METODI STATISTICI E COMPUTAZIONALI

Stefania Spagnolo

Dipartimento di Matematica e Fisica, Univ. del Salento

ESEMPIO SIMULAZIONE

Spettro di energia di una sorgente Originale e Misurato

ESEMPIO SIMULAZIONE

- In http://www.dmf.unisalento.it/~spagnolo/MSC_aa21_22/EsempioSimulazione/
  - **EsempioSimulazione.C**
    - # genera uno spettro di energia vera, e uno di energia misurata emulando efficienza (dipendente dall'energia) e smearing gaussiano secondo una certa risoluzione
    - Si esegue con
      - root [0] .x EsempioSimulazione.C
  - Produce
    - **EsempioSimulazione.root**
    - Contiene un TTree T->>>>
      - chiamato Simulazione

```
 TFile*         EsempioSimulazione.root
  KEY: TTree    T;1     Simulazione
root [3] T->Print()
******************************************************************************
*Tree    :T         : Simulazione                                             *
*Entries :  1000000 : Total =        36104659 bytes  File  Size =   14298295 *
*        :          : Tree compression factor =   2.53                       *
******************************************************************************
*Br    0 :Id        : Id/I                                                    *
*Entries :  1000000 : Total  Size=    4011129 bytes  File Size  =     176002 *
*Baskets :      126 : Basket Size=      32000 bytes  Compression=  22.77     *
*.............................................................................*
*Br    1 :EVera     : EVera/D                                                 *
*Entries :  1000000 : Total  Size=    8022652 bytes  File Size  =    7522322 *
*Baskets :      251 : Basket Size=      32000 bytes  Compression=   1.07     *
*.............................................................................*
*Br    2 :EMisurata : EMisurata/D                                             *
*Entries :  1000000 : Total  Size=    8023672 bytes  File Size  =    6029446 *
*Baskets :      251 : Basket Size=      32000 bytes  Compression=   1.33     *
*                                                                            *
```

ESEMPIO SIMULAZIONE

- In http://www.dmf.unisalento.it/~spagnolo/MSC_aa21_22/EsempioSimulazione/
  - **ProcessaSimulazione.h, ProcessaSimulazione.C**
  - # processa il TTRee Simulazione
  - Prodotta da Root con
    - root EsempioSimulazione.root
      - root [3] T->MakeClass("ProcessaSimulazione")
  - Modificata da noi per aggiungere
    - Nuovi dati della classe ProcessaSimulazione
      - Contatori, puntatori a istogrammi, ecc
    - Nuovi metodi:
      - AtTheStart()
      - AtTheEnd()
- Il metodo Loop() lancia il loop sugli eventi del TTree “Simulazione”
- Il TTree e’ fissato come “input” della classe nel costruttore

ESEMPIO SIMULAZIONE

- In http://www.dmf.unisalento.it/~spagnolo/MSC_aa21_22/EsempioSimulazione/
  - **ProcessaSimulazione.h, ProcessaSimulazione.C**

```
#ifdef ProcessaSimulazione_cxx
ProcessaSimulazione::ProcessaSimulazione(TTree *tree) : fChain(0)
{
// if parameter tree is not specified (or zero), connect the file
// used to generate this class and read the Tree.
   if (tree == 0) {
      TFile *f = (TFile*)gROOT->GetListOfFiles()->FindObject("EsempioSimulazione.
      if (!f || !f->IsOpen()) {
         f = new TFile("EsempioSimulazione.root");
      }
      f->GetObject("T",tree);

   }
   Init(tree);
}
```

In ProcessaSimulazione.h

- Il metodo Loop() lancia il loop sugli eventi del TTree “Simulazione”
- Il TTree e’ fissato come “input” della classe nel costruttore

ESEMPIO SIMULAZIONE

- In http://www.dmf.unisalento.it/~spagnolo/MSC_aa21_22/EsempioSimulazione/
  - **ProcessaSimulazione.h, ProcessaSimulazione.C**
  - Una classe che consente l'analisi dei dati nel TTRee Simulazione evento per evento
    - Tutte le "operazioni di servizio" (apertura file, accesso alle branch del TTree, ecc) sono gestite (letteralmente "scritte") da Root nello scheletro della classe (prodotto da TTree::MakeClass())
    - Le "operazioni analisi" sono da implementare nel metodo ProcessaSimulazione::Loop()
  - Si usa in questo modo:
    - // root> **.L ProcessaSimulazione.C++ (con ++ il codice e' *compilato* in anticipo, piuttosto che *interpretato* run-time)**
    - // root> ProcessaSimulazione t
    - // root> t.GetEntry(12); // Fill t data members with entry number 12
    - // root> t.Show(); // Show values of entry 12
    - // root> t.Show(16); // Read and show values of entry 16
    - // root> t.Loop(); // Loop on all entries

ESEMPIO SIMULAZIONE. ROOT 5

- In http://www.dmf.unisalento.it/~spagnolo/MSC_aa21_22/EsempioSimulazione/
  - **ProcessaSimulazione.h, ProcessaSimulazione.C**
  - Una classe che consente l'analisi dei dati nel TTRee Simulazione evento per evento
    - Tutte le "operazioni di servizio" (apertura file, accesso alle branch del TTree, ecc) sono gestite (letteralmente "scritte") da Root nello scheletro della classe (prodotto da TTree::MakeClass())
    - Le "operazioni analisi" sono da implementare nel metodo ProcessaSimulazione::Loop()
  - Si usa in questo modo:
    - // root> **.L ProcessaSimulazione-r5.C++ (con ++ il codice e' *compilato* in anticipo, piuttosto che *interpretato* run-time)**
    - // root> ProcessaSimulazione t
    - // root> t.GetEntry(12); // Fill t data members with entry number 12
    - // root> t.Show(); // Show values of entry 12
    - // root> t.Show(16); // Read and show values of entry 16
    - // root> t.Loop(); // Loop on all entries

UNFOLDING

- Da
  - G. D'Agostini, Nuclear Instruments and Methods in Physics Research A 362 (1995) 487

If one observes $n(\mathrm{E})$ events with effect E, the expected number of events assignable to each of the causes is

$$\hat{n}(\mathrm{C}_i) = n(\mathrm{E})\,P(\mathrm{C}_i|\mathrm{E}). \qquad (2)$$

As the outcome of a measurement one has several possible effects E_j $(j = 1, 2, \cdots, n_\mathrm{E})$ for a given cause C_i. For each of them the Bayes formula (1) holds, and $P(\mathrm{C}_i|\mathrm{E}_j)$ can be evaluated. For simplicity we will refer to conditional probabilities $P(\mathrm{C}_i|\mathrm{E}_j)$ as *smearing matrix* **S**, even if they describe cell-to-cell migration. Let us write Eq. (1) again in the case of n_E possible effects [1], indicating the initial probability of the causes with $P_0(\mathrm{C}_i)$:

$$P(\mathrm{C}_i|\mathrm{E}_j) = \frac{P(\mathrm{E}_j|\mathrm{C}_i)P_0(\mathrm{C}_i)}{\sum_{l=1}^{n_\mathrm{C}} P(\mathrm{E}_j|\mathrm{C}_l)P_0(\mathrm{C}_l)}. \qquad (3)$$

One has to note that:

— $\sum_{i=1}^{n_\mathrm{C}} P_0(\mathrm{C}_i) = 1$, as usual. Notice that if the probability of a cause is initially set to zero it can never change, i.e. if a cause does not exist it cannot be invented;

— $\sum_{i=1}^{n_\mathrm{C}} P(\mathrm{C}_i|\mathrm{E}_j) = 1$: this normalization condition, mathematically trivial since it comes directly from Eq. (3), tells that each effect must come from one or more of the causes under examination. This means that if the observables contain also a non-negligible amount of background, this needs to be included among the causes;

— $0 \le \epsilon_i \equiv \sum_{j=1}^{n_\mathrm{E}} P(\mathrm{E}_j|\mathrm{C}_i) \le 1$: there is no need for each cause to produce at least one of the effects taken under consideration. ϵ_i gives the *efficiency* of detecting the cause C_i in any of the possible effects.

UNFOLDING

- Da
  - G. D'Agostini, Nuclear Instruments and Methods in Physics Research A 362 (1995) 487

After N_{obs} experimental observations one obtains a distribution of frequencies $\boldsymbol{n}(\mathrm{E}) \equiv \{n(\mathrm{E}_1), n(\mathrm{E}_2), \cdots, n(\mathrm{E}_{n_\mathrm{E}})\}$. The expected number of events to be assigned to each of the causes and only due to the observed events can be calculated applying Eq. (2) to each effect:

$$\hat{n}(\mathrm{C}_i)\,|_{\mathrm{obs}} = \sum_{j=1}^{n_\mathrm{E}} n(\mathrm{E}_j)\, P(\mathrm{C}_i|\mathrm{E}_j).$$

Taking into account the inefficiency [2], the best estimate of the true number of events is then

$$\hat{n}(\mathrm{C}_i) = \frac{1}{\epsilon_i} \sum_{j=1}^{n_\mathrm{E}} n(\mathrm{E}_j)\, P(\mathrm{C}_i|\mathrm{E}_j) \quad \epsilon_i \neq 0. \qquad (4)$$

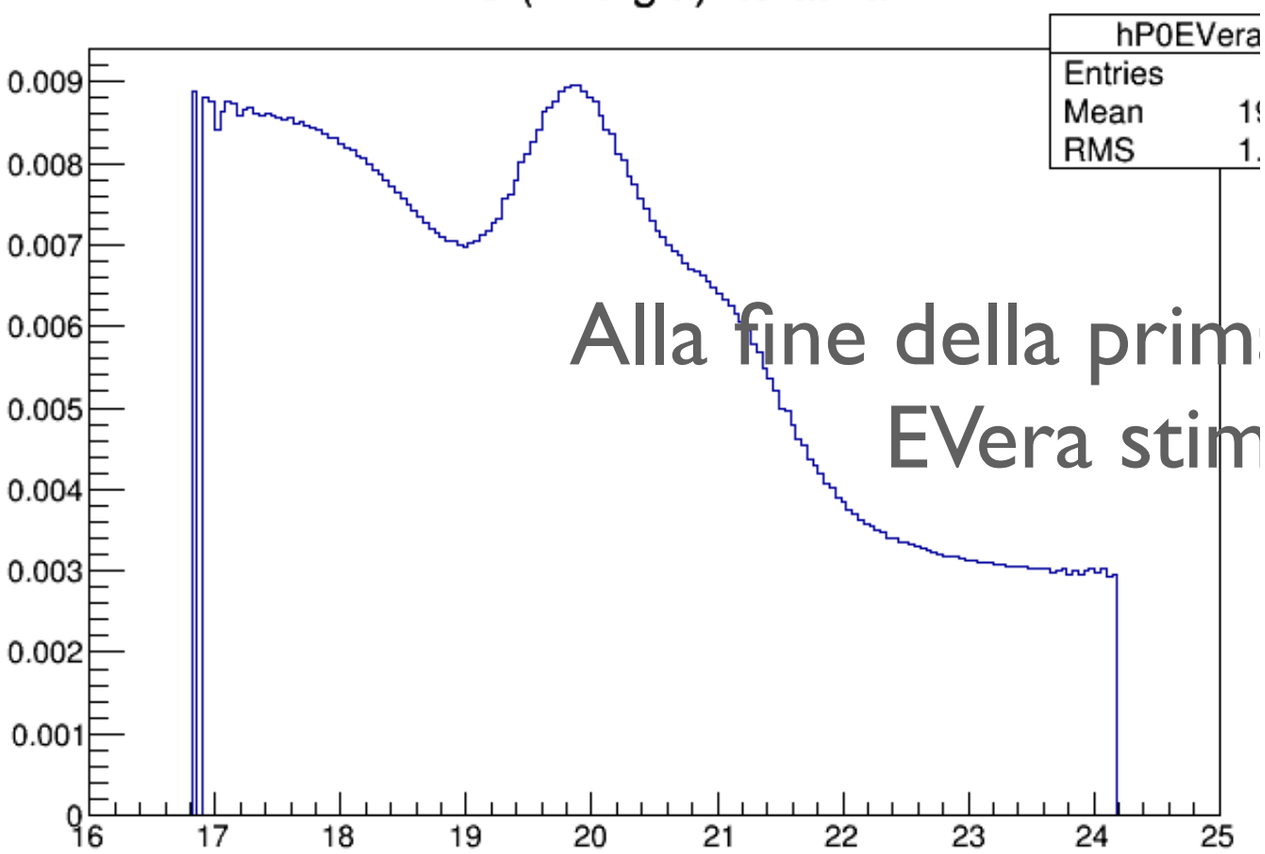
From these unfolded events we can estimate the true total number of events, the final probabilities of the causes and the overall efficiency:

$$\hat{N}_{\mathrm{true}} = \sum_{i=1}^{n_\mathrm{C}} \hat{n}(\mathrm{C}_i),$$

$$\hat{P}(\mathrm{C}_i) \equiv P(\mathrm{C}_i\,|\,\boldsymbol{n}(\mathrm{E})) = \frac{\hat{n}(\mathrm{C}_i)}{\hat{N}_{\mathrm{true}}},$$

$$\hat{\epsilon} = \frac{N_{\mathrm{obs}}}{\hat{N}_{\mathrm{true}}}.$$

If the initial distribution $\boldsymbol{P}_0(\mathrm{C})$ is not consistent with the data, it will not agree with the final distribution $\hat{\boldsymbol{P}}(\mathrm{C})$. The closer the initial distribution is to the true distribution, the better the agreement is. One can easily verify for simulated data (see e.g. the non-trivial cases shown in Section 6) that the distribution $\hat{\boldsymbol{P}}(\mathrm{C})$ lies between $\boldsymbol{P}_0(\mathrm{C})$ and the true one. This suggests to proceed iteratively. So



Alla fine della prim
EVera stim

Punto di partenza

If the initial distribution $P_0(C)$ is not consistent with the data, it will not agree with the final distribution $\hat{P}(C)$. The closer the initial distribution is to the true distribution, the better the agreement is. One can easily verify for simulated data (see e.g. the non-trivial cases shown in Section 6) that the distribution $\hat{P}(C)$ lies between $P_0(C)$ and the true one. This suggests to proceed iteratively. So

UNFOLDING

- Da
  - G. D'Agostini, Nuclear Instruments and Methods in Physics Research A 362 (1995) 487

the unfolding can be performed through the following steps:

1) choose the initial distribution of $P_0(C)$ from the best knowledge of the process under study, and hence the initial expected number of events $n_0(C_i) = P_0(C_i)N_{obs}$; in case of complete ignorance, $P_0(C)$ will be just a uniform distribution: $P_0(C_i) = 1/n_C$;

2) calculate $\hat{n}(C)$ and $\hat{P}(C)$;

3) make a χ^2 comparison between $\hat{n}(C)$ and $n_0(C)$;

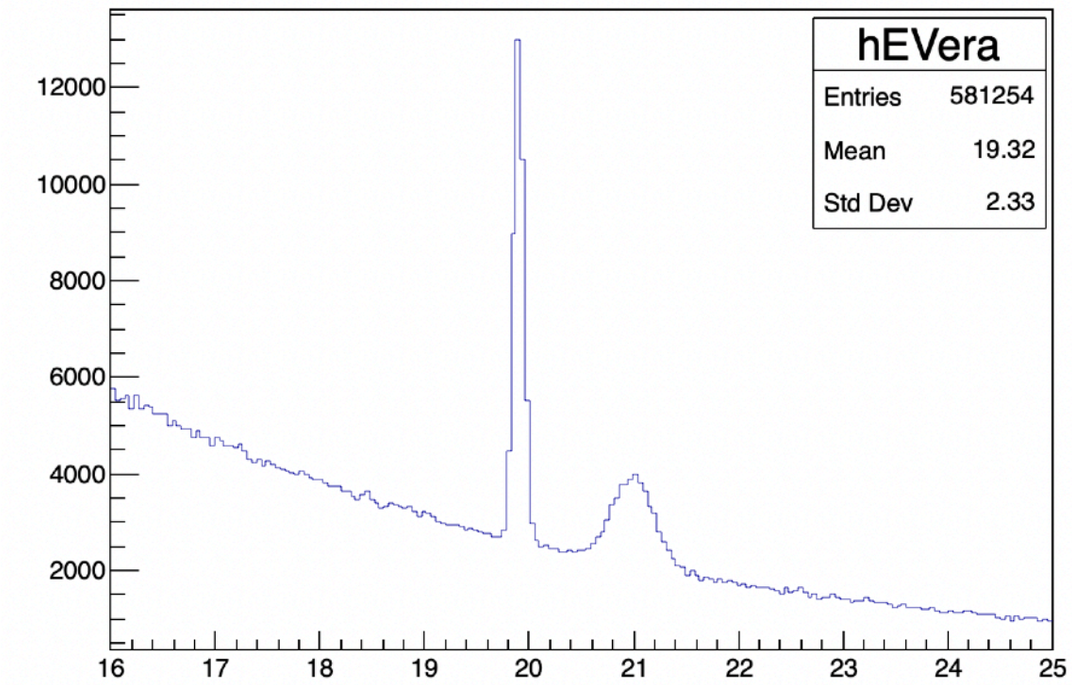
4) replace $P_0(C)$ by $\hat{P}(C)$, and $n_0(C)$ by $\hat{n}(C)$, and start again; if, after the second iteration the value of χ^2 is ''small enough'', stop the iteration; otherwise go to step 2. Some criteria about the optimum number of iterations will be discussed later.

UNFOLDING

- Da
  - G. D'Agostini, Nuclear Instruments and Methods in Physics Research A 362 (1995) 487

  - I passi preliminari (da fare una sola volta) che dipendono dalla nostra conoscenza della procedura sperimentale (quantità mai modificate dalla procedura iterativa) :
    - **Definizione di cause ed effetti**
      - **Ci (= EVera nel bin i-esimo)**
      - **Ej (=EMisurata nel bin j-esimo)**
      - $n_C \quad n_E$ => numero di cause, numero di effetti
    - **Calcolo delle probabilita' degli effetti condizionate alle cause per ogni causa ed effetto**
      - $P(E_j \mid C_i)$
    - **Calcolo delle efficienze (probabilita' che una causa produca uno qualunque degli effetti considerati) per ogni causa**
      - $\epsilon(C_i) = \sum_{j}^{n_E} P(E_j \mid C_i)$

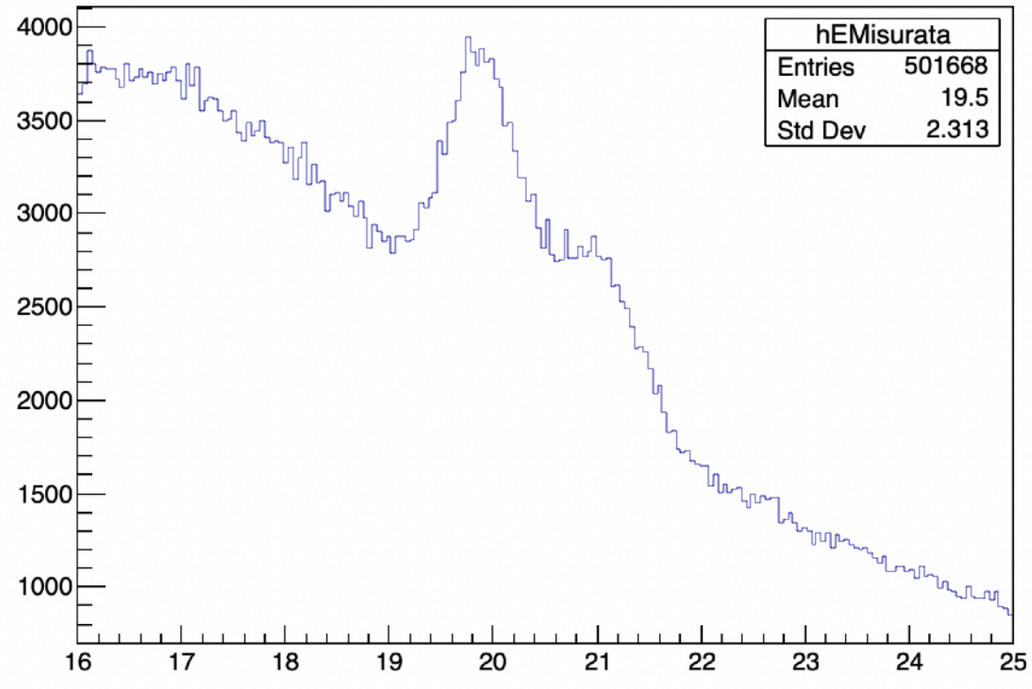
EVera di riferimento



hEVera->Draw()

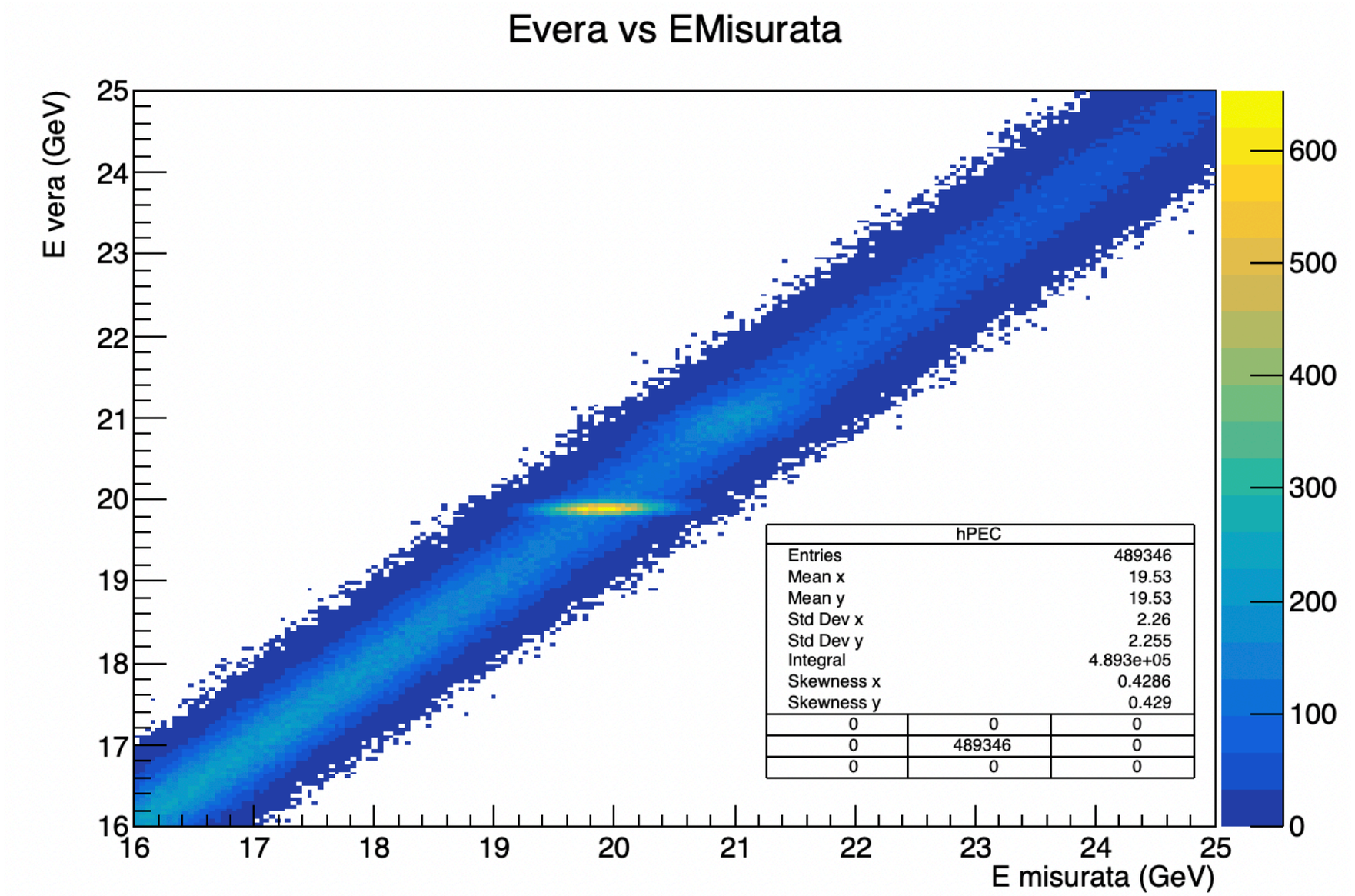
EVera in 200 bin
(da 1 a 200) da 16 a 25 GeV

EMisurata



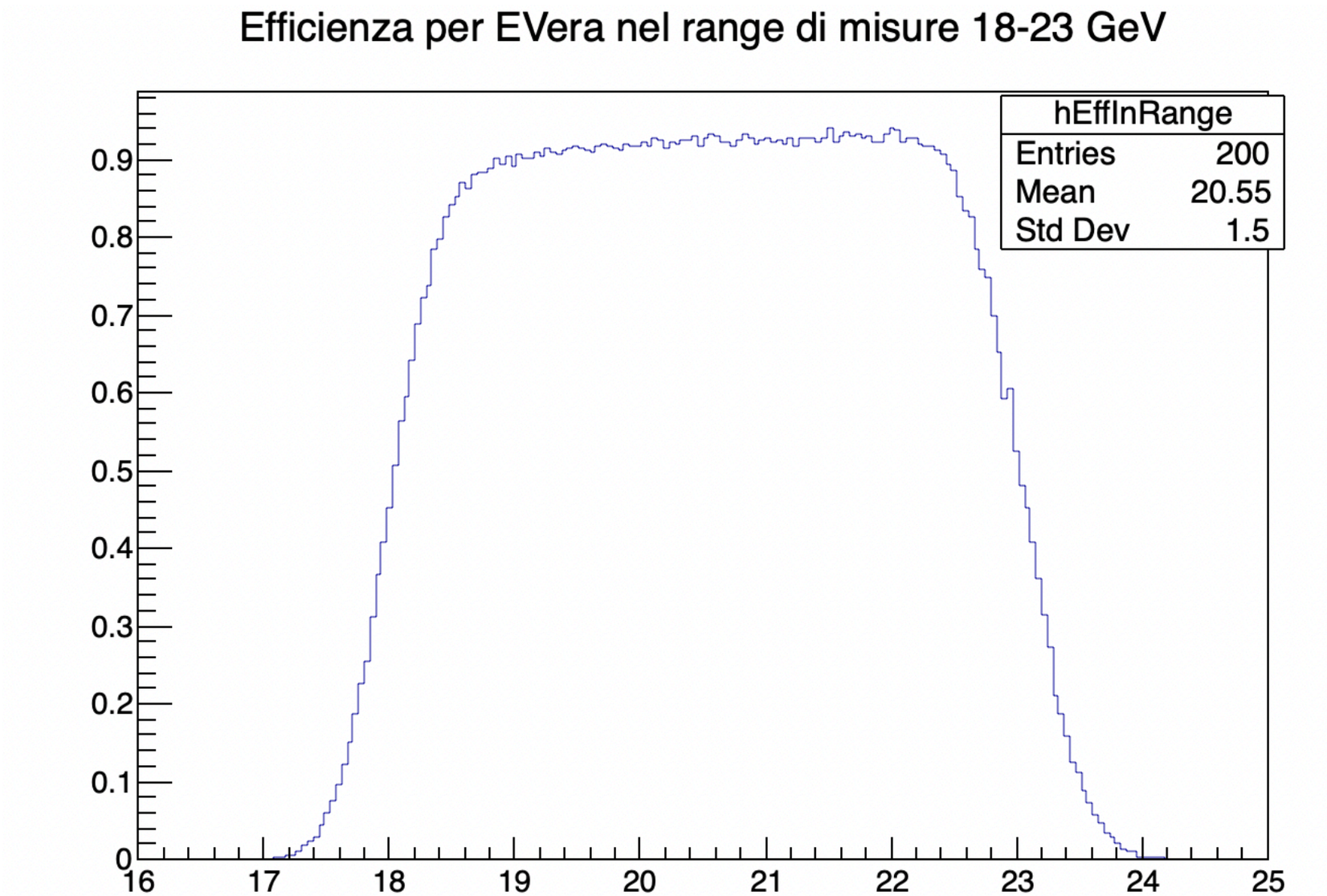
hEMisurata->Draw()

EMisurata in da 18 a 23 GeV
(Dal bin 45 al bin 157)



hPEC->Draw("zcol")

- Calcolo delle efficienze (probabilita' che una causa produca uno qualunque degli effetti considerati) per ogni causa
  - $\epsilon(C_i) = \sum_{j}^{n_E} P(E_j | C_i)$



5 E 1000 ITERAZIONI