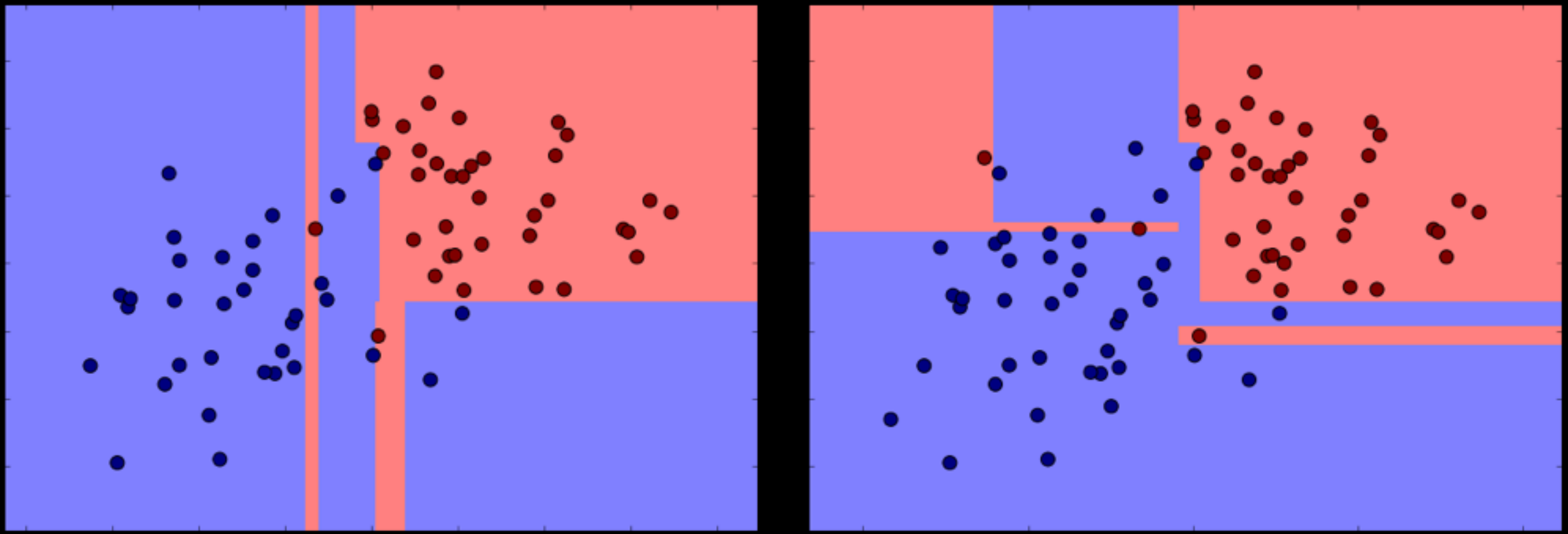
Machine learning in HEP

Likhomanenko Tatiana

Summer school on Machine Learning in High Energy Physics

Trees instability



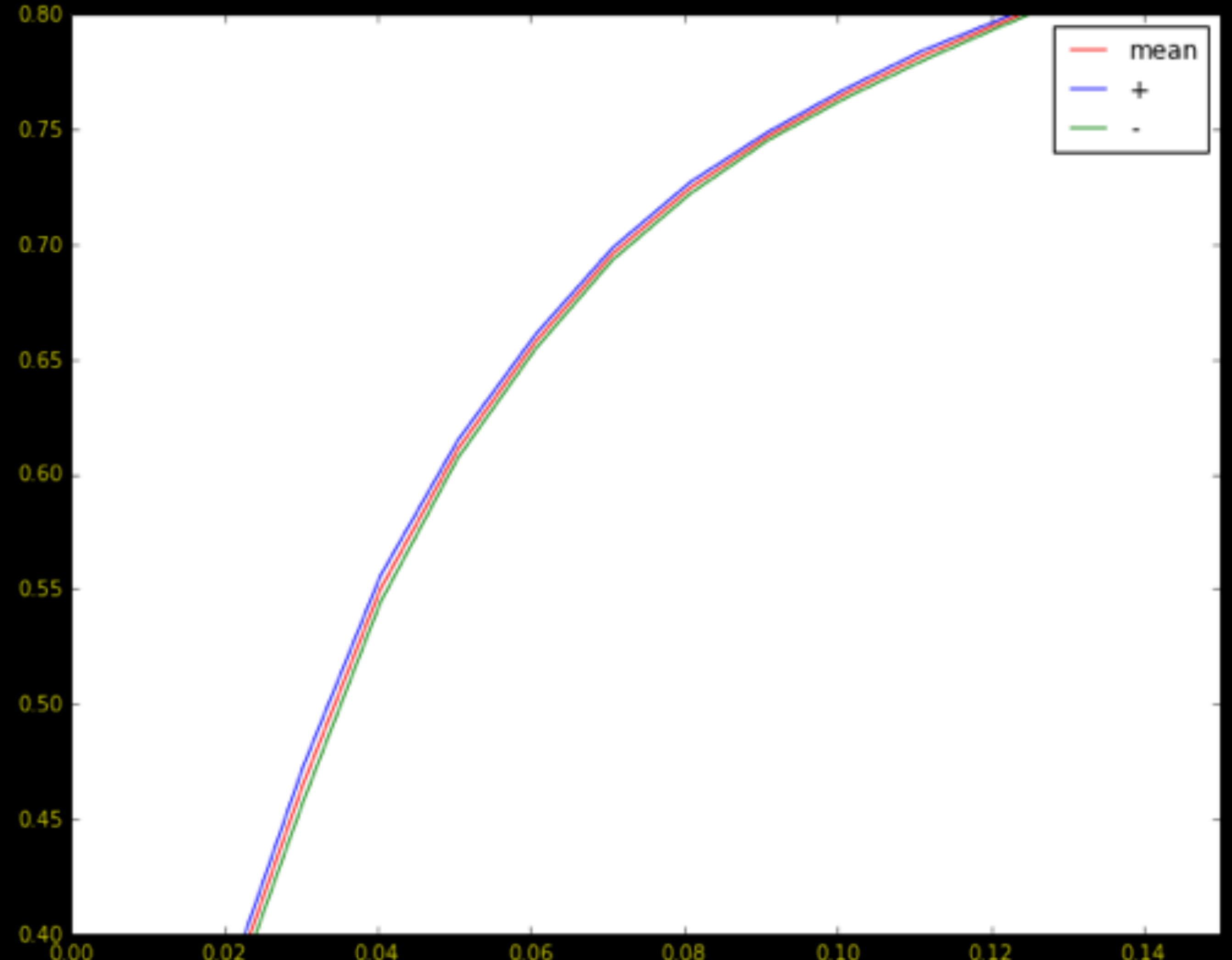
Trees ensemble is more stable, but the quality has small fluctuations during different subsampling in the ensemble

ROC curve confidence interval

To estimate the actual quality you can do

- run an ensemble algorithm many times (if bagging is used), or run on the different subsamples of the training dataset
- compute for all models ROC curves and plot the confidence interval

It is sufficiently to train ~30 different models to obtain the confidence interval



AUC: 0.91307 +/- 0.00022

HEPML Problems-solutions

- Correlation: throw mass feature and others correlated with the mass (feature selection).
- Disagreement of a classifier’s output on MC and real data: throw features which are disagree on MC and real data (feature selection).

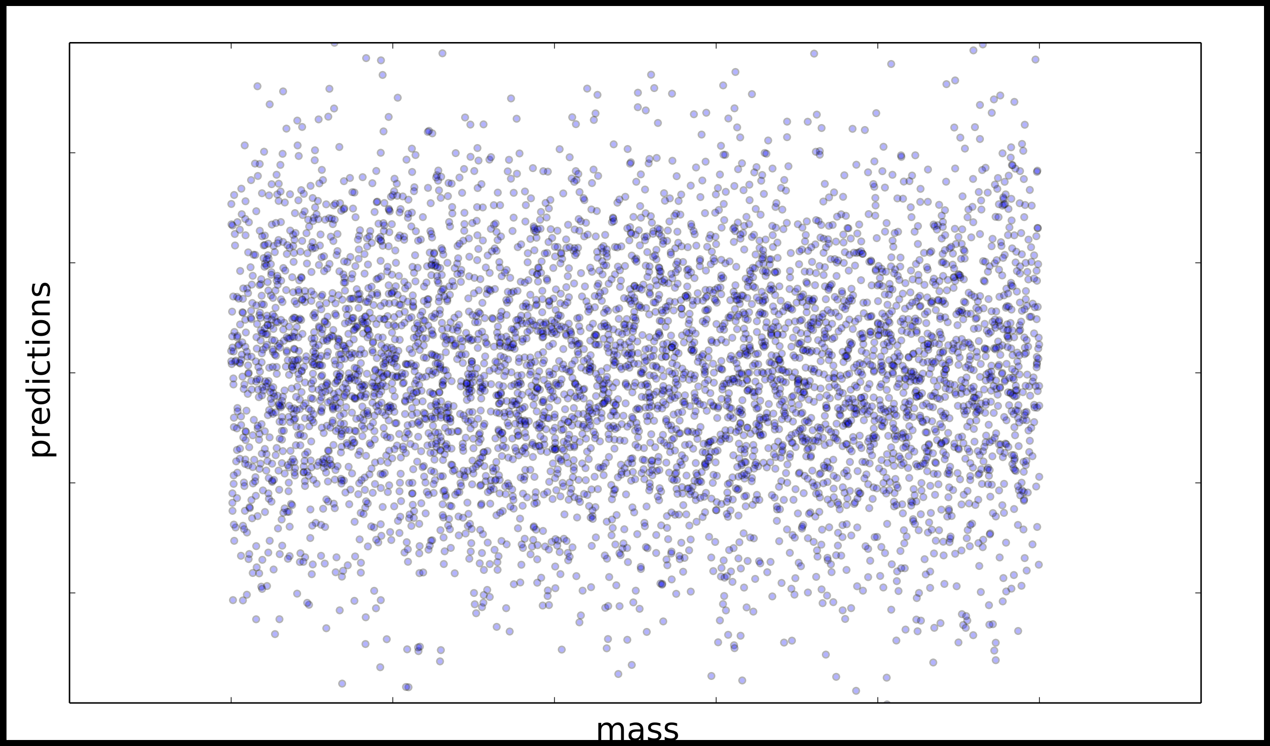
- Maybe there are another solutions how to construct the better model with the correlation and agreement restrictions?

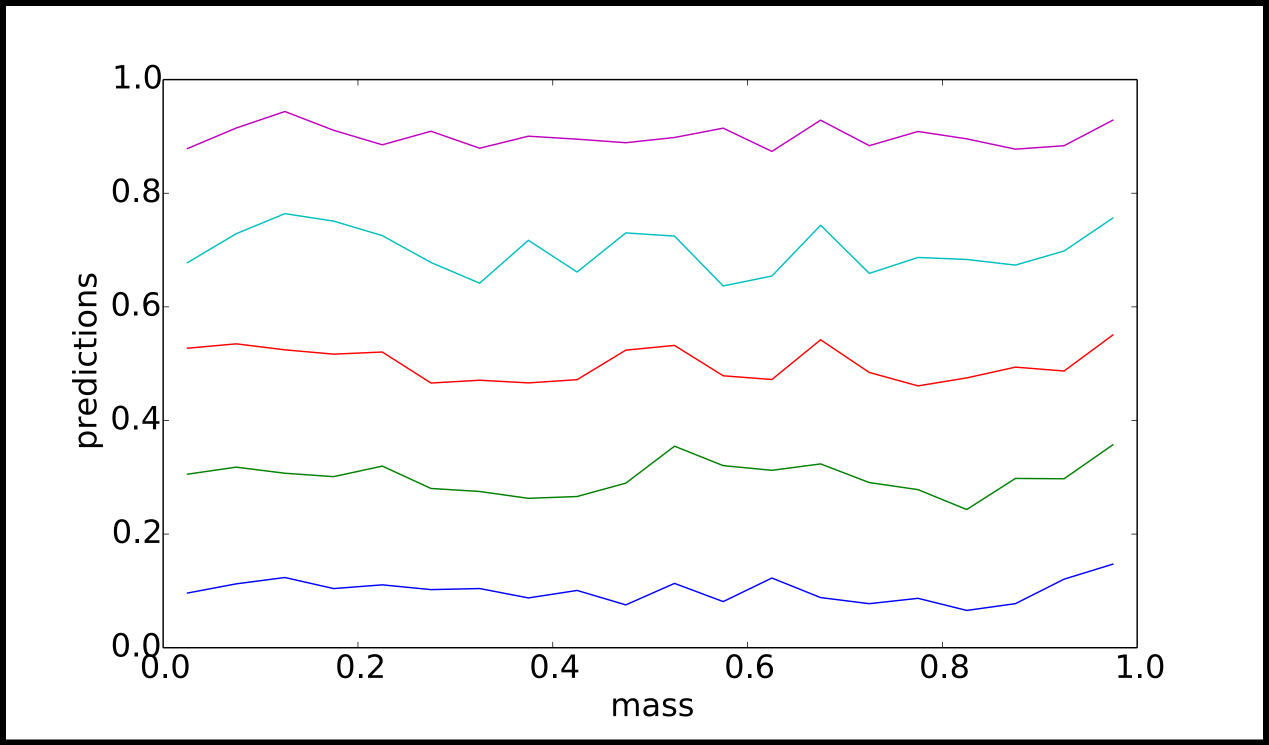
Uniformity

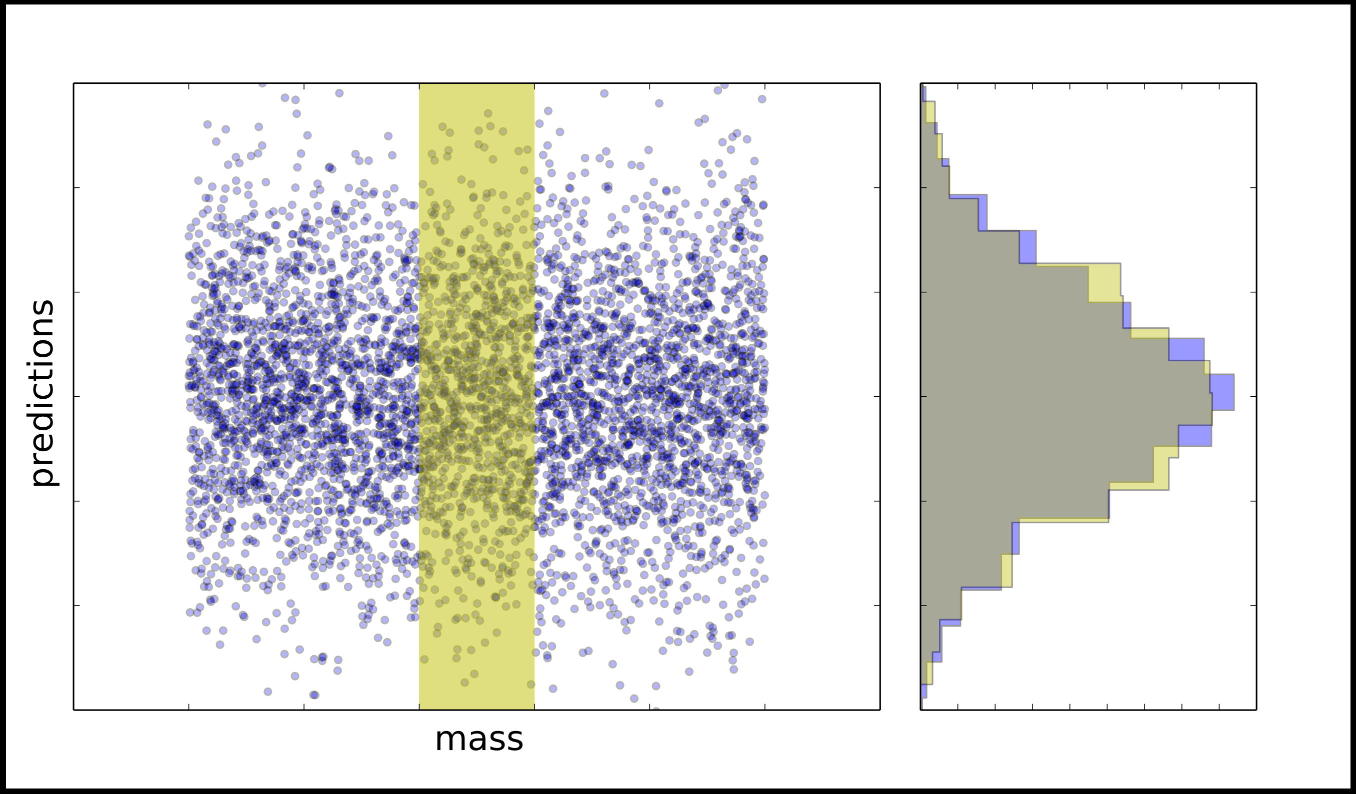
- Predictions of some classifier are called uniform in variables var1, var2, …, varn if the prediction and set of this variables are statistically independent.

- This (and only this) guarantees that any cut of the prediction will produce the same efficiency in every region over var1, var2, …, varn.

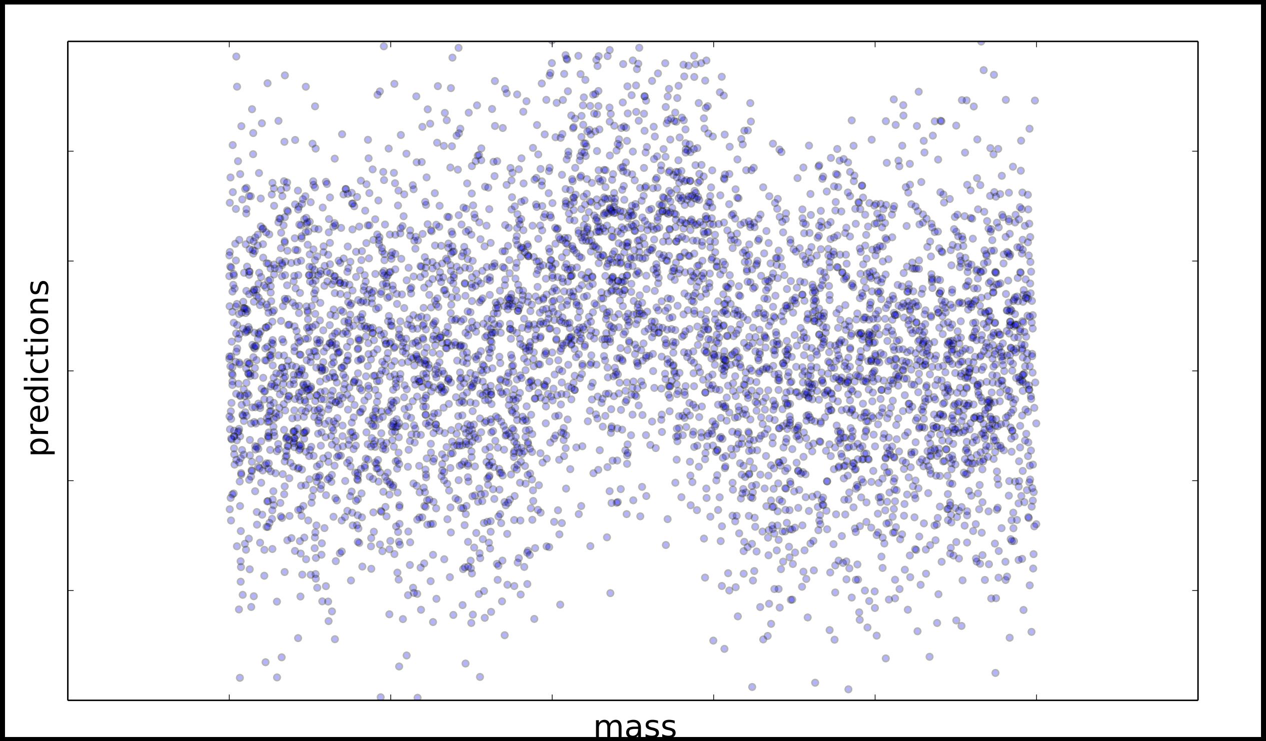
Uniformity

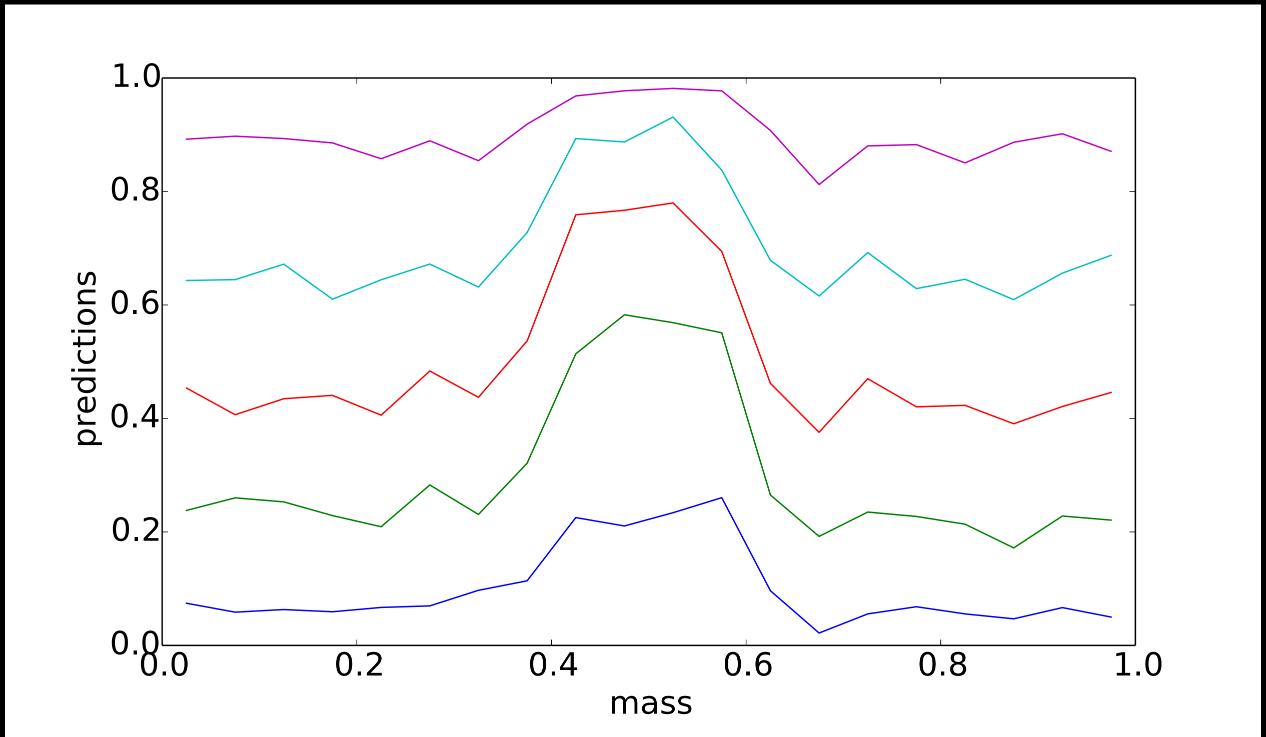

predictions
mass

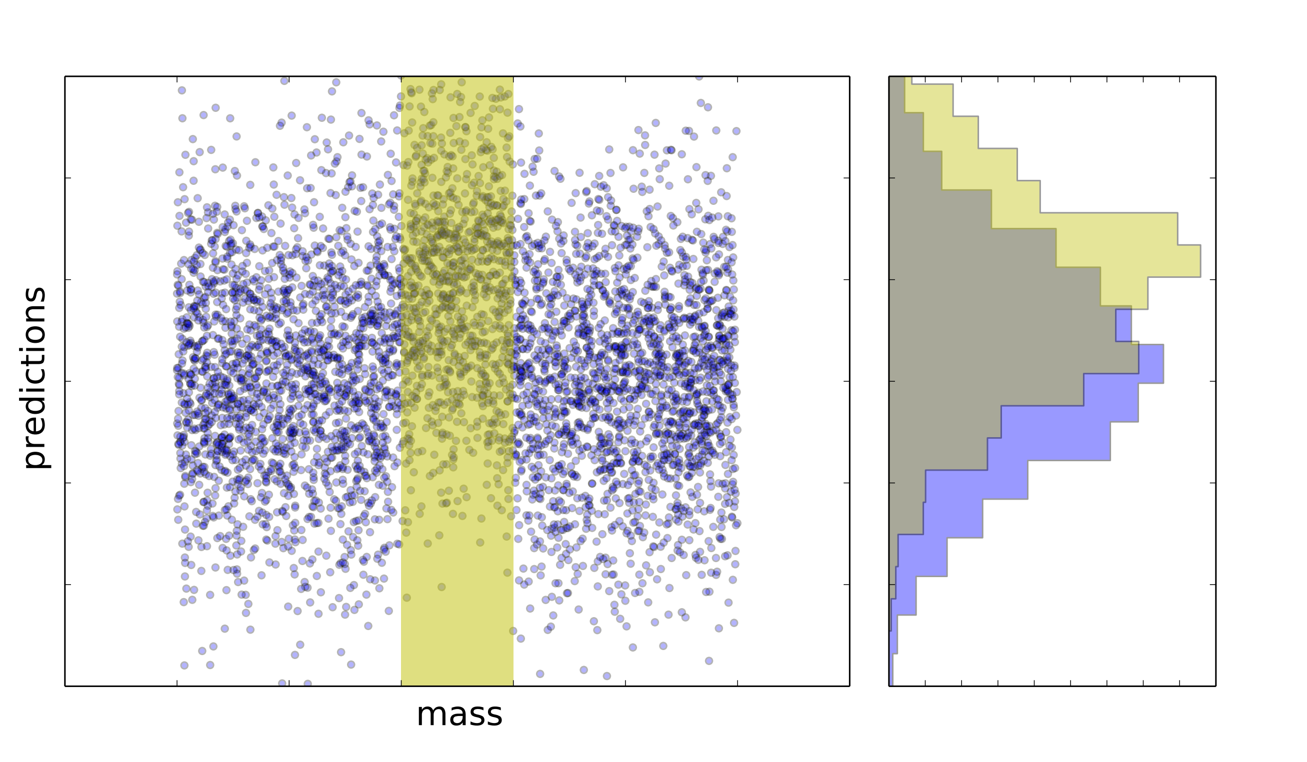

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
predictions
mass


predictions
mass

Non-Uniformity







*knn*AdaBoost

- Usual AdaBoost reweighing procedure:

$$w_i' = w_i * \exp[-y_i p_i]$$

- *knn*AdaBoost uses mean of the prediction of the neighbors (which are of the same class):

$$w_i' = w_i * \exp\left[-y_i \frac{1}{k} \sum_{j \in knn(i)} p_j\right]$$

Thus boosting focuses not on the events that were poorly classified, but on the regions with poor classification

*knn*AdaLoss

- Usual AdaLoss:

$$L_{ada} = \sum_{i \in events} w_i * \exp\left[-score_i y_i\right]$$

- *knn*AdaLoss:

$$L_{knn-ada} = \sum_{i \in events} w_i * \exp\left[-y_i \frac{1}{k} \sum_{j \in knn(i)} score_j\right]$$

FlatnessLoss

- CvM metric can be written as:

$$\sum_{bin} weight_{bin} \int |F_{bin}(x) - F(x)|^p dF(x)$$

- Lets modify it:

$$FL = \sum_{bin} weight_{bin} \int |F_{bin}(x) - F(x)|^p dx$$

Thus, it becomes differentiable:

$$\frac{\partial}{\partial score_i} FL \sim w_i p |F_{bin(i)}(x) - F(x)|^{(p-1)} sgn[F_{bin(i)}(x) - F(x)]|_{x=score_i}$$

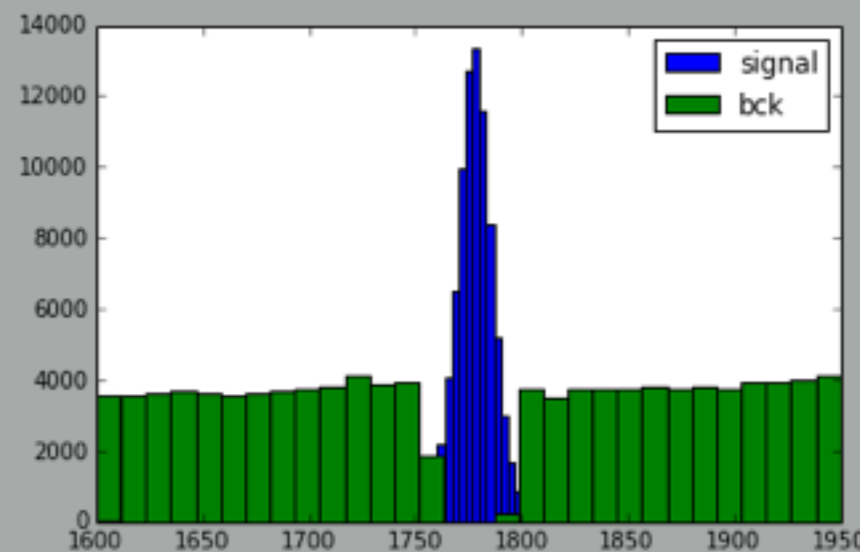
GB with FlatnessLoss (uGBFL)

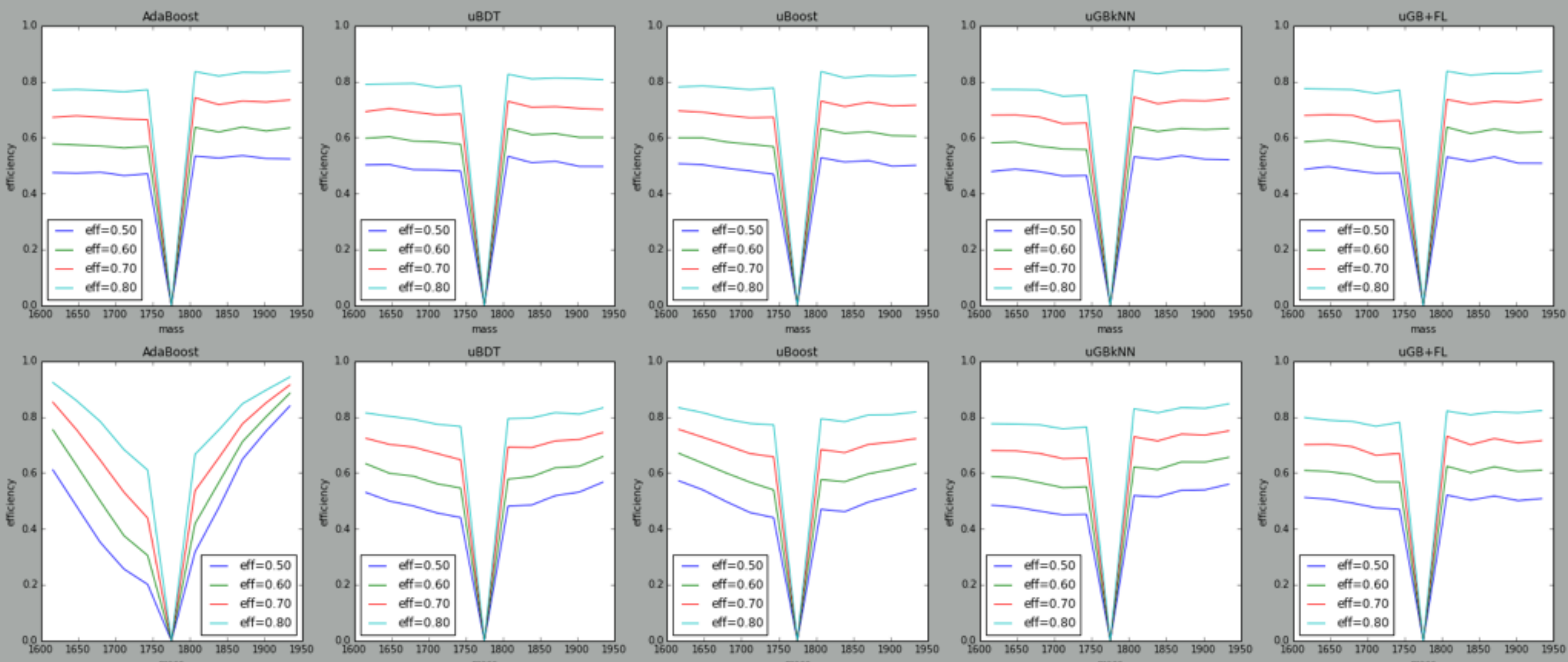
- FL doesn't into account the quality of predictions, only uniformity. That is why in practice we use the linear combination of FlatnessLoss and AdaLoss:

$$loss = \alpha FL + L_{ada}$$

- First one penalizes non-uniformity, second one - the poor predictions
- α is a way to regulate quality vs uniformity tradeoff
- http://arxiv.org/abs/1410.4140

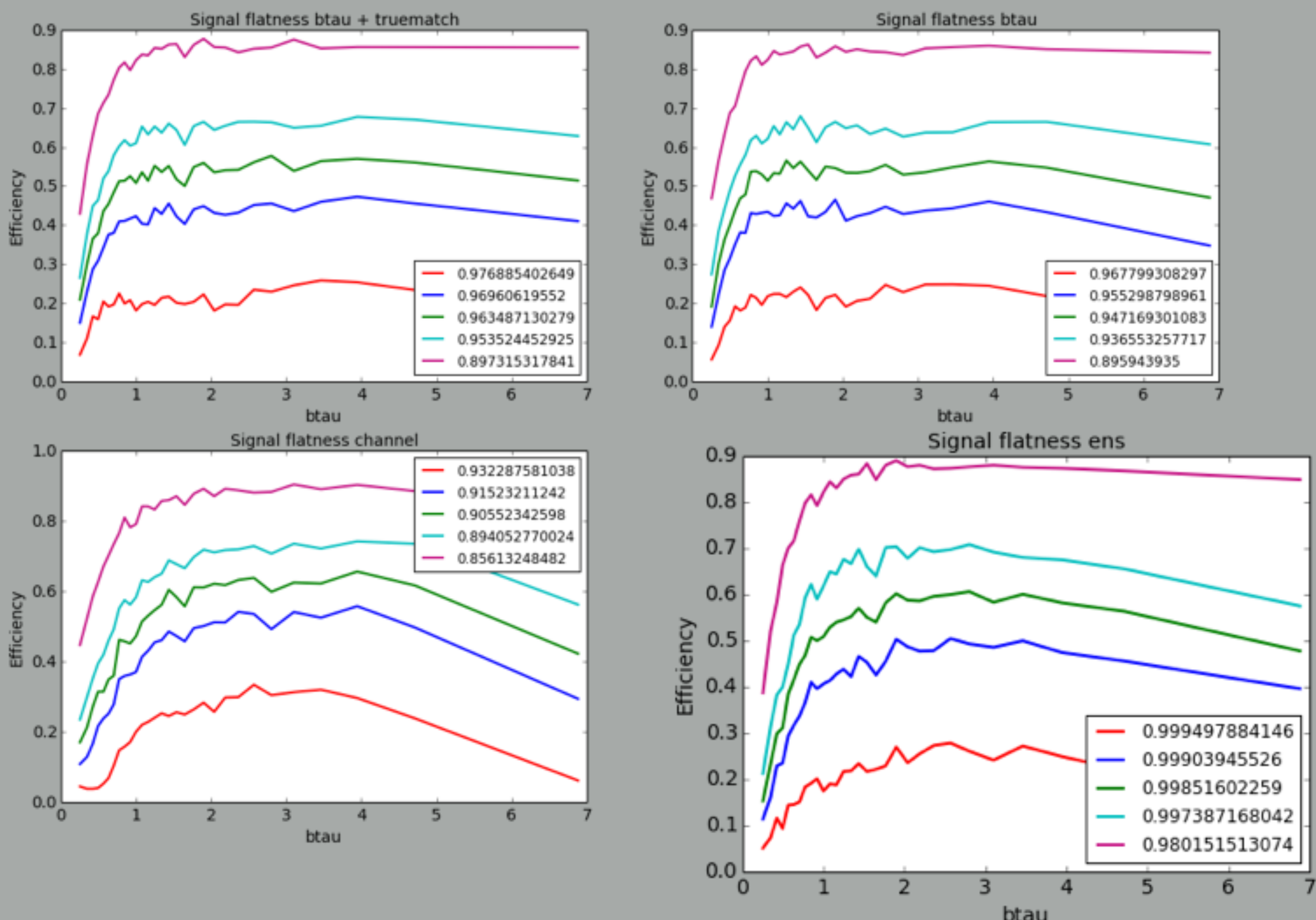
Flatness vs Standard models



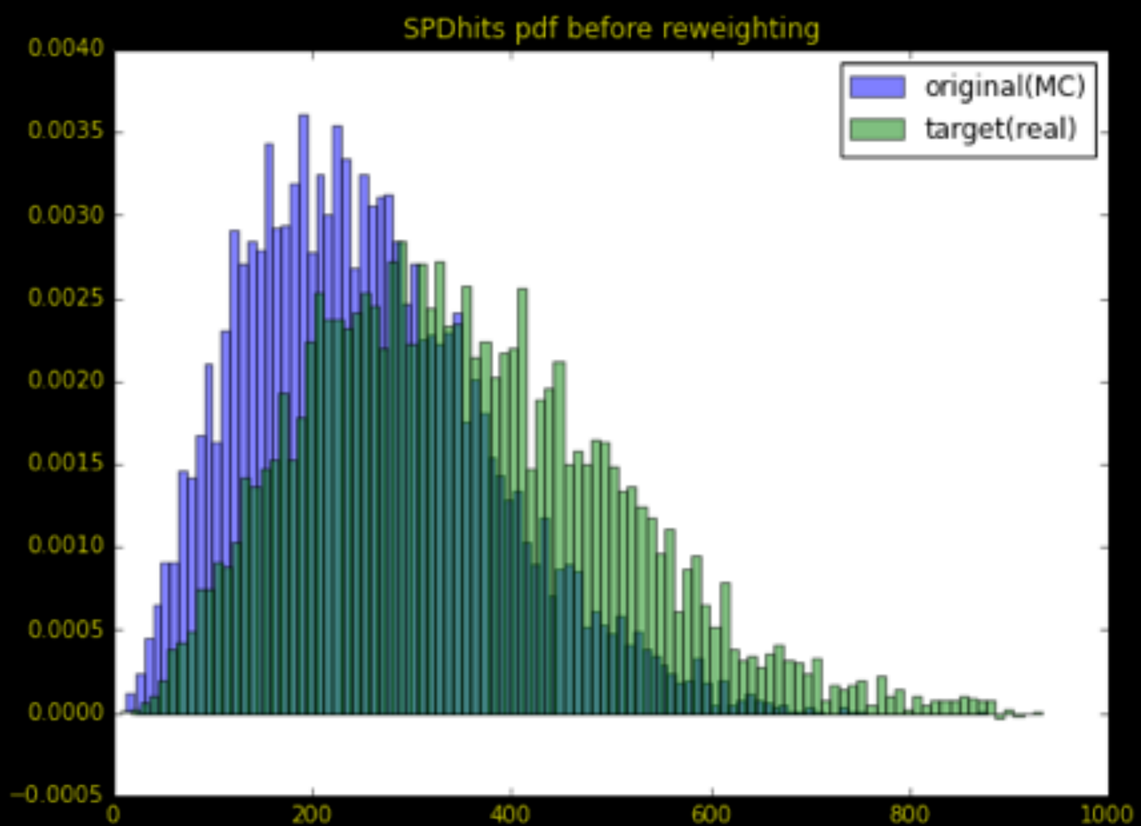


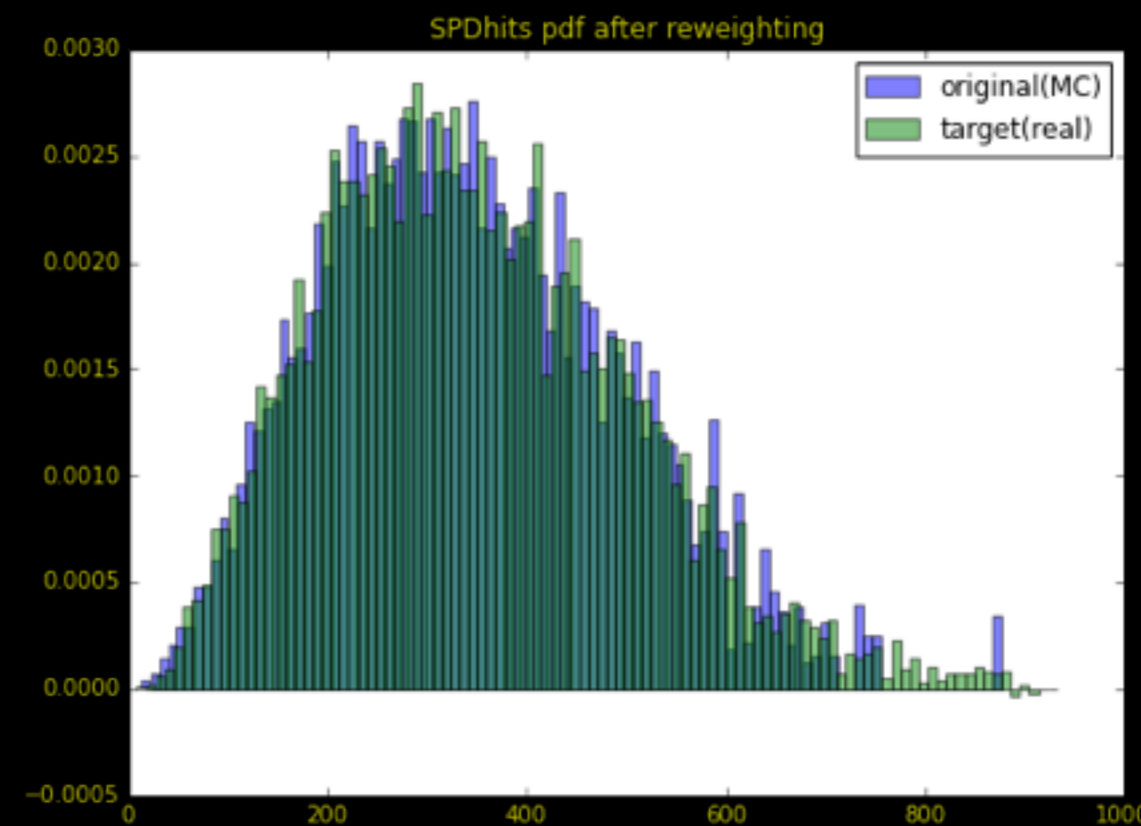
Flatness model: another application

- Trigger system: flatness for btau variable (B-meson life time) to select short lived particles

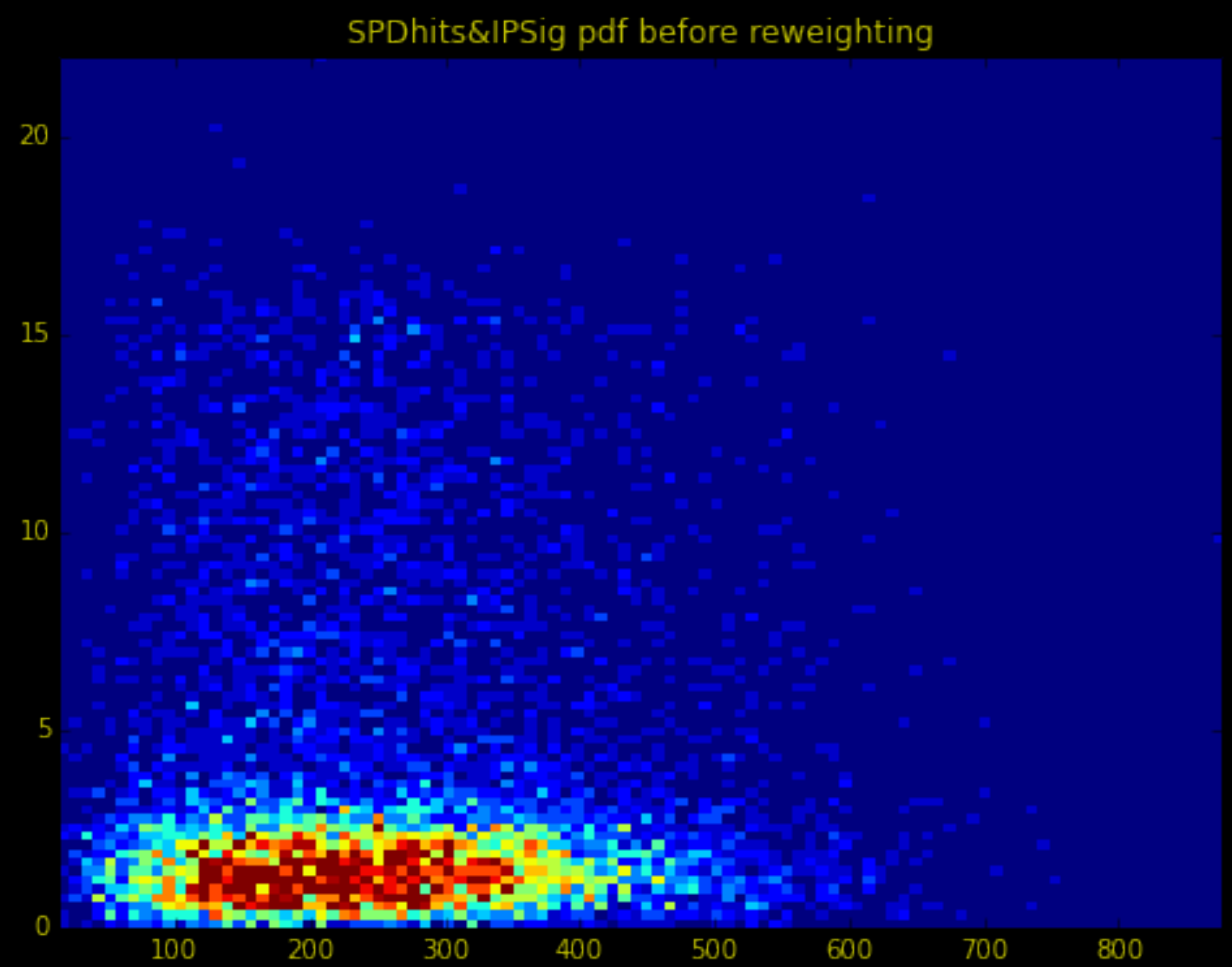


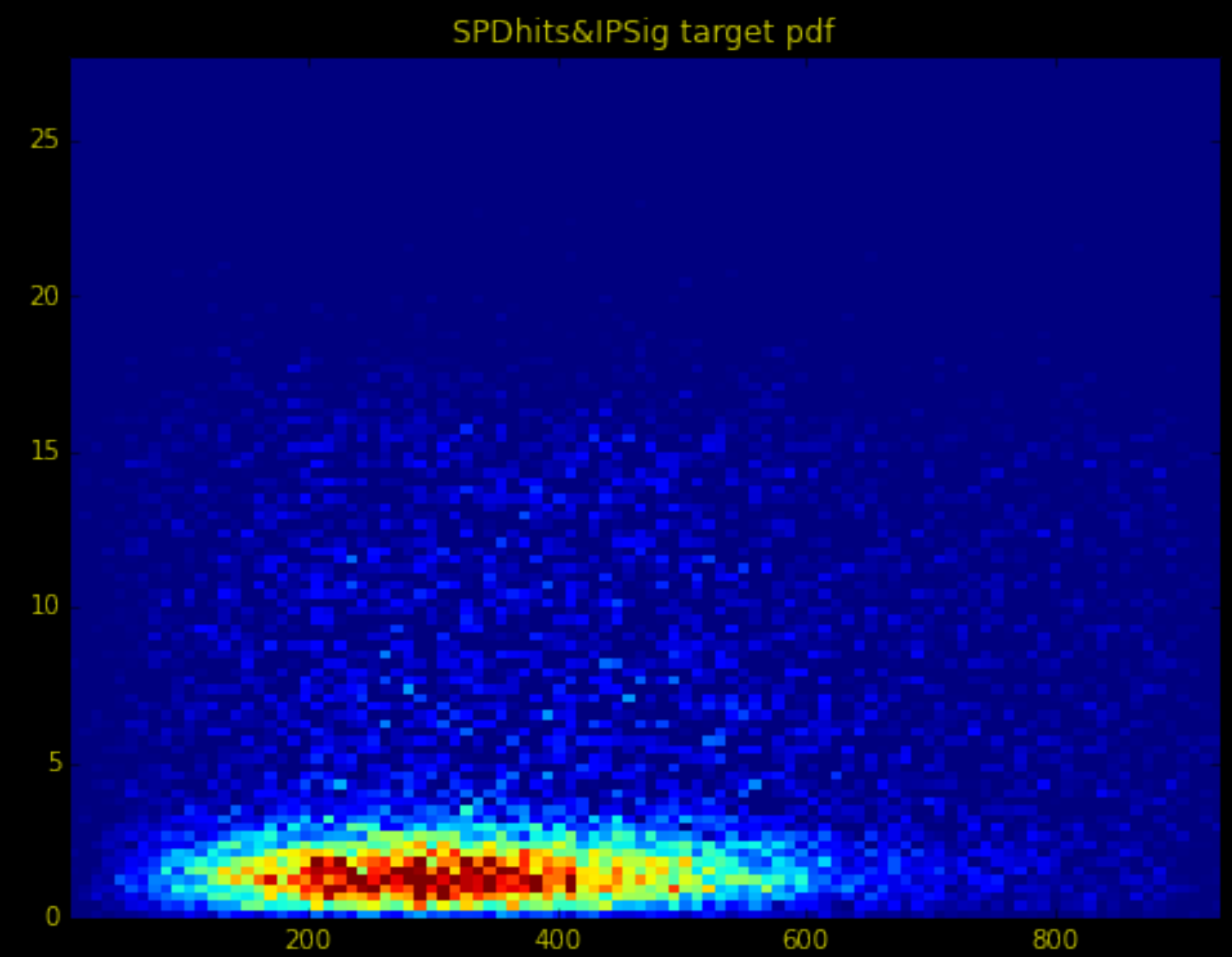
Reweighting procedure: 1D



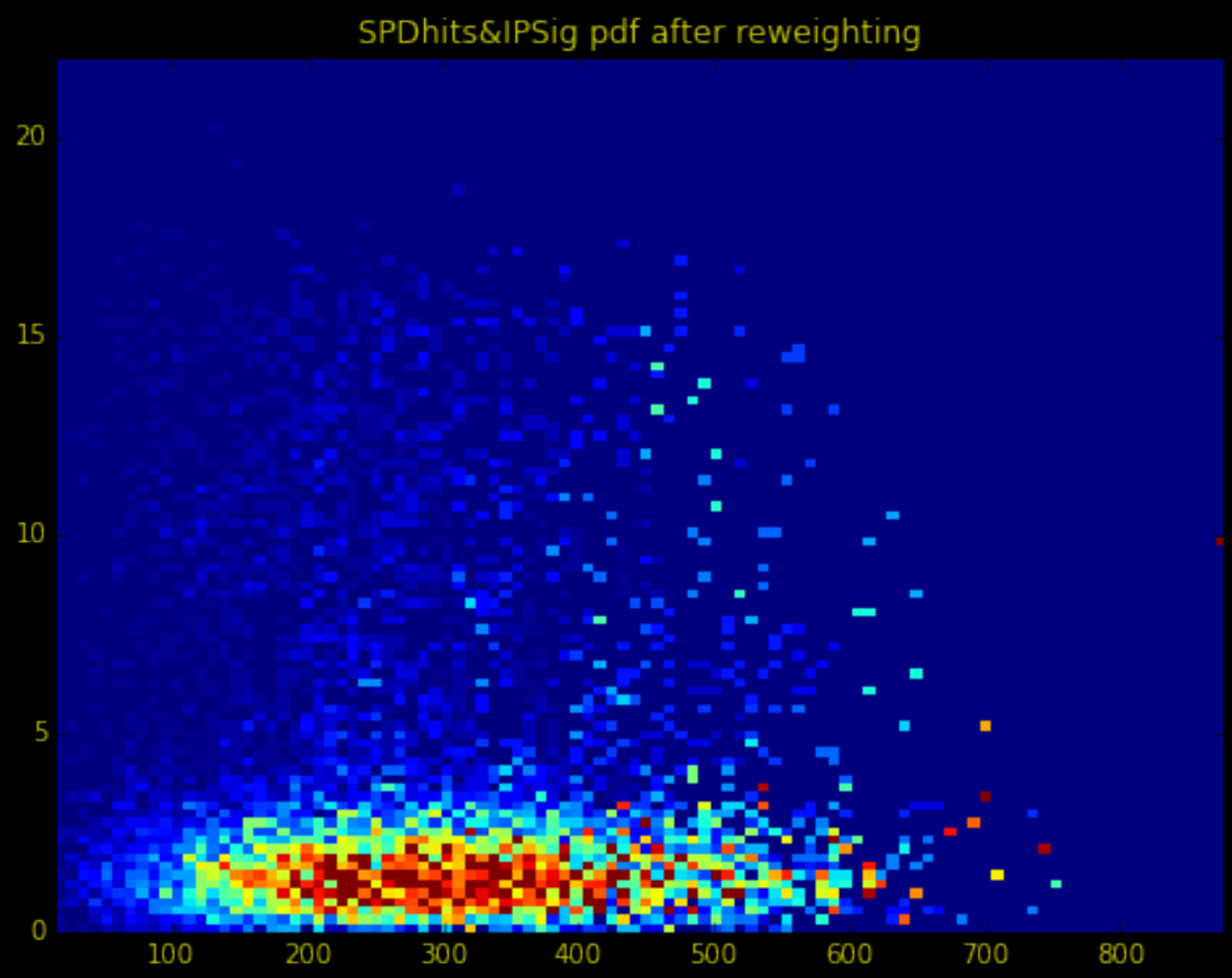


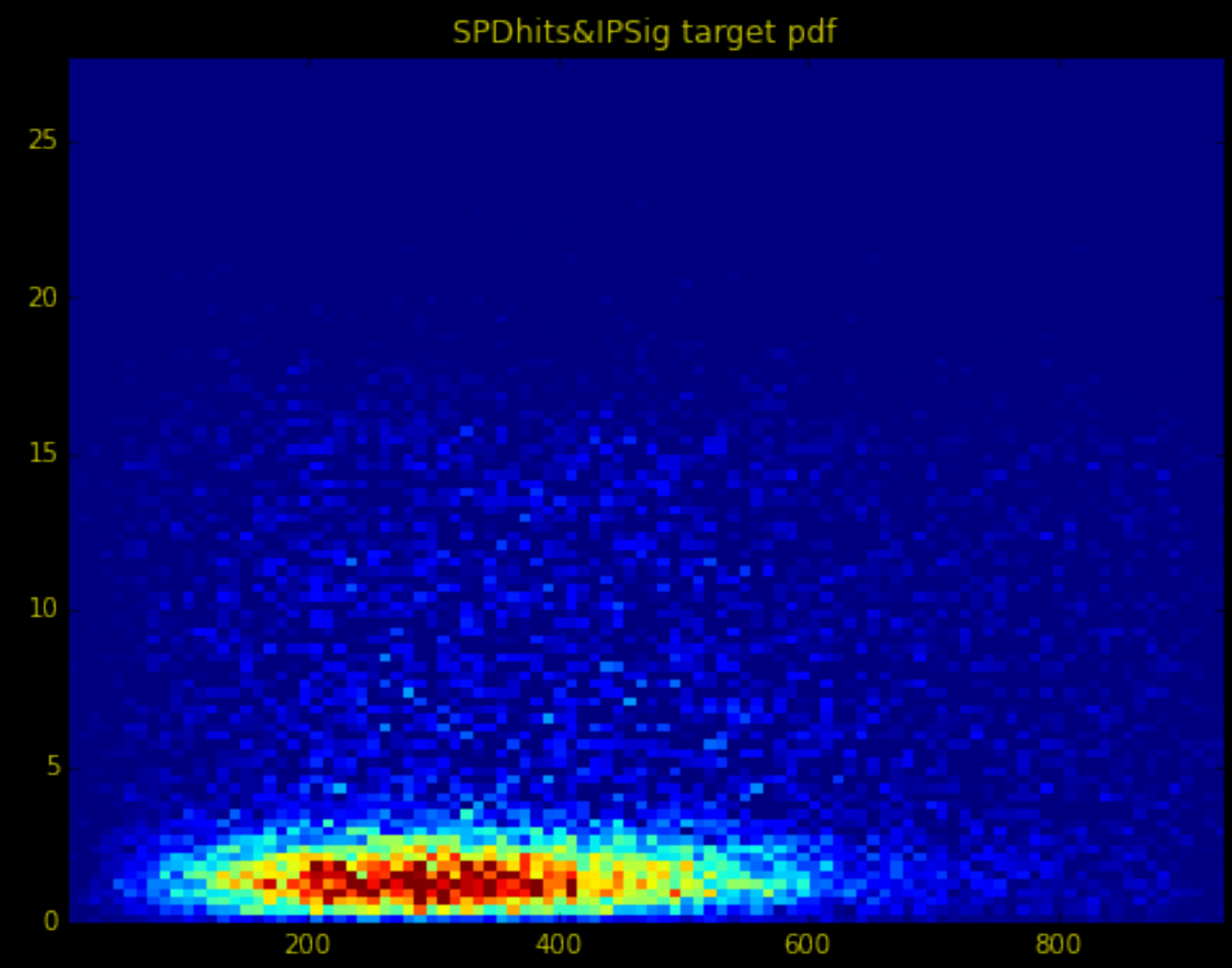
Reweighting procedure: 2D





Reweighting procedure: 2D





Reweighting procedure

- Divide values into bins
- Compute weight for each bin => Weight(bin) function
- To what data we should apply Weight(bin)?

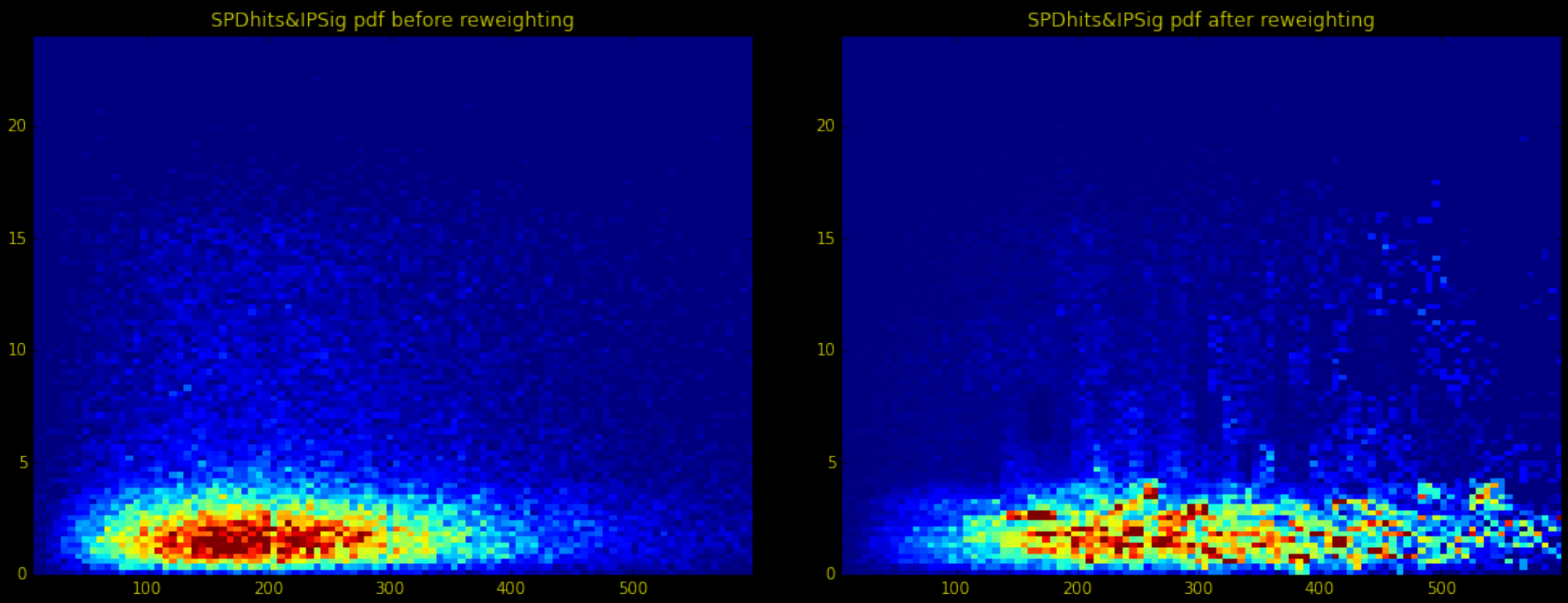
$$TPR^{S}_{real\ data}/TPR^{S}_{MC} = TPR^{N}_{real\ data}/TPR^{N}_{MC}$$

if we reweight MC to real in the control channel we should also apply a weight function to MC in the signal channel to preserve the equation.

if we reweight MC (control channel) to MC (signal channel) we should apply a weight function to real data (control channel): just multiply sPlot weights on this new weights, because we suppose that mass and features are not correlated.

Any reweighting procedure should save the equation.

Reweighting procedure: 2D
apply to signal MC



Bin reweighter is already unstable in 2D

Problems

- Bin reweighting procedure is unstable in multidimensional case
- Can we apply ml to reweight?
- KS is appropriate to compare 1-D pdfs
- How to compare the similarity for multidimensional pdf?
- Can we use ml in this area?

Boosting as reweighter?

- Will train Gradient Boosting (AdaBoost like) over regression trees to reweight an original pdf to the target pdf.

- An update rule for reweighting GB (like in AdaBoost algorithm):

$$w = \begin{cases} w \text{ event from the target distribution} \\ e^{pred} w, \text{event from an original distribution} \end{cases}$$

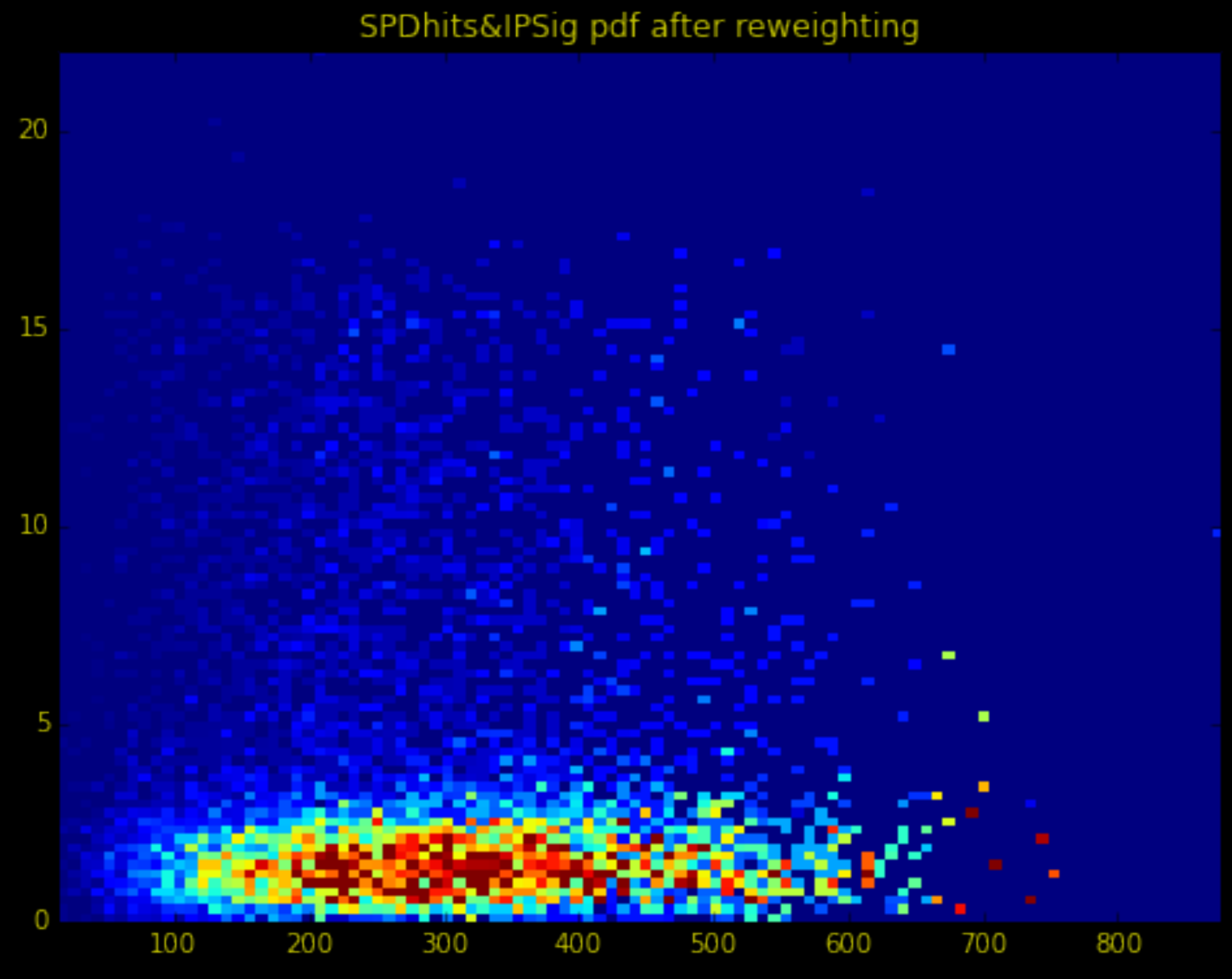
- Splitting criterion during tree construction - maximize binned chi-squared statistic (choose a bit more symmetric way):

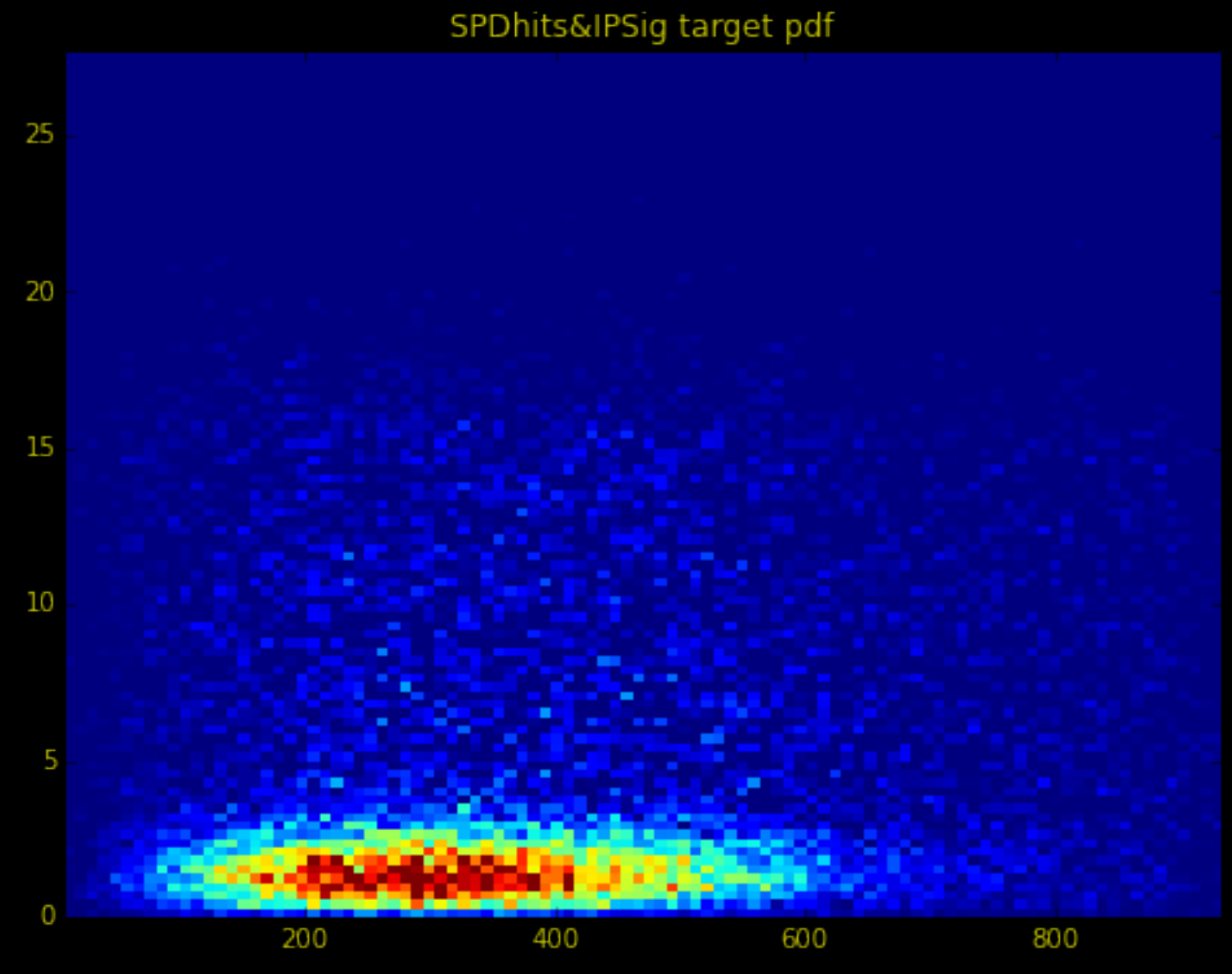
$$\chi^2 = \sum_{\text{bins}} \frac{(w_{target} - w_{original})^2}{w_{target} + w_{original}}$$

- Compute optimal value in the leaf (loss function cannot be calculated at all):

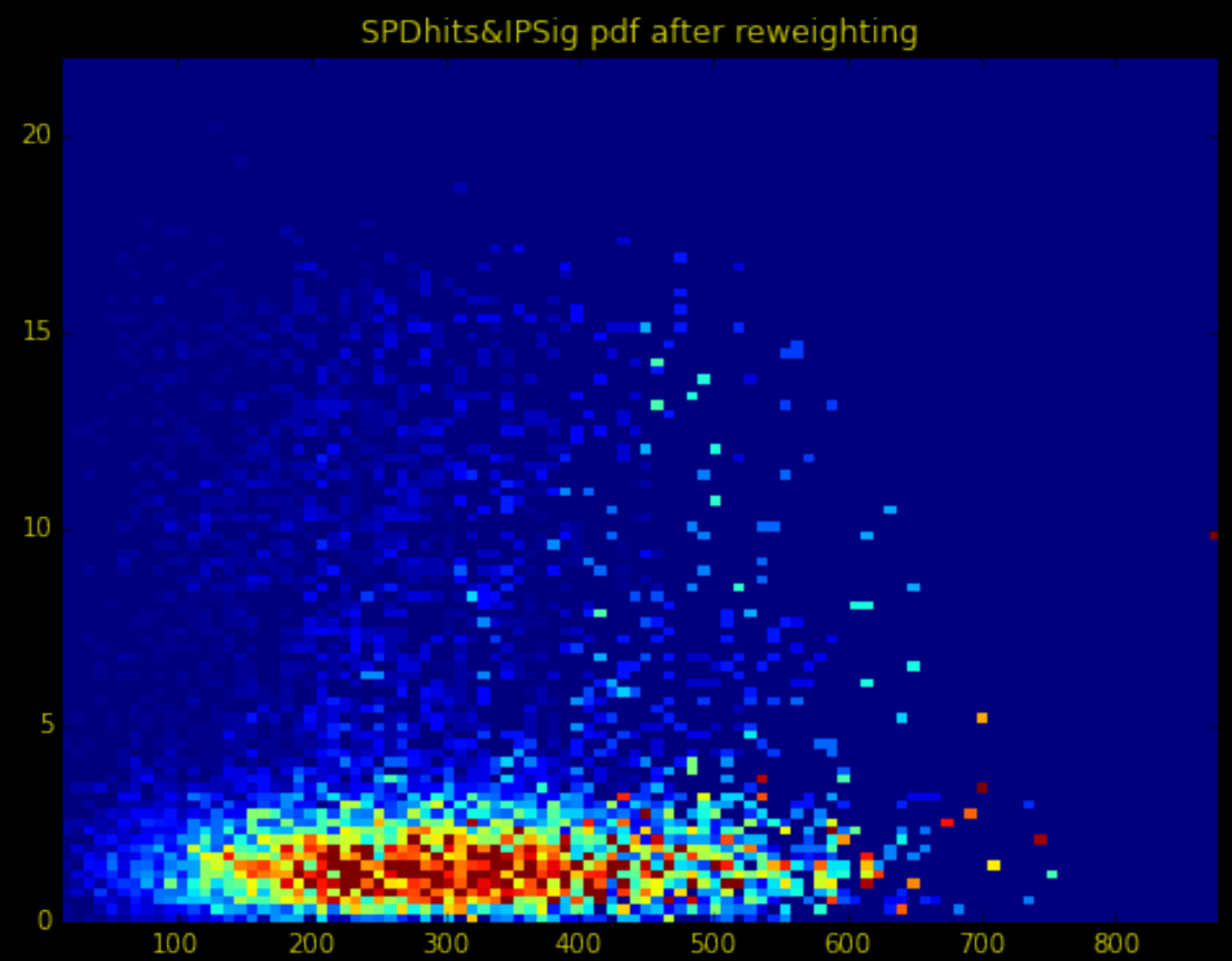
$$\text{leaf\_value} = \log \frac{w_{target}}{w_{original}}$$

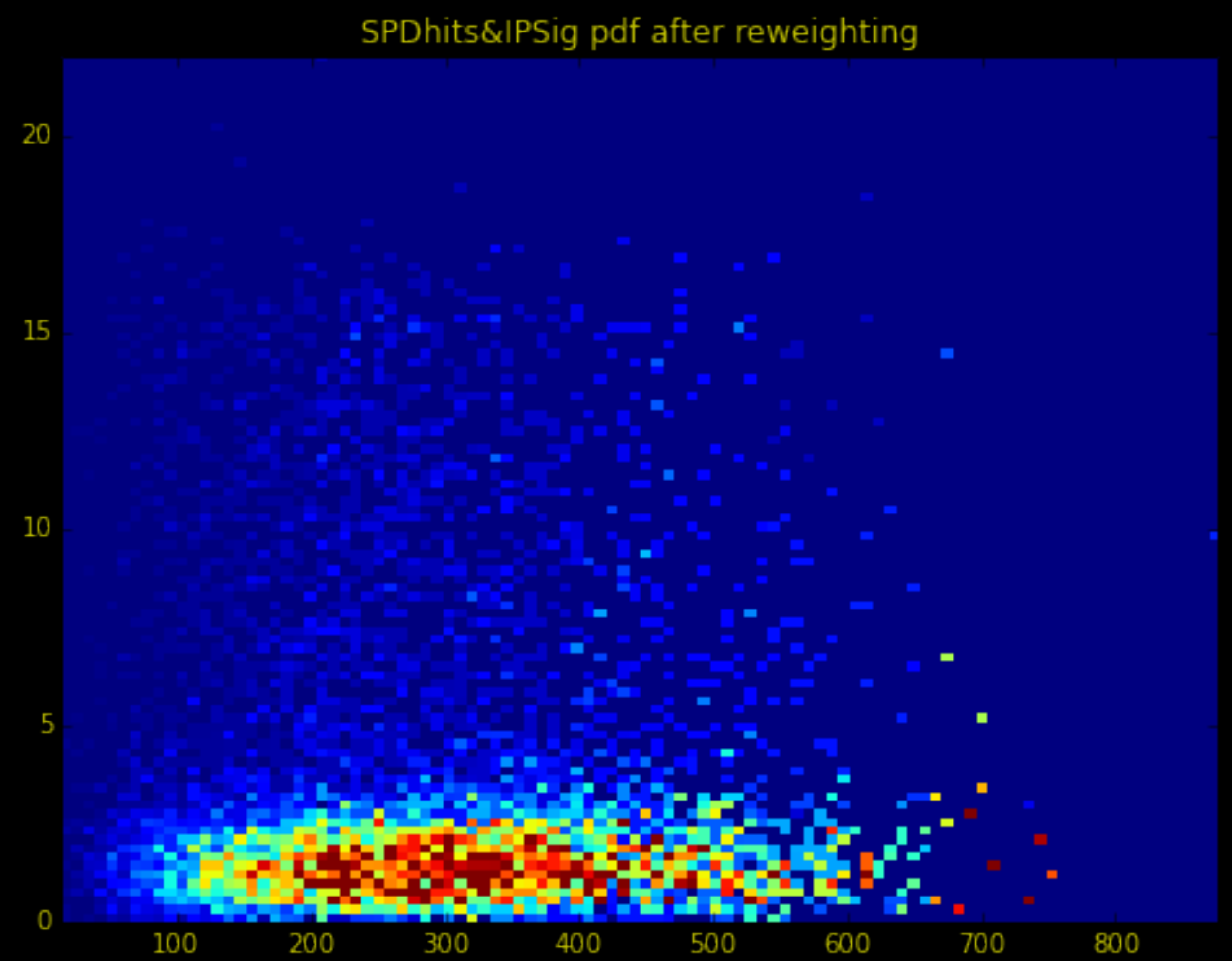
GB reweighter



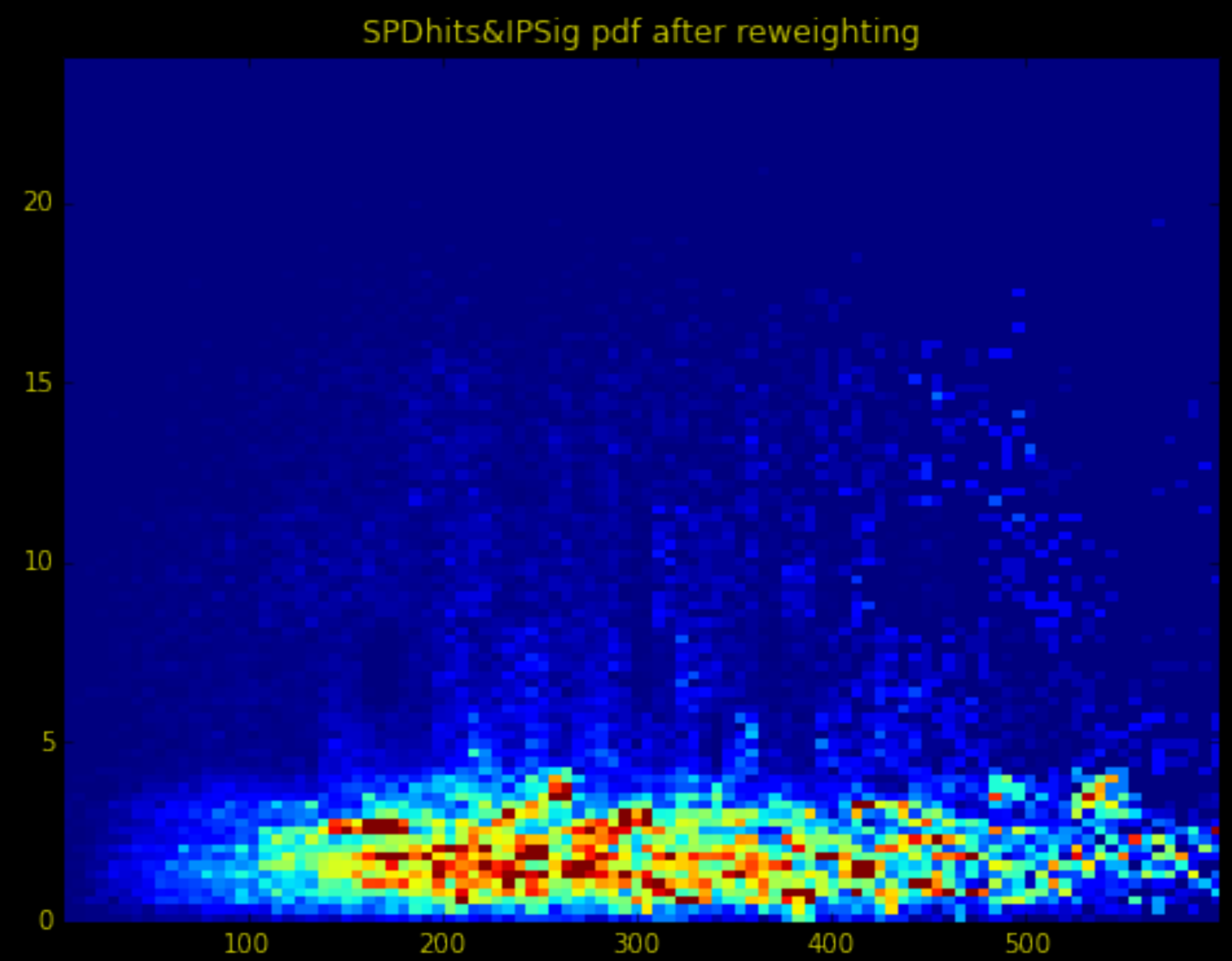


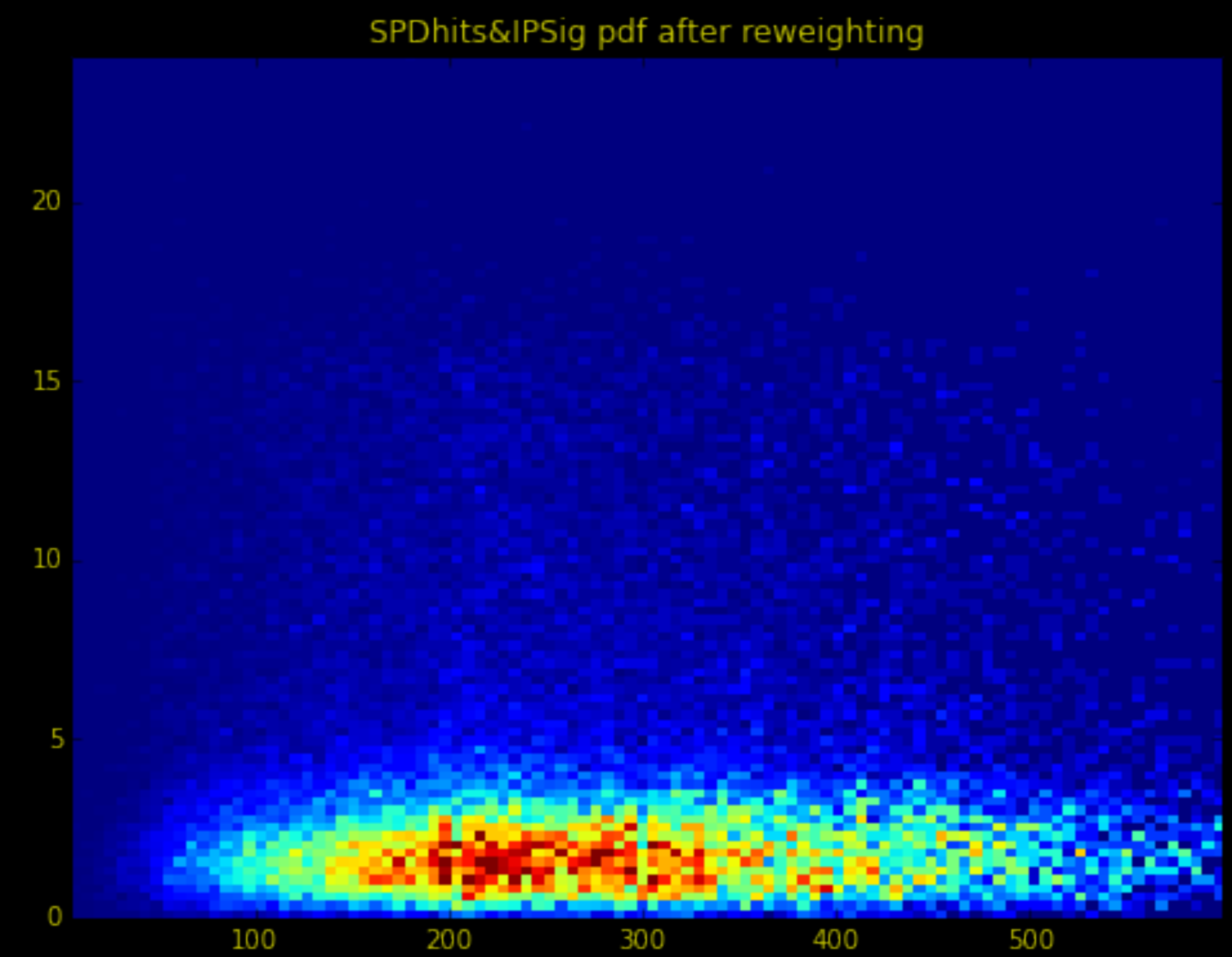
Bin reweighter vs GB reweighter



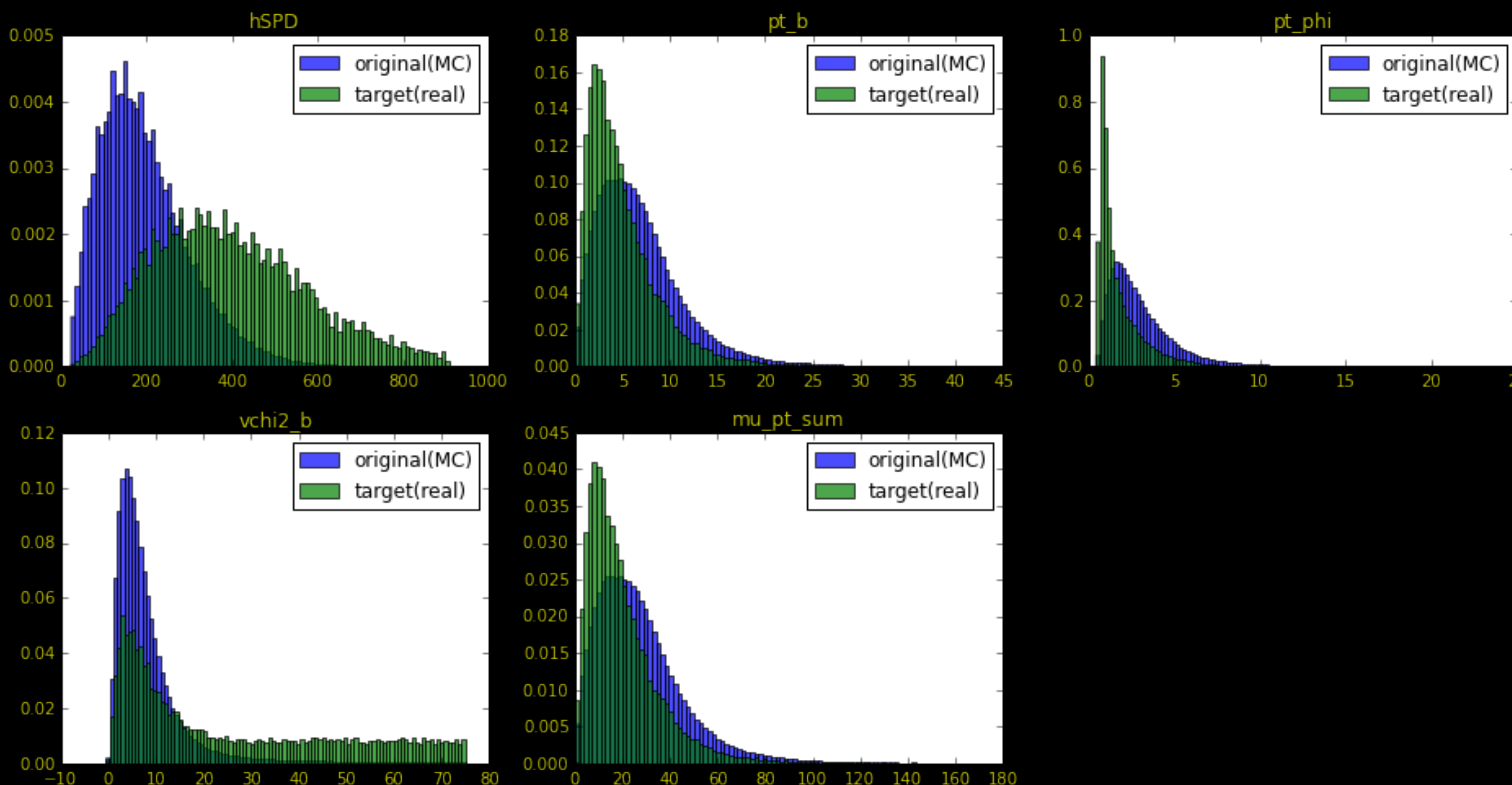


Bin reweighter vs GB reweighter apply to singal MC

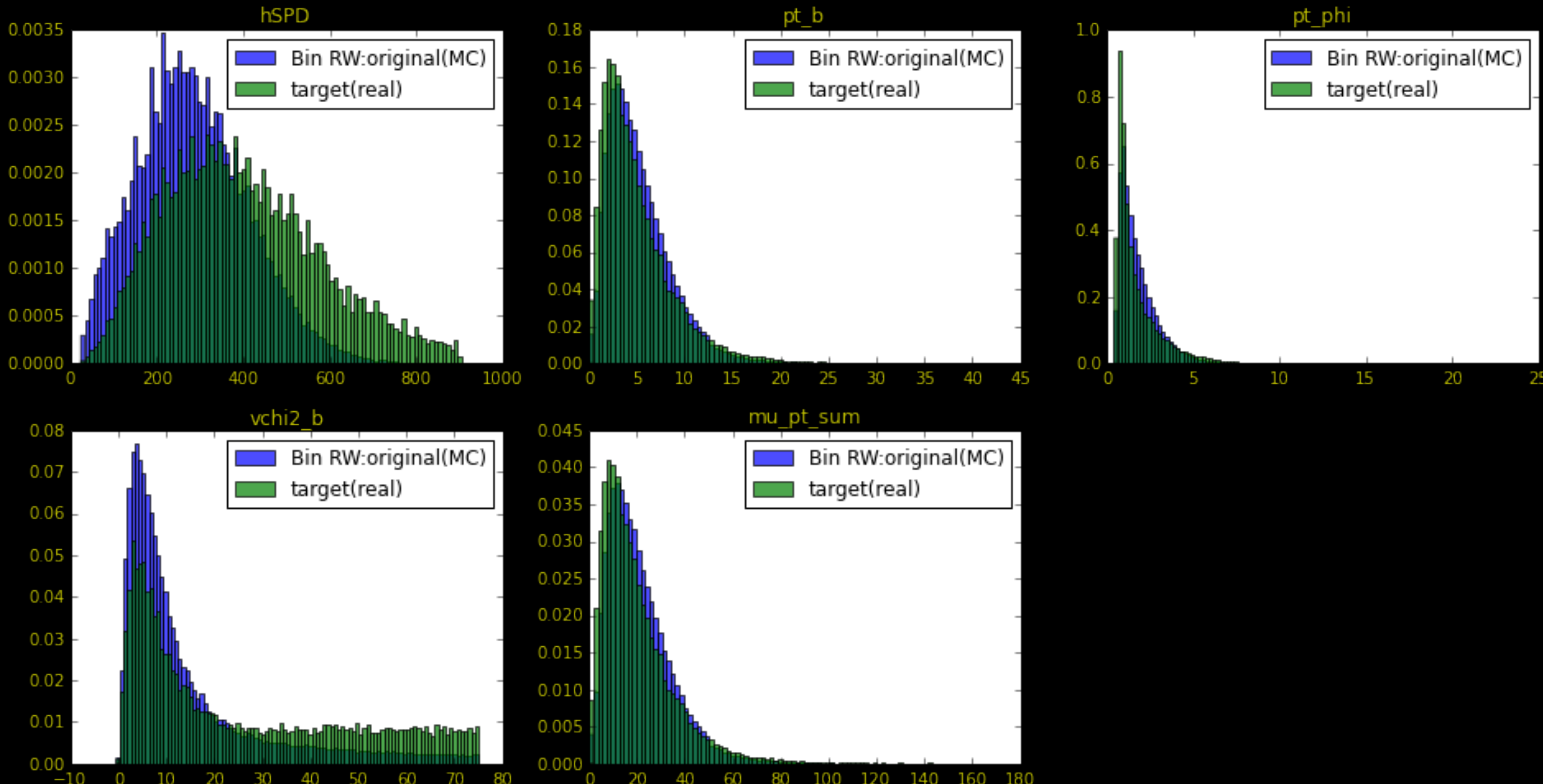




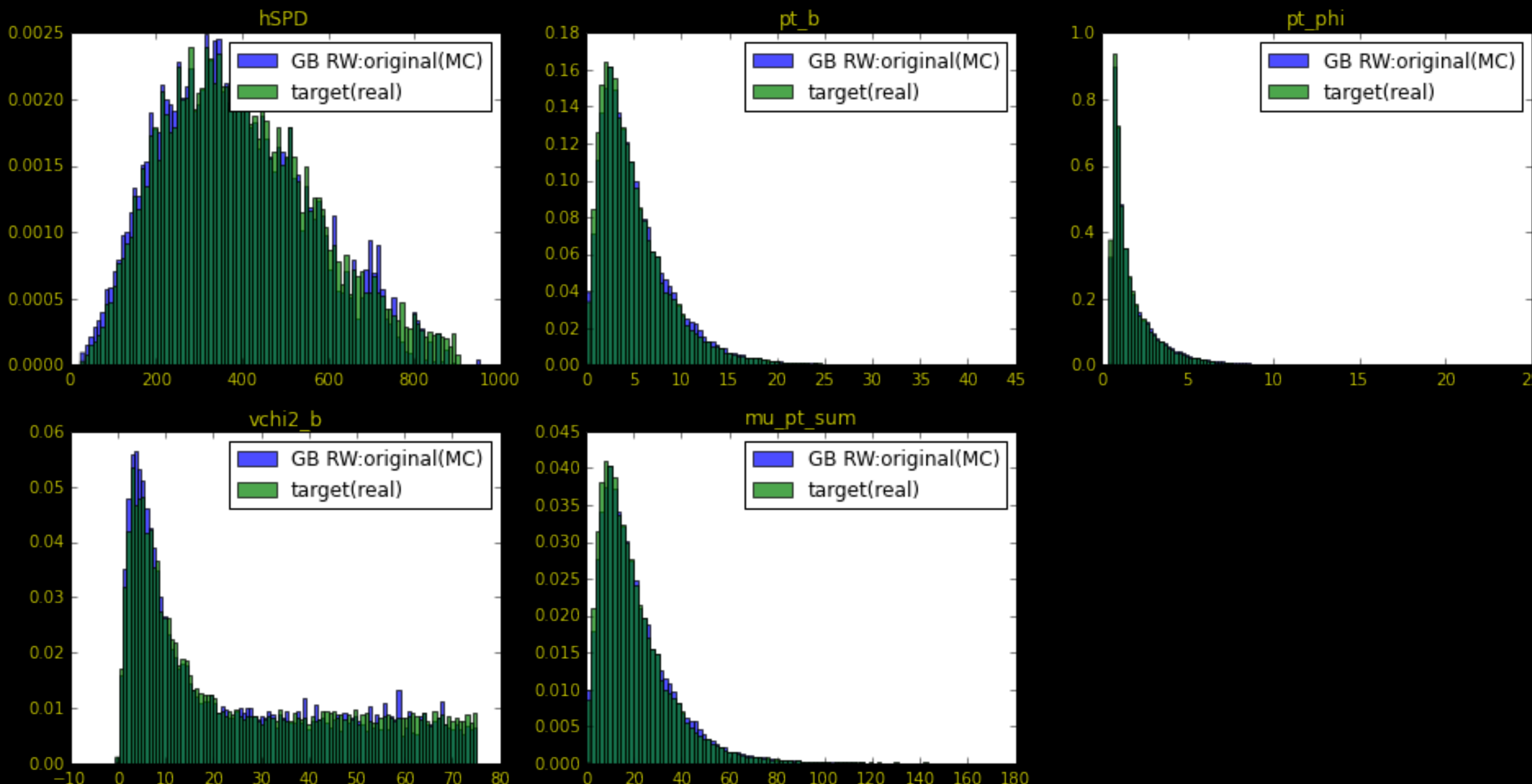
Bin reweighter vs GB reweighter: ND initial



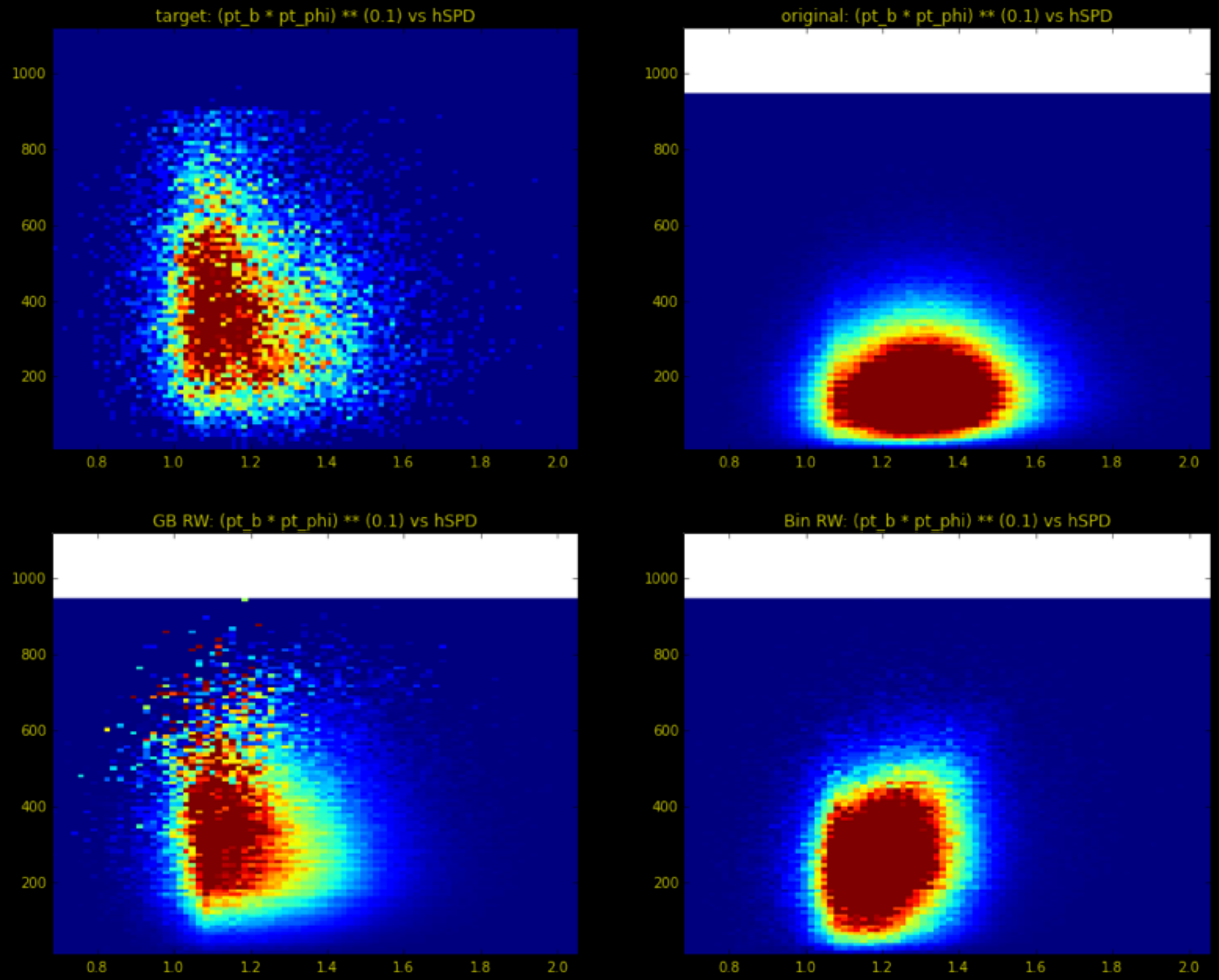
Bin reweighter vs GB reweighter: ND bin RW



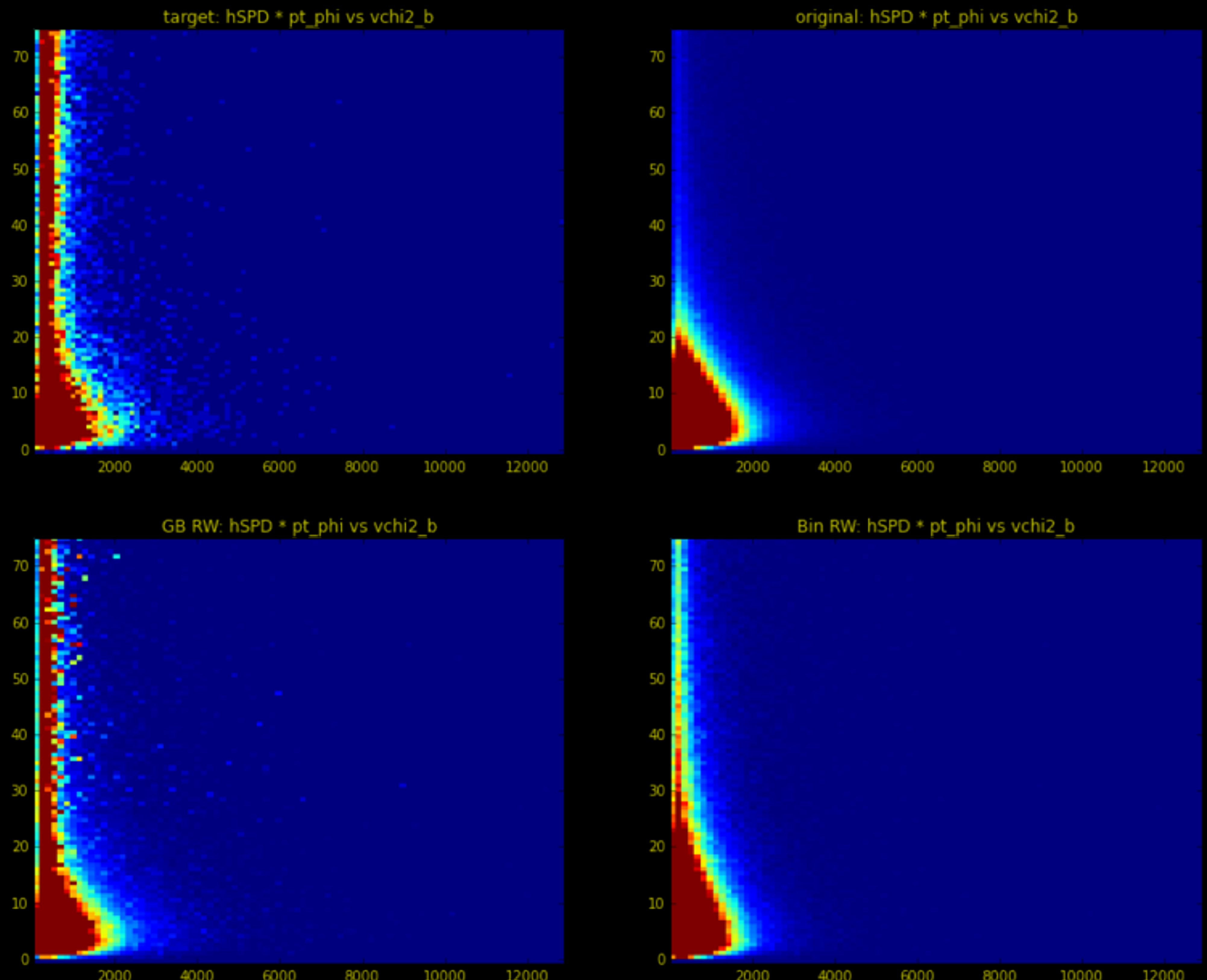
Bin reweighter vs GB reweighter: ND
GB RW



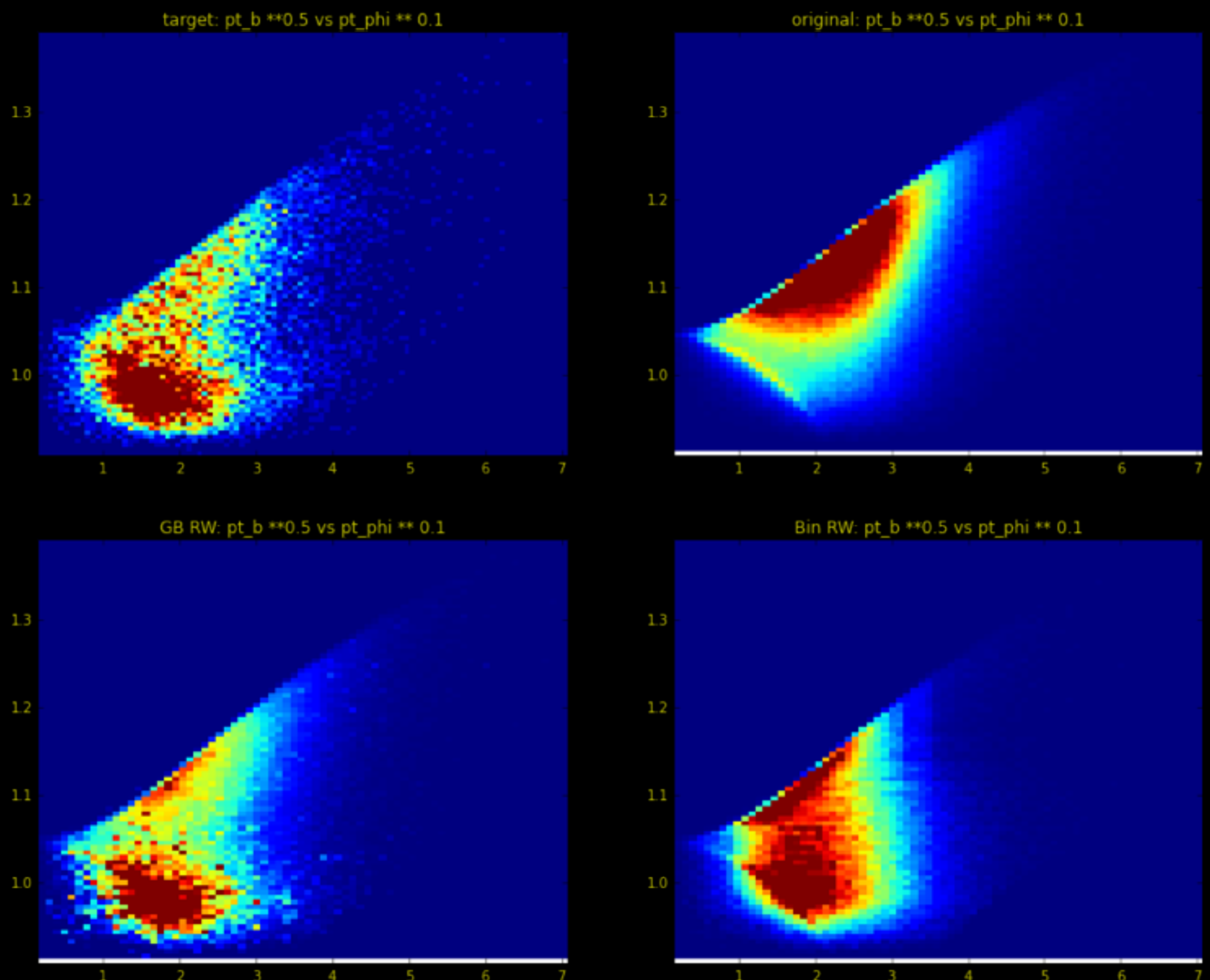
Bin reweighter vs GB reweighter: ND feature combination



Bin reweighter vs GB reweighter: ND feature combination



Bin reweighter vs GB reweighter: ND feature combination



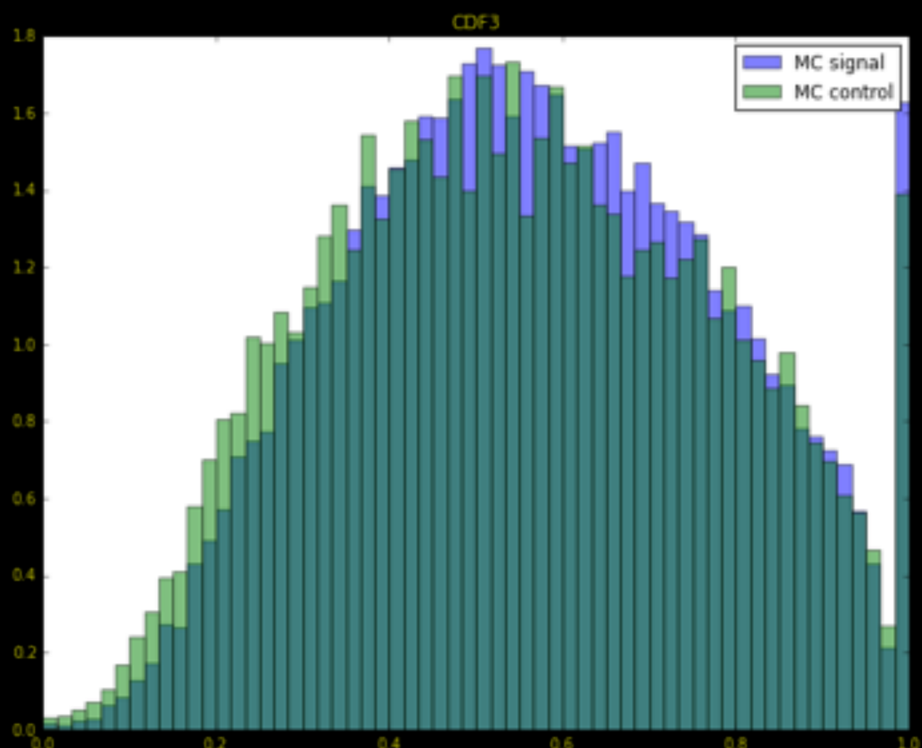
How to compare ND pdf?

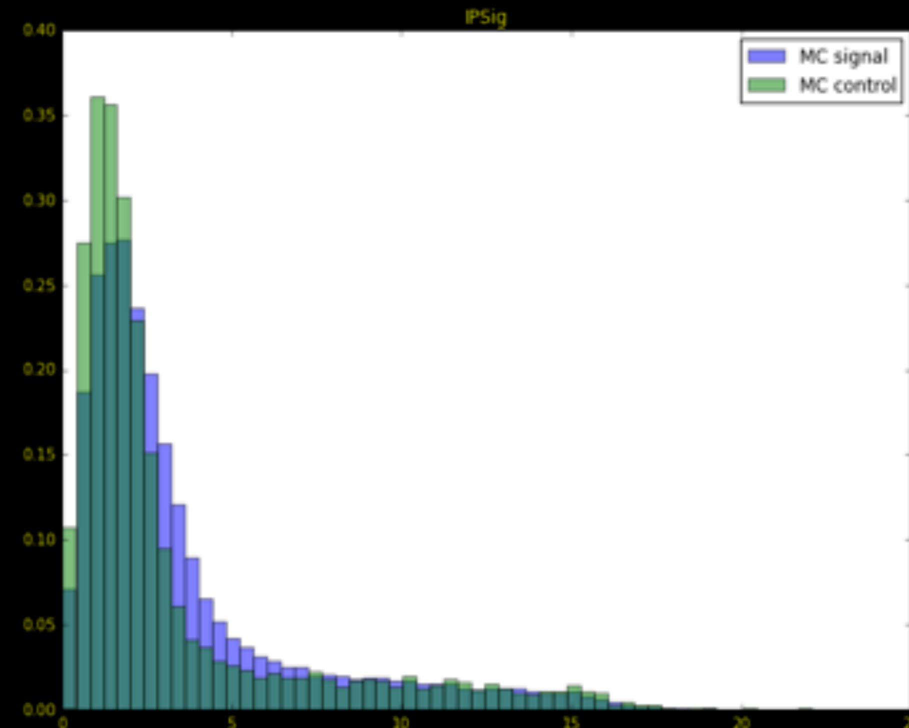
- Idea: two pdfs are equal if we cannot distinguish them at all.
- It means that any classifier trained on data (1-label the first pdf, 0-label for the second) cannot distinguish them. (AUC ~ 0.5).
- Thus machine learning can be applied to compare ND pdfs.
- For more details: http://statweb.stanford.edu/~jhf/ftp/gof (He find p-value during hypothesis testing using ML)
- In application: train several different models (different nature of algorithms, like trees, linear and NN); if AUC is similar to a random classifier in all cases then your ND pdfs are similar.
- To compare reweighting algorithms you can train classifier on reweighted data to understand which one is the best.

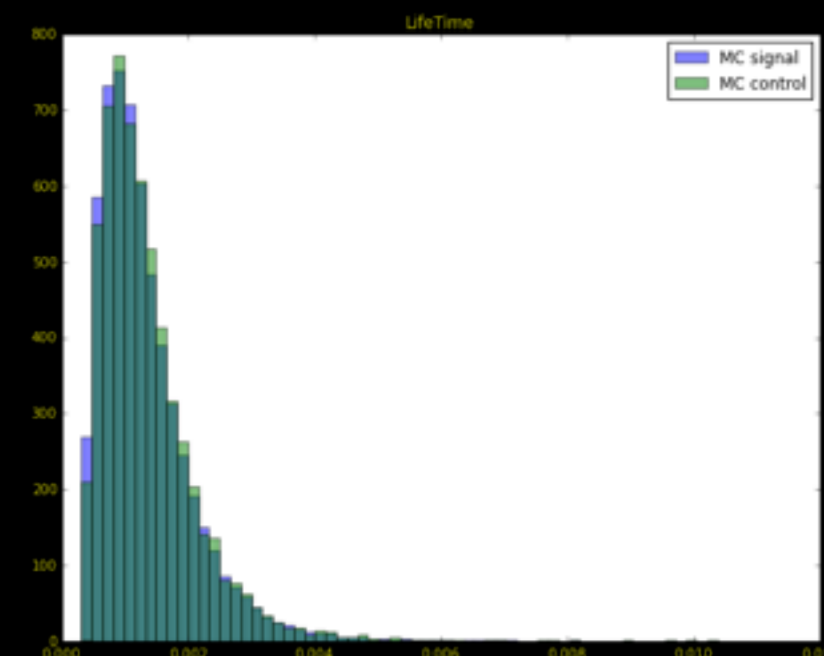
Systematic error source: MC vs MC

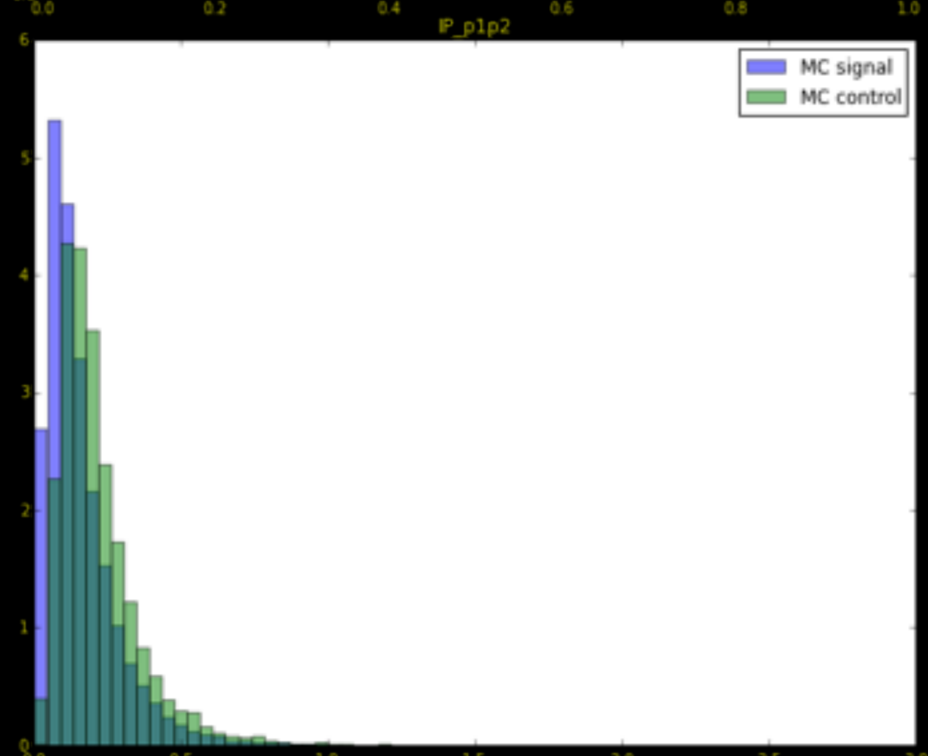
- One of systematic sours is the MC different for the signal and the control channel.
- Simple way to make sure is to train a model to distinguish them
- Usually the classifier ignore this difference and we can check it computing KS between predictions for both MC

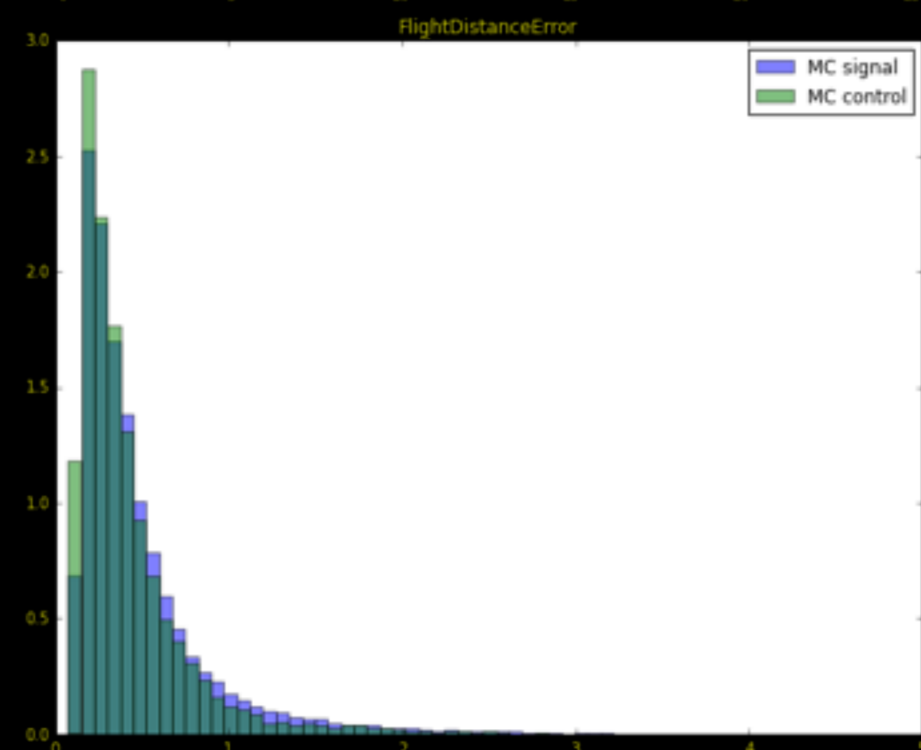
Systematic error source: MC vs MC



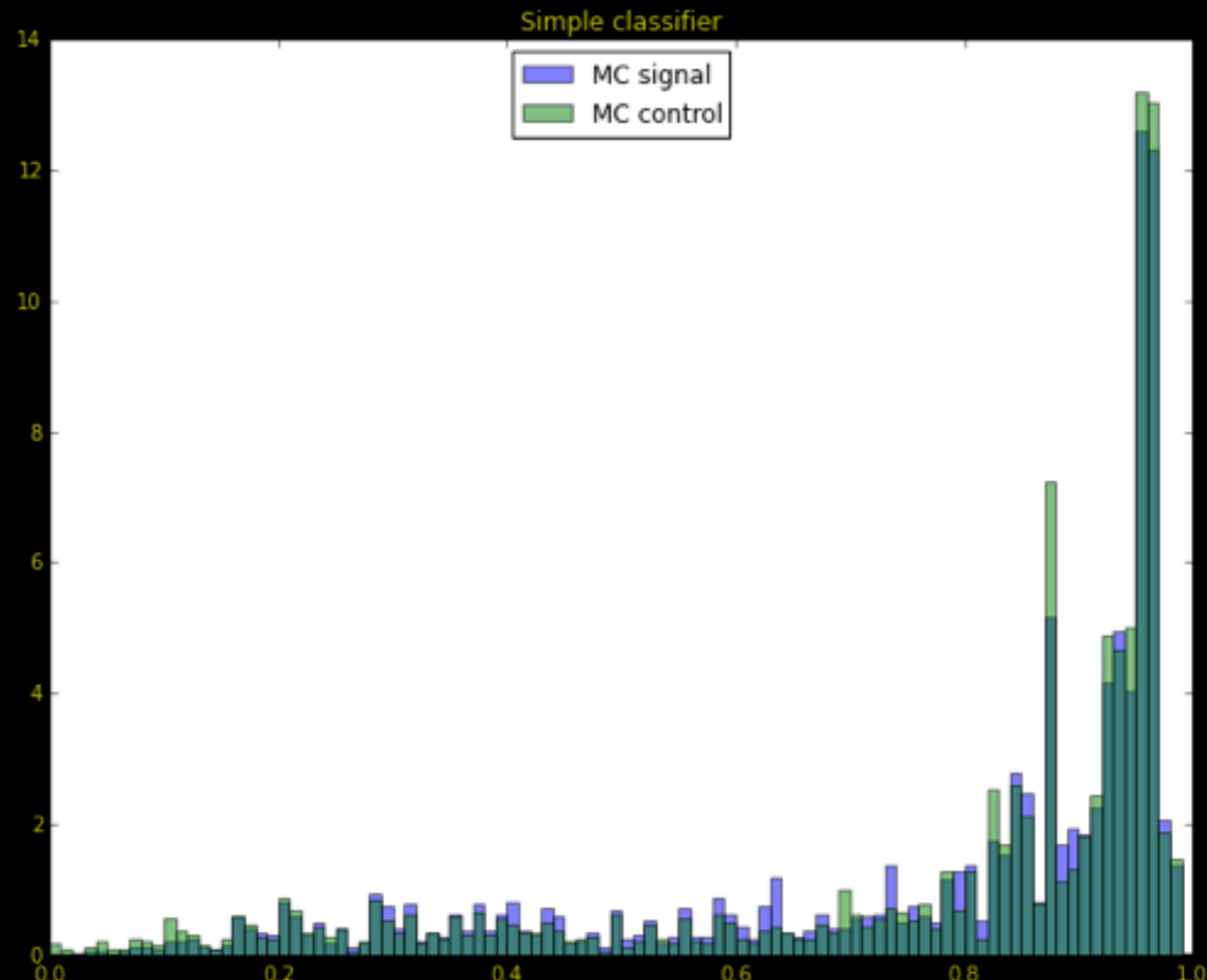


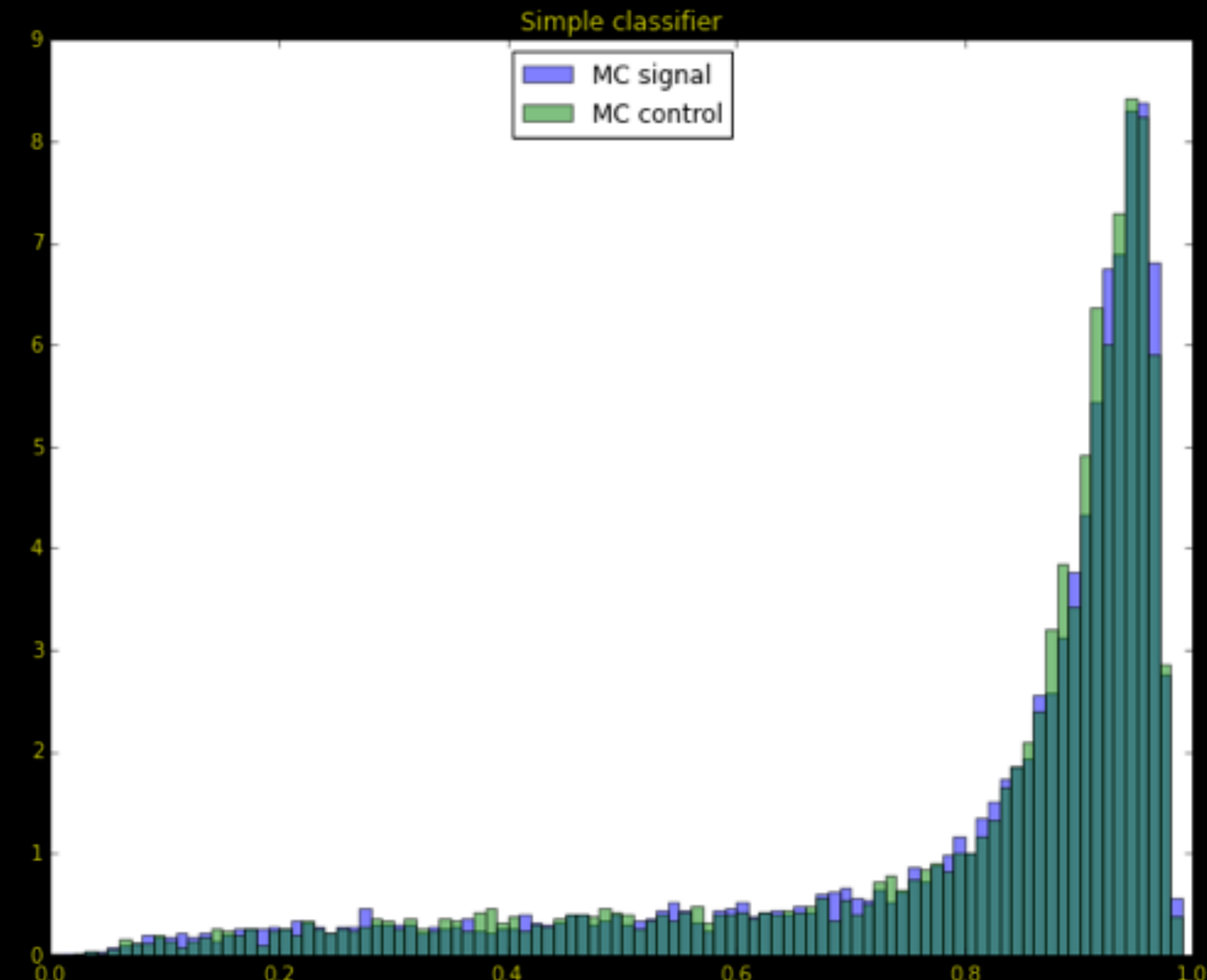






Systematic error source: MC vs MC

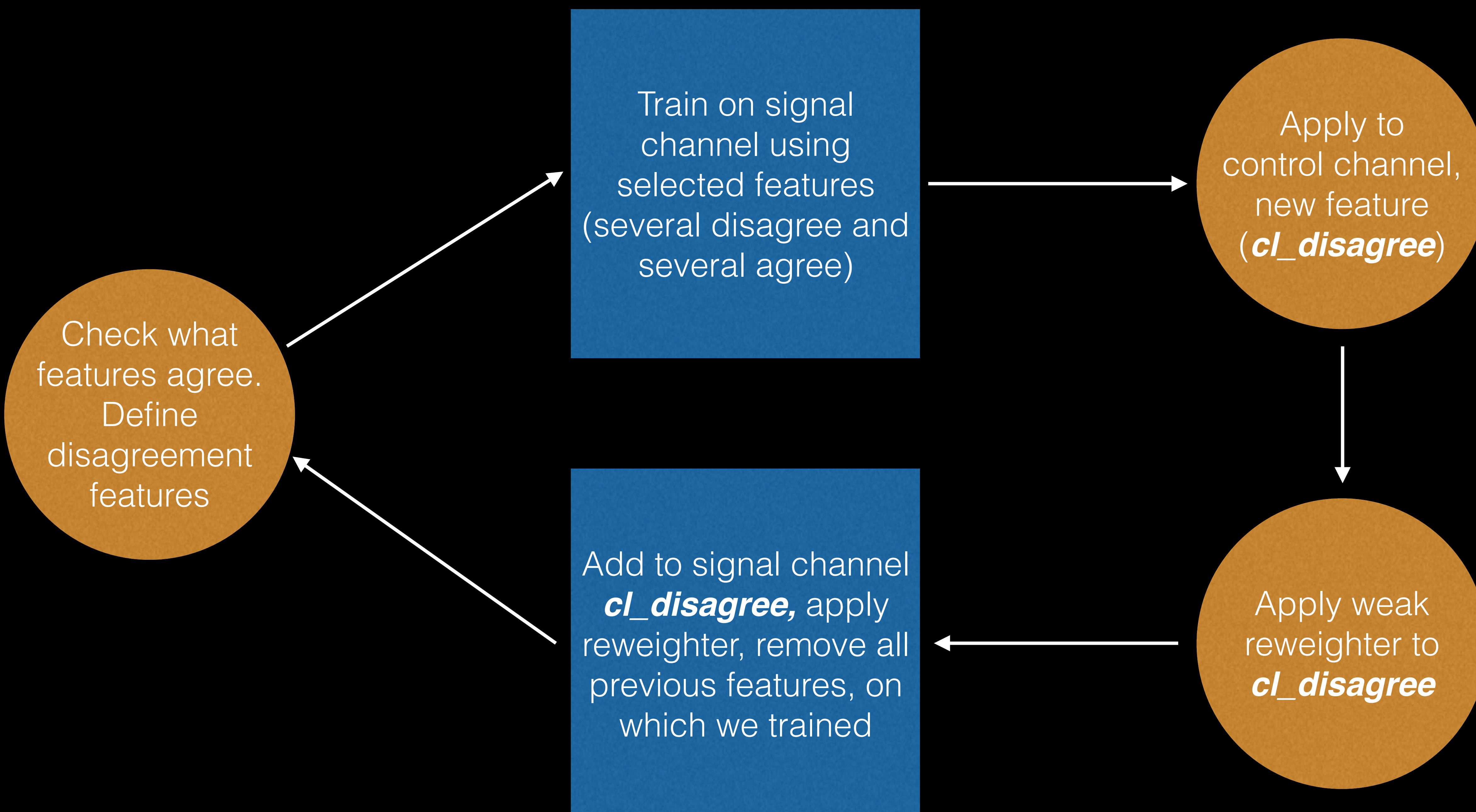




`statistic=0.0498 pvalue=2.4e-10`

`statistic=0.0154, pvalue=0.226`

Iterative learning (feature extraction)



Feature selection in HEP

- In HEP feature selection is often connected to select those which can help find new physics (like those which are not influence on the mass correlation).
- It is actual problem for trigger system (only save interesting events).
- In trigger system feature using can improve model but it will select only some region of interesting events.
- Often you try to remove some features (to save regions with possible new physics).
- Here is tradeoff between removing and saving the same quality for the basic regions.