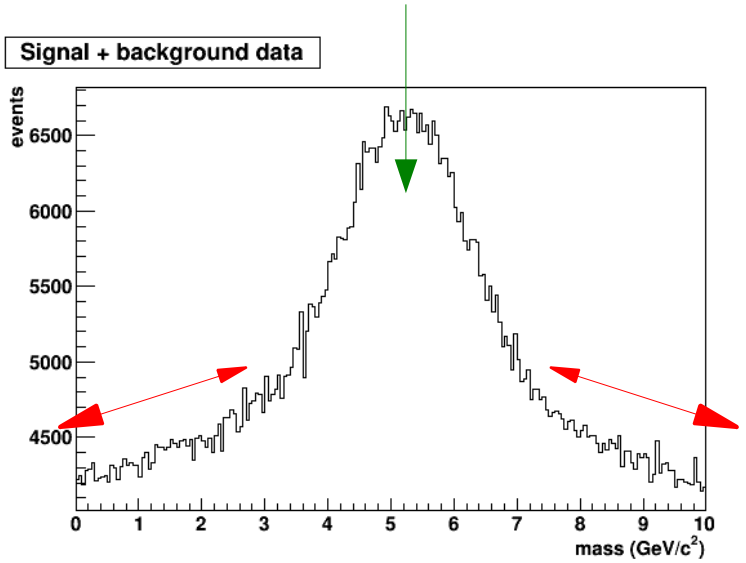
Introduzione a ROOT (parte III)

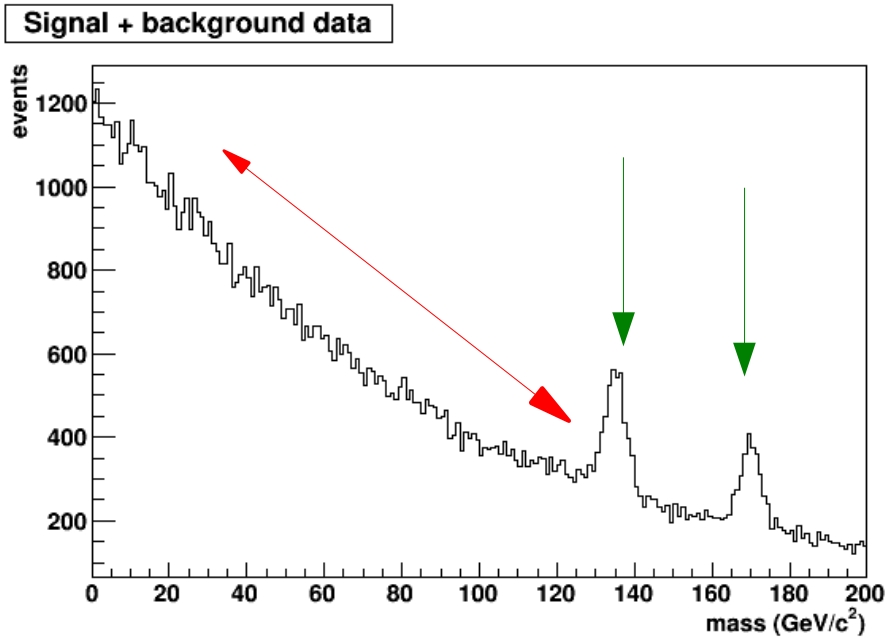
- *Fitting* in ROOT

- TProfile

Segnale più fondo

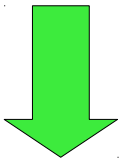
- Supponiamo che il risultato di un esperimento fornisca un *set* di dati (ad es. la massa di una particella), che rappresentiamo nei seguenti istogrammi 1D. Si distinguono due contributi:
  - "eventi non interessanti", che contaminano il *sample* (**background**)
  - "eventi interessanti", per i quali è stato condotto l'esperimento (**segnale**)





Estrarre il segnale

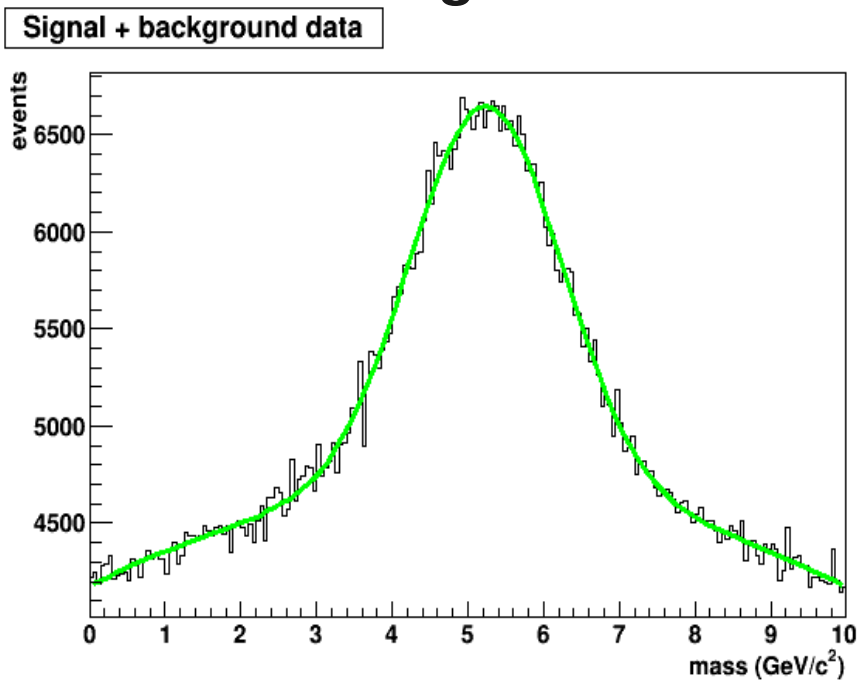
- Può essere necessario **isolare il contributo del segnale**, per esempio per identificare le caratteristiche del picco (la posizione, la larghezza…) e per contare il numero di eventi di segnale. Perciò è necessario:
  - Conoscere a priori la forma attesa per segnale e fondo
  - Parametrizzare segnale e fondo con opportune funzioni



- È necessario un ***fit*!** Vediamo due approcci differenti:
  - *Fit* con funzione S+B
  - Metodo del *sideband*

FIT Segnale più fondo

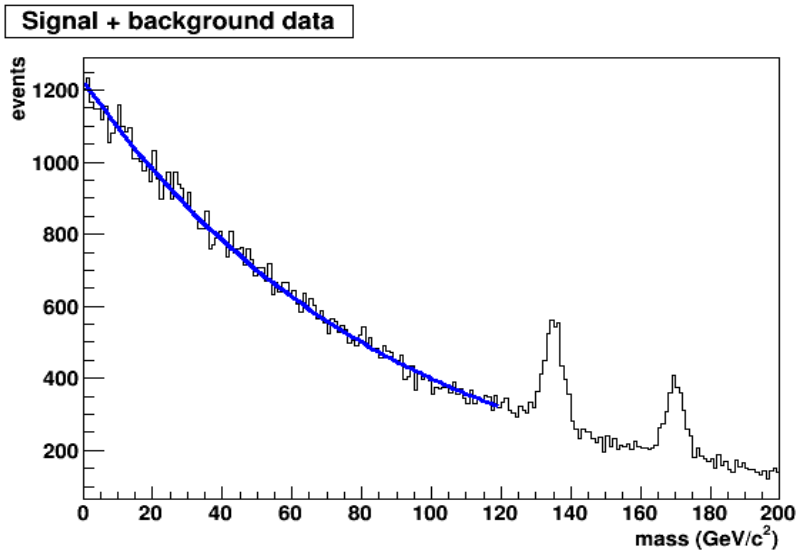
- Se i contributi di segnale e di fondo non sono facilmente distinguibili
  - si costruisce la **funzione somma** dei contributi di **segnale** e **fondo**
  - per ciascuna componente si introducono i parametri liberi di *fit*
  - Dopo aver *fittato* l'istogramma, i parametri della componente di segnale danno informazioni sul picco

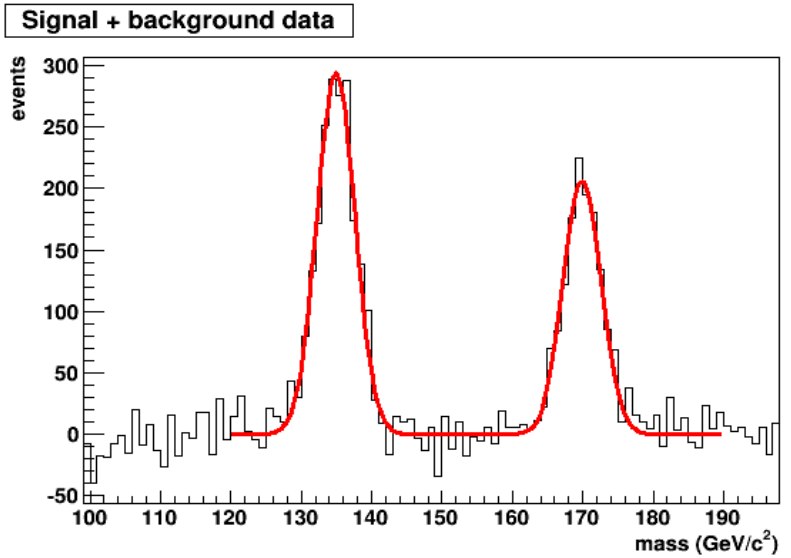


- N.B.: con un alto numero di parametri liberi, il *fit* dell'istogramma diventa instabile e **spesso non converge**
- Maggiore è il numero di gradi di libertà e più difficile è la convergenza del fit -> inizializzare i parametri!!!

FIT Sideband

- Se è facile individuare una regione in cui solo il fondo è presente:
  - si esegue il *fit* di quella regione con la funzione che parametrizza il fondo (con un **numero ridotto** di **parametri** liberi)
  - si sottrae al contenuto di ciascun *bin* dell'istograma il valore della funzione calcolata nel centro del *bin*
  - quello che rimane è un **istogramma di solo segnale**





Reminder per il Fit in Root

- Potete definire l'epressione matematica della TF1 che usate per il fit
  - Al momento della creazione della TF1

```
TF1* user = new TF1("user", "[0]*(1. - exp(-x/[1]))", 0., 0.01);
```

  - Come funzione di C++ che poi viene richiamata dal costruttore delle TF1

Definizione delle funzioni → **Loro utilizzo nel main/macro di fit**

```
// The gaussian function
double gaussian (double* x, double* par)
{
  // gaussiana
  double arg = (x[0] - par[1])/par[2] ;
  double val = par[0] * exp (-0.5*arg*arg) ;

  return val ;
}

// The signal function
double signal (double* x, double* par)
{
  return gaussian(x, par) + gaussian(x, &par[3]) + gaussian(x, &par[6]) ;
}
```

```
double signalPar[9] = {500., 9.5., 0.1,
                       100., 10., 0.1,
                       20., 10.35, 0.1} ;

TF1* signalfunc = new TF1 ("signalfunc", signal, xMin, xMax, 9) ;
signalfunc -> SetParameters (signalPar) ;
signalfunc -> SetLineColor (kRed) ;
```

Ora tocca a voi

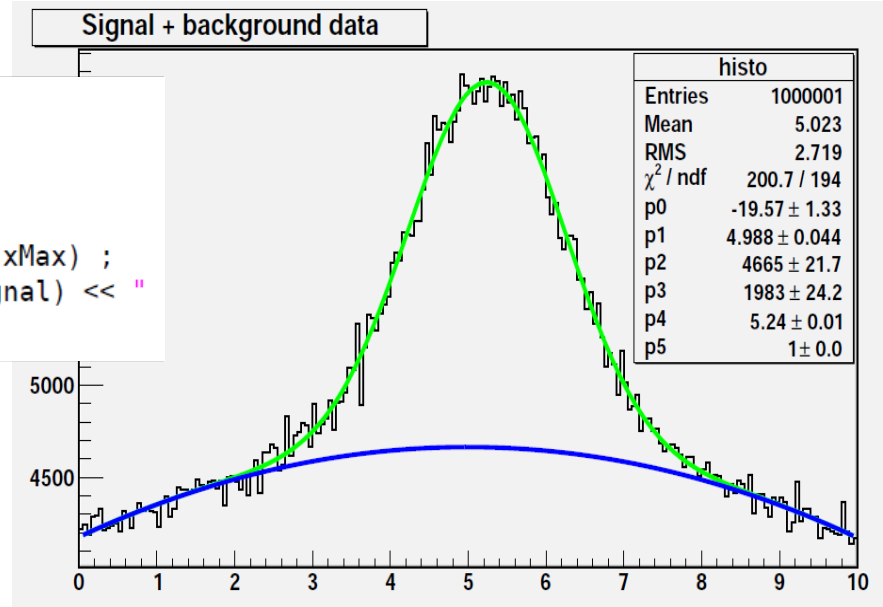
- Scaricate dal blog i *file* che contengono i risultati di un ipotetico esperimento di conteggio segnale+fondo
  - `data_funzione.txt`
  - `data_sidebands.txt`
- Scoprite quali sono i segnali in essi celati, cioè trovate:
  - I valori di ascissa su cui sono centrati i picchi e la loro larghezza (sigma gaussiana)
  - Il numero di eventi di segnale contenuti nei picchi
- Si sa inoltre che i picchi attesi sono di forma gaussiana, mentre il fondo concorrente all'osservazione del segnale segue una distribuzione parabolica, nel primo caso, ed esponenziale nel secondo.

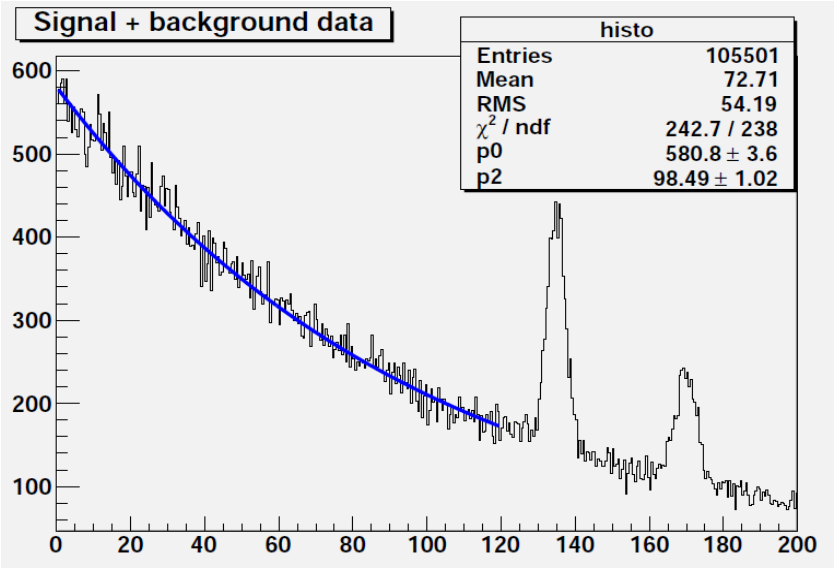
Calcolo conteggi segnale

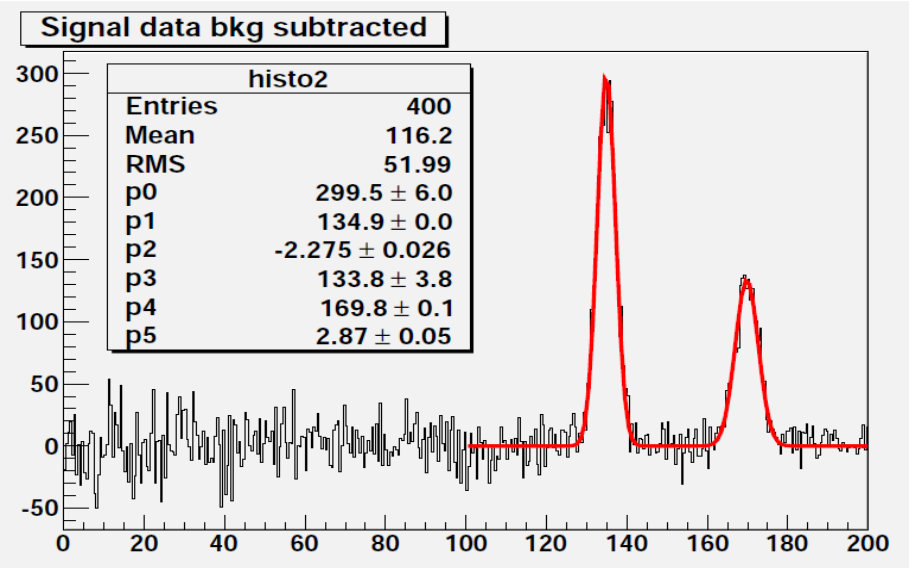
```
  // Count signal events - Integrale netto della gaussiana
  signalfunc -> SetParameter (0, sumfunc -> GetParameter (3)) ;
  signalfunc -> SetParameter (1, sumfunc -> GetParameter (4)) ;
  signalfunc -> SetParameter (2, sumfunc -> GetParameter (5)) ;
  double nSignal = nBin / (xMax - xMin) * signalfunc -> Integral (xMin, xMax) ;
  std::cout << "\n****** Signal + Background Function ---> " << int(nSignal) << "
signal events" << std::endl ;
```

CASO1: Signal + Background Function ---> 99448 signal events

CASO2: Sidebands ---> 5341 signal events

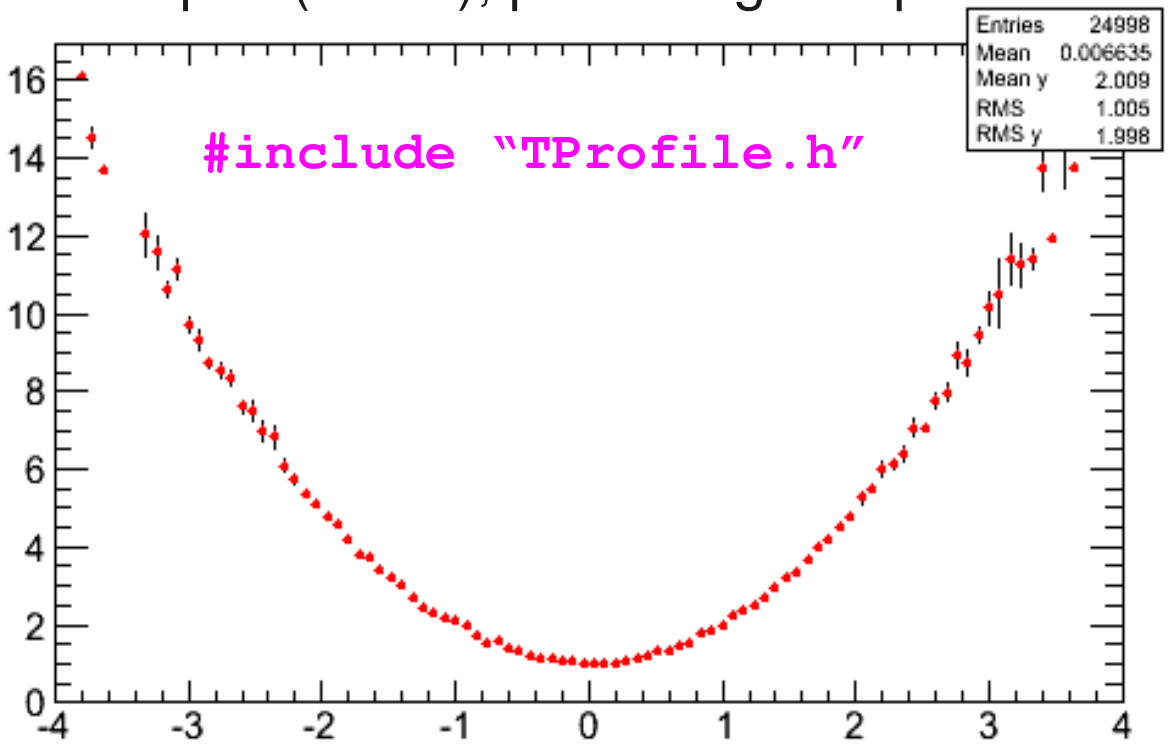






TProfile

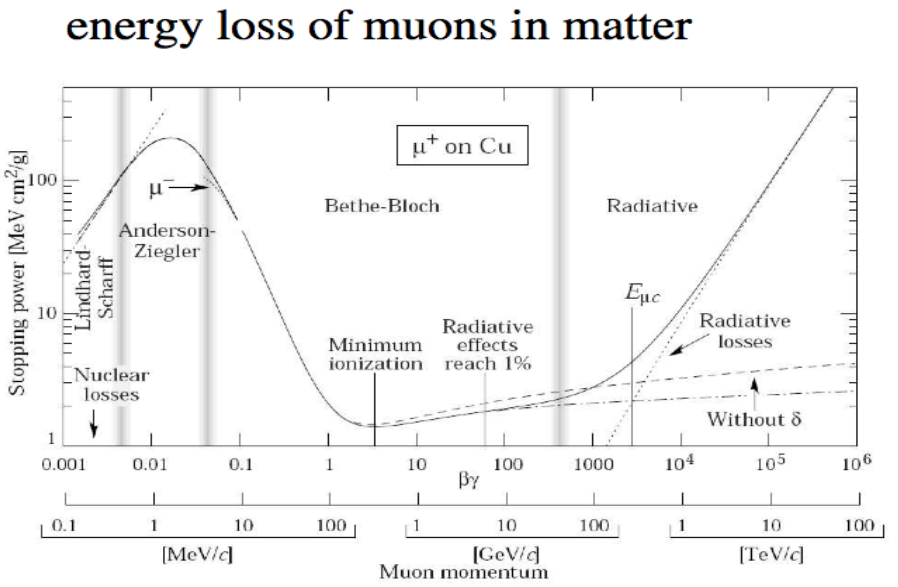
- Rappresenta Ymedio e Yrms, per ogni bin in X
- Consente di vedere correlazioni tra le variabili
  - quando Y è approssimativamente una funzione di X è da preferirsi ad uno scatter-plot (TH2F), per la migliore precisione



TProfile : Stopping Power

- Utilizzo di un `TProfile` in un contesto di fisica:
  - costruiamo la curva di <*dE/ρdx*> che descrive la **perdita di energia media nella materia** per particelle cariche (pesanti) in funzione della quantità di moto (momento) della particella.
- La perdita di energia è dovuta principalmente a
  - irraggiamento (per particelle leggere di alto momento)
  - collisioni con elettroni del mezzo
- <*dE/ρdx*>
  - è ~ indipendente

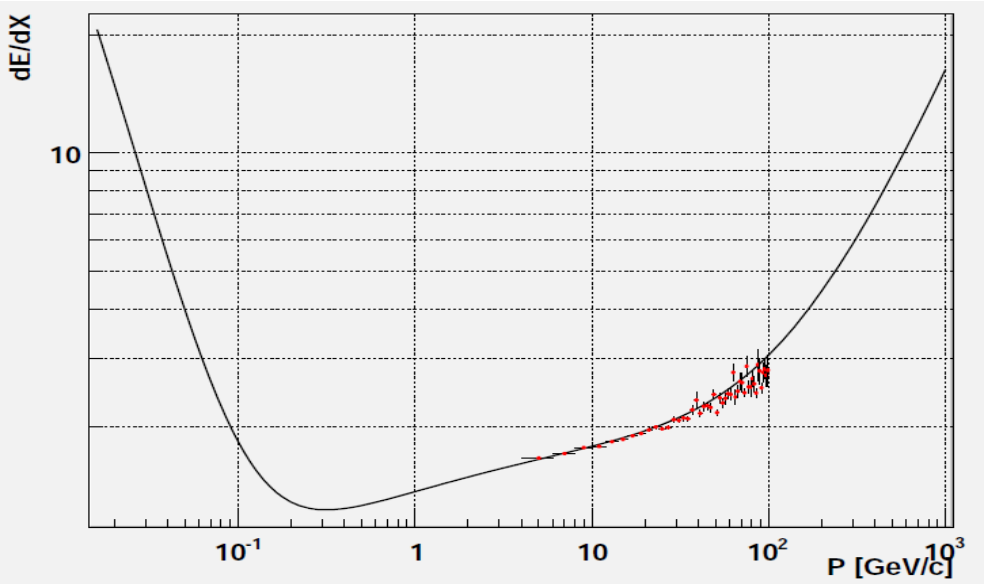
    dal mezzo attraversato
  - ha una forma caratteristica



TProfile : Stopping Power

- Esercizio per voi per testare il `TProfile` :
  - Costruite direttamente il Tprofile di <*dE/ρdx*> riempiendolo con i dati sperimentali contenuti nel file “stoppingPower_exp.dat”
  - Confrontate il dati sperimentali con la curva teorica descritta in “stoppingPower_th.dat”

```
TProfile* StopPow_profile = new TProfile ("StopPow_profile", "StopPow_profile", 500, 0., 1000.) ;

std::ifstream in_FileExp (input,std::ios::in);
while(!in_FileExp.eof())
{
  float muonP, dEdX;
  in_FileExp >> muonP >> dEdX;
  StopPow_profile->Fill(muonP,dEdX);
}
in_FileExp.close();

StopPow_profile->SetMarkerStyle(20);
StopPow_profile->SetMarkerColor(kRed);
StopPow_profile->SetMarkerSize(0.4);

StopPow_profile->Draw("same");
```



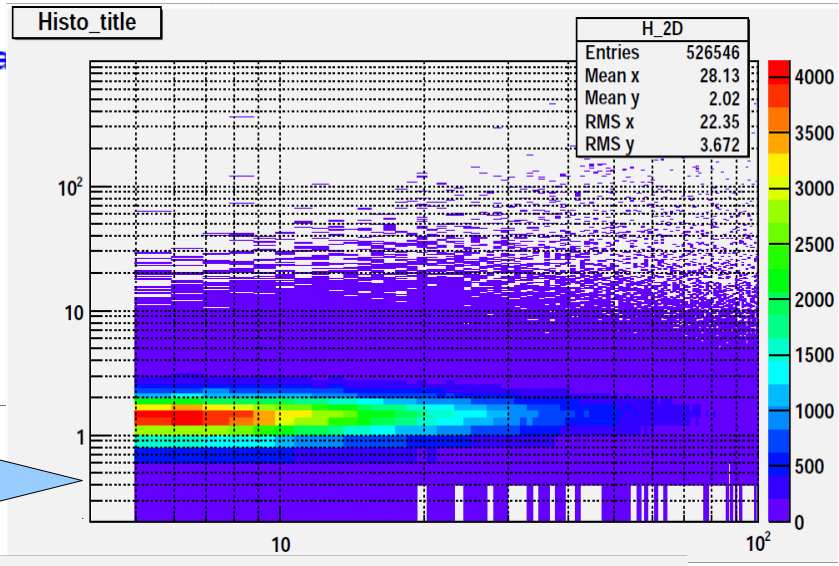
TProfile : Stopping Power

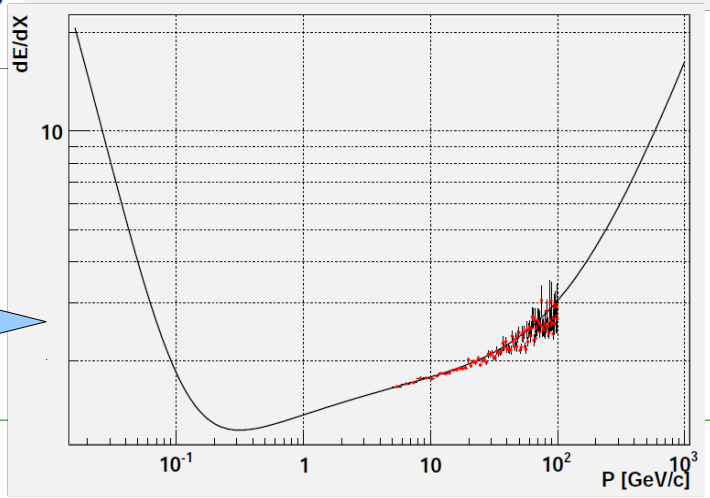
- Esempio 2: Costruire un Tprofile a partire da un TH2F generato dai dati salvati in una Ntupla

```
// Costruisco una NTupla per contenere i dati sperimentali
TNtuple* NTu = new TNtuple ("Ntu_data","Ntu_title","muonP:dEdX");

// Apro il file di dati sperimentali e riempio la NTupla
std::ifstream in_FileExp (input,std::ios::in);
while(!in_FileExp.eof())
{
  float x,y;
  in_FileExp >> x >> y;
  NTu->Fill(x,y);
}
in_FileExp.close();
```

```
//Creo un TH2F a partire dalla NTupla
TH2F *isto2d= new TH2F ("H_2D","Histo_title",100,4,100,5000,0.,1000);
NTu->Draw("dEdX:muonP >> H_2D","","COLZ");
```

```
// OPERAZIONE DI PROFILIZZAZIONE DI UN TH2F
TProfile *isto2d_profilized = new TProfile();
isto2d_profilized = isto2d-> ProfileX();

isto2d_profilized->Draw("same");
```





TProfile : Stopping Power

- Se si prova a visualizzare il TH2F dei dati con l'opzione “SURF3”...

```
//Creo un TH2F a partire dalla NTupla
TH2F *isto2d= new TH2F ("H_2D","Histo_title", 100,4,100,5000,0.,1000);
NTu->Draw("dEdX:muonP >> H_2D","","SURF3");
```