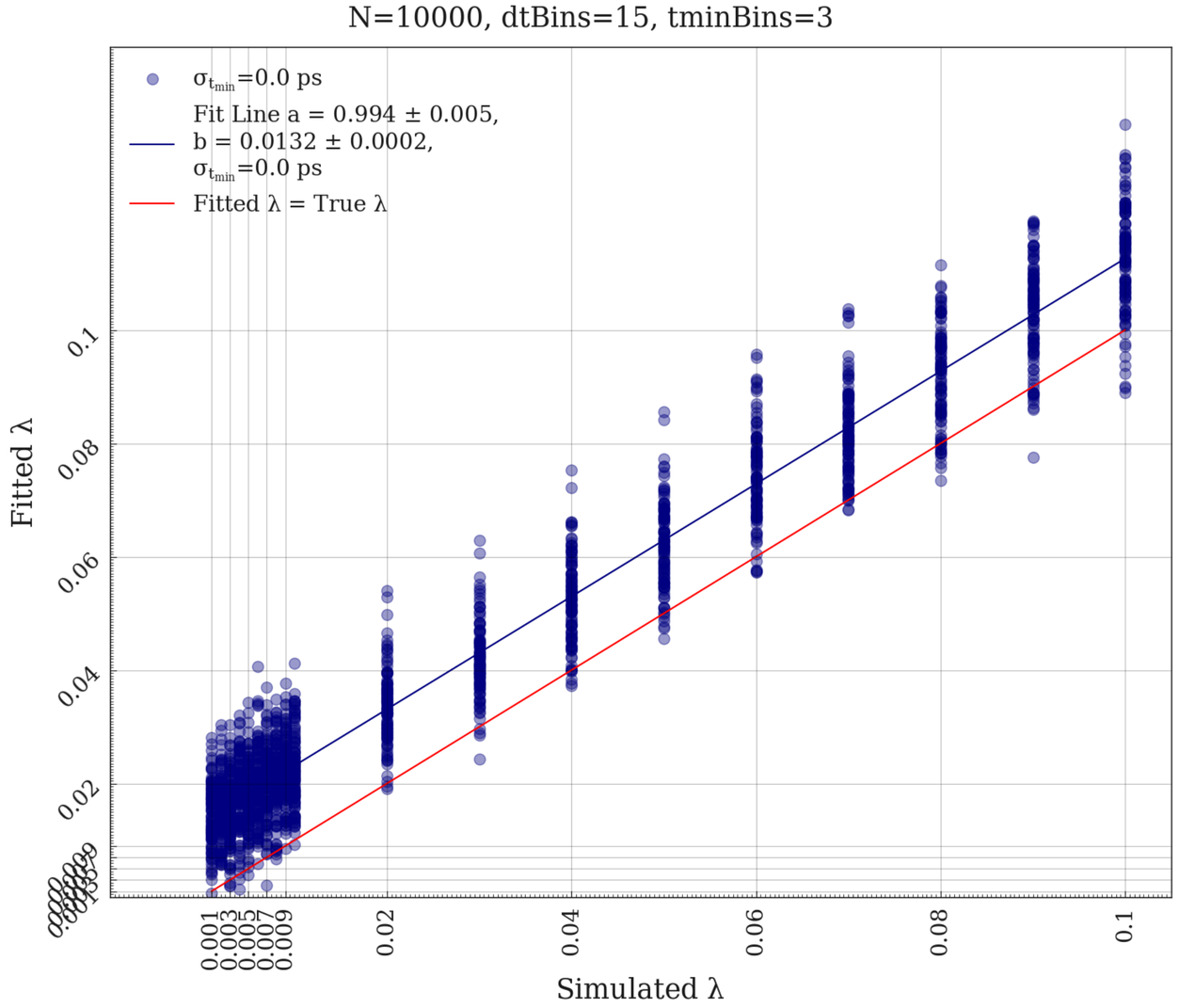
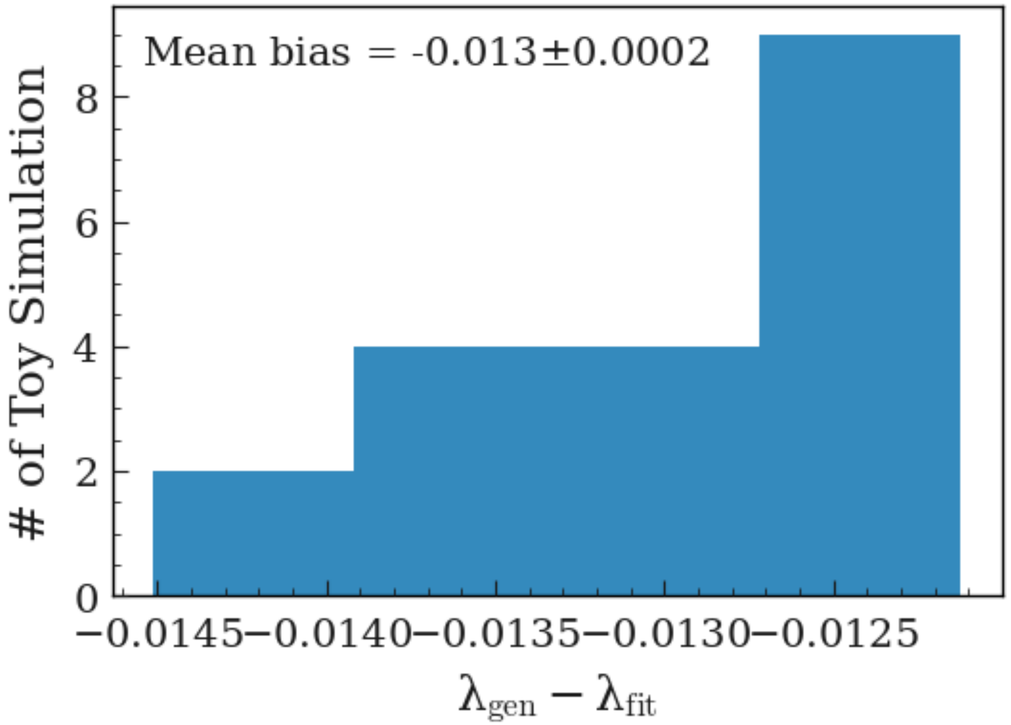
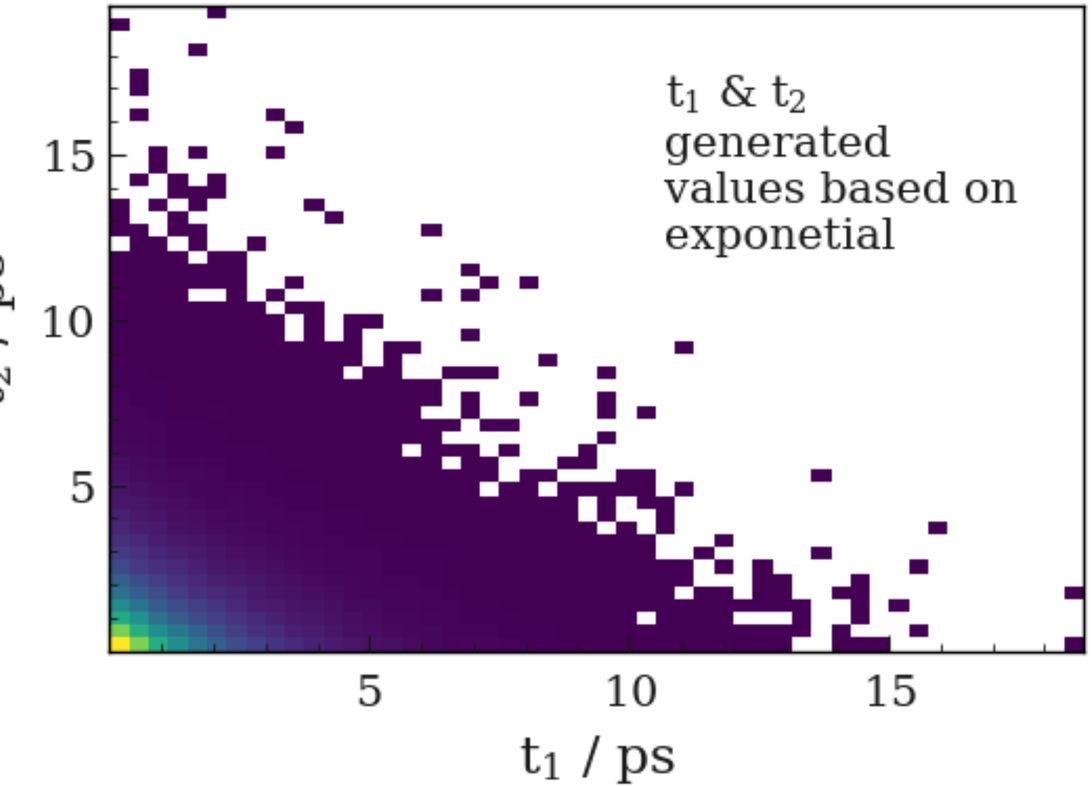
Status Update - Decoherence Fitter



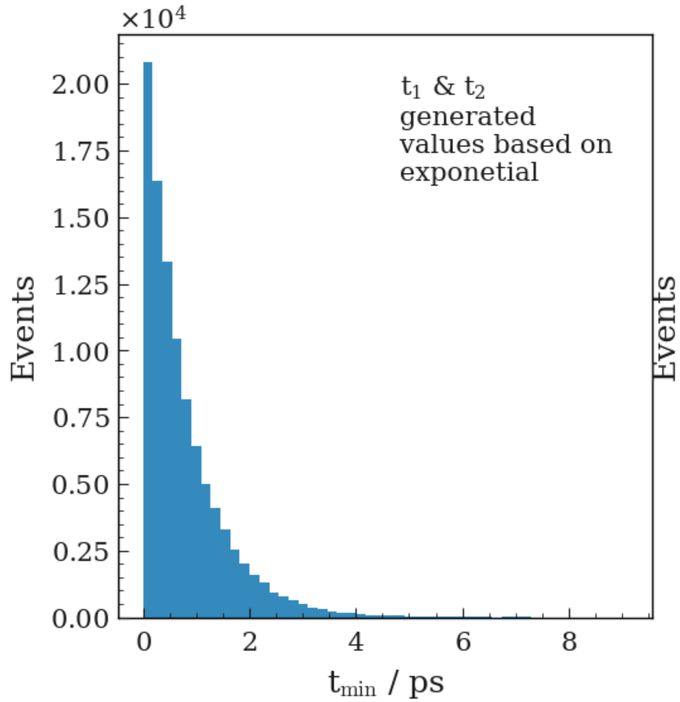


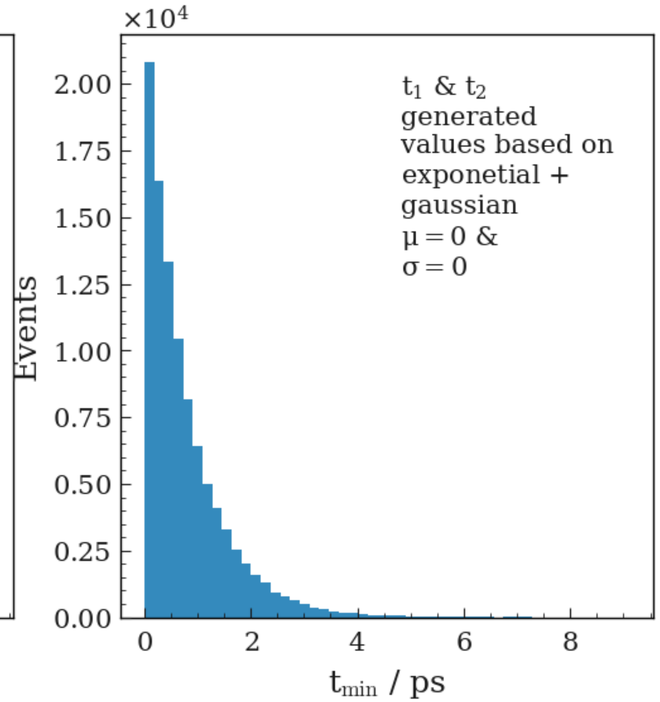
- Observe a constant bias for all simulated λ values
- Need to find the error in the fitter now!

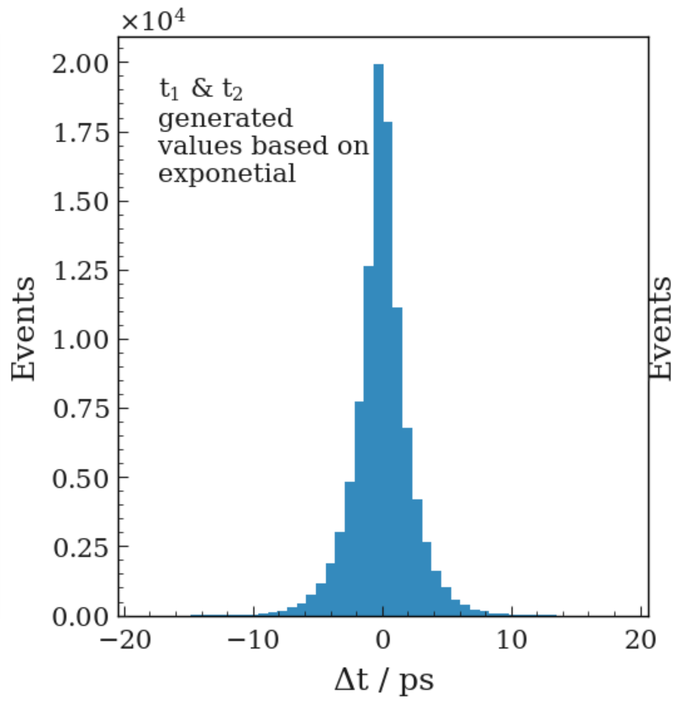
Step by step analysis: Δt, $t_{\min}$ and $\mathrm{sf}_{\mathrm{prob}}$

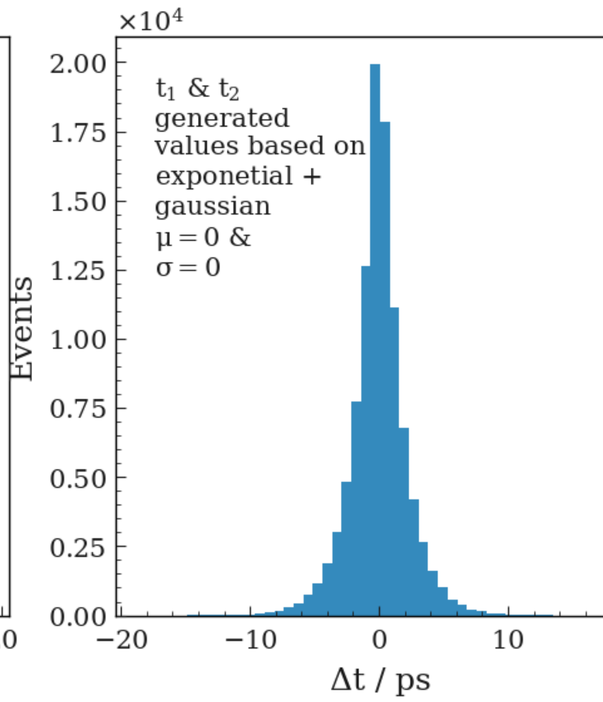



```
t1 = np.random.exponential(scale=tau, size=N) * 1.04
t2 = np.random.exponential(scale=tau, size=N) * 1.04

deltaT = t2 - t1

MinimumTimeCM = np.minimum(t1, t2)

MinimumTimeCM_Error = 0.0
deltaT_Error_forGauss = 0.0

# When we generate same flavor and opposite flavor events, do we need to account for mint and dt erro
MinimumTimeCM_withGauss, MinimumTimeCM_withGaussError = AddGaussianNoise(MinimumTimeCM,
                                                                         MinimumTimeCM_Error, mu=mu,
                                                                         sigma=sigma)
deltaT_withGauss, deltaT_withGaussError = AddGaussianNoise(deltaT, deltaT_Error_forGauss,
                                                           mu=mu, sigma=sigma)

SFProb, SFProb_error = CumulativeProb(keyword='NSF', t1=t1, t2=t2, lam=theolam, dt=deltaT_withGauss,
                                      dt_error=0, tmin=MinimumTimeCM_withGauss, tmin_error=0)
LogicalComparison = np.random.uniform(0, 1, N)

SameFlavorCM= SFProb > LogicalComparison # Gives true or false, wheather to count event as SF or not
OppositeFlavorCM = ~SameFlavorCM
```

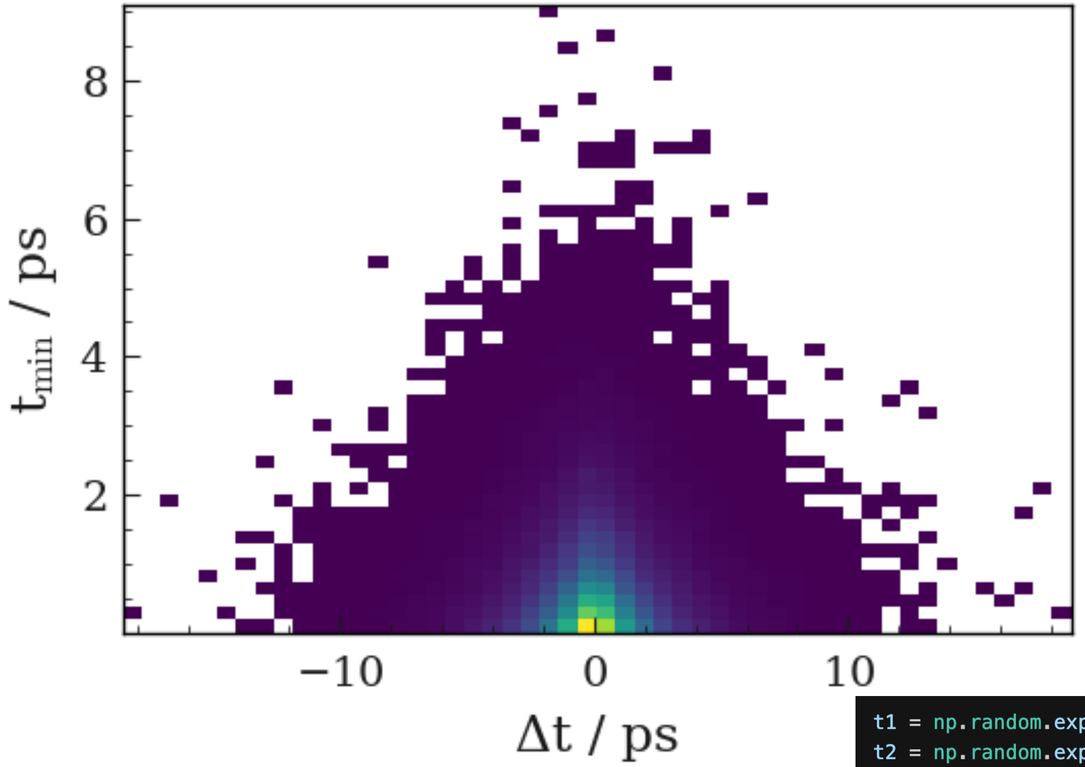


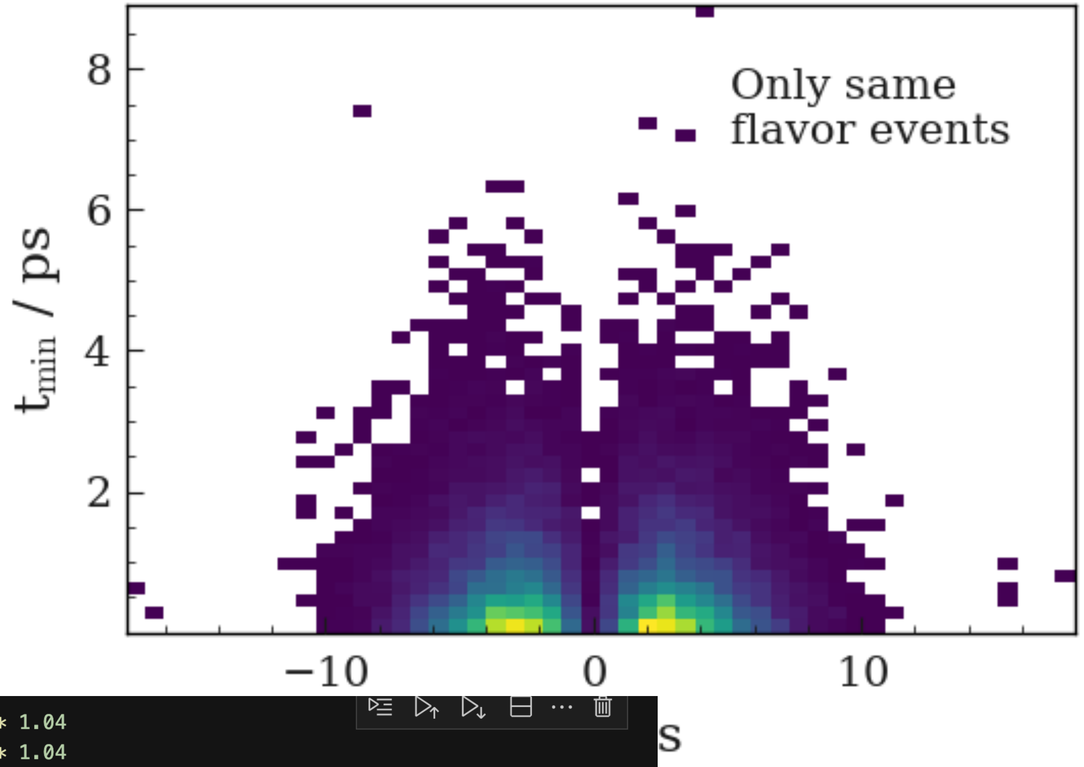




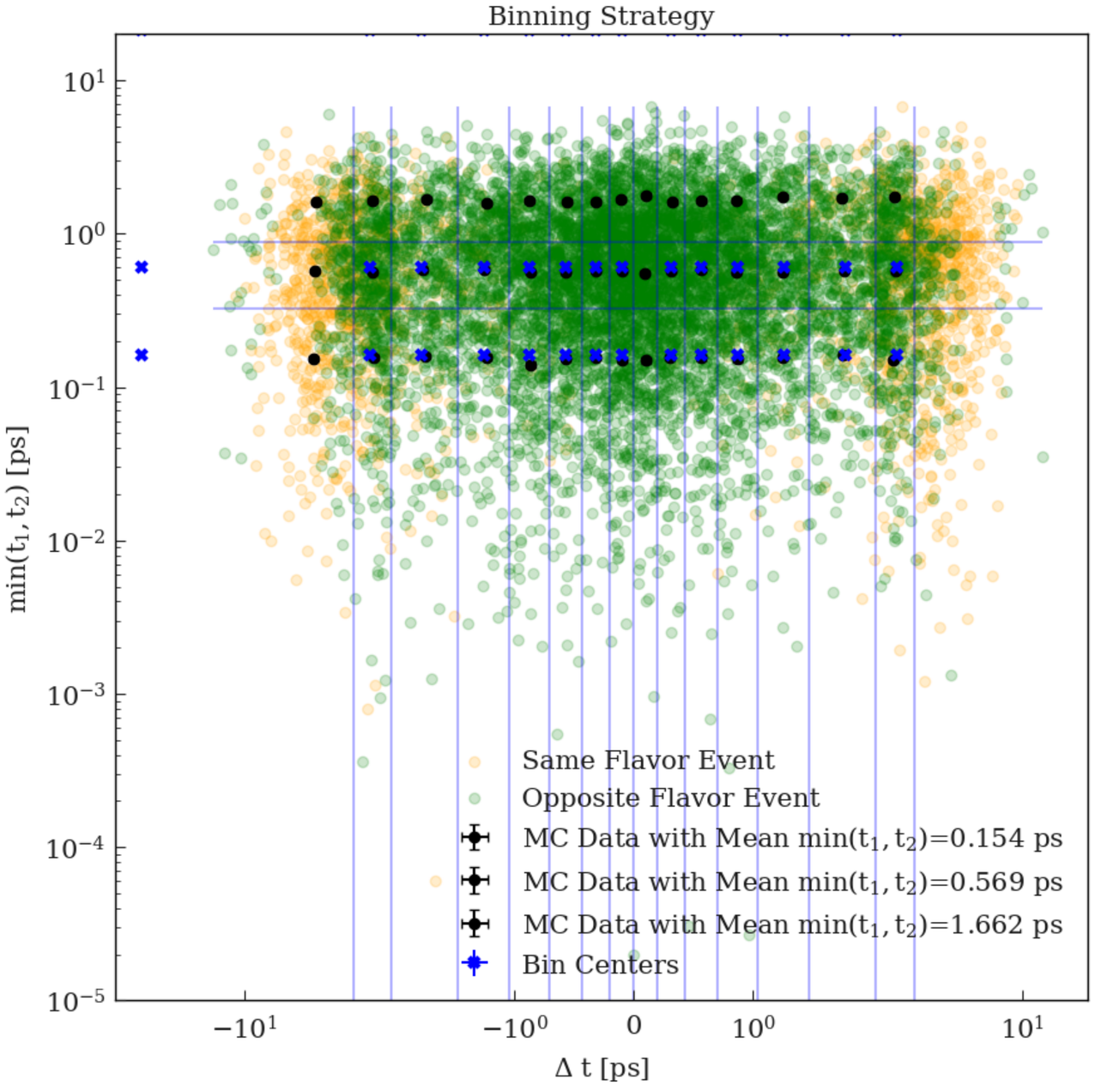


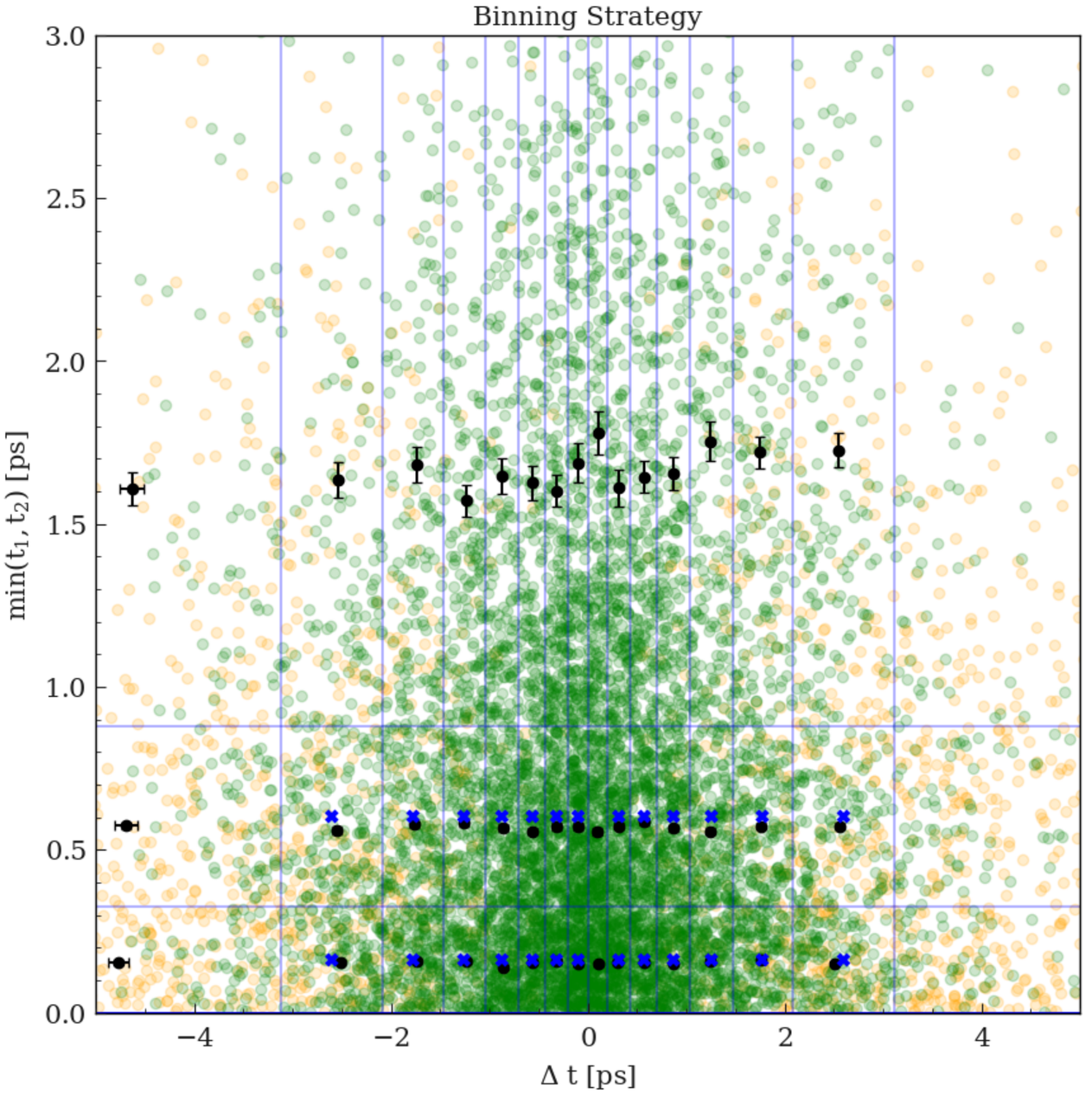

```
t1 = np.random.exponential(scale=tau, size=N) * 1.04
t2 = np.random.exponential(scale=tau, size=N) * 1.04

deltaT = t2 - t1

MinimumTimeCM = np.minimum(t1, t2)

MinimumTimeCM_Error = 0.0
deltaT_Error_forGauss = 0.0

# When we generate same flavor and opposite flavor events, do we need to account for mint and dt err
MinimumTimeCM_withGauss, MinimumTimeCM_withGaussError = AddGaussianNoise(MinimumTimeCM,
                                                                         MinimumTimeCM_Error, mu=mu,
                                                                         sigma=sigma)
deltaT_withGauss, deltaT_withGaussError = AddGaussianNoise(deltaT, deltaT_Error_forGauss,
                                                           mu=mu, sigma=sigma)

SFProb, SFProb_error = CumulativeProb(keyword='NSF', t1=t1, t2=t2, lam=theolam, dt=deltaT_withGauss,
                                      dt_error=0, tmin=MinimumTimeCM_withGauss, tmin_error=0)
LogicalComparison = np.random.uniform(0, 1, N)

SameFlavorCM= SFProb > LogicalComparison # Gives true or false, wheather to count event as SF or not
OppositeFlavorCM = ~SameFlavorCM
```

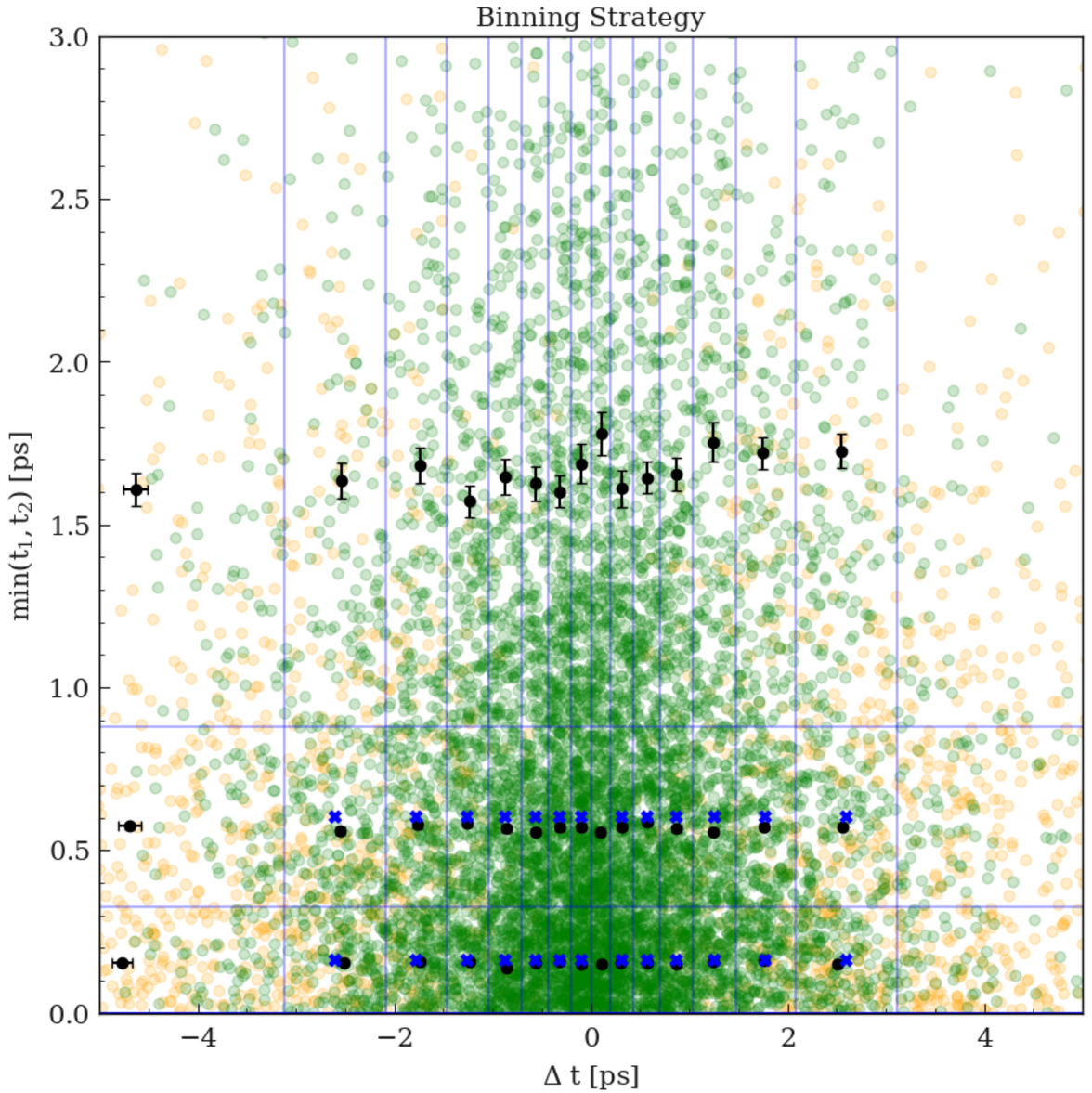





```
t1 = np.random.exponential(scale=tau, size=N) * 1.04
t2 = np.random.exponential(scale=tau, size=N) * 1.04

deltaT = t2 - t1

MinimumTimeCM = np.minimum(t1, t2)

MinimumTimeCM_Error = 0.0
deltaT_Error_forGauss = 0.0

# When we generate same flavor and opposite flavor events, do we need to account for mint and dt erro
MinimumTimeCM_withGauss, MinimumTimeCM_withGaussError = AddGaussianNoise(MinimumTimeCM,
                                                                         MinimumTimeCM_Error, mu=mu,
                                                                         sigma=sigma)
deltaT_withGauss, deltaT_withGaussError = AddGaussianNoise(deltaT, deltaT_Error_forGauss,
                                                           mu=mu, sigma=sigma)

SFProb, SFProb_error = CumulativeProb(keyword='NSF', t1=t1, t2=t2, lam=theolam, dt=deltaT_withGauss,
                                      dt_error=0, tmin=MinimumTimeCM_withGauss, tmin_error=0)
LogicalComparison = np.random.uniform(0, 1, N)

SameFlavorCM= SFProb > LogicalComparison # Gives true or false, wheather to count event as SF or not
OppositeFlavorCM = ~SameFlavorCM
```


Binning Strategy
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
$\min(t_1, t_2)$ [ps]
-10^{1}
-10^{0}
0
10^{0}
10^{1}
Δ t [ps]
Same Flavor Event
Opposite Flavor Event
MC Data with Mean $\min(t_1, t_2)$=0.154 ps
MC Data with Mean $\min(t_1, t_2)$=0.569 ps
MC Data with Mean $\min(t_1, t_2)$=1.662 ps
Bin Centers


Binning Strategy
3.0
2.5
2.0
1.5
1.0
0.5
0.0
$\min(t_1, t_2)$ [ps]
−4
−2
0
2
4
Δ t [ps]



- Errors of $t_{\min}$ and Δt are calculated as the "Standard error of the mean":

$$SEM_i = \frac{\text{Standard deviation of sample in bin } i}{\sqrt{\text{Sample Size in bin } i}}$$

```
dtErrors[ind, mtind] = np.std(deltaT[FullCondition]) / np.sqrt(EvtPerBin[mtind, ind])

minTErrors[ind, mtind] = np.std(MinimumTime[FullCondition]) / np.sqrt(EvtPerBin[mtind, ind])
```

B2D*lnu —> 0.049
D*2Dpi0 —> 0.677
1) D2K+pi- —> .03947 or 2) D2K+pi-pi0 —> 0.144 or 3) D2K+pi+pi- —> 0.0280

Run1 data set —> 0.5ab^-1

BBbar XSection —> 0.5346 nb ~ 0.5346e9 ab

Expected Number of events (no tagging efficiency):
1) 615379.71
2) 2553737.916
3) 496560.00

With hadronic tagging efficiency (~0.01):
1) 6153.8
2) 25537.4
3) 4965.6

With leptonic tagging efficiency (~0.10):
1) 61537.97
2) 255373.79
3) 49656.00