JETS FAKING PHOTONS 2019

Cartella Skipper

La cartella contiene:

2 TSelector

2 .h e 2 .C, uno per i dati e uno per i MC, le simulazioni hanno infatti delle branch per i pesi che i dati non hanno e solo quindi necessari 2 Selector. Sono lo stesso Selector a cui è stato cambiato il nome e sono state tolte le variabili per i pesi.

Ogni Selector prende i parametri di input (calo e track isolation, divisioni di met, bin di tightness) dal file selector_parameters.txt e può essere eseguito con il comando TTreeXY->Process("SelectorXY.C") (fatto in automatico da routine.sh) a cui non si possono passare argomenti, sono quindi definite due variabili nel .h:

```
#define sample "Znunugamma"
#define directory " 15-16"
```

che vengono modificate da routine.sh per ciclare sui vari MC e su dati e MC di diversi anni. IMPORTANTE: bisogna lasciare queste variabili nelle righe 18 e 19, in modo che routine.sh riscriva le variabili nelle righe giuste.

I due Selector calcolano le popolazioni da usare nel metodo ABCD, producendo lo stesso formato di risultati organizzato come un dizionario di Python e facilmente leggibile tramite il modulo "yaml" di Python. La gerarchia è del dizionario è:

SR o CR —> ISR o ESR —> tightness —> isolamento

Queste matrici tengono conto delle popolazione necessarie per il calcolo delle sistematiche su tightness e isolamento. Questo viene fatto calcolando tutte le popolazioni con le selezioni Tight, Tight-3, Tight-4 e Tight-5, e variando il gap di isolamento come definito in selector_parameters.txt (iso_gap[i]).

NB: nel Selector dei MC il nome del sample stabilisce se il campione è di segnale o di background. Il campione di gammajets può essere usato per tutti e due ed è necessario cercare il fotone o jet anche nell'oggetto subleading, questo viene fatto alle righe 96-107 di SelectorMC.C, ma non sono totalmente sicuro che sia corretta la ricerca nel leading o subleading object, andrebbe rivista.

routine.sh

Eseguibile che apre ROOT e lancia i selector. La prima parte (righe 7-27) crea la cartella “dati” se non esiste, e all’interno crea tre cartelle in cui vengono salvate le matrici prodotte. La seconda parte (righe 33-104) ciclo sui vari campioni MC e su tutti i dati, sia per i MC sia per i dati si cicla anche sugli anni 2015-16, 2017 e 2018.

derivate_complete.py

File che contiene tutte le derivate necessari per la propagazione degli errori sulla purezza, generate con Mathematica.

functions.py

Funzioni necessarie per l’analisi dati:

- merge_years: mergia le popolazioni di diversi anni, è l’unico da toccare per analizzare diversi anni
- merge: mergi SRs e CRs opportune
- correlation_factor e leakage_coefficient: calcola i coefficienti
- purezza: calcola le purezze e gli errori associati
- analyze_coefficient: analizza i coefficienti

Kowalsky

Notebook eseguibile con Python3 che analizza i dati della cartella “dati”. Calcola tutte le purezze con relativi errori statistici e sistematici e calcola poi le sistematiche dovute ai coefficienti e infine salva i dati. Alla fine c’è anche una parte di confronto con i risultati dell’analisi precedente (solo per dati 2015-16).

NB: ci sono problemi di statistica dei dati 2017 nel calcolo delle purezze usate per stimare le sistematiche, nella tightness non si può fare il calcolo con Tight-3 nella 2muCR - ISR3 e nelle sistematiche su c3 in tutte le ISR3. Ci sono quindi un paio di righe (commentate) da scommentare solo per analizzare i dati

solo 2017.

##########################################################################
Aggiornamento 19/12/19

Per definire i bin di met:

1. Definire il numero e i bin nel file selector_parameter.txt
2. Modificare il valore di numberOfMetBins nel file functions.py

##########################################################################