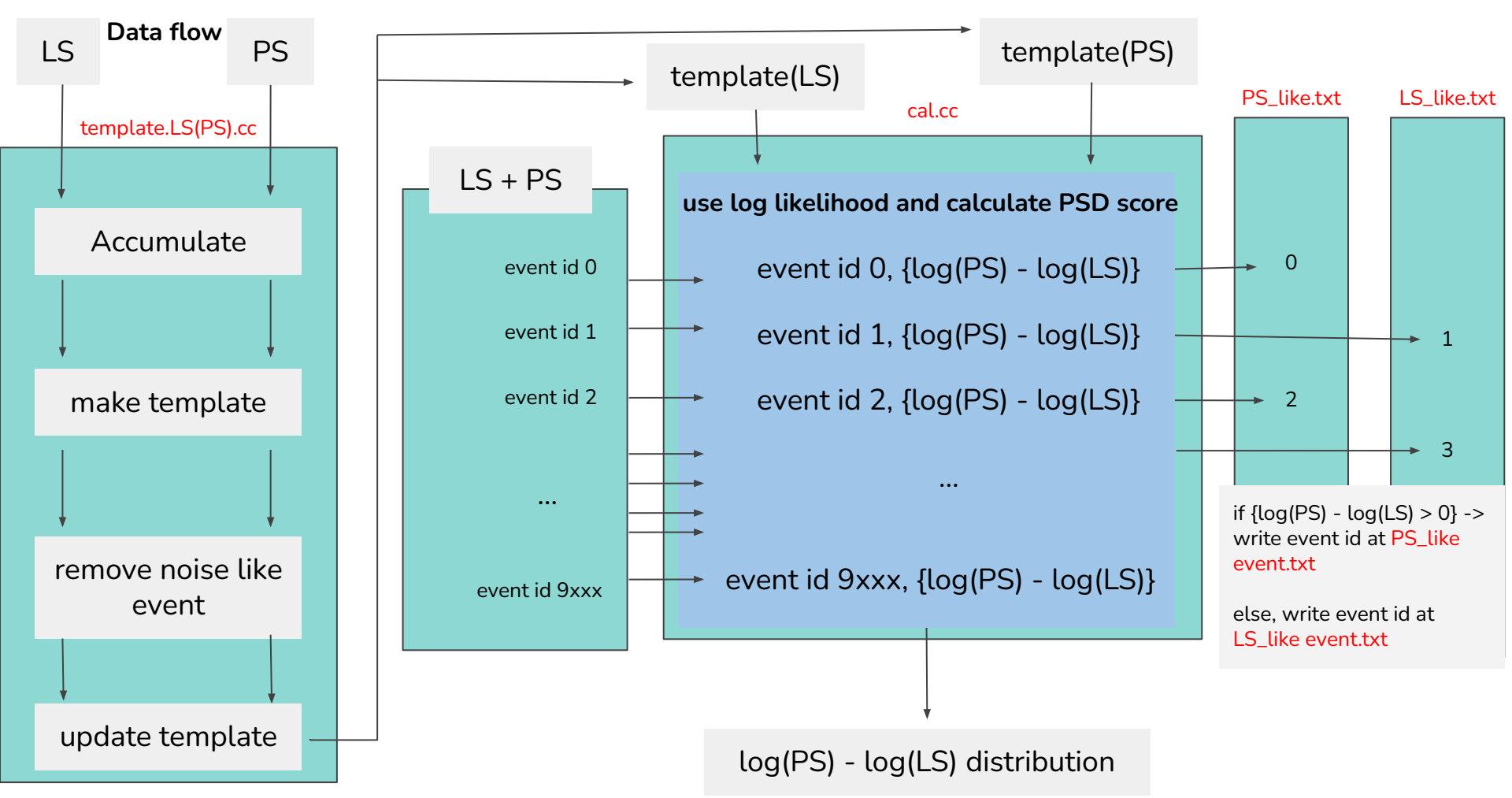
데이터분석및실험
기말과제

2022220206 김종엽

Content

- Accumulate & Make template
- Remove noise like event & update template
  - convolution
  - total charge
  - update template
- Log likelihood
- PSD score


LS
Data flow
PS
template.LS(PS).cc
Accumulate
make template
remove noise like event
update template
template(LS)
template(PS)
cal.cc
LS + PS
event id 0
event id 1
event id 2
...
event id 9xxx
use log likelihood and calculate PSD score
event id 0, {log(PS) - log(LS)}
event id 1, {log(PS) - log(LS)}
event id 2, {log(PS) - log(LS)}
...
event id 9xxx, {log(PS) - log(LS)}
PS_like.txt
LS_like.txt
0
1
2
3
if {log(PS) - log(LS) > 0} -> write event id at PS_like event.txt
else, write event id at LS_like event.txt
log(PS) - log(LS) distribution

Accumulate & Make template

Accumulate & Make template

```
// event loop
for(int i = 0; i < total ; i++){

        vector<double> tempList(40);
        chain->GetEntry(i);
        ped = 0 ; Qtotal = 0 ;
        ped_count = 0; peak_bin = 0;
        cnt = 0 ;
        value = 0 ;

        // pedestal loop
        for(int jj = 0 ; jj < hist_point; jj++){
                his->SetBinContent(jj+1,fadc0[jj]);

                if (jj >=0 && jj < 1100){
                ped += fadc0[jj];
                ped_count++;
                }
        }
        ped /= (double)ped_count; // normalize pedestal
        peak_bin = his->GetMaximumBin();

        //Qtotal loop
        for(int jj = peak_bin - 10 ; jj < peak_bin + 30 ; jj++) Qtotal += fadc0[jj] - ped;
        Qtotal_list[i] = Qtotal ;

        // calculate the bin content using pedestal
        for(int jj = peak_bin - 10 ; jj < peak_bin + 30 ; jj++){
                value = (double)(fadc0[jj] - ped) ;
                tempList[cnt] = value ;
                cnt ++ ;
        }

        vecOfVecs[i] = tempList;
}
```

- 각 이벤트마다 pedestal 제외.
- 전압 최대치가 되는 bin 을 기준으로 -10 에서 +30 만큼 템플릿을 제작할 범위를 설정.
- 이후 각 이벤트의 bin content 에 대한 정보를 vector 에 저장.

Accumulate & Make template

```
//make template
for (int j = 0 ; j < 40 ; j ++){
        sum = 0 ;
        for (int i = 0 ; i < total ; i++) sum += vecOfVecs[i][j];
        LS_accu->SetBinContent(j+1,sum);
}
LS_accu->Scale(1.0/LS_accu->Integral());

for (int i = 0 ; i < 40 ; i++) temp[i] = LS_accu->GetBinContent(i+1);
```

각 이벤트의 waveform 정보를 저장한

2차원 벡터(vecOfVecs 변수)

accumulate

-> 첫번째 bin content 의 총합.

Remove noise like event & Update template

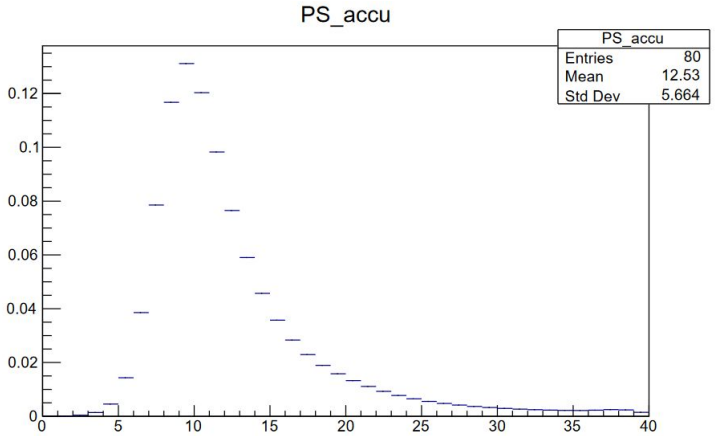
Remove nose like event & Update template

- convolution 과 , total charge 에 대한 하한선을 LS , PS 데이터 파일에 적용하여 noise like 이벤트를 제거.

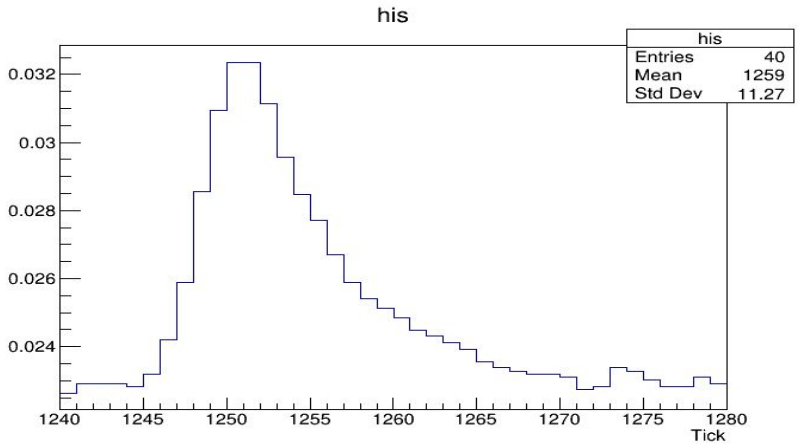
- 이후, signal like event 의 정보만을 이용하여 템플릿을 업데이트.

Convolution

signal like event의 경우, 탬플릿과의 Convolution 형태가 일정하고 높은 피크를 보이는 것을 이용하여, Convolution 최대값이 낮은 이벤트를 제외하고 탬플릿을 다시 제작.

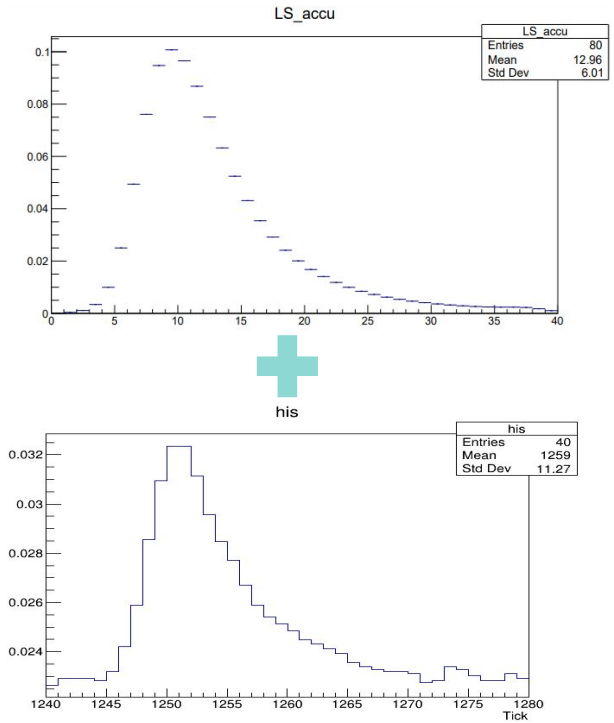


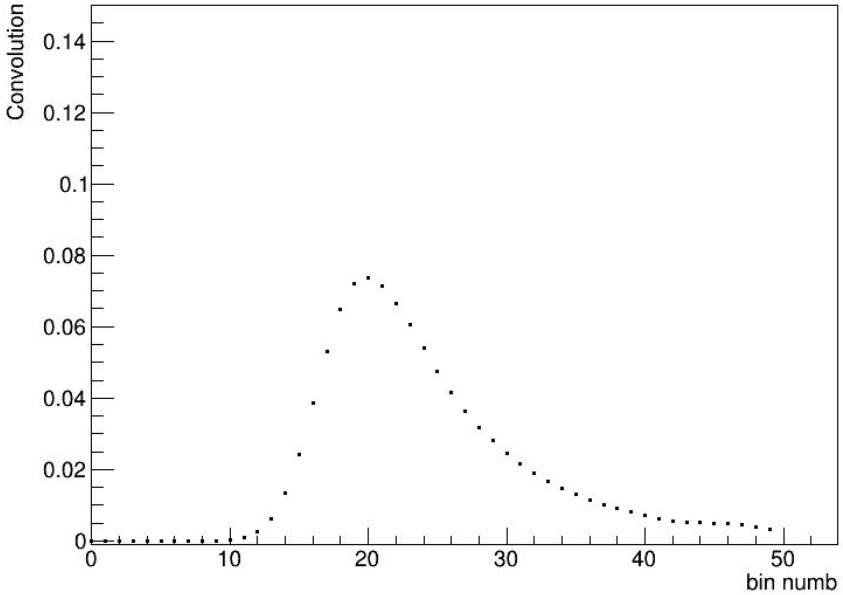
각 이벤트마다 (peak bin -10) ~ (peak bin + 30) 범위 내의 waveform 을 축적하여 만든 PS 데이터의 탬플릿.



PS 데이터 의 특정 이벤트의 (peak bin -10) ~ (peak bin + 30) 범위 내의 waveform

Convolution

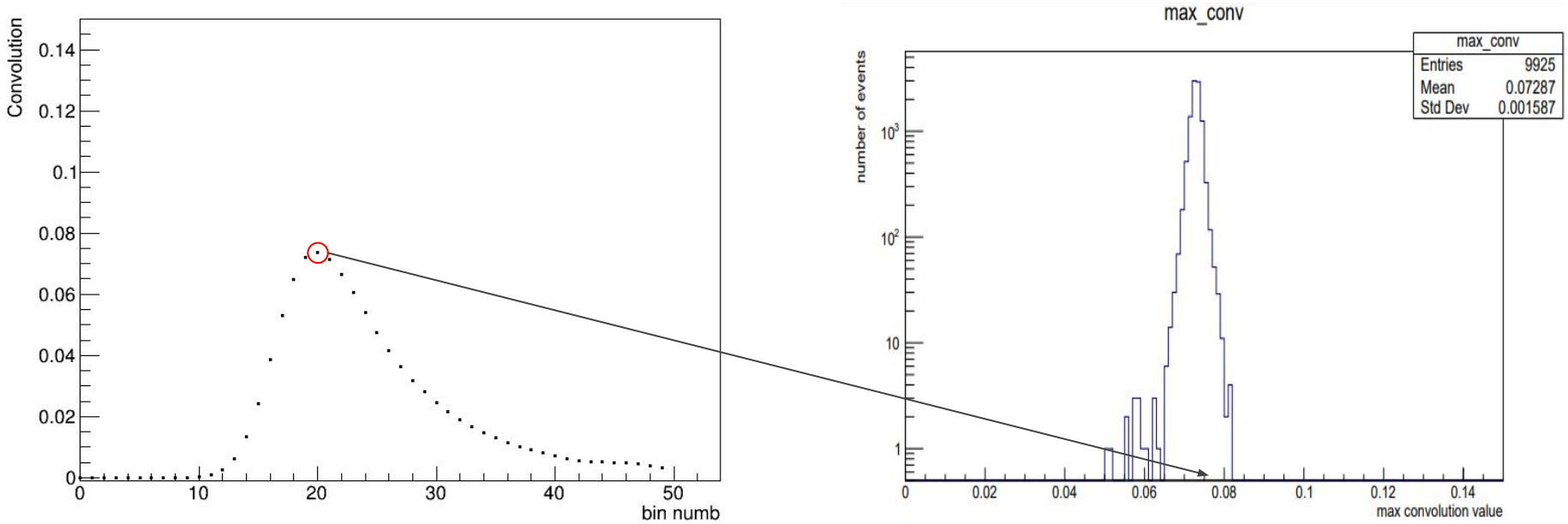




Convolution 결과

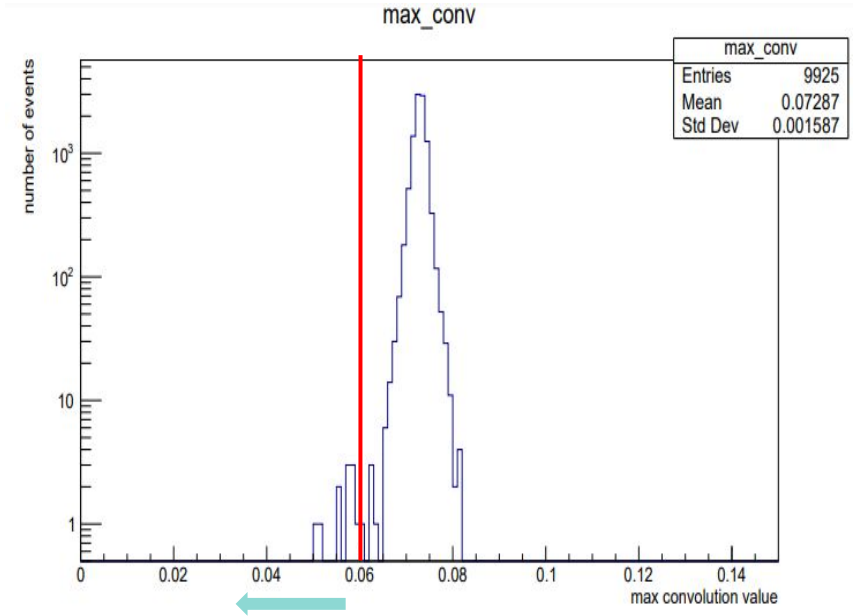
Convolution

Convolution 최대값의 분포도를 제작.



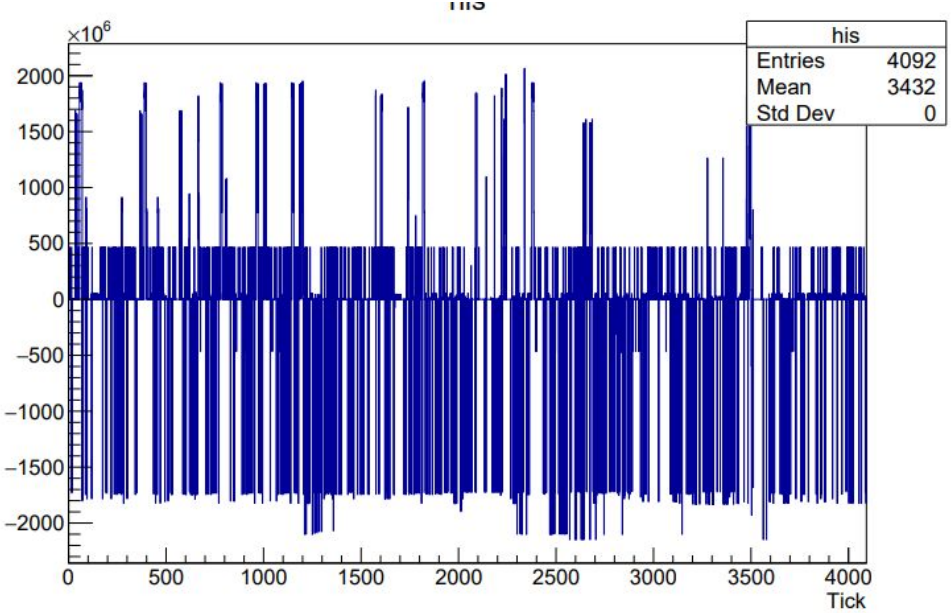
Convolution

낮은 Convolution 최대값을 가지는 이벤트들을 제외 (0.6 이하)

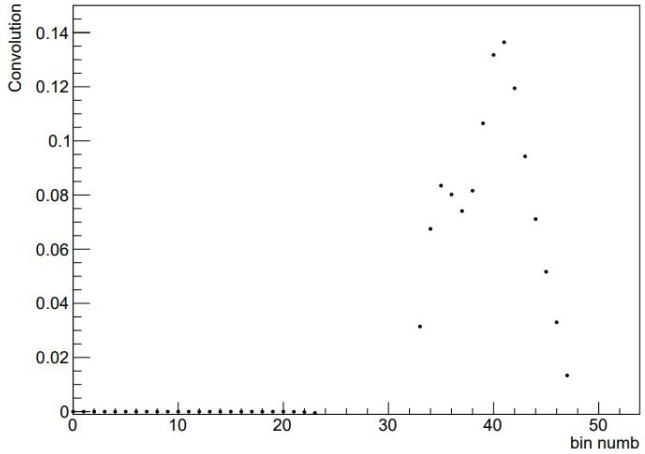


Convolution

PS 파일의 event id = 7711 데이터

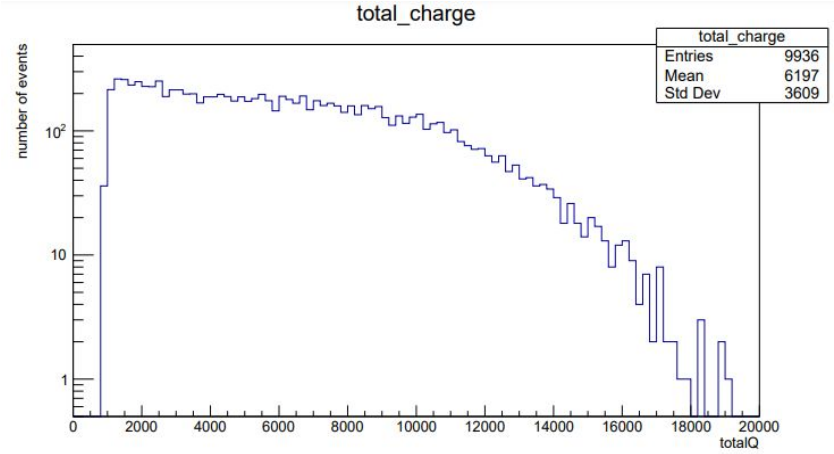


Convolution 패턴이 다른 경우.

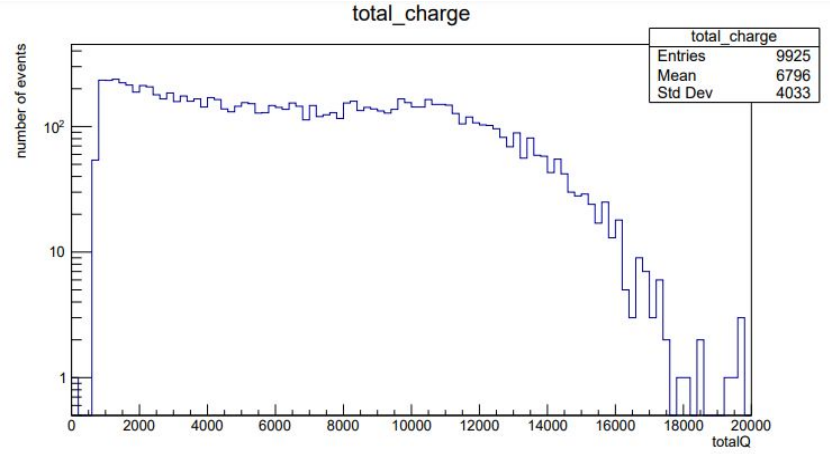


Total charge

하위 5% 의 total charge 를 가지는 이벤트를 제외.(LS 의 경우, 1400 이하, PS 의 경우, 1200 이하)



LS 데이터의 total charge 분포도

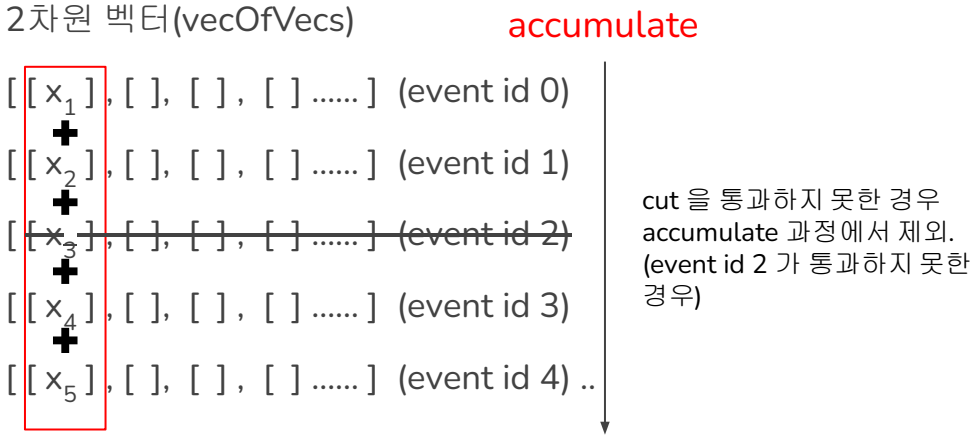


PS 데이터의 total charge 분포도

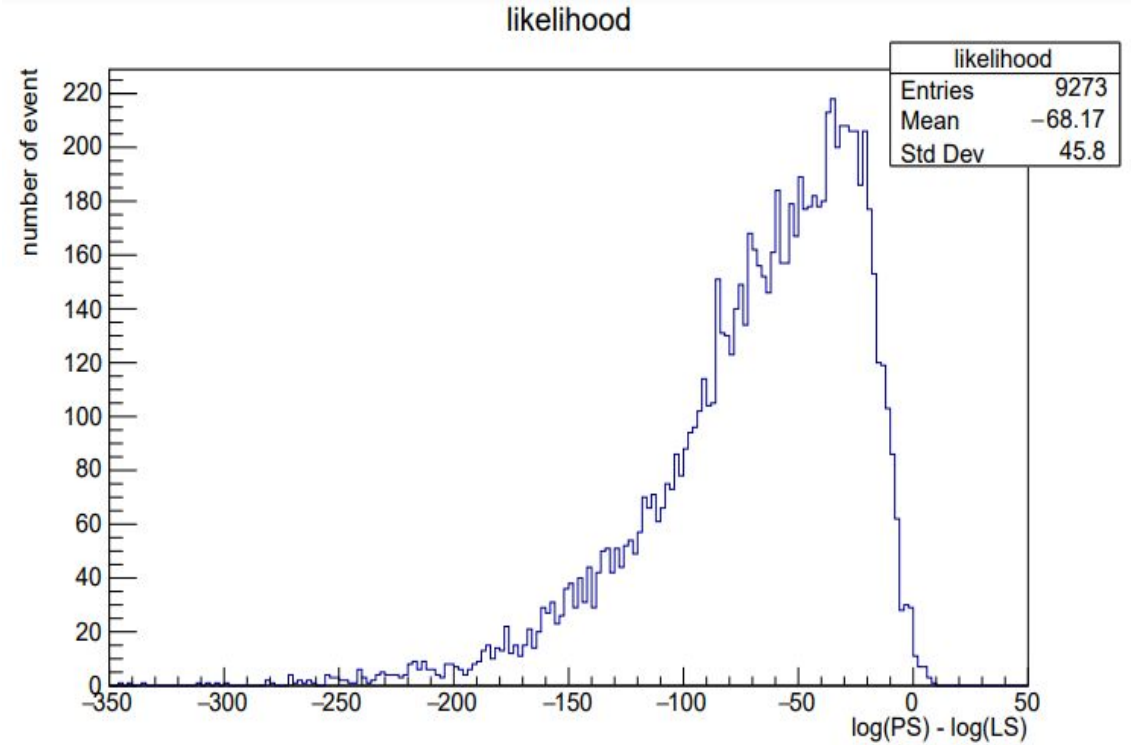
Update Template

```
// apply cut
for (int i = 0 ; i < total ; i++){
        if ((conv_list[i] < 0.05) || (Qtotal_list[i] < 1400)){
                for (int j = 0 ; j < 40 ; j++) vecOfVecs[i][j] = 0 ;
        }
}

//update template
for (int j = 0 ; j < 40 ; j ++){
        sum = 0 ;
        for (int i = 0 ; i < total ; i++) sum += vecOfVecs[i][j];
        LS_accu->SetBinContent(j+1,sum);
}
LS_accu->Scale(1.0/LS_accu->Integral());
```



Update Template



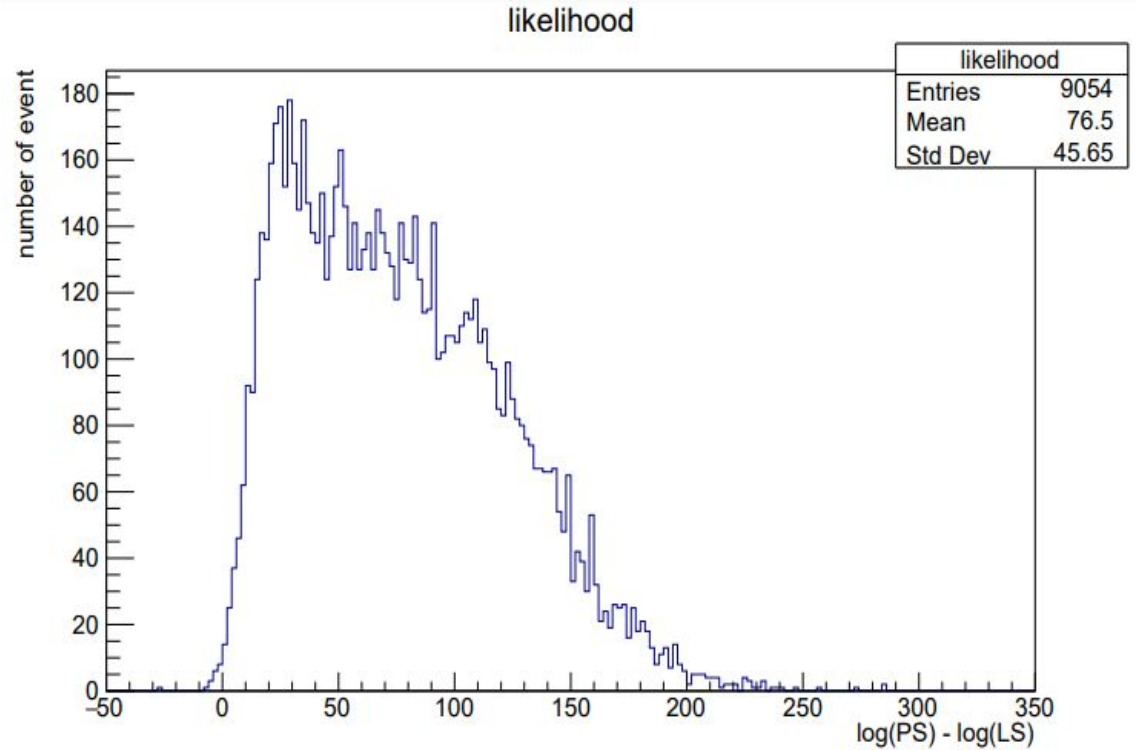
LS 데이터 의 PSD 점수.

Qtotal cut = 1500,

오판율 = 0.31 %

root cal_LS.cc 입력 후,
result_LS.root 파일 에서 확인가능

Update Template



PS 데이터 의 PSD 점수

Qtotal cut = 1500,

오판율 = 0.43 %

root cal_PS.cc 입력 후,
result_PS.root 파일에서 확인가능

Log Likelihood

Log Likelihood

$$log[L] = N_i \sum_{i=1}^{nbin} log[P_i]$$

log(L) = log likelihood 값, i = bin 번호

N_i = LS+PS 데이터의 i 번째 bin content 값,

P_i = LS, PS 탬플릿의 i 번째 bin content 값.

Log Likelihood

$$\log(PS) = \sum_{i}^{\text{bin number}} (\text{test event bin content}) * \log_{10}(\text{PS data template bin content})$$

```
if (Qtotal < charge_cut ) continue ;
// likelihood calculation
for (int jj = 0 ; jj < 40 ; jj++){
        total_LS += content[jj] * log10( LS_accu→GetBinContent(jj+1));
       }

for (int jj = 0 ; jj < 40 ; jj++){
        total_PS += content[jj] * log10(PS_accu→GetBinContent(jj+1));
       }
```

log likelihood 계산을 다음과 같이 수행.

Log Likelihood

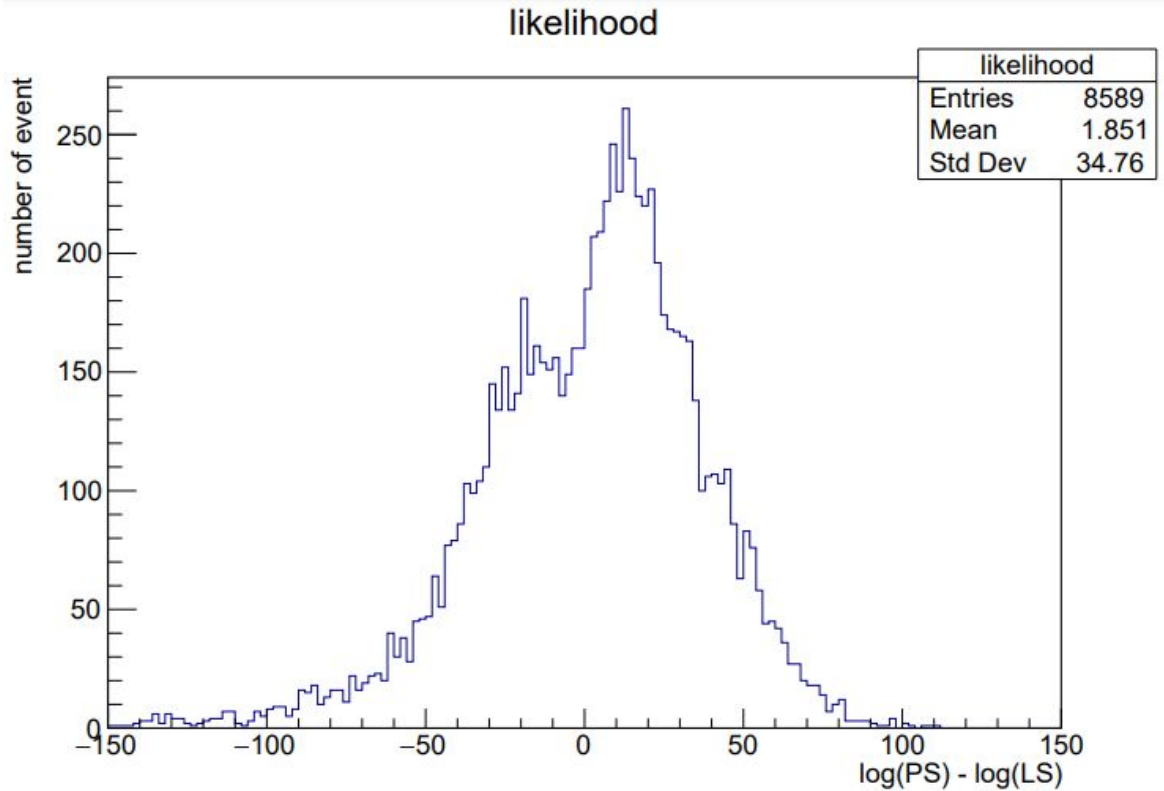
PSD 점수 = log(PS) - log(LS)

```
if ((total_PS - total_LS) > 0) {
        file1 << i << endl;
        PS_like += 1 ;
}
else{
        file2 << i << endl;
        LS_like += 1 ;
}
```

PSD 점수를 계산하여 PS like 한 이벤트와 LS like한 이벤트의 갯수를 확인.

PSD score distribution

PSD score



- LS + PS 데이터에 대한 Qtotal 컷 : 1500.
- 전체 event 갯수 : 8589
- log(PS) - log(LS) > 0 인 event 수 : 4875.
- log(PS) - log(LS) < 0 인 event 수 : 3714.
- event id 에 대한 정보는 txt 파일에 저장.

파일 설명 (Final_exam 디렉토리 위치 기준)

template_LS.cc : LS 디렉토리에 LS 탬플릿 생성

template_PS.cc : PS 디렉토리에 PS 탬플릿 생성

cal.cc : LS + PS 데이터 log likelihood 계산

cal_LS.cc : LS 데이터 log likelihood 계산

cal_PS.cc : PS 데이터 log likelihood 계산

파일 설명 (Final_exam 디렉토리 위치 기준)

result_LS.root, result_PS.root : LS, PS 데이터의 PSD 점수 분포도

result.root : LS + PS 데이터의 PSD 점수 분포도

LS/LS_accu.root : LS 탬플릿, Qtotal, Convolution 최대치 분포도

PS/PS_accu.root : PS 탬플릿, Qtotal, Convolution 최대치 분포도

convolution/conv_shape_LS.cc : LS 데이터의 Convolution 확인용 파일

convolution/conv_shape_PS.cc : PS 데이터의 Convolution 확인용 파일

실행 순서

Final_exam 디렉토리에서 다음과 같이 명령어 입력

root template_LS.cc

root template_PS.cc

-> PS, LS 디렉토리에 각각 PS, LS 탬플릿 생성

이후 root cal.cc 명령어로

PSD 점수 분포도 확인 가능.