



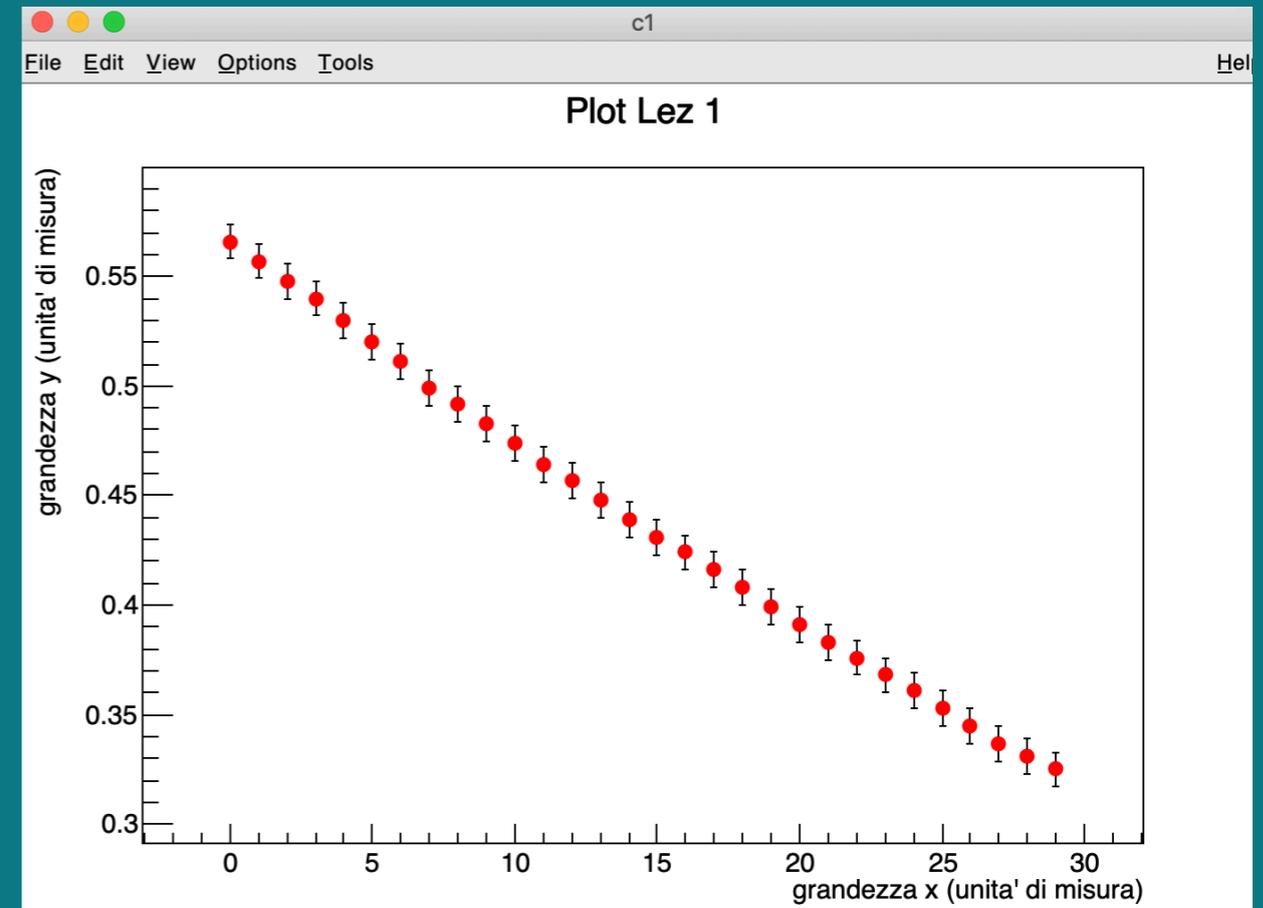
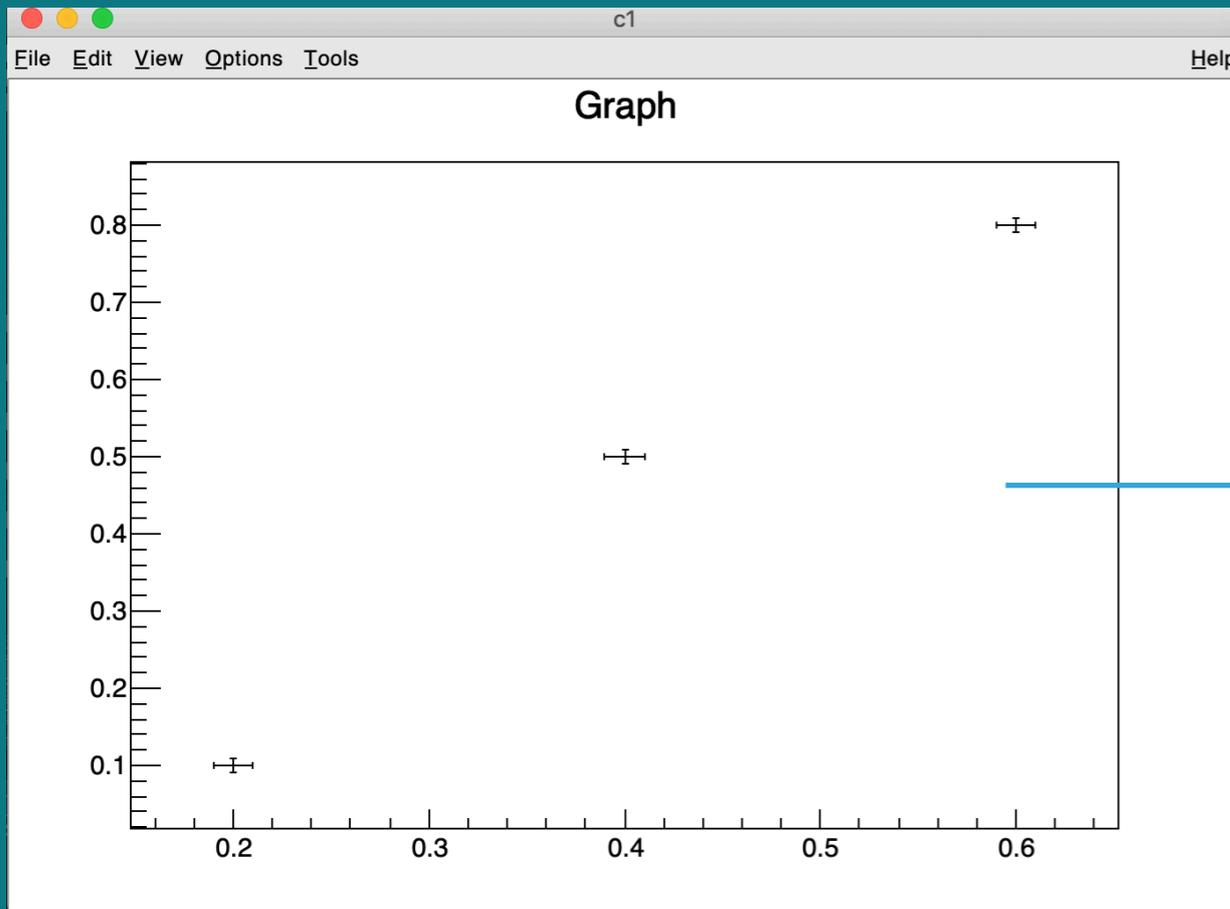
AISF - COMITATO LOCALE DI PERUGIA

GRAFICI PARTE 2

ISTOGRAMMI

- ▶ **Recap della lezione 1**
- ▶ **Root è un pacchetto software fornito dal Cern, contenente una serie di funzioni raggruppate in Classi (TGraph, TGraphErrors, TCanvas,.....)**
- ▶ **Ogni classe ha una serie di funzioni corrispondenti chiamate in termini informatici: Metodi della classe.**
- ▶ **se vogliamo ad esempio creare un Istogramma si dovrà creare un file (chiamato Oggetto) appartenente alla classe corrispondente. Nel nostro caso TH1**

► Rivediamo l'estetica del grafico con errori



- inserire titoli degli assi e del grafico principale
- “evidenziare” i punti sperimentali

► analizziamo i metodi

```
TGraphErrors gr (30,x,y,ex,ey);

//impostiamo il colore del marker 4=celeste la tavola complessiva si puo trovare sul sito cern
gr.SetMarkerColor(2); //uso il metodo setmarkerstyle della classe TGraphErrors
gr.SetMarkerStyle(20); //scelgo lo stile 21 del marker
gr.SetTitle("Plot Lez 1"); //titolo grafico

gr.GetAxis()->SetTitle("grandezza x (unita' di misura)");
gr.GetAxis()->SetTitle("grandezza y (unita' di misura)");

gr.Draw("AP"); //stampa asse punti

}
```

SetMarkerColor -> stabilisce i colori dei punti sperimentali

SetTitle(".....") titolo principale

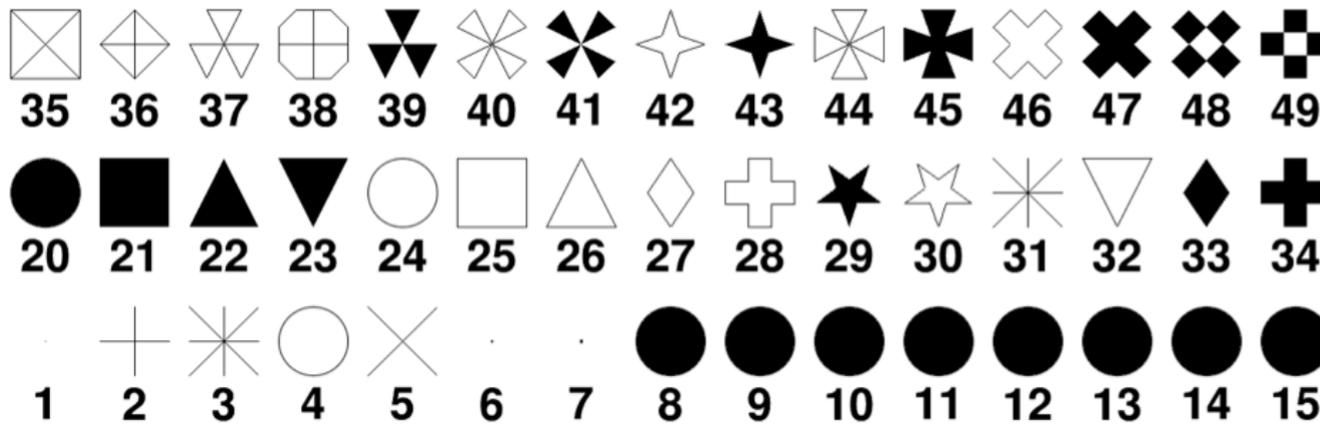
GetXaxis()->SetTitle("") titoli assi

Draw("AP") con il parametro AP stampa solo punti sperimentali e assi

parametri dei metodi stilistici



▶ **colori dati sperimentali**



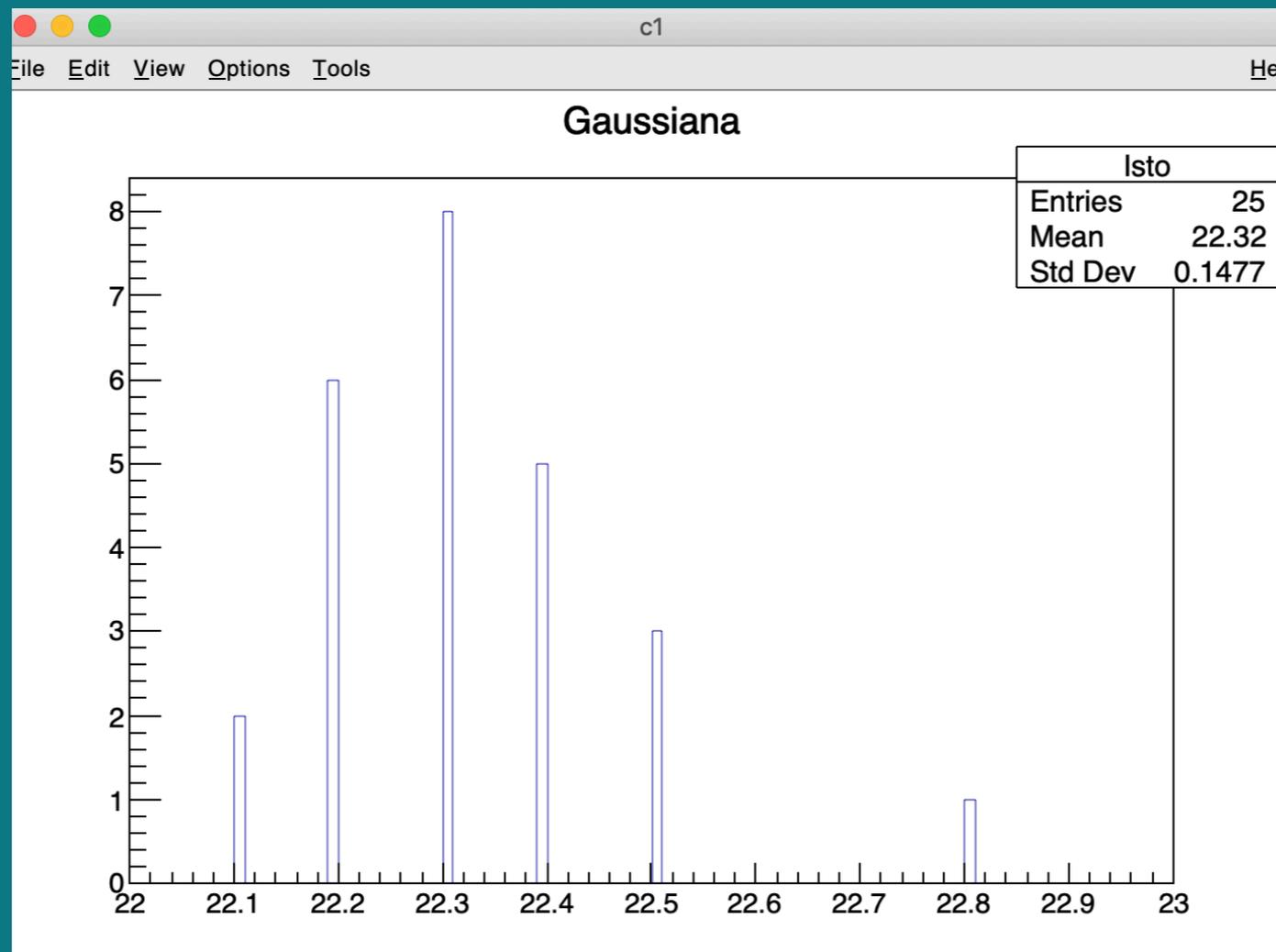
▶ **stile del marker**

► Istogrammi

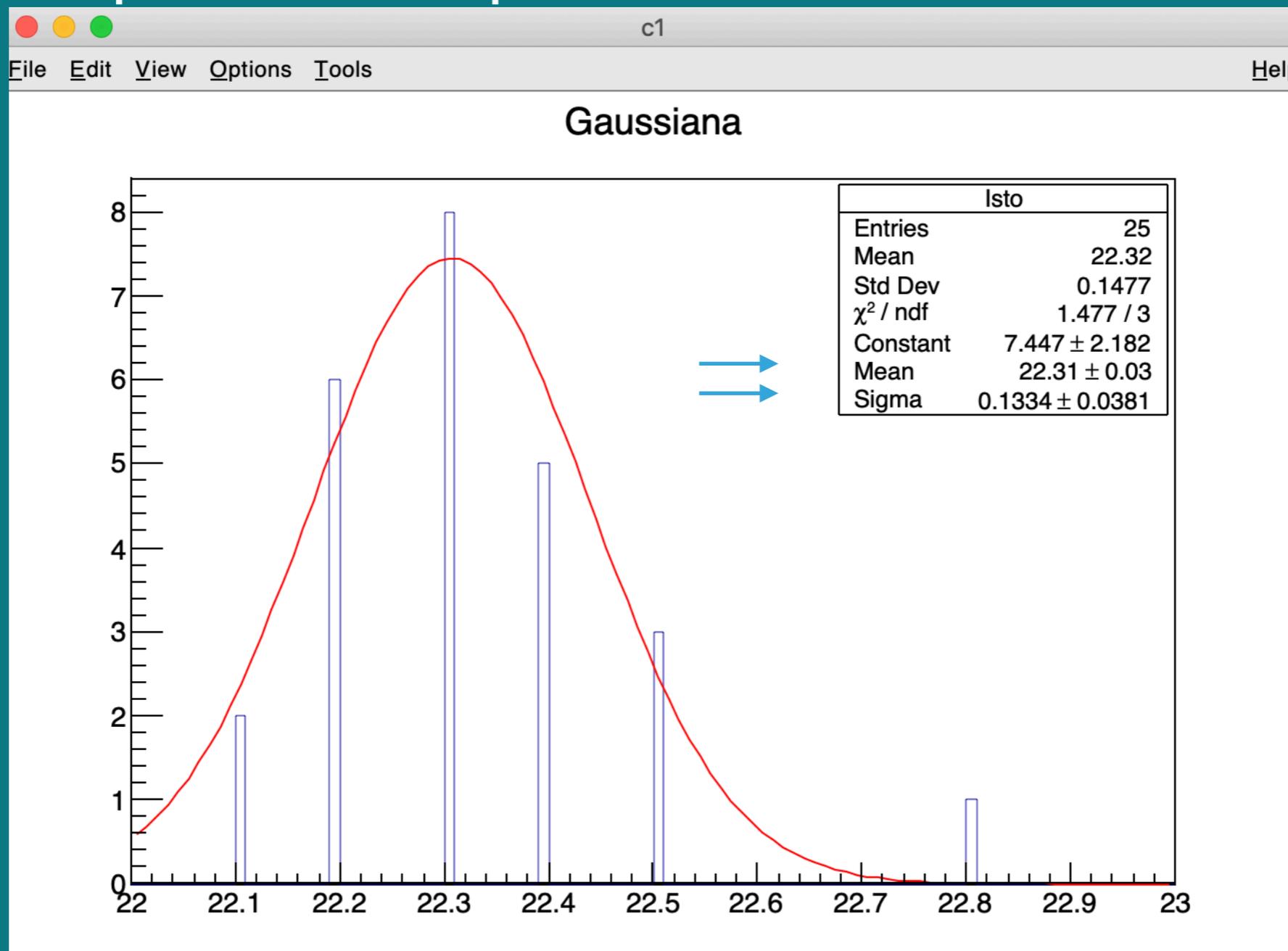
- se ripetiamo la misura di una grandezza x più volte e volessimo osservarne la distribuzione ricorriamo ad istogrammi
- es: valori misurati della grandezza x

```
{22.3, 22.5, 22.4, 22.8, 22.3, 22.3, 22.5, 22.1, 22.2, 22.4, 22.1,  
22.2, 22.5, 22.3, 22.3, 22.3, 22.2, 22.2, 22.3, 22.2, 22.4, 22.4, 22.2,  
22.3, 22.4}
```

- l'obiettivo è inserirli in un istogramma



- ▶ la distribuzione è "tipicamente" gaussiana. In genere si fornisce il valore della grandezza x come: $X(\text{medio}) \pm \text{stdev}$
- ▶ stdev = deviazione standard della Gaussiana
- ▶ Dall'istogramma si può eseguire un Fit Guassiano tramite il metodo "dinamico" della lezione 1, stavolta impostando *gaus* dal pannello FitFunction per ricavare i valori necessari



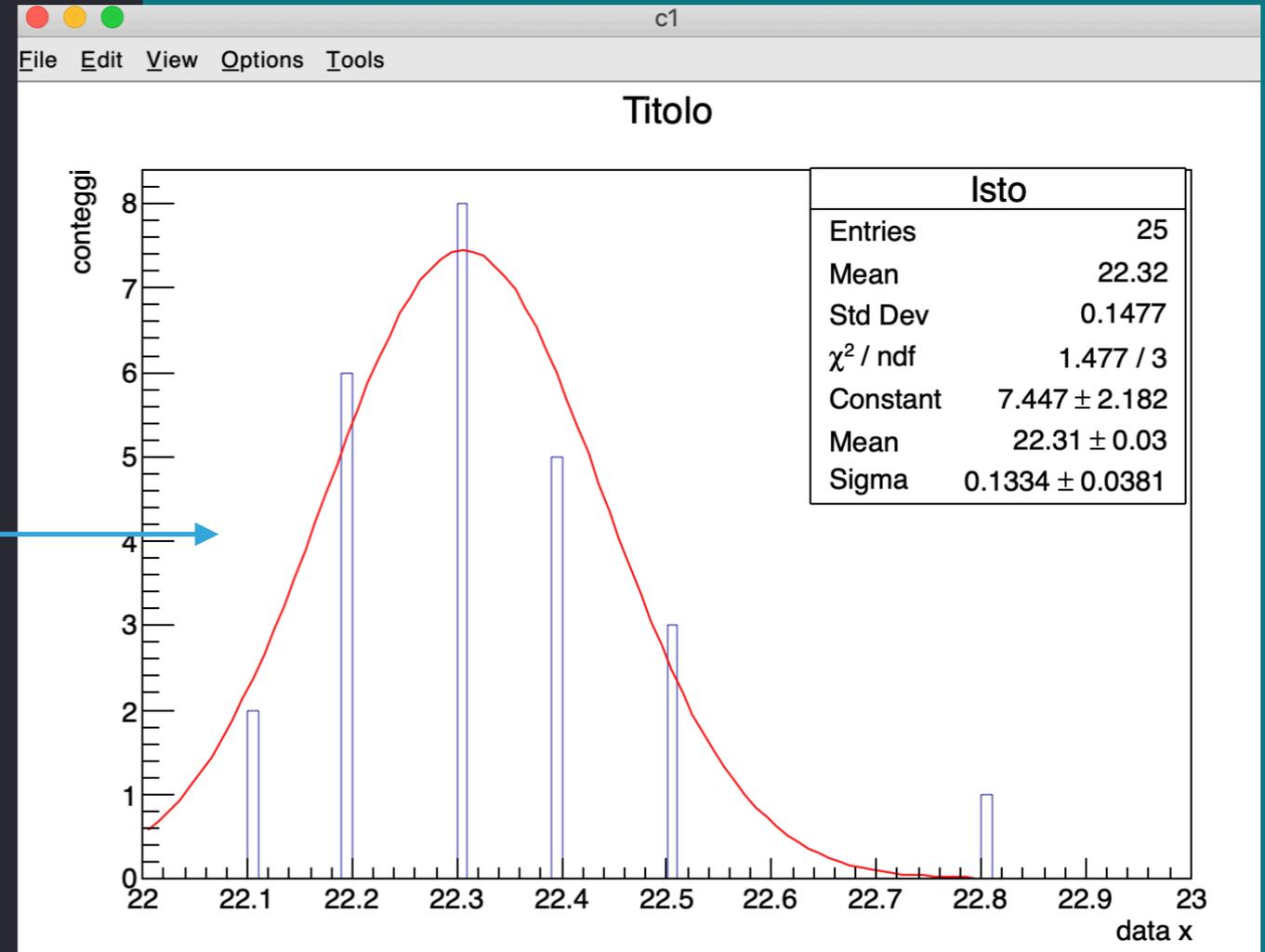
- ▶ risultato finale sarà : $x = \text{Mean} \pm \text{Sigma}$

- ▶ **usiamo la classe TH1F**
- ▶ **parametri: "titolo", numero di bin, range**

```
Isto.cpp > No Selection
1  {
2  TH1F hist5("Isto", "Gaussiana", 100, 22, 23);
3  //0.70-0.80 intervallo dell'asse x
4
5      hist5.Fill(22.3);
6      hist5.Fill(22.5);
7      hist5.Fill(22.4);
8      hist5.Fill(22.8);
9      hist5.Fill(22.3);
10     hist5.Fill(22.3);
11     hist5.Fill(22.5);
12     hist5.Fill(22.1);
13     hist5.Fill(22.2);
14     hist5.Fill(22.4);
15     hist5.Fill(22.1);
16     hist5.Fill(22.2);
17     hist5.Fill(22.5);
18     hist5.Fill(22.3);
19     hist5.Fill(22.3);
20     hist5.Fill(22.3);
21     hist5.Fill(22.2);
22     hist5.Fill(22.2);
23     hist5.Fill(22.3);
24     hist5.Fill(22.2);
25     hist5.Fill(22.4);
26     hist5.Fill(22.4);
27     hist5.Fill(22.2);
28     hist5.Fill(22.3);
29     hist5.Fill(22.4);
30
31
32     hist5.Draw();
33 }
34
35
```

▶ inseriamo i titoli

```
Isto.cpp
Isto.cpp > No Selection
1 {
2 TH1F hist5("Isto", "Titolo",100,22,23);
3 //0.70-0.80 intervallo dell asse x
4
5 hist5.Fill(22.3);
6 hist5.Fill(22.5);
7 hist5.Fill(22.4);
8 hist5.Fill(22.8);
9 hist5.Fill(22.3);
10 hist5.Fill(22.3);
11 hist5.Fill(22.5);
12 hist5.Fill(22.1);
13 hist5.Fill(22.2);
14 hist5.Fill(22.4);
15 hist5.Fill(22.1);
16 hist5.Fill(22.2);
17 hist5.Fill(22.5);
18 hist5.Fill(22.3);
19 hist5.Fill(22.3);
20 hist5.Fill(22.3);
21 hist5.Fill(22.2);
22 hist5.Fill(22.2);
23 hist5.Fill(22.3);
24 hist5.Fill(22.2);
25 hist5.Fill(22.4);
26 hist5.Fill(22.4);
27 hist5.Fill(22.2);
28 hist5.Fill(22.3);
29 hist5.Fill(22.4);
30
31
32 hist5.GetAxis()->SetTitle("data x");
33 hist5.GetAxis()->SetTitle("conteggi");
34
35
36 hist5.Draw();
37 }
38
39
```



- ▶ esiste una versione differente per Fill per velocizzare: **Fill(22.3,3)**
- ▶ **Fill(22.3,3)** inserisce 3 volte lo stesso valore

▶ “rebinning”

- ▶ vediamo una piccola variazione nella macros per inserire il numero di bin che vogliamo direttamente da terminale

```
{  
  
    double xmin;  
    double xmax;  
    int n;  
  
    cout<<"Inserisci l'estremo xmin "<<endl;  
    cin>>xmin;  
  
    cout<<"Inserisci l'estremo xmax "<<endl;  
    cin>>xmax;  
  
    cout<<"BINNING-->Inserisci numero di bin "<<endl;  
    cin>>n;  
  
    TH1F hist5("Parameters", "h", n, xmin, xmax);  
    //0.70-0.80 intervallo dell'asse x  
    hist5.SetTitle("Titolo istogramma");  
  
    //Torna a TH1F (se non è già un TH1F)  
    TH1F *h1f = dynamic_cast<TH1F*>(hist5);  
    if (h1f) h1f->SetType("HISTOGRAM");  
}
```

- ▶ ****NB** il risultato del Fit dipende dal numero (ossia dalla dimensione) dei bin. La scelta del numero di Bin è personale

- ▶ parametri di TH1F sono “dinamici” -> inserimento da terminale
- ▶ “Parameters” titolo dei parametri fittati
- ▶ “h” -> nome subcanvas parametri

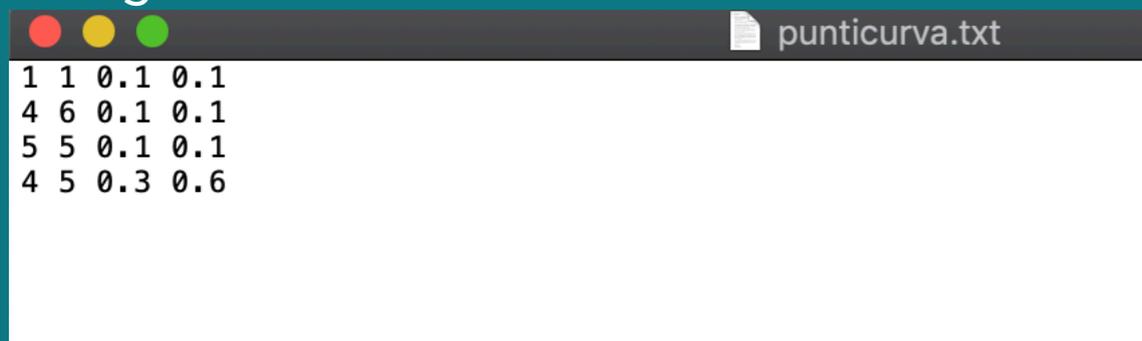
► inserimento dati per TGraphErrors direttamente da file.txt

Notiamo la nuova sintassi (*g = new) per dichiarare un oggetto. Da ricordare solo il fatto che così facendo si definisce un oggetto dinamico. Il risultato è un'efficienza maggiore per il codice ma "dal punti di vista pratico" non cambia nulla. **Differenza sintattica si usa -> e non "." per chiamare i metodi.

```
{
  TString nomefile = "/Users/David/Desktop/punticurva.txt"; //percorso
  TGraphErrors *g = new TGraphErrors(nomefile);
  g->GetXaxis()->SetTitle("x");
  g->GetYaxis()->SetTitle("f(x)");
  g->GetXaxis()->CenterTitle();
  g->GetYaxis()->CenterTitle();

  g->SetMarkerColor(4); //Markers...
  g->SetMarkerStyle(20);
  g->SetTitle("titolo");
  g->Draw("ap");
}
```

- si può passare direttamente il "path" del file(txt) contenente i dati
- il grafico viene creato esattamente allo stesso modo delle macros precedenti



```
1 1 0.1 0.1
4 6 0.1 0.1
5 5 0.1 0.1
4 5 0.3 0.6
```

scrivo i punti in 4 colonne-> nell'ordine(x ,y ,errx ,erry)

- ▶ esecuzione macros dalla cartella specifica in cui dovrete salvare tutte le macros
- ▶ per lanciare la macros:
- ▶ salvare nella cartella giusta in cui Root ricerca
- ▶ una volta aperto Root: { `.x NomeMacros.cpp` }
- ▶ nella prossima lezione parleremo di:
- ▶ importazione dati per velocizzare l'inserimento(istogrammi)
- ▶ sovrapposizione grafici (+ fit in una sola canvas)