



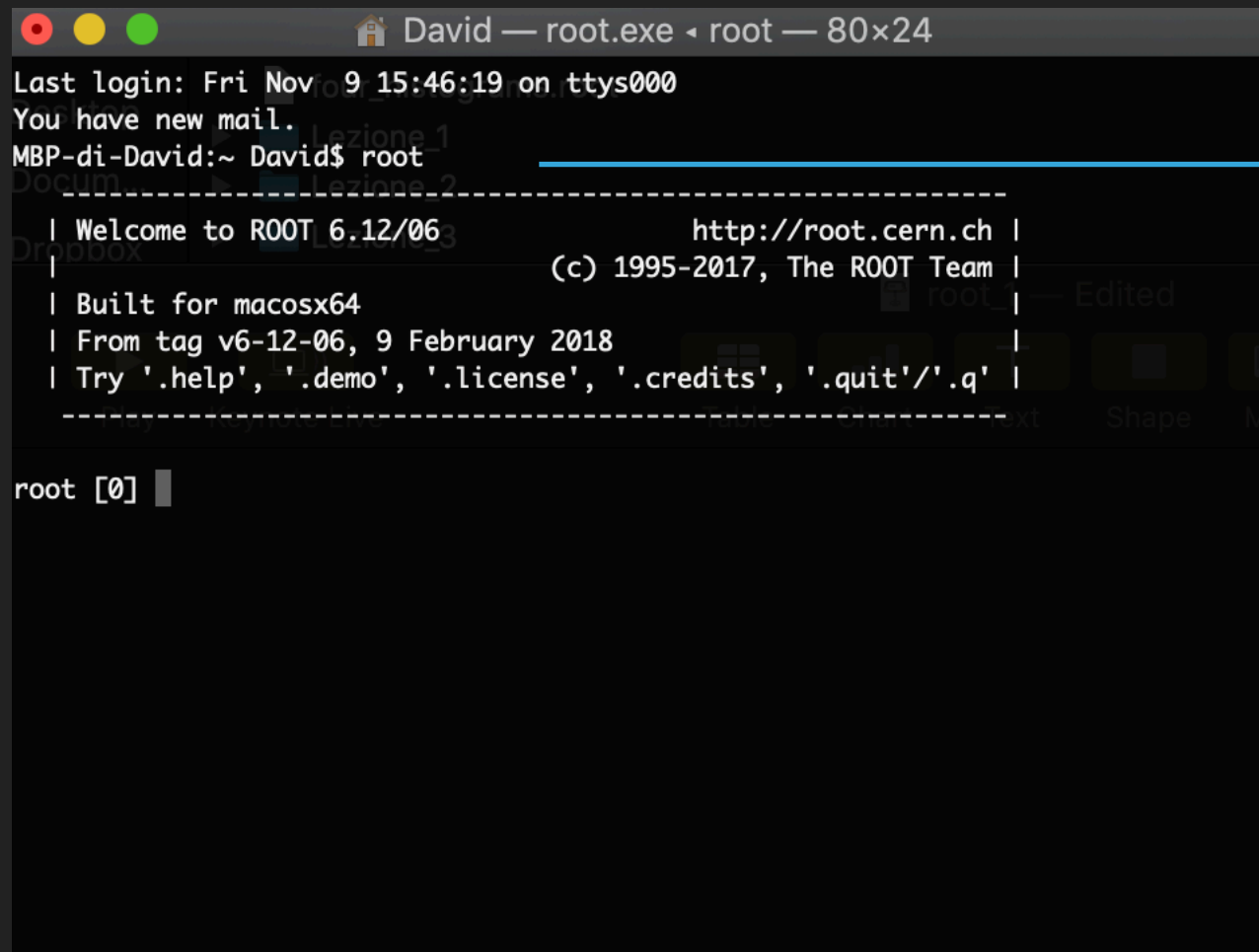
AISF - COMITATO LOCALE DI PERUGIA

CORSO ROOT

INTRO COMANDI BASE ROOT

- ▶ 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 grafico con Errori si dovrà creare un file (chiamato Oggetto) appartenente alla classe corrispondente.

- ▶ Schermata principale di Root, una volta installato correttamente si può accedere a root dal terminale eseguendo il comando: "root".



```
David — root.exe - root — 80x24
Last login: Fri Nov 9 15:46:19 on ttys000
You have new mail.
MBP-di-David:~ David$ root
-----
| Welcome to ROOT 6.12/06                      http://root.cern.ch |
| (c) 1995-2017, The ROOT Team                 |
| Built for macosx64                          |
| From tag v6-12-06, 9 February 2018          |
| Try '.help', '.demo', '.license', '.credits', '.quit'/'.' |
|-----|
root [0]
```

comando di
apertura
software

- ▶ vediamo come creare un oggetto grafico per visualizzare la funzione " $\sin(x)/x$ "

► analizziamo i comandi

```
David — root.exe ◀ root — 80x24

Last login: Fri Nov  9 15:46:19 on ttys000
You have new mail.
[MBP-di-David:~ David$ root]

-----
| Welcome to ROOT 6.12/06                               http://root.cern.ch |
|                                                         (c) 1995-2017, The ROOT Team |
| Built for macosx64                                     |
| From tag v6-12-06, 9 February 2018                     |
| Try '.help', '.demo', '.license', '.credits', '.quit'/'.' |
|                                                         |
-----

[root [0] TF1 f1("f1", "sin(x)/x",0,10)
(TF1 &) Name: f1 Title: sin(x)/x
[root [1] f1.Draw()
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [2] ]
```

TF1:nome classe
dell'oggetto

f1:nome
oggetto

parametri
(.....)

i parametri dipendono dal tipo di classe scelta, per la classe TF1 i parametri sono:

"f1" = nome oggetto , "sin(x)/x" = espressione analitica che vogliamo graficare

0,10 = range i cui visualizzare il grafico

► premendo invio creiamo il grafico

```
David — root.exe ◀ root — 80x24

Last login: Fri Nov 9 15:46:19 on ttys000
You have new mail.
[MBP-di-David:~ David$ root

-----
| Welcome to ROOT 6.12/06                               http://root.cern.ch |
|                                                         (c) 1995-2017, The ROOT Team |
| Built for macosx64                                     |
| From tag v6-12-06, 9 February 2018                     |
| Try '.help', '.demo', '.license', '.credits', '.quit'/'.' |
|                                                         |
-----

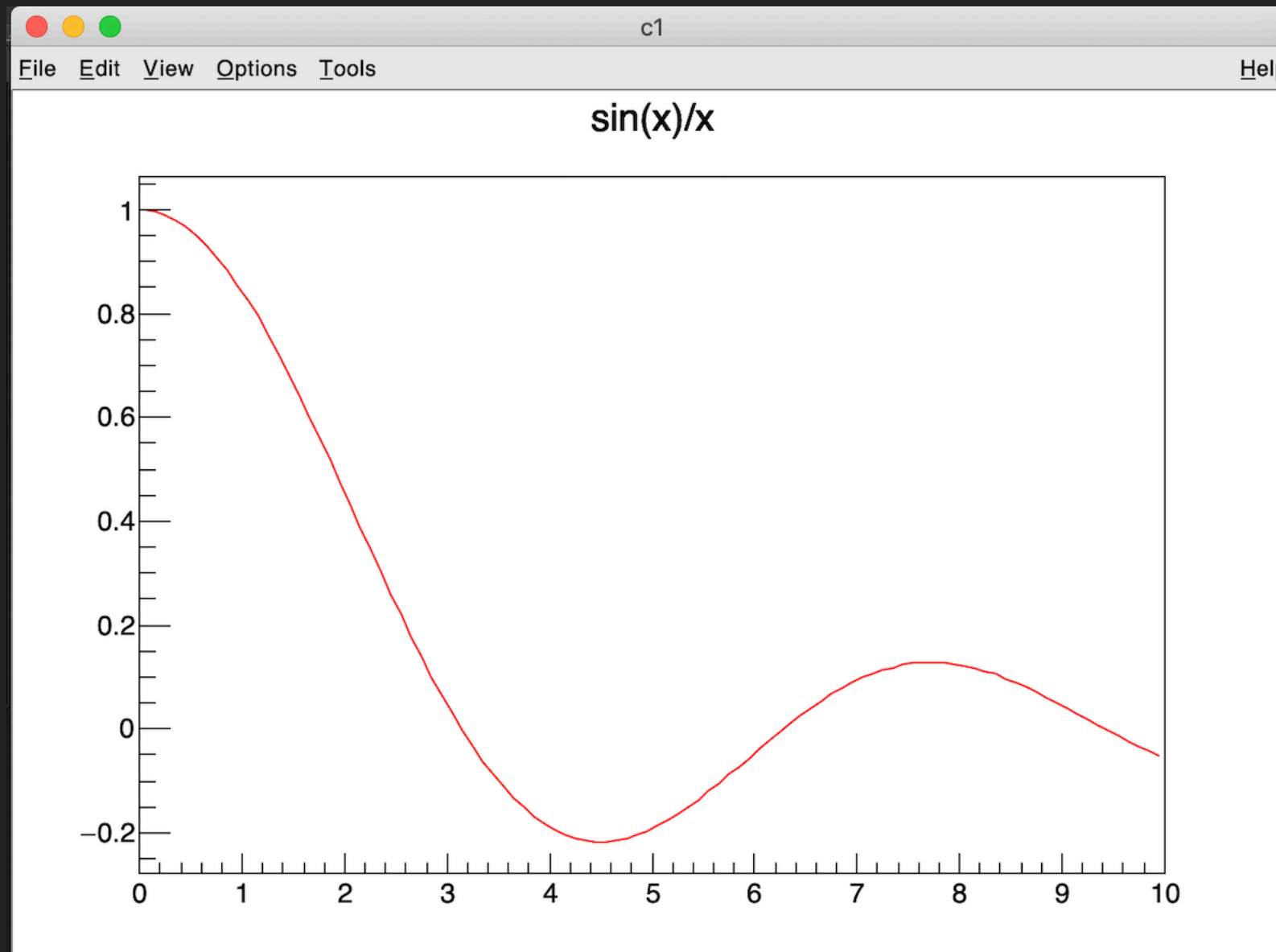
[root [0] TF1 f1("f1", "sin(x)/x",0,10)
(TF1 &) Name: f1 Title: sin(x)/x
[root [1] f1.Draw()
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [2] ]
```

premando invio, creiamo correttamente il grafico

visualizzo il grafico

Per visualizzare il grafico devo usare il metodo Draw della classe TF1

- ▶ con il comando Draw() questo è ciò che appare, ossia la canvas contenente il grafico.



- ▶ con lo stesso ragionamento possiamo creare anche grafici più interessanti come grafici con punti sperimentali(con relativi errori).
- ▶ useremo un classe diversa TGraphErrors
- ▶ prima dobbiamo creare gli array contenenti i punti sperimentali ed errori
- ▶ in particolare 4 array: (dati x , dati y , errori su x , errori su y)

creazione singola
dei vari
array(double) di
dimensione 3

```
David — root.exe - root — 80x24
MBP-di-David:~ David$
MBP-di-David:~ David$
MBP-di-David:~ David$
MBP-di-David:~ David$
MBP-di-David:~ David$
MBP-di-David:~ David$
MBP-di-David:~ David$ root
-----
| Welcome to ROOT 6.12/06                               http://root.cern.ch |
|                                                         (c) 1995-2017, The ROOT Team |
| Built for macosx64                                     |
| From tag v6-12-06, 9 February 2018                     |
| Try '.help', '.demo', '.license', '.credits', '.quit'/'.' |
-----

root [0] double x[3] = {0.2 , 0.4,0.6}
(double [3]) { 0.20000000, 0.40000000, 0.60000000 }
root [1] double y[3] = {0.1 , 0.5,0.8}
(double [3]) { 0.10000000, 0.50000000, 0.80000000 }
root [2] double erry[3] = {0.01 , 0.01,0.01}
(double [3]) { 0.01000000, 0.01000000, 0.01000000 }
root [3] double errx[3] = {0.01 , 0.01,0.01}
(double [3]) { 0.01000000, 0.01000000, 0.01000000 }
root [4]
```

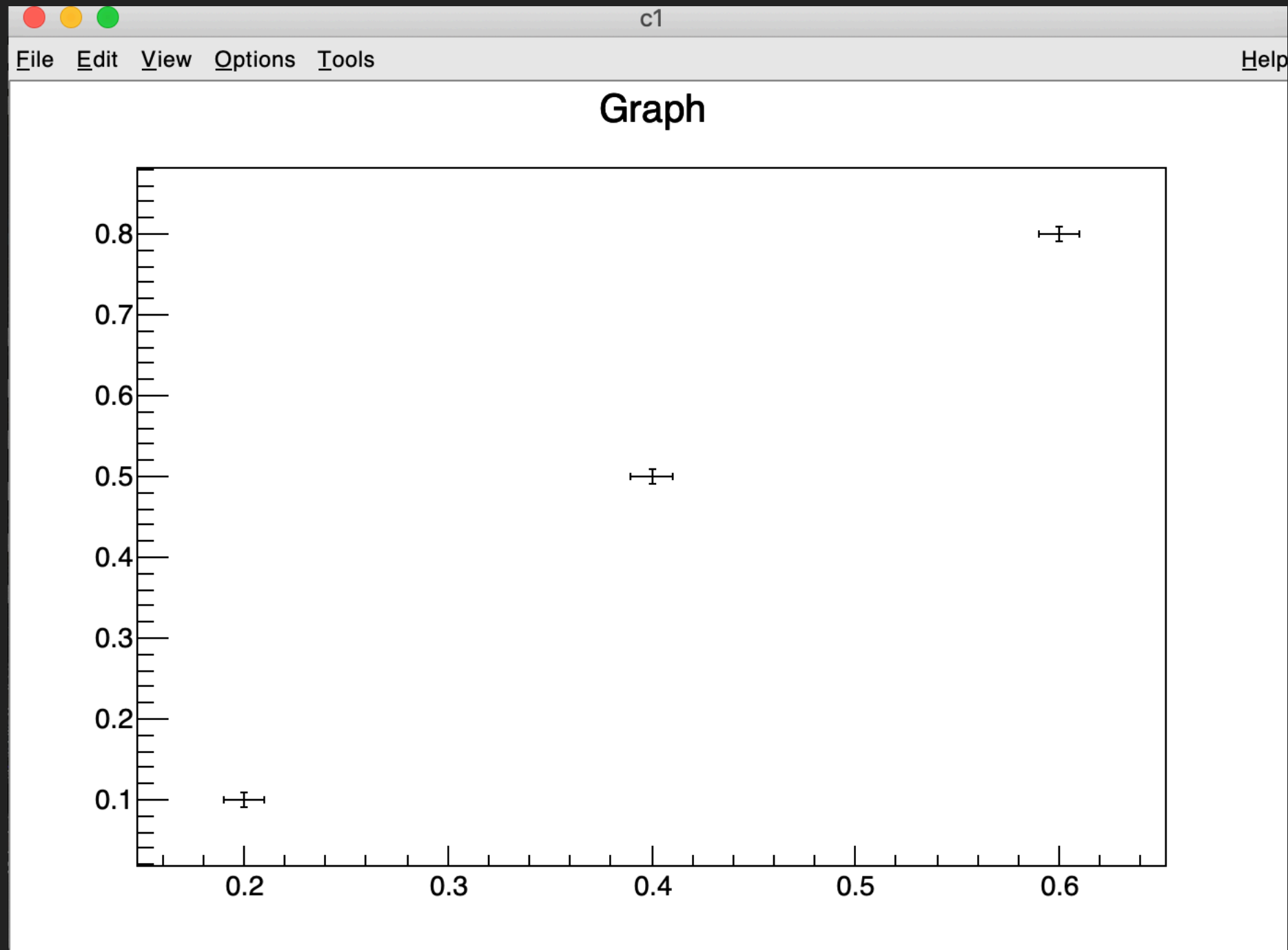

- ▶ chiamiamo la funzione TGraphErrors, creando l'oggetto g
- ▶ vediamo i nuovi parametri: (3,x,y,errx,erry)
- ▶ 3 = lunghezza degli array
- ▶ (x,y,errx,erry) rispettivamente gli array contenente dati e errori

```
David — root.exe - root — 80x24
[MBP-di-David:~ David$
[MBP-di-David:~ David$
[MBP-di-David:~ David$ root
-----
| Welcome to ROOT 6.12/06                      http://root.cern.ch |
|                                           (c) 1995-2017, The ROOT Team |
| Built for macosx64                      |
| From tag v6-12-06, 9 February 2018      |
| Try '.help', '.demo', '.license',       |
| '.credits', '.quit'/'.'q'              |
-----

[root [0] double x[3] = {0.2 , 0.4,0.6}
(double [3]) { 0.20000000, 0.40000000, 0.60000000 }
[root [1] double y[3] = {0.1 , 0.5,0.8}
(double [3]) { 0.10000000, 0.50000000, 0.80000000 }
[root [2] double erry[3] = {0.01 , 0.01,0.01}
(double [3]) { 0.01000000, 0.01000000, 0.01000000 }
[root [3] double errx[3] = {0.01 , 0.01,0.01}
(double [3]) { 0.01000000, 0.01000000, 0.01000000 }
[root [4] TGraphErrors g(3,x,y,errx,erry)
(TGraphErrors &) Name: Graph Title: Graph
[root [5] g.Draw("AP")
Info in <TCanvas::MakeDefCanvas>: created default TCanvas with name c1
root [6] 
```

con il metodo g.Draw() visualizzo il grafico

► questo è ciò che si visualizza



► i 3 punti sperimentali con i relativi errori su asse x e asse y

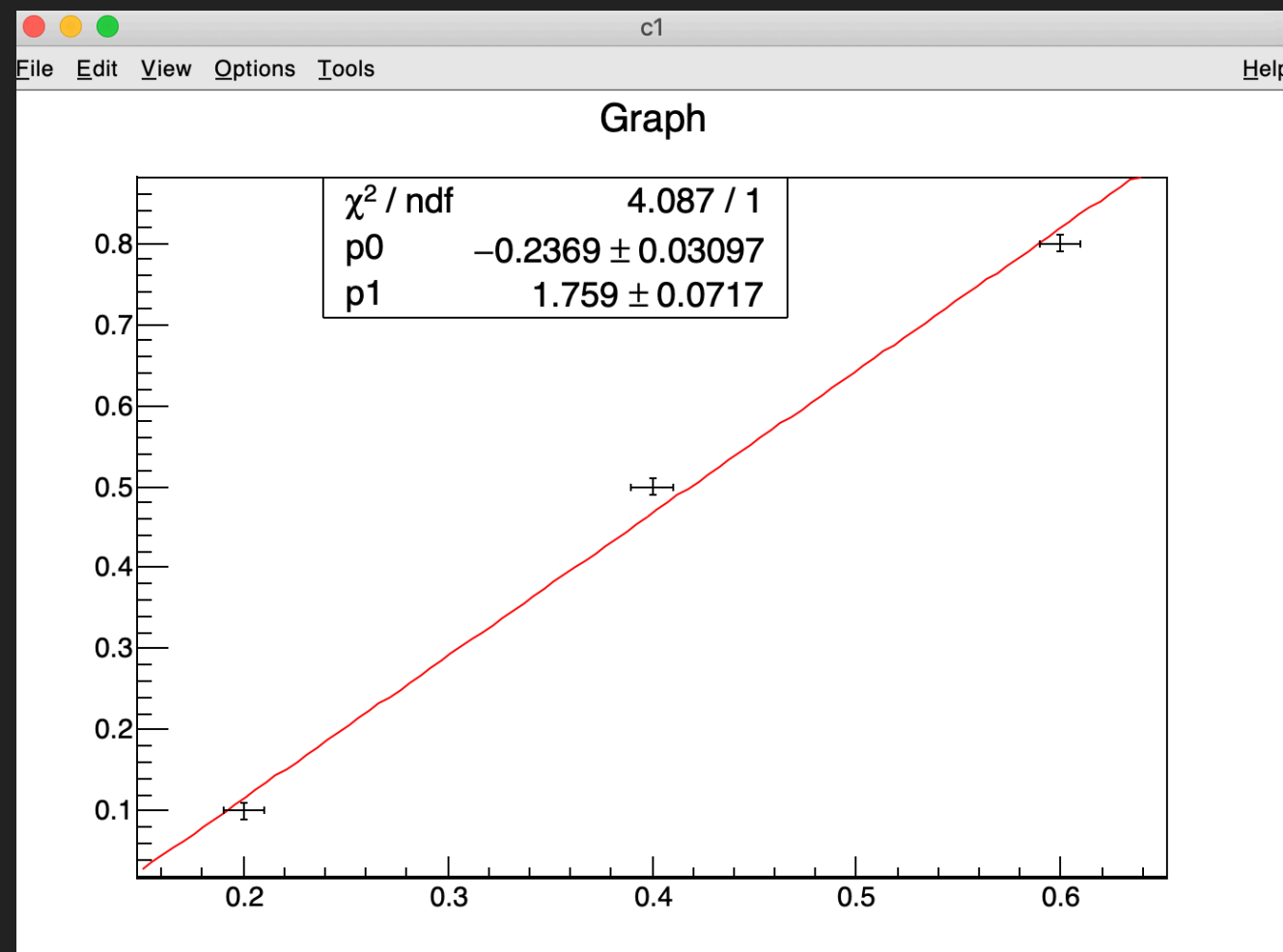
se volessi fare un fit lineare su questi dati per cercare la migliore retta che approssima i dati

dalla schermata di root (Canvas) che si apre, andare su

Tools -> fit panel -> impostare pol1 (polinomiale grado 1 ossia retta) dal pannello fitfunction

per visualizzare i risultati del Fit Lineare sul grafico : option -> Fit parameter

La prossima volta vedremo alcuni comandi per migliorare l' "estetica " del grafico.....



**TUTTI I COMANDI VISTI PER GENERARE IL GRAFICO:
CREATION ARRAY E CHIAMATA DELL'OGGETTO TGRAPHERRORS, POSSONO ESSERE IN REALTÀ ACCORPATI IN UN SOLO COMANDO CREANDO UNA MACROS (OSSIA UNA COLLEZIONE DI COMANDI ROOT CHE VENGONO ESEGUITI IN SINGOLA SESSIONE)**

in allegato trovate un esempio di macros reale(riferita ad una vera esperienza di Lab1) sulla creazione di un grafico con dati sperimentali e relativi errori.

Salvando la macros in un path specifico, che capirete una volta installato Root,

la macros può essere eseguita una volta aperto Root attraverso il comando:

.x Nomemacros

es: Bern003.cpp (la macros allegata)

.x Bern003.cpp