How I improved Quantum Gravity Previewer

that is, how easily you can help science

Dominik Ostrogórski

AGH University of Science and Technology

CREDO School, 2018

Introduction

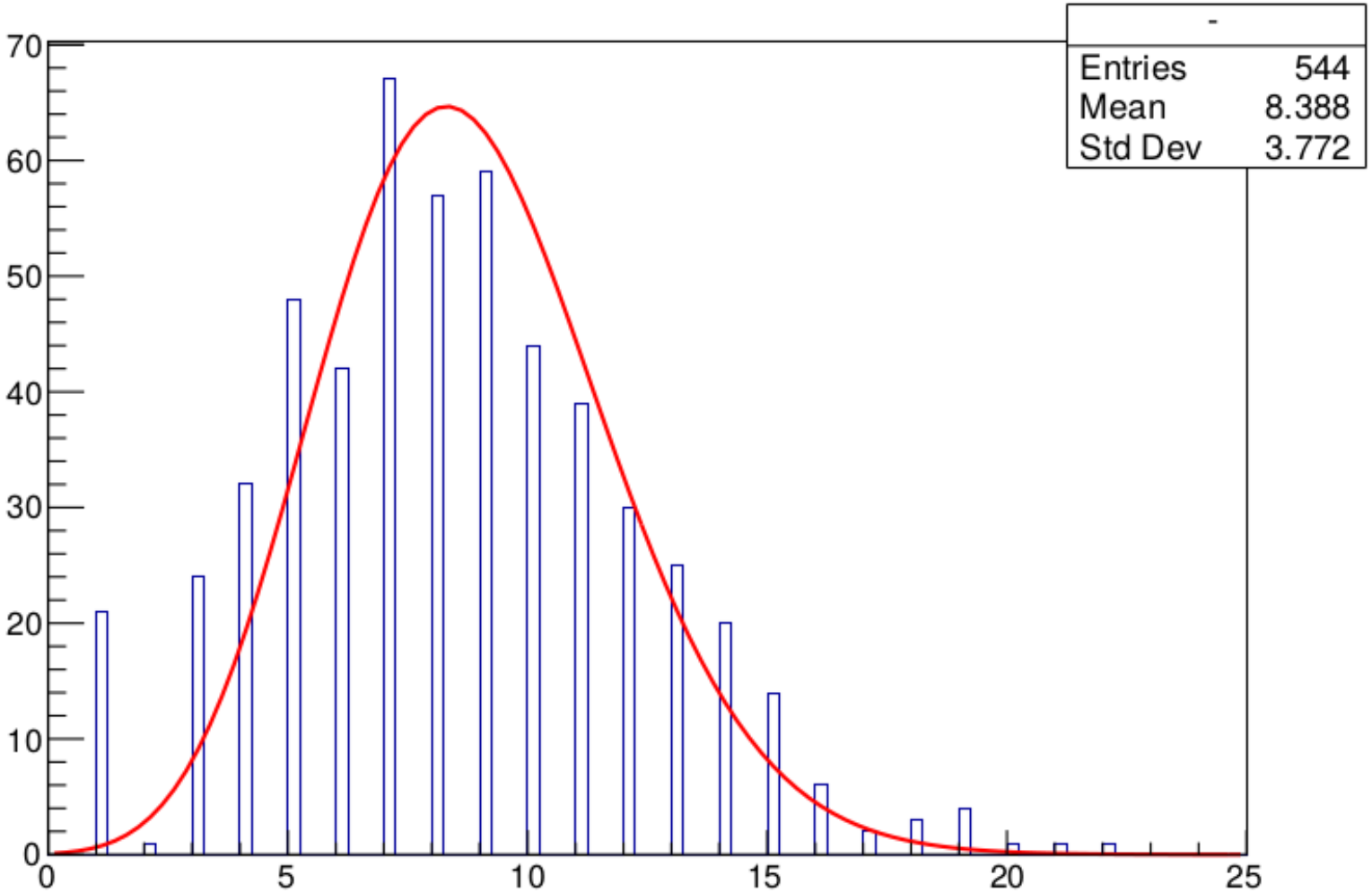
- AGH, Faculty of Physics and Applied Computer Science
- Institute of Nuclear Physics
- CREDO

Getting started

```
for(int i=0;i<tab.size;i++)
        h1->Fill(tab[i]);


cout<<endl;
TCanvas *c1 = new TCanvas();
h1->Draw();

TF1 *pois = new TF1("pois","[0]*TMath::Poisson(x,[1])",0,h_num*25);
pois->SetParameter(0,h1->GetMean(1));
pois->SetParameter(1,h1->GetRMS(1));
h1-> Fit(pois);
c1->Print("example.pdf");
```


-
Entries 544
Mean 8.388
Std Dev 3.772
70
60
50
40
30
20
10
0
0
5
10
15
20
25

https://github.com/credo-science/CREDO-monitor-TimeClusteringAlgo/

Simple solutions - the best benefits

```
//calculate time difference between consecutive events, given the number of consecutive events one wishes to look at (mut
double timeDiff_funct(const int numOfEvents, double timeStamp[], int firstEvent, int multiplet)
{
        double deltaT;

    if (multiplet==2)
    {
        deltaT = timeStamp[firstEvent+1] - timeStamp[firstEvent];
    }

    else if (multiplet==3)
    {
        deltaT = 2*(timeStamp[firstEvent+2] - timeStamp[firstEvent]);
    }

    else if (multiplet==4)
    {
        deltaT = 3*(timeStamp[firstEvent+3] - timeStamp[firstEvent])+(timeStamp[firstEvent+2]-timeStamp[firstEvent+1]);
    }

    else if (multiplet==5)
    {
        deltaT = 4*(timeStamp[firstEvent+4] - timeStamp[firstEvent])+2*(timeStamp[firstEvent+3] - timeStamp[firstEvent+1]
    }

    else if (multiplet==6)
    {
        deltaT = 5*(timeStamp[firstEvent+5] - timeStamp[firstEvent])+3*(timeStamp[firstEvent+4] - timeStamp[firstEvent+1]
    }

    else if (multiplet==7)
    {
        deltaT = 6*(timeStamp[firstEvent+6] - timeStamp[firstEvent])+4*(timeStamp[firstEvent+5] - timeStamp[firstEvent+1]
    }
```

```
//calculate time difference between consecutive events, given the number of consecutive events one wishes to look at (mutiplet)
double timeDiff_funct(const int numOfEvents, double timeStamp[], int firstEvent, int multiplet)
{
        int k = 0;
    double deltaT = 0;

    while ((2*k+1) < multiplet)
    {
        deltaT += (multiplet - (2*k+1))*(timeStamp[firstEvent + (multiplet - k - 1)] - timeStamp[firstEvent + k]);
        k++;
    }
        return deltaT;
}
```

Changing the sorting method – operating time 100 times shorter

```
//sort array in increasing values
void sort_array(int numOfEvents, double array[])
{
    for(int j = 0; j < numOfEvents - 1; j++)
    {
        double currentMin = array[j];
        int currentMinIndex = j;

        for(int k = j+1; k < numOfEvents; k++)
        {
            if(currentMin > array[k])
            {
                currentMin = array[k];
                currentMinIndex = k;
            }
        }

        if(currentMinIndex != j)
        {
            array[currentMinIndex] = array[j];
            array[j] = currentMin;
        }
    }
}
```

```
int partition(double tablica[], int p, int r) // divide the tables into two parts, in the first all numbers are
{
    double x = tablica[p],w; // we choose x
    int i = p, j = r; // i, j - indexes in the array
    while (true) // infinite loop - leave it only by return j
        {
        while (tablica[j] > x)
            j--;
        while (tablica[i] < x)
            i++;
        if (i < j) // we change places when i <j
        {
            w = tablica[i];
            tablica[i] = tablica[j];
            tablica[j] = w;
            i++;
            j--;
        }
        else // when i> = j we return j as the partition point of the array
            return j;
        }
}

void quicksort(double tablica[], int p, int r) // qsort
{
    int q;
    if (p < r)
    {
        q = partition(tablica,p,r); // we divide the arrays into two parts; q is the partition point
        quicksort(tablica, p, q); // we recursively call the quicksort for the first part of the array
        quicksort(tablica, q+1, r); // we recursively call the quicksort for the second part of the array
    }
}
//////////////////////////////////////////////////
```

Summary

- The possibility of **help** is available to everyone

- The possibility of **help** is available to everyone
- **Help** is simple

- The possibility of **help** is available to everyone
- **Help** is simple
- This **help** contributes to a better understanding of the universe