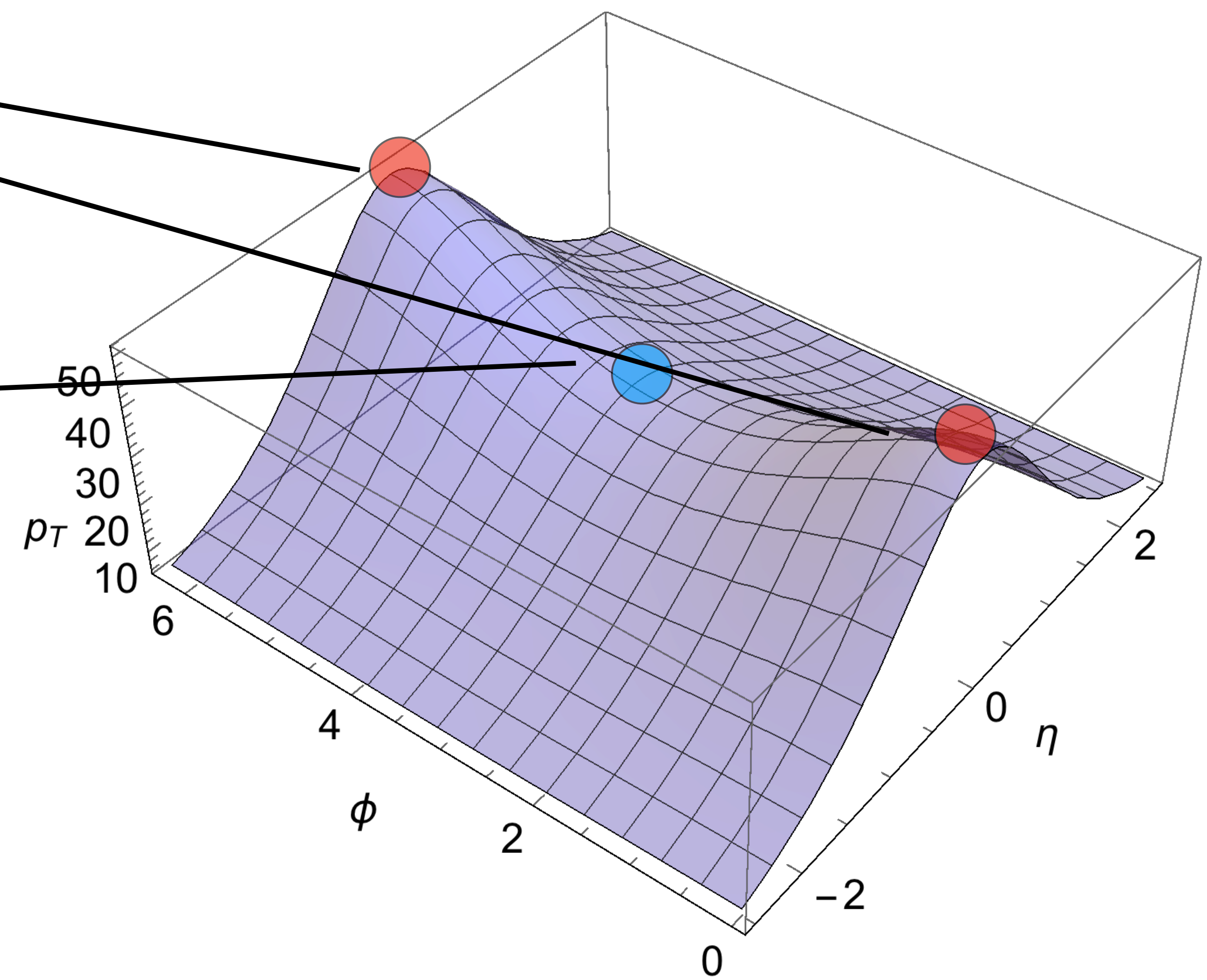
however the three variables have a constraint (in rest frame $E^*= M/2$)



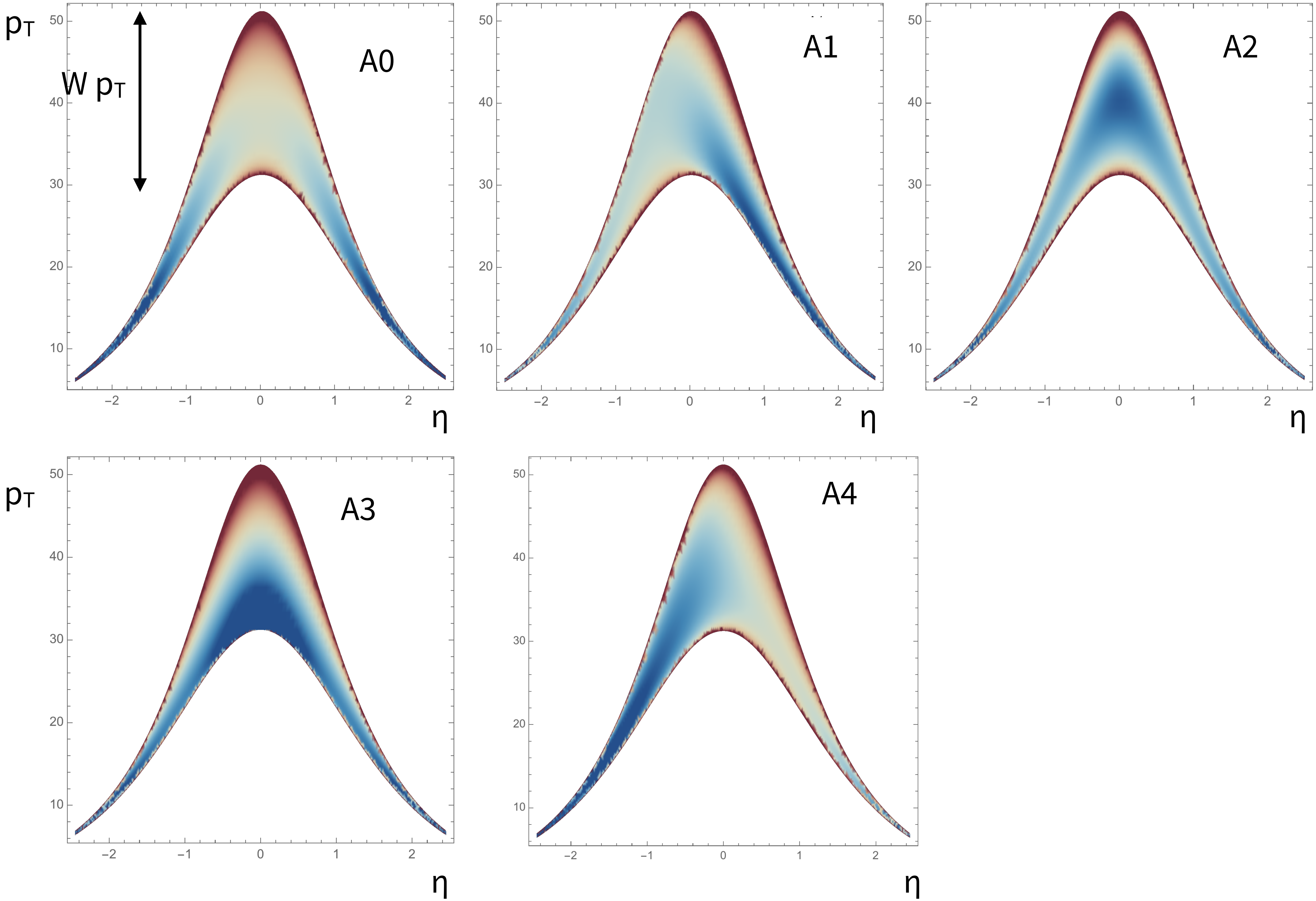
y = 0, W p_T = 20 GeV

y = 0, W p_T = 20 GeV





angular coefficients templates



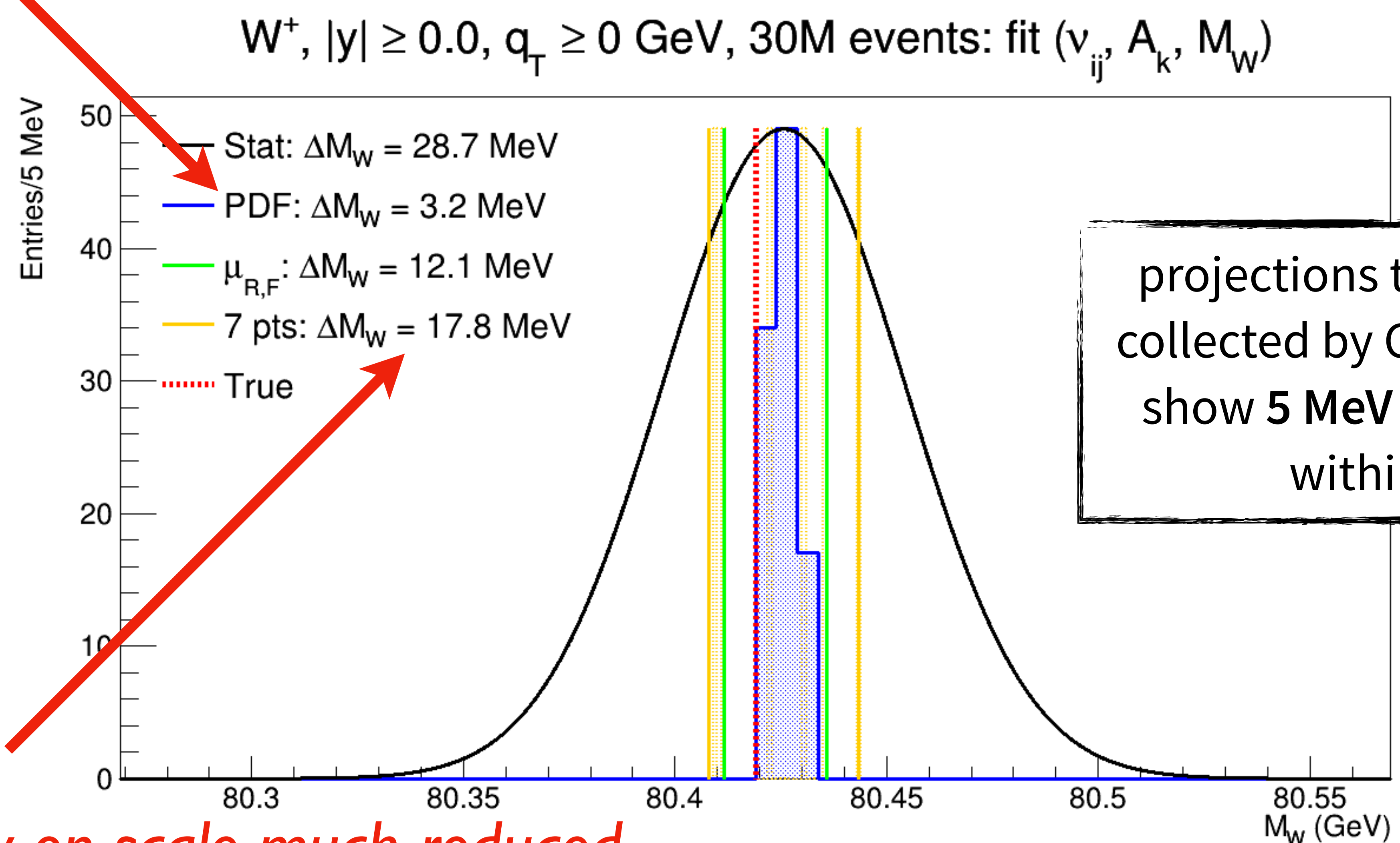
changing basis from helicity amplitudes to angular coefficients

$$\frac{d\sigma_{W^\pm}}{d|y|}_{L,R,0} \quad \rightarrow \quad \frac{d\sigma^{UL}_{W^\pm}}{d|y|\,dq_T},\ A^{\pm}_{0,1,2,3,4}(q_T,|y|),\quad M_W$$

and adding a finer binning in W pT

it is possible to measure simultaneously the double differential distribution of the W rapidity and pT ***and its mass***

Uncertainty on PDF negligible (<3 MeV)



projections to data sample collected by CMS in full Run2 show **5 MeV uncertainty** is within reach

Uncertainty on scale much reduced (residual unc. likely due to stat. fluctuation)

some numbers

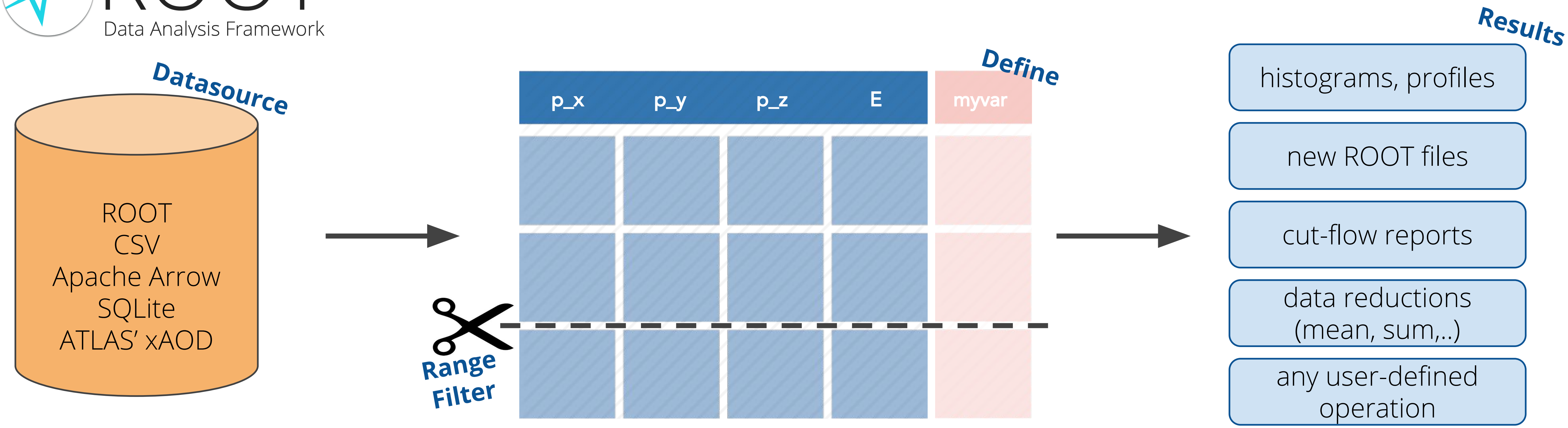
CMS data taking period	**2016**	**2017**	**2018**
number of events into acceptance	35 fb^{-1} ~100 M	45 fb^{-1} ~130 M	65 fb^{-1} ~185 M

NB: for the analysis to work, MC must have ~10 more statistics than data!

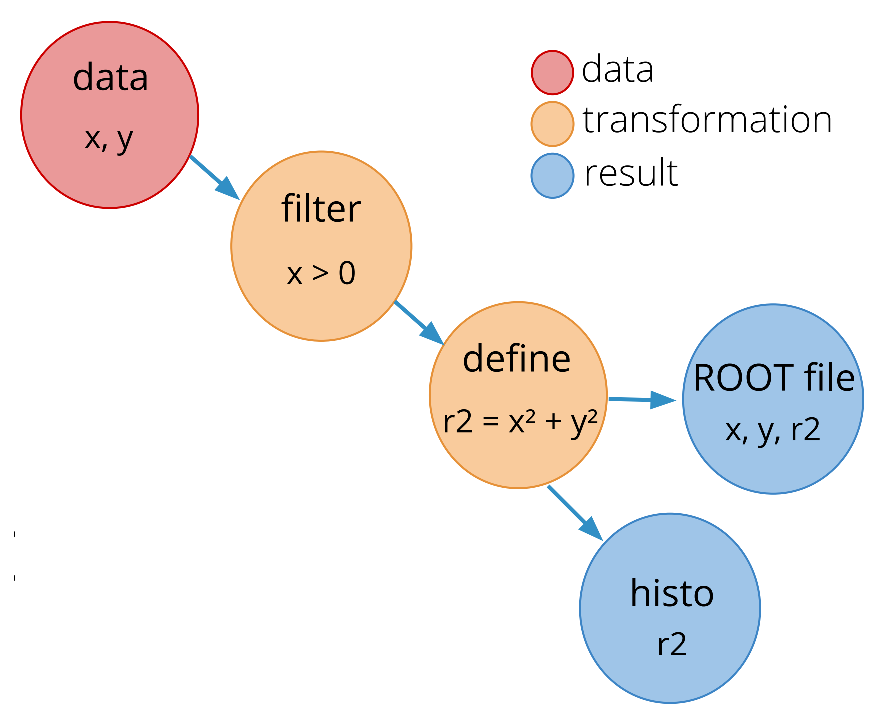
do we have the tools to process such a huge number of events?

ROOT
Data Analysis Framework

RDataFrame



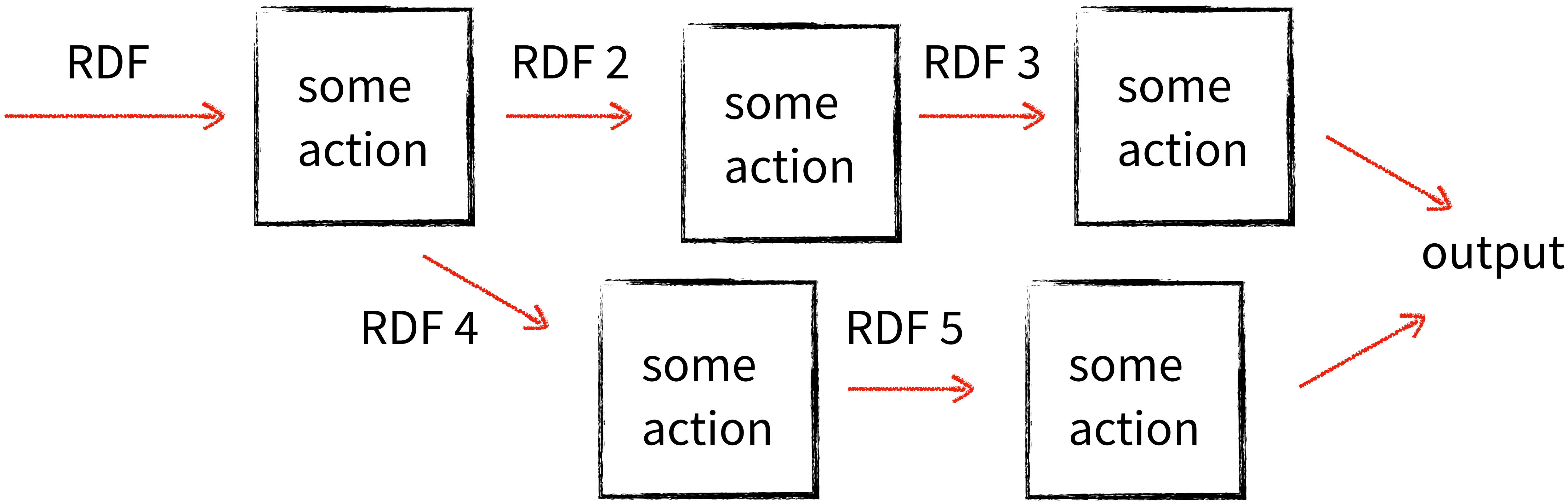
your analysis as a graph of operations



lazy execution guarantees that the event loop is run only once

developed to be transparently parallelisable through Implicit MultiThreading

the idea: wrap RDF into a mini-framework that executes some modules following a graph path



some action = a group of actions on RDF
i.e. apply a calibration, filter events,
fill histograms….

some action

= a python class

= a C++ class (inherits from a virtual mother class)

```python
class filter:

    def __init__(self, string):

        self.myTH1 = []
        self.myTH2 = []
        self.myTH3 = []
        self.string = string

    def run(self,d):

        self.d = d.Filter(self.string)
        return self.d

    def getTH1(self):

        return self.myTH1

    def getTH2(self):

        return self.myTH2

    def getTH3(self):

        return self.myTH3
```

```cpp
#include “filter.h"

RNode filter::run(RNode d){

   auto d1=d.Filter("Mystring_");

   return d1
}

std::vector<ROOT::RDF::RResultPtr<TH1D>> filter::getTH1(){

   return _h1List;

}

std::vector<ROOT::RDF::RResultPtr<TH2D>> filter::getTH2(){

   return _h2List;

}

std::vector<ROOT::RDF::RResultPtr<TH3D>> filter::getTH3(){

   return _h3List;

}
```

an example of action: extraction of the "Angular Coefficients" from a WJets Montecarlo

spherical harmonics 2nd order (W has spin 1!)

task: compute

$$m = \frac{\sum P_k(\cos\theta, \phi) w_i}{\sum w_i}$$

$$\sigma_m = \sqrt{\sigma^2_{P_k} \frac{\sum w_i^2}{(\sum w_i)^2}}$$

for each bin of W p_T and y
for each harmonics k = 0,…,7

implementation:

- 1 TH2 filled with *w*

for each harmonics (0 to 7):

- 1 TH2 filled with P_k and weighted with *w*
- 1 TH2 filled with $P_k{}^2$ (to compute variance)

in RDF language:
about 10 Filters and 10 Defines

results

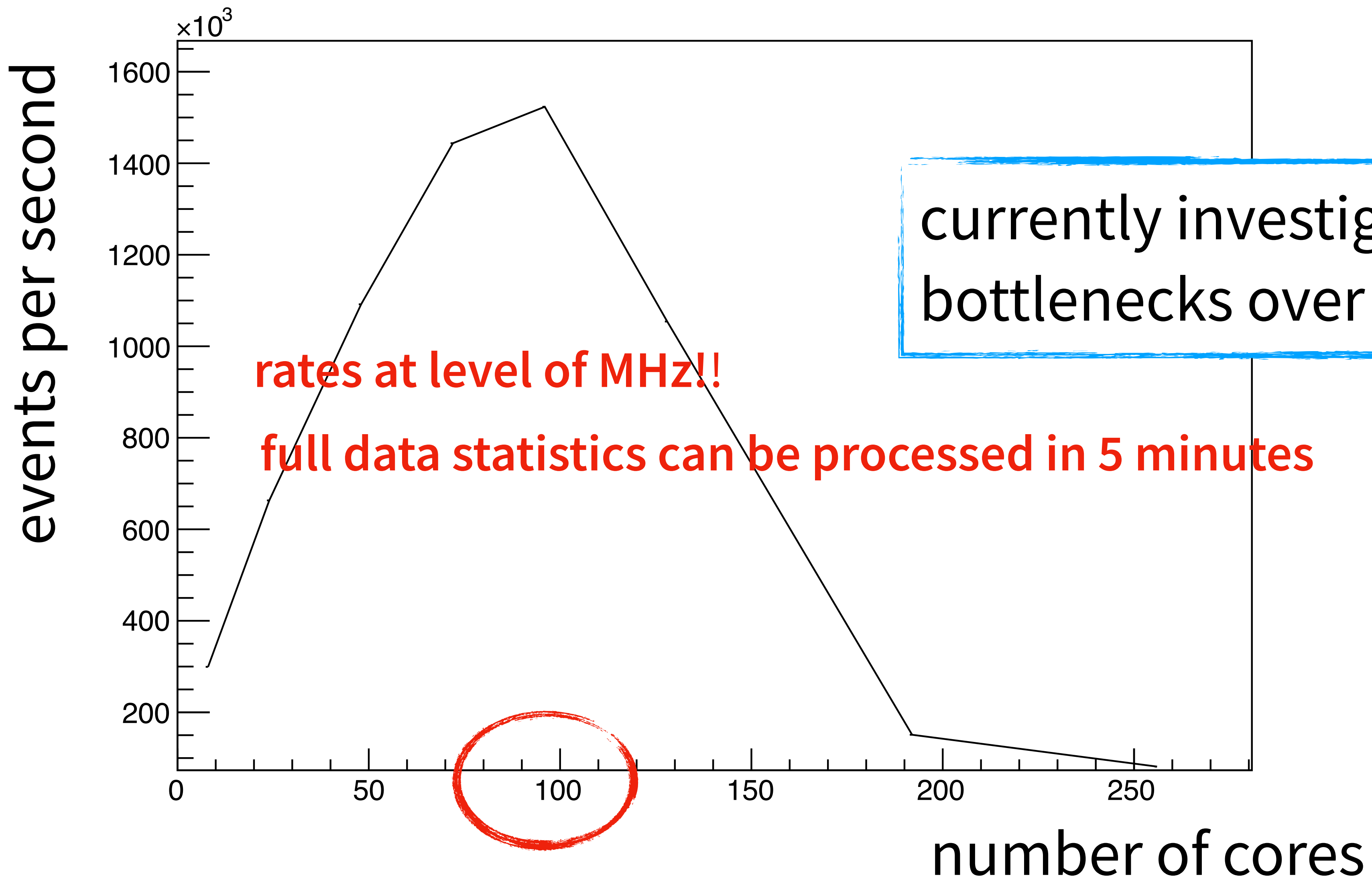
a scaling plot:

2*192 cores Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz

Lots of SSD storage, bleeding edge hardware

@ **Scuola Normale Superiore**

27 GB input data (~4000 cluster)



currently investigating bottlenecks over 100 cores

rates at level of MHz!!

full data statistics can be processed in 5 minutes