SMoment 1.0 Analysis Toolkit Manual

Prithwish Tribedy

BROOKHAVEN NATIONAL LABORATORY

Download link : https://github.com/ptribedy/SMoment

For any issues write to : prithwish2005@gmail.com, ptribedy@bnl.gov

SMoment 1.0 : Objectives

Major challenges in multiplicity fluctuation analysis are :

- Methods of efficiency correction becomes complicated
- Algebra of higher moments is cumbersome
- Error estimation in analytic approaches is very involved

Goal of `SMoment` is to **standardize** and **automate** such operations

Different higher moment observables

Higher moments :

The popular ones

Mean (μ_1)= κ_1

Variance (σ^2) = κ_2

Skewness (S) = $\dfrac{\kappa_3}{\kappa_2^{3/2}}$

Kurtosis (k) = $\dfrac{\kappa_4}{\kappa_2^2}$

Cross correlations :

$$C_{XY\cdots}^{i,j,\cdots} = \left\langle (X - \langle X \rangle)^i \, (Y - \langle Y \rangle)^j \cdots \right\rangle$$

Koch *et al* nucl-th/0505052

Ratio fluctuations :

$$\nu_{dyn}^{i} = \sum_r (-1)^r \binom{i}{r} \frac{f_{ir}}{f_{10}^{i}\, f_{01}^{r}}$$

Christiansen *et al* 0902.4788

Sangaline 1505.00261

Strongly intensive cumulants:

$$E_{i,0} = \frac{1}{\mu_{0,1}} \left(\mu_{i,0} - \sum_{r=0}^{i-1} \binom{i-1}{r-1} \mu_{i-r,1} E_r \right)$$

Ladrem *et al* 1509.00954

......And many more

You can design & implement your own

The fundamental ones

N —> The quantity measured in every event

Any complicated observable can be expressed in terms of :

Moments

$$\mu_i = \langle N^i \rangle \qquad \mu_{i,j,\cdots} = \langle N_1^i N_2^j \cdots \rangle$$

Central Moments

$$C_i = \langle (N_1 - \langle N_1 \rangle)^i \rangle \qquad C_{i,j,\cdots} = \langle (N_1 - \langle N_1 \rangle)^i (N_2 - \langle N_2 \rangle)^j \cdots \rangle$$

Cumulants

$$\kappa_i = C_i - \mathcal{F}(C_{i-1}, C_{i-2}, \cdots)$$

$\mathcal{F}$—> function of lower cumulants

Factorial moments

$$f_i = \langle N(N-1)\cdots(N-i+1) \rangle \qquad f_{i,j,\cdots} = \left\langle \frac{N_1!}{(N_1-i)!} \frac{N_2!}{(N_2-j)!} \cdots \right\rangle$$

Introduction to SMoment 1.0

Iterative algorithms that allow for a number of observables :

- Moment conversion up to **any order**
- Efficiency correction for **any number of bins**
- Error estimation using **delta theorem**
- **Includes** Binwidth other analysis related corrections

Two headers :

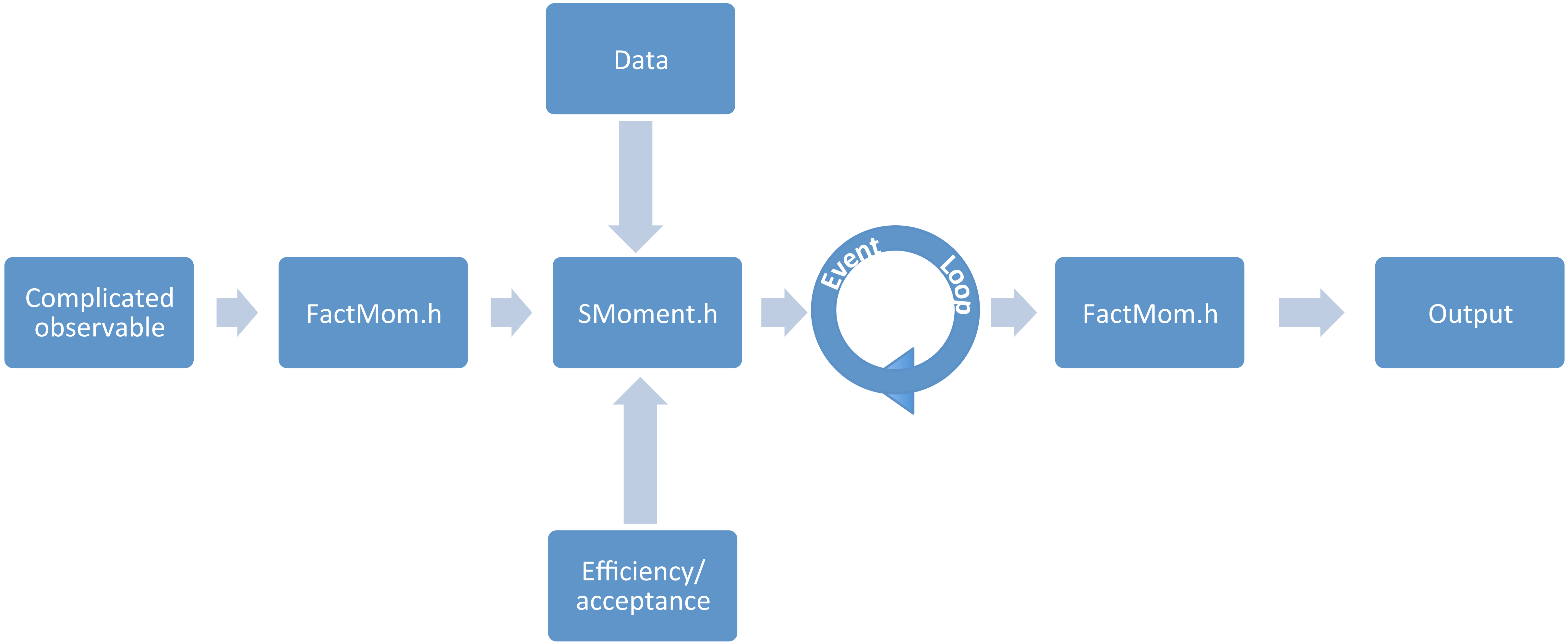
I. FactMom.h : contains different classes that performs algebra

II. Smoment.h : contains single class that performs analysis

SMoment can be downloaded from : https://github.com/ptribedy/SMoment

Let me know : prithwish2005@gmail.com for any issues

SMoment : Flowchart



SMoment : Classes

For algebra

- `CentVecN` : → Central Moments (N-order, N-variate)
- `MomVecN` : → Moments (N-order, N-variate)
- `FactMomN` : → Factorial Moments (N-order, N-variate)
- `CumulantVec` : → Cumulants (N-order, bi-variate)
- `CumulantRatio` : → Cumulant Ratios (N-order, bi-variate)
- `SICumulantVec` : → Strong. Int. Cumulants (N-order, bi-variate)
- `NudynVecN` : → ν-dynamic (N-order, bi-variate)

↓

For analysis

- `SMomentN` : → E-by-E analysis (error, efficiency, bin-width)

SMoment : example of moment conversion

Convert $\mu_{42} = \langle N^4 \bar{N}^2 \rangle$ into factorial moments

Incase you want to use this code:

```
#include "SMoment.h"
{
       MomVecN mnt_; //Class to handle multi-variate moments

        vector<int> mth;   //specify orders of different variates
        mth.push_back(4);  //order of first variate
        mth.push_back(2);  //order of second variate

        mnt_.Mom2FactN(mth); //convert to factorial moment
}
```

Output :

```
Func to convert moments μ_m =<N^m> to factorial moments f_m = <N(N-1)..(N-m+1)>
μ_42= f11 +  f12 + 7 f21 + 7 f22 + 6 f31 + 6 f32 +  f41 +  f42
```

`SMoment` : example of error estimation

Find expression for **variance** of $\kappa_6(\Delta N)$ using **delta theorem**

```
#include "SMoment.h"
{
   const int mth =6; //Order of cumulant of ΔN

   CumulantVec  clt__; //Class to handle bi-variate cumulants

   clt__.CalcVariance(mth); //Find variance in terms of central moments
}
```

Output :

```
Var(K_6)= ( Ç_12,  -  Ç_6^2  - 12 Ç_7  Ç_5  + 36 Ç_5^2  Ç_2 )  + 2 ( - 15 Ç_4 + 90 Ç_2^2 ) ( Ç_8  -
Ç_6  Ç_2  - 6 Ç_5  Ç_3 )  + 2 ( - 15 Ç_2 ) ( Ç_10,  -  Ç_6  Ç_4  - 6 Ç_5^2  - 4 Ç_7  Ç_3  + 24 Ç_5
Ç_3  Ç_2 )  + 2 ( - 20 Ç_3 ) ( Ç_9  -  Ç_6  Ç_3  - 6 Ç_5  Ç_4  - 3 Ç_7  Ç_2  + 18 Ç_5  Ç_2^2 )  + (
- 15 Ç_4 + 90 Ç_2^2 )^2  ( Ç_4  -  Ç_2^2 )  + 2 ( - 15 Ç_2 )( - 15 Ç_4 + 90 Ç_2^2 ) ( Ç_6  -  Ç_4
Ç_2  - 4 Ç_3^2 )  + 2 ( - 20 Ç_3 )( - 15 Ç_4 + 90 Ç_2^2 ) ( Ç_5  - 4 Ç_3  Ç_2 )  + ( - 15 Ç_2 )^2
( Ç_8  -  Ç_4^2  - 8 Ç_5  Ç_3  + 16 Ç_3^2  Ç_2 )  + 2 ( - 15 Ç_2 )( - 20 Ç_3 ) ( Ç_7  - 5 Ç_4  Ç_3
- 3 Ç_5  Ç_2  + 12 Ç_3  Ç_2^2 )  + ( - 20 Ç_3 )^2  ( Ç_6  -  Ç_3^2  - 6 Ç_4  Ç_2  + 9 Ç_2^3 )
```

SMoment : example of efficiency correction

Convert κ_3 into factorial moments & correct for efficiency (2 bins)

```
#include "SMoment.h"
{
   const int mth =3; //Order of cumulant the K(ΔN)

   CumulantVec  clt__; //Class to handle bi-variate cumulants

   vector<int> abinN;  //vector for no. of efficiency bins
   abinN.push_back(2); //N       (e.g. no. of protons)
   abinN.push_back(2); //\bar{N} (e.g. no. of anti-protons)

   clt__.Cumulant2FactMoment(mth,abinN); //cumulants --> factorial moments with efficiency
}
```

Output :

```
Func to convert cumulant K_m to factorial moments f_m = <N(N-1)..(N-m+1)>
 K_3= ( ((f_{1000}/€1  + f_{0100}/€2) + 3(f_{2000}/€1^2    + f_{1100}/€1/€2    + f_{1100}/€1/€2    + f_{0200}/€2^2) + (f_{3000}/
€1^3    + f_{2100}/€1^2/€2+ f_{2100}/€1^2/€2   + f_{1200}/€1/€2^2+ f_{2100}/€1^2/€2  + f_{1200}/€1/€2^2  + f_{1200}/€1/€2^2  +
f_{0300}/€2^3) - 3(f_{1000}/€1 + f_{0100}/€2)^2 - 3(f_{2000}/€1^2     + f_{1100}/€1/€2    + f_{1100}/€1/€2    + f_{0200}/€2^2)
(f_{1000}/€1    + f_{0100}/€2) + 2(f_{1000}/€1 + f_{0100}/€2)^3)  - 3 ((f_{1010}/€1/€3    + f_{1001}/€1/€4    + f_{0110}/€2/€3    +
f_{0101}/€2/€4) - (f_{1000}/€1 + f_{0100}/€2)(f_{0010}/€3   + f_{0001}/€4) + (f_{2010}/€1^2/€3 + f_{2001}/€1^2/€4  + f_{1110}/
€1/€2/€3    + f_{1101}/€1/€2/€4+ f_{1110}/€1/€2/€3     + f_{1101}/€1/€2/€4 + f_{0210}/€2^2/€3  + f_{0201}/€2^2/€4) - (f_{2000}/€1^2
    + f_{1100}/€1/€2    + f_{1100}/€1/€2    + f_{0200}/€2^2)(f_{0010}/€3    + f_{0001}/€4) - 2(f_{1010}/€1/€3  + f_{1001}/€1/€4    +
f_{0110}/€2/€3  + f_{0101}/€2/€4)(f_{1000}/€1  + f_{0100}/€2) + 2(f_{1000}/€1    + f_{0100}/€2)^2(f_{0010}/€3  + f_{0001}/€4))  + 3
((f_{1010}/€1/€3    + f_{1001}/€1/€4    + f_{0110}/€2/€3+ f_{0101}/€2/€4) + (f_{1020}/€1/€3^2   + f_{1011}/€1/€3/€4 + f_{1011}/
€1/€3/€4    + f_{1002}/€1/€4^2  + f_{0120}/€2/€3^2 + f_{0111}/€2/€3/€4 + f_{0111}/€2/€3/€4 + f_{0102}/€2/€4^2) - 2(f_{1010}/€1/€3  +
f_{1001}/€1/€4  + f_{0110}/€2/€3    + f_{0101}/€2/€4)(f_{0010}/€3 + f_{0001}/€4) + 2(f_{1000}/€1      + f_{0100}/€2)(f_{0010}/€3    +
f_{0001}/€4)^2 - (f_{1000}/€1  + f_{0100}/€2)(f_{0010}/€3   + f_{0001}/€4) - (f_{0020}/€3^2    + f_{0011}/€3/€4    + f_{0011}/€3/€4
    + f_{0002}/€4^2)(f_{1000}/€1    + f_{0100}/€2))  -  ((f_{0010}/€3  + f_{0001}/€4) + 3(f_{0020}/€3^2   + f_{0011}/€3/€4    +
f_{0011}/€3/€4  + f_{0002}/€4^2) + (f_{0030}/€3^3  + f_{0021}/€3^2/€4  + f_{0021}/€3^2/€4  + f_{0012}/€3/€4^2  + f_{0021}/€3^2/€4  +
f_{0012}/€3/€4^2    + f_{0012}/€3/€4^2+ f_{0003}/€4^3) - 3(f_{0010}/€3+ f_{0001}/€4)^2 - 3(f_{0020}/€3^2 + f_{0011}/€3/€4    +
f_{0011}/€3/€4  + f_{0002}/€4^2)(f_{0010}/€3   + f_{0001}/€4) + 2(f_{0010}/€3    + f_{0001}/€4)^3)  )
```

SMoment : example of efficiency correction

Find expression for **variance** of $Var(\kappa_6/\kappa_2)$ using **delta theorem**

```
#include "SMoment.h"
{
        CumulantRatio cr_("K6/K2"); //Class to handle bi-variate cumulant ratio

        cr_.CumulantRatio2CorrMoment(); //convert to central moments of ΔN

        cr_.CalcVariance(); //Find variance in terms of central moments of ΔN
}
```

Output :

```
   ( Ç_6 - 15 Ç_2 Ç_4 - 10 Ç_3^2 + 30 Ç_2^3)
k =-----------------------------------------
   ( Ç_2)

Var(k)=((1)/(( Ç_2)) ( + 1) )^2  ( Ç_12,  -  Ç_6^2  - 12 Ç_7  Ç_5  + 36 Ç_5^2  Ç_2 )  + 2 ((1)/(( Ç_2)) ( + 1) )
((1)/(( Ç_2)) ( - 15 Ç_4 + 90 Ç_2^2) + (-1)( Ç_6 - 15 Ç_2 Ç_4 - 10 Ç_3^2 + 30 Ç_2^3)/(( Ç_2)^2) ( + 1) ) ( Ç_8  -
Ç_6  Ç_2  - 6 Ç_5  Ç_3 )  + 2 ((1)/(( Ç_2)) ( + 1) )((1)/(( Ç_2)) ( - 15 Ç_2) ) ( Ç_10,  -  Ç_6  Ç_4  - 6 Ç_5^2
- 4 Ç_7  Ç_3  + 24 Ç_5  Ç_3  Ç_2 )  + 2 ((1)/(( Ç_2)) ( + 1) )((1)/(( Ç_2)) ( - 20 Ç_3) ) ( Ç_9  -  Ç_6  Ç_3  - 6
Ç_5  Ç_4  - 3 Ç_7  Ç_2  + 18 Ç_5  Ç_2^2 )  + ((1)/(( Ç_2)) ( - 15 Ç_4 + 90 Ç_2^2) + (-1)( Ç_6 - 15 Ç_2 Ç_4 - 10
Ç_3^2 + 30 Ç_2^3)/(( Ç_2)^2) ( + 1) )^2  ( Ç_4  -  Ç_2^2 )  + 2 ((1)/(( Ç_2)) ( - 15 Ç_2) )((1)/(( Ç_2)) ( - 15
Ç_4 + 90 Ç_2^2) + (-1)( Ç_6 - 15 Ç_2 Ç_4 - 10 Ç_3^2 + 30 Ç_2^3)/(( Ç_2)^2) ( + 1) ) ( Ç_6  -  Ç_4  Ç_2  - 4 Ç_3^2
)  + 2 ((1)/(( Ç_2)) ( - 20 Ç_3) )((1)/(( Ç_2)) ( - 15 Ç_4 + 90 Ç_2^2) + (-1)( Ç_6 - 15 Ç_2 Ç_4 - 10 Ç_3^2 + 30
Ç_2^3)/(( Ç_2)^2) ( + 1) ) ( Ç_5  - 4 Ç_3  Ç_2 )  + ((1)/(( Ç_2)) ( - 15 Ç_2) )^2  ( Ç_8  -  Ç_4^2  - 8 Ç_5  Ç_3
+ 16 Ç_3^2  Ç_2 )  + 2 ((1)/(( Ç_2)) ( - 15 Ç_2) )((1)/(( Ç_2)) ( - 20 Ç_3) ) ( Ç_7  - 5 Ç_4  Ç_3  - 3 Ç_5  Ç_2
+ 12 Ç_3  Ç_2^2 )  + ((1)/(( Ç_2)) ( - 20 Ç_3) )^2  ( Ç_6  -  Ç_3^2  - 6 Ç_4  Ç_2  + 9 Ç_2^3 )
```

SMoment : example of e-by-e analysis

Calculate κ_6 & its error with efficiency corrections for any no. of bins

```
STEP-I
CumulantVec * clt__ =new CumulantVec(); //Class to handle bi-variate cumulants
vector<FactVec> fvv=clt__->CalcFactVec(6); //Set the moment-vector which includes all expressions

STEP-II

//Set the SMoment class for e-by-e analysis
SMomentN * smt = new SMomentN(1,fvv,abinN);//(centrality-bins, moment-vector, efficiency-bin-vector)
STEP-III
//Event loop
{
   smt->Fill(Np_,eff_,0,1.); //multiplicity-vector, efficiency-vector, centrality-bin, event-weight
}

STEP-IV
//print output
 smt->CalcCumulant(Centrality,1,clt__,IncMom_,ObsMom_,"LOUD");
 smt->CalcCumulantError(Centrality,1,clt__,IncErr_,ObsErr_,"LOUD");

 cout<<Corrected Mom,Err= "<<IncMom_<<"±"<<IncErr_<<" Un-Corrected Mom,Err= "<<ObsMom_<<"±"<<ObsErr_<<endl;
```

SMoment : example of e-by-e analysis

```
const int mth =3; //Order of cumulant the K(ΔN)

CumulantVec * clt__ =new CumulantVec(); //Class to handle bi-variate cumulants

vector<FactVec> fvv=clt__->CalcFactVec(mth); //Set the moment-vector which includes all expressions

//Set the no. of bins for efficiency
vector<int> abinN;
abinN.push_back(2); //(No. of efficiency bins for protons)
abinN.push_back(2); //(No. of efficiency bins for anti-protons)

//Set the values for efficiency
vector<double> eff_;
eff_.push_back(0.5); eff_.push_back(0.5);//(efficiency values for protons)
eff_.push_back(0.5); eff_.push_back(0.5);//(efficiency values for anti-protons)

//Set the SMoment class for e-by-e analysis
SMomentN * smt = new SMomentN(1,fvv,abinN);//(centrality-bins, moment-vector, efficiency-bin-vector)


//Event loop
for(int i=0; i<1e5; i++){

   vector<int> Np_; //multiplicity-vector
   Np_.push_back(gRandom->Binomial(gRandom->Poisson(20),0.5));
   Np_.push_back(gRandom->Binomial(gRandom->Poisson(20),0.5));
   Np_.push_back(gRandom->Binomial(gRandom->Poisson(10),0.5));
   Np_.push_back(gRandom->Binomial(gRandom->Poisson(10),0.5));

   smt->Fill(Np_,eff_,0,1.); //multiplicity-vector, efficiency-vector, centrality-bin, event-weight
}

for(int i=0; i<3; i++){
 smt->CalcCumulant(Centrality,1,clt__,IncMom_,ObsMom_,"LOUD");
 smt->CalcCumulantError(Centrality,1,clt__,IncErr_,ObsErr_,"LOUD");
 cout<<" K("<<i<<") : Corrected= "<<IncMom_<<"±"<<IncErr_<<" Un-Corrected = "<<ObsMom_<<"±"<<ObsErr_<<endl;
}
```

SMoment : example of efficiency correction

Find expression for **variance** of $Var(\kappa_4/\kappa^2)$ using **delta theorem**

```
#include "SMoment.h"
{
        CumulantRatio cr_("K4/K2^2"); //Class to handle bi-variate cumulant ratio

        cr_.CumulantRatio2CorrMoment(); //convert to central moments of ΔN

        cr_.CalcVariance(); //Find variance in terms of central moments of ΔN
}
```

Output :

```
input=K4/K2^2
 order=4 expo=1
 order=2 expo=-2
   ( Ç_4 - 3 Ç_2^2)
k =--------------------
   ( Ç_2)^2

Var(k)=((1)/(( Ç_2)^2) ( + 1) )^2  ( Ç_8  -  Ç_4^2  - 8 Ç_5  Ç_3  + 16 Ç_3^2  Ç_2 )  + 2
((1)/(( Ç_2)^2) ( + 1) )((1)/(( Ç_2)^2) ( - 6 Ç_2) + (-2)( Ç_4 - 3 Ç_2^2)/(( Ç_2)^3) ( +
1) ) ( Ç_6  -  Ç_4  Ç_2  - 4 Ç_3^2 )  + ((1)/(( Ç_2)^2) ( - 6 Ç_2) + (-2)( Ç_4 - 3 Ç_2^2)/((
Ç_2)^3) ( + 1) )^2  ( Ç_4  -  Ç_2^2 )
```