

# Keeping Track of MC Weights: Why and What Do We Need?

---

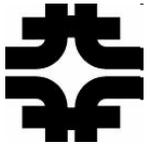
- Examples of MC Weights
- What do we need stored?
- MC Weight Object Proposal



Charles Plager (UCLA/FNAL)



# Weighting MC



- Whenever we want to compare more than one MC sample together, they need to be properly weighted.
  - All samples should be normalized to correspond to the same amount of integrated luminosity.

$$\text{weight}(\int \mathcal{L} dt) = \frac{\int \mathcal{L} dt \cdot \text{cross section}}{\text{number of events considered}}$$

- For example, I want to compare distributions of  $W + \text{jets}$  and  $\text{top}$ . Assuming an integrated luminosity of  $1 \text{ fb}^{-1}$ , I get:

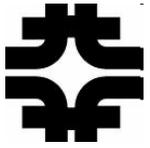
Sample	Cross section	Number of MC Events	Weight
$W + \text{jets}$	24 nb	1,204,434	19.926
Top	165 pb	221,331	0.745

(Remember:  $1 \text{ nb} = 1 \cdot 10^6 \text{ fb}$  and  $1 \text{ pb} = 1 \cdot 10^3 \text{ fb}$ )

- After running analysis jobs, make sure we scale/multiply all histograms/numbers by the proper weight.
- This information is currently **not** stored in CMS MC.



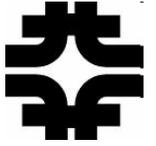
# More Complicated Example



- Instead of using Madgraph samples for  $W + \text{jets}$ , what if we want to use the Alpgen sample?
  - Instead of a single sample, we now have to properly combine 20 different pieces ( = 4 pt bins  $\cdot$  5 parton pieces) together.
- OK, we can do this. We just need to :
  - Calculate 20 different weights (because each sample has a different number of counts and cross section)
  - Run 20 sets of jobs to produce whatever histograms, numbers, etc. we want
  - Successfully combine the right histograms/numbers with the right weights.
- *“Isn’t  $Z + \text{jets}$  a background, too?”*
  - So we need to do the same thing for  $Z + \text{jets}$ , dibosons, three variations single top, ...
- Is there an easier, more robust way?
  - ⇒ Store necessary information **per event**.



# “Wait... This Looks Like a MC Soup??!!”



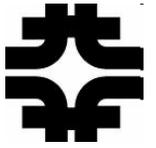
- In the past, the data-ops group tried to make a “MC soup” and hand it to analysts.
  - “MC soup” is a generic term for mixing different MC samples into a single job/file.
- This did not work well for data ops, nor for the analysts.
  - Among other issues, it is not clear that enough information was stored per event to make this feasible.
  - Large headaches for data-ops...
- This is **NOT** what we are proposing here:
  - Any “souping” will NOT be done by data ops, but rather **only** by individual analysts as well as analysis groups.
  - I have talked to members of data-ops\* and they are okay with this plan.



\* Names available upon request...



# MC Weight Object Proposal



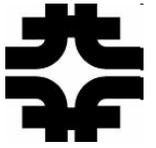
- I want to make a small object that stores:
  - Unique identifier for each sample:
    - “Process Parameter Set Hash” – effectively a checksum of all tracked parameters used to generate MC/process data
  - Cross section for sample
  - Number of events “considered” for this sample.
    - If an analysis filter is used, then one needs to know the number **before** the filter step.
- Each event could then have a weight:

$$\text{weight}(\int \mathcal{L} dt) = \frac{\int \mathcal{L} dt \cdot \text{cross section}}{\text{number of events considered}}$$

- After weighting the events, one can now *soup* the MC with no problems.



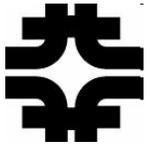
# Why Not Just Store a Weight?



- *“If all we need to do is use the weight, why store the other pieces as well?”*
- The number of events considered *may* be different for different people:  
*E.g.*,
  - Crab jobs crash
  - May deliberately choose to only run over fraction of a sample
- If people are considering only a fraction of a MC phase space, then the cross section will be smaller as well.
  - *E.g.*, making a high statistics  $\hat{p}_T$  sample by combining several inclusive samples.
- By storing the pieces, we make it possible to update part of the calculation without throwing away all of the information.



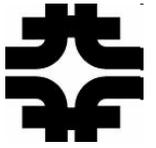
# Where do we add this object?



- When originally envisioned, I thought people would add this information when making their “PATtuples.” ⇒ **Would be user-level object only.**
- Even though all of our MC samples are made in two passes (*i.e.*, generation and then reconstruction), we will **NOT** have all necessary information after the generation step.
  - The merging step only knows about several other jobs and not ALL of the jobs. ⇒ **We will not know total event counts.**
- Object producer will be designed that one can update part of the information with keeping the rest of the information. For example:
  - Updating the counts to reflect the sample size you actually ran over while keeping the cross sections the same.
  - Updating cross section numbers when stitching different  $\hat{p}_T$  samples together.



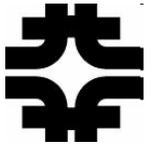
# Where are Cross Section and Counts Stored?



- We want users to be able to easily create/update the information stored in these objects.
- We want to track via provenance where this information came from.
- First solution: Text files
  - Keep text file name and (non-comment) checksum in provenance.
  - Users can post these text files to the UserCode CVS repository.
- Second solution: ???



# Summary



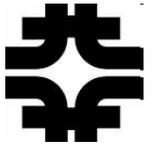
- Creating and using a MC Weight Object would be:
  - Easy to do
  - Allow users to create MC soups
  - Make my mother happy.





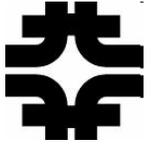
# Backup/Old Slides

---





# *“Wait... I don't want to soup...”*

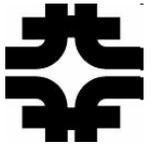


- *“Wait... I don't use Alpgen and I only have a few different samples. Does any of this help me?”*  
⇒ Yes!
- Even if you are only processing one sample at a time, you still need to normalize each MC sample to the amount of integrated luminosity you are analyzing.
- This is much easier to do if your MC already knows how to normalize itself.





# MC Soup



- “MC soup” is a generic term for mixing different MC samples into a single job/file.
- Why would we want to use a soup? For example:
  - W + light flavor jets Alpgen sample currently made up of many different pieces (4 pt bins by 5 n parton pieces).
  - Why run 20 jobs when you can just make it into a “soup?”
    - This can be done either with or without actually merging the samples together into a single file.
- *“O.k. You sold me on using MC soups? What’s the problem?”*
  - Each one of the 20 samples above has a different integrated luminosity equivalent. We have to make sure they are properly weighted before we can “soup” them.
  - With the current technology, this can be painful.
  - Proposal: Store necessary information in each event.