# $\chi^2$ of Tracks in GlueX

Kei Moriya

2011.04.06

# Motivation

- understand origin of $\chi^2$ of each track
- will be used in kinematic fitting of events
- currently, strong hints that the numbers are unreliable...
  - efficiency of reconstruction
  - large number of events with $\chi^2$ too low/high

# Line of Attack

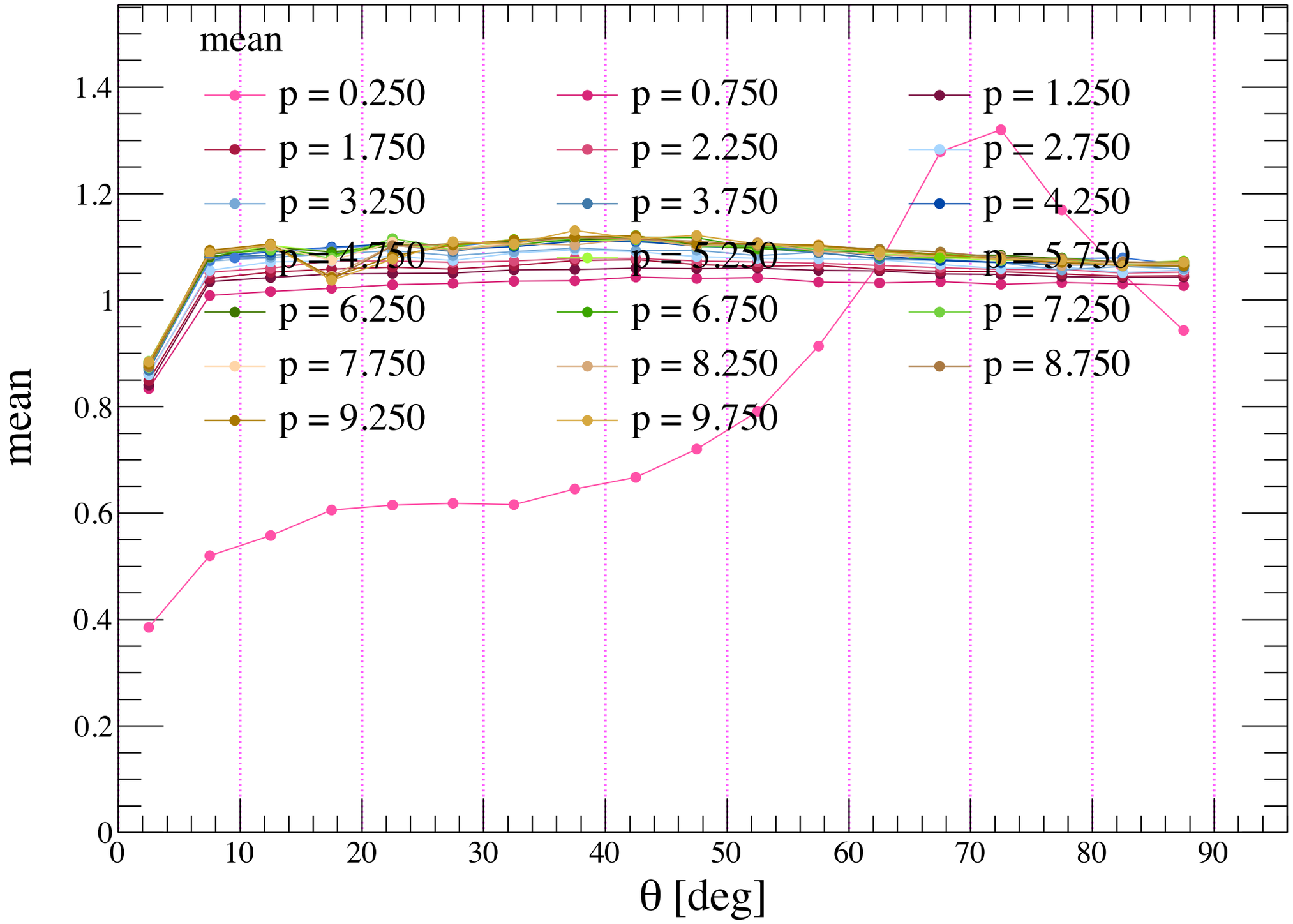
- previously used multi-particle events
- complicated to look at individual tracks due to multiplicity
- now using particle gun
- now possible to match up generated vs. reconstructed momentum

# Events

- generate single particle events in bins of p, θ
- 20k events in
  - 0.5 GeV/c bins for 0< p <10
  - 5° bins for 0°< θ <90°
- total: 360 bins
- generated for p, $\pi^+$, $\pi^-$

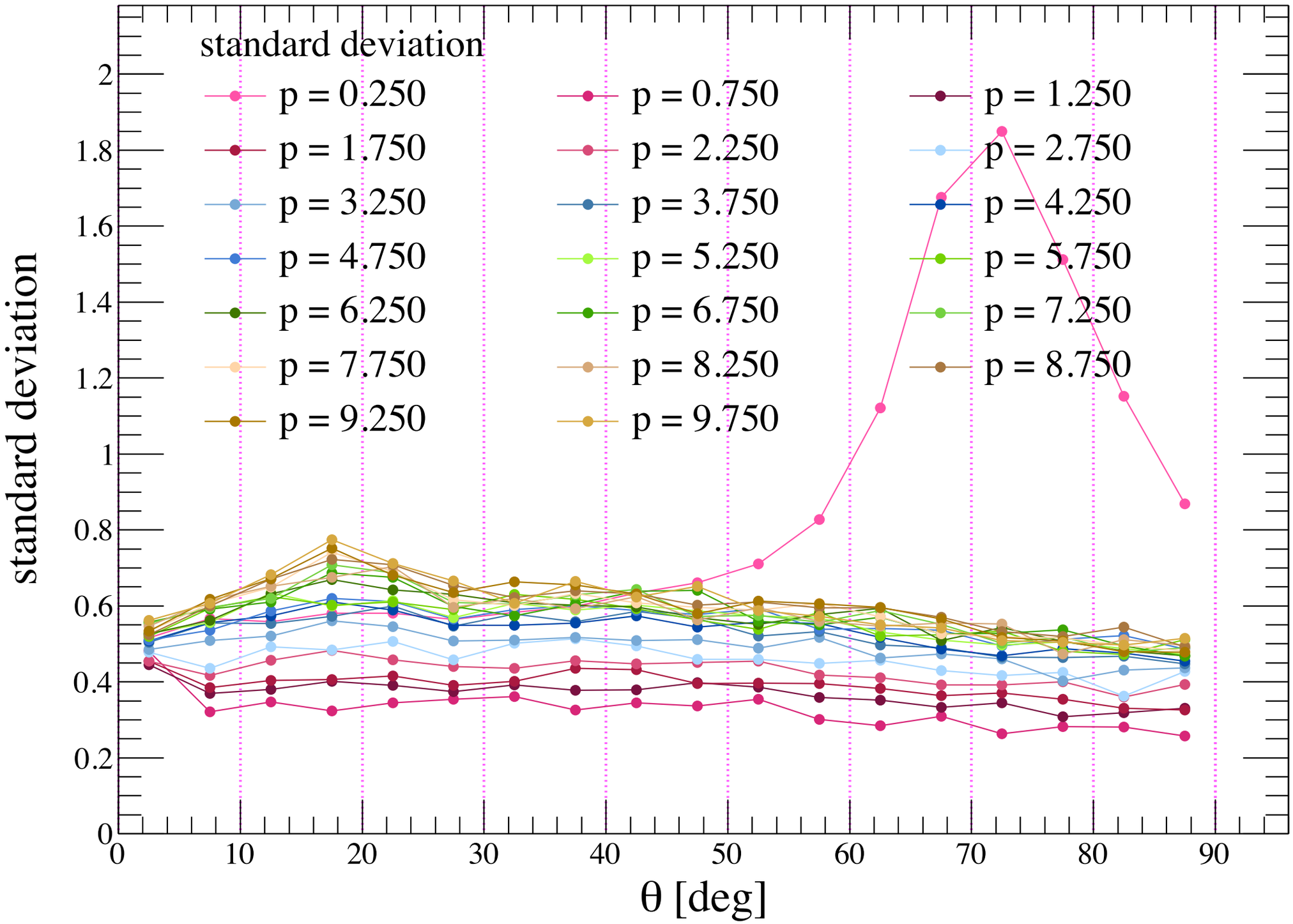
# Distributions

- mean number of tracks in each bin (20k generated)



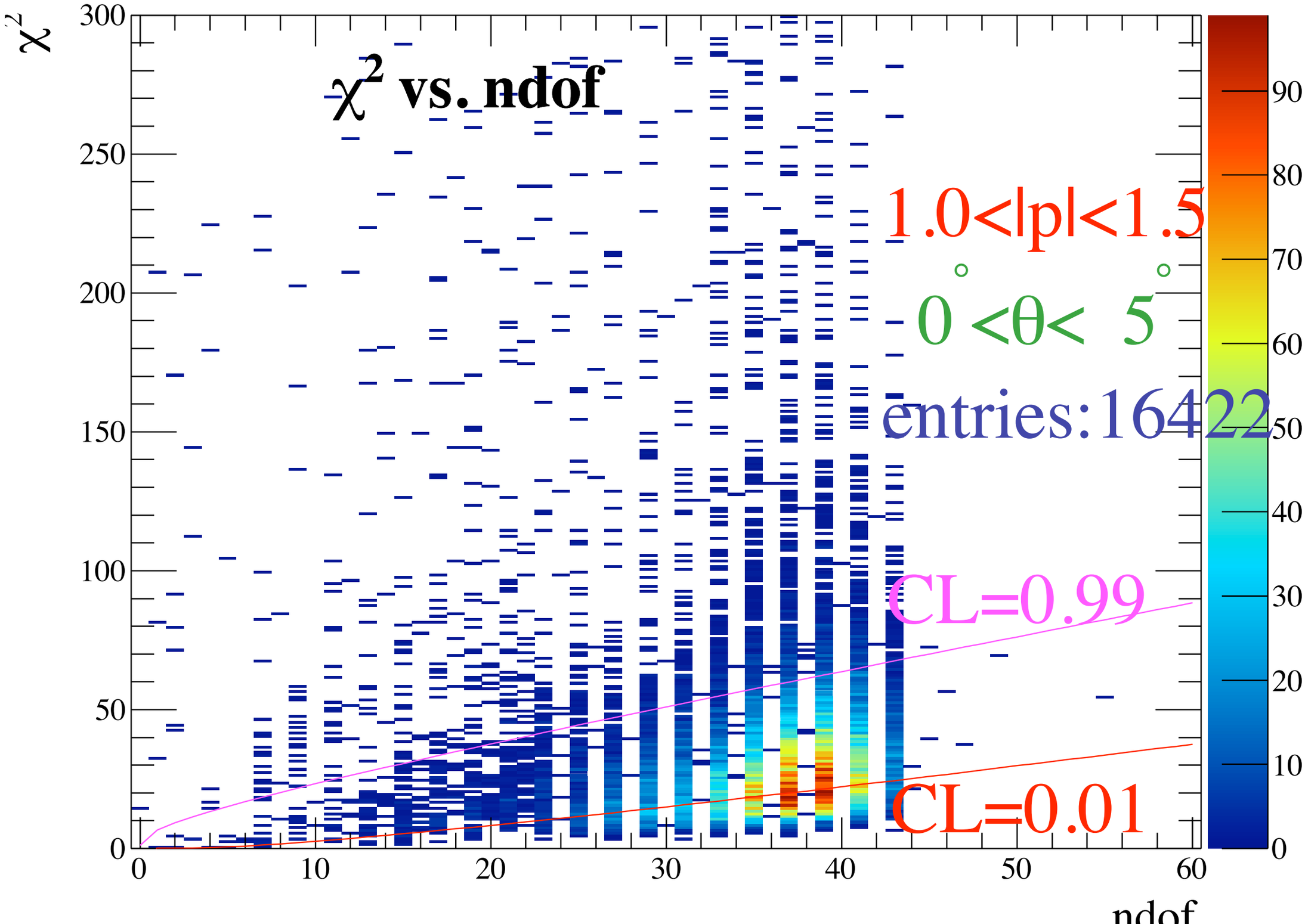
# Distributions

- standard deviation of number of tracks



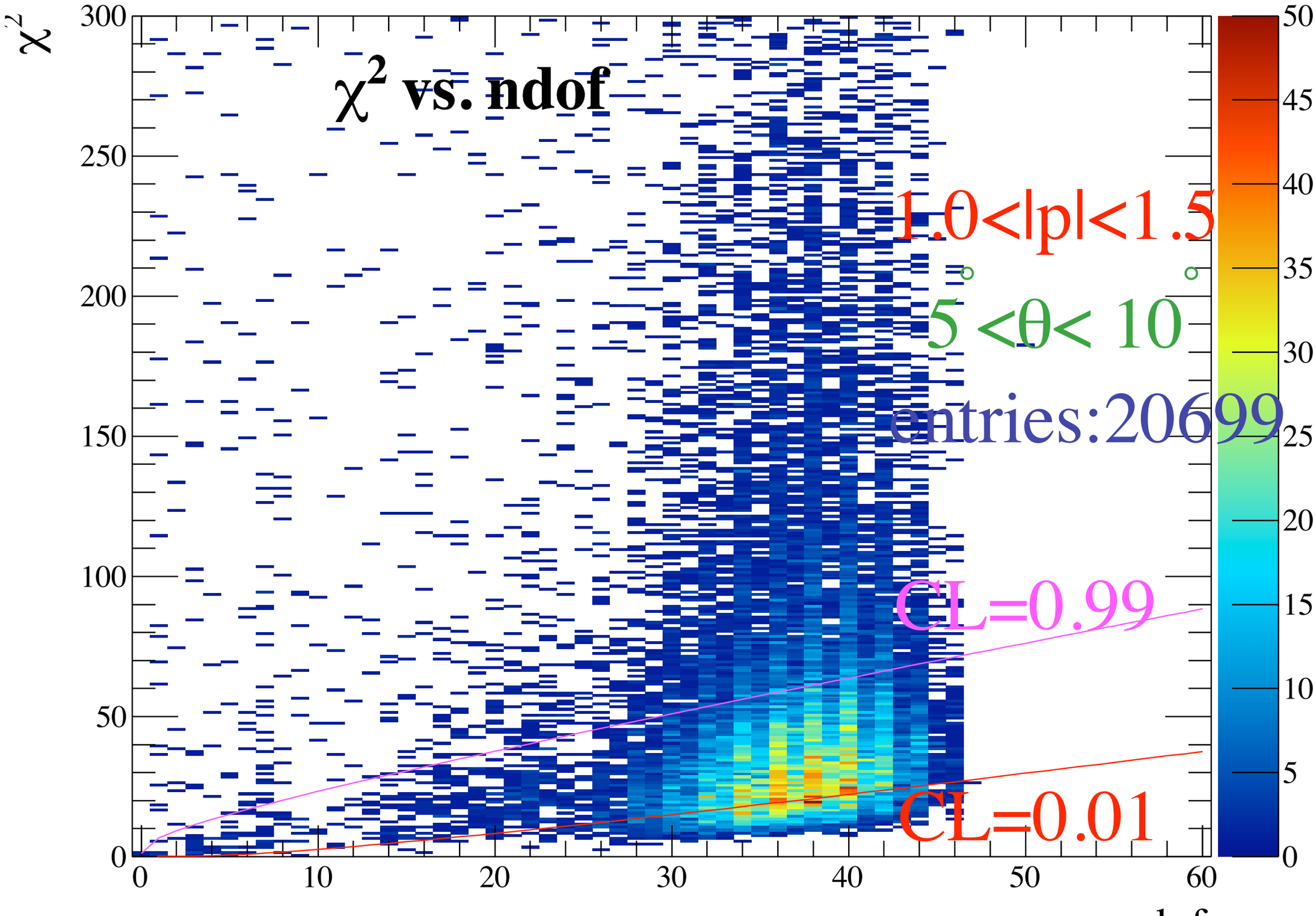
# Distributions

- $\chi^2$ vs. ndof for each track



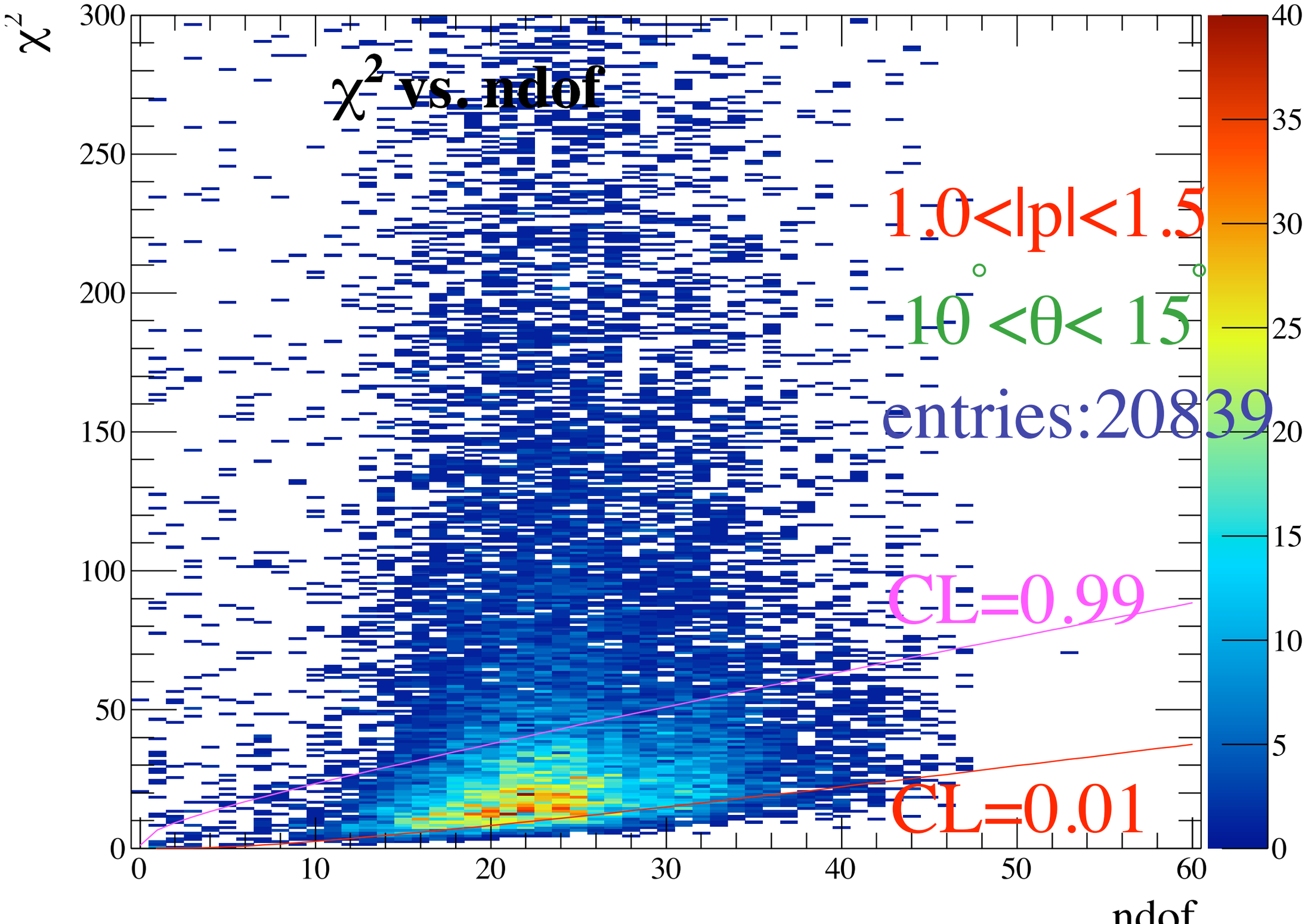
# Distributions

- $\chi^2$ vs. ndof for each track



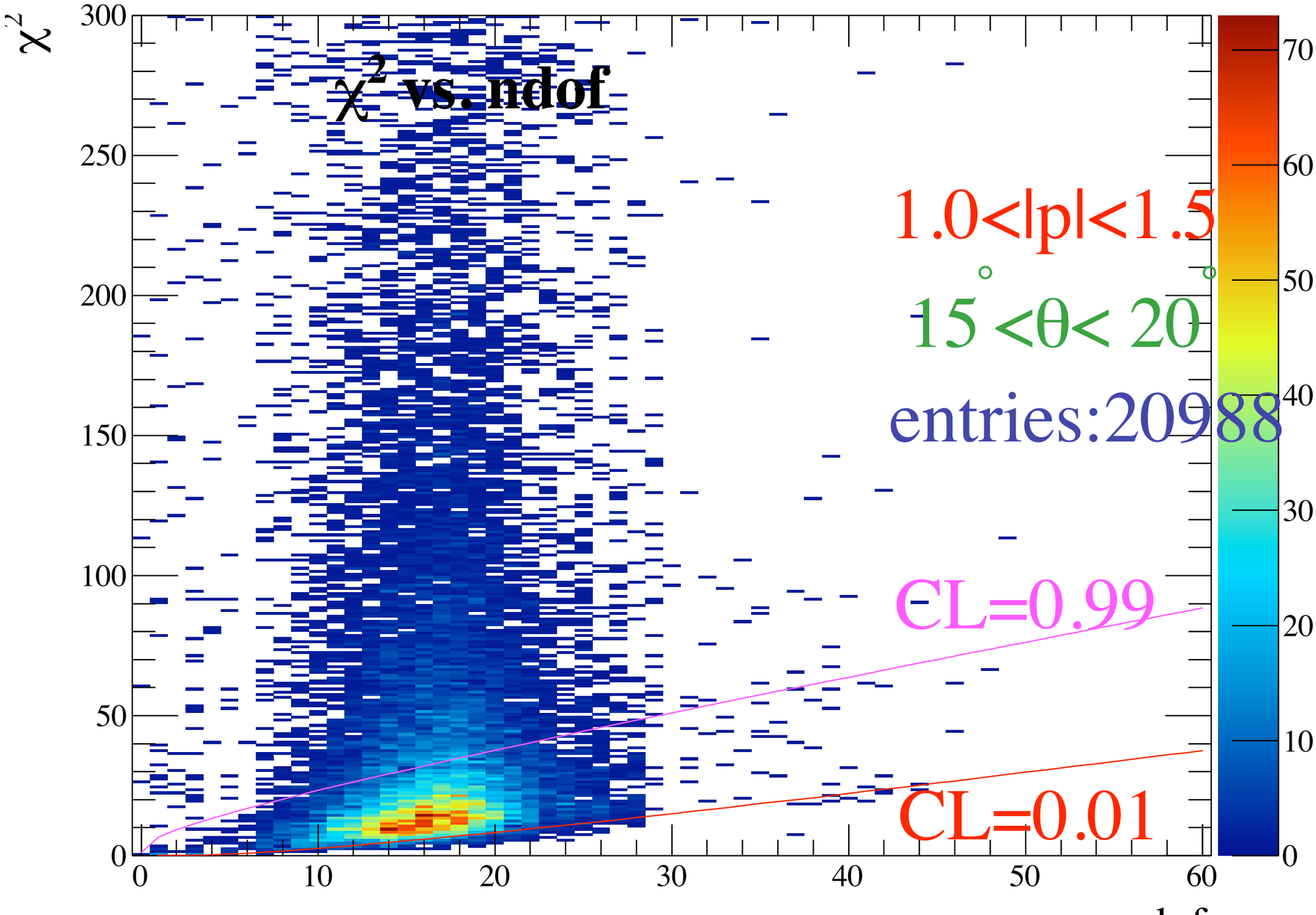
# Distributions

- $\chi^2$ vs. ndof for each track



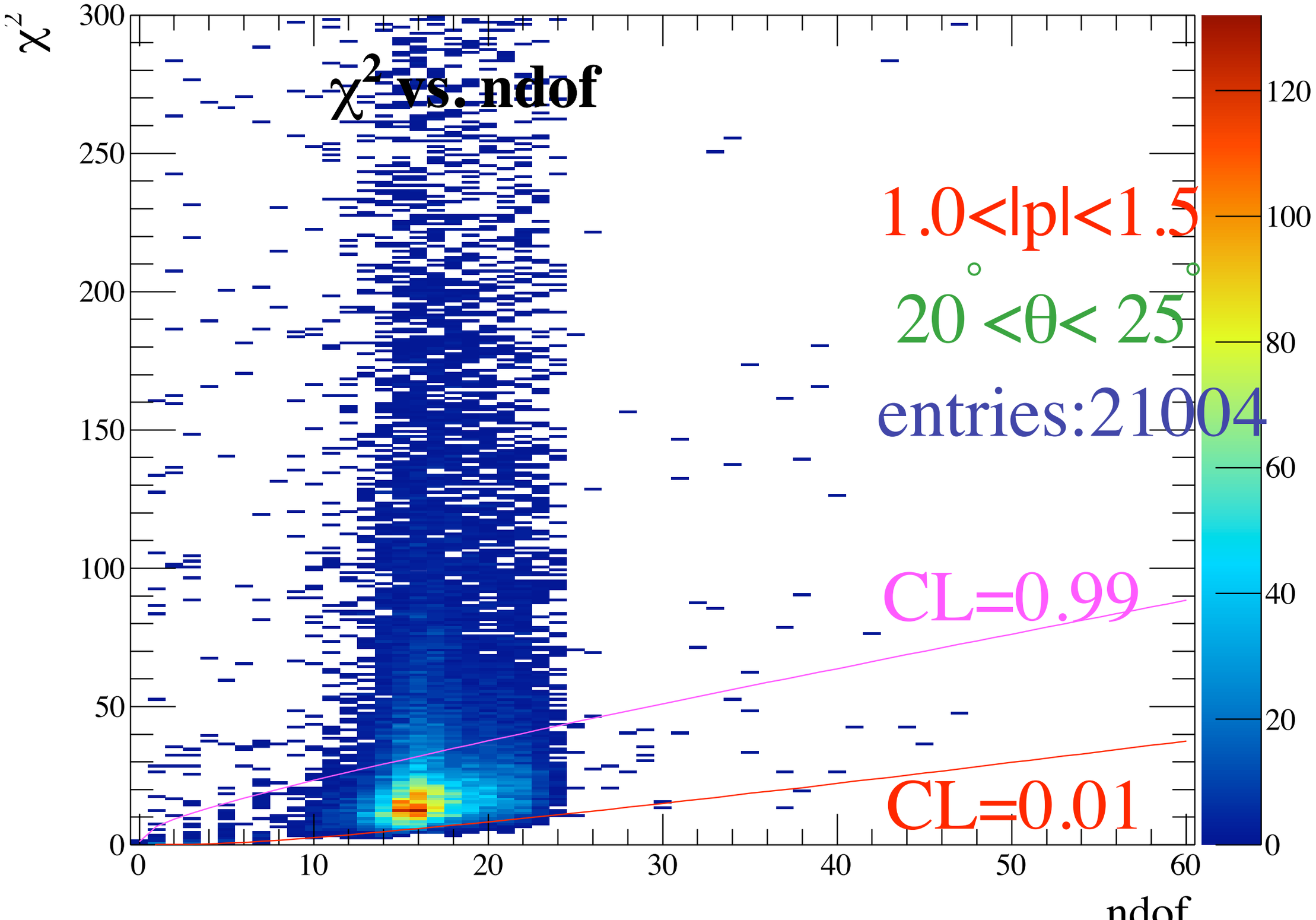
# Distributions

- $\chi^2$ vs. ndof for each track



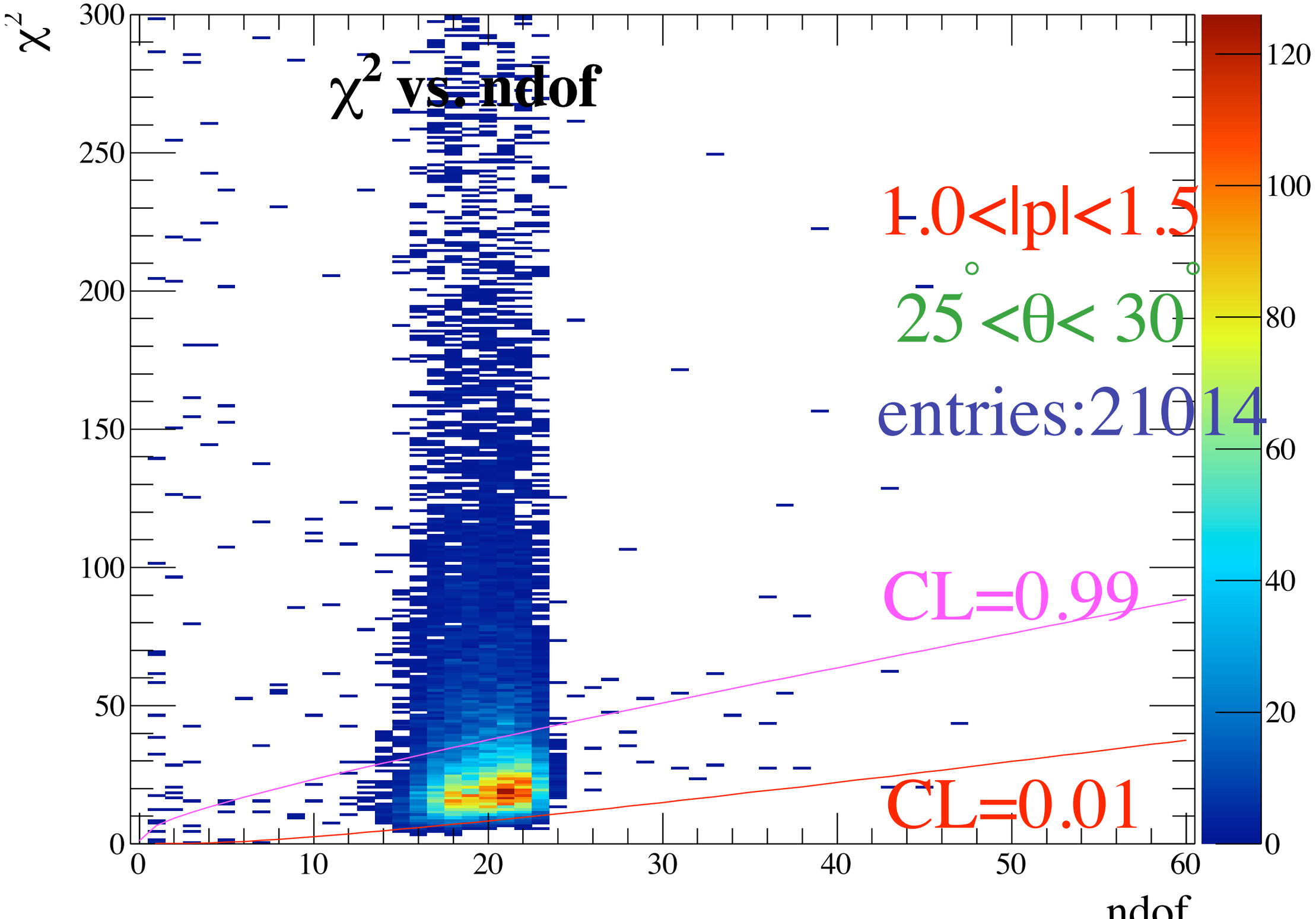
# Distributions

- $\chi^2$ vs. ndof for each track



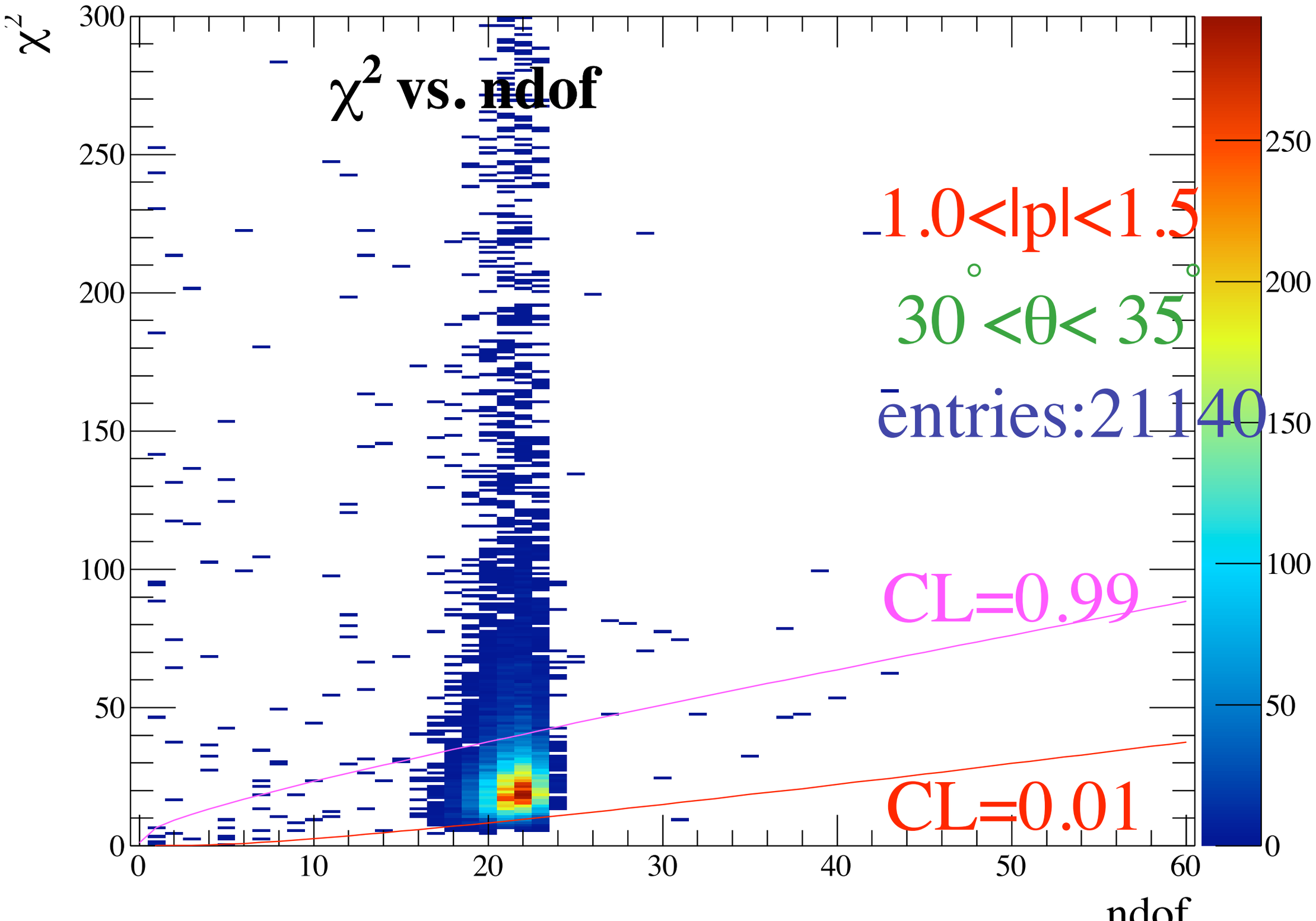
# Distributions

- $\chi^2$ vs. ndof for each track



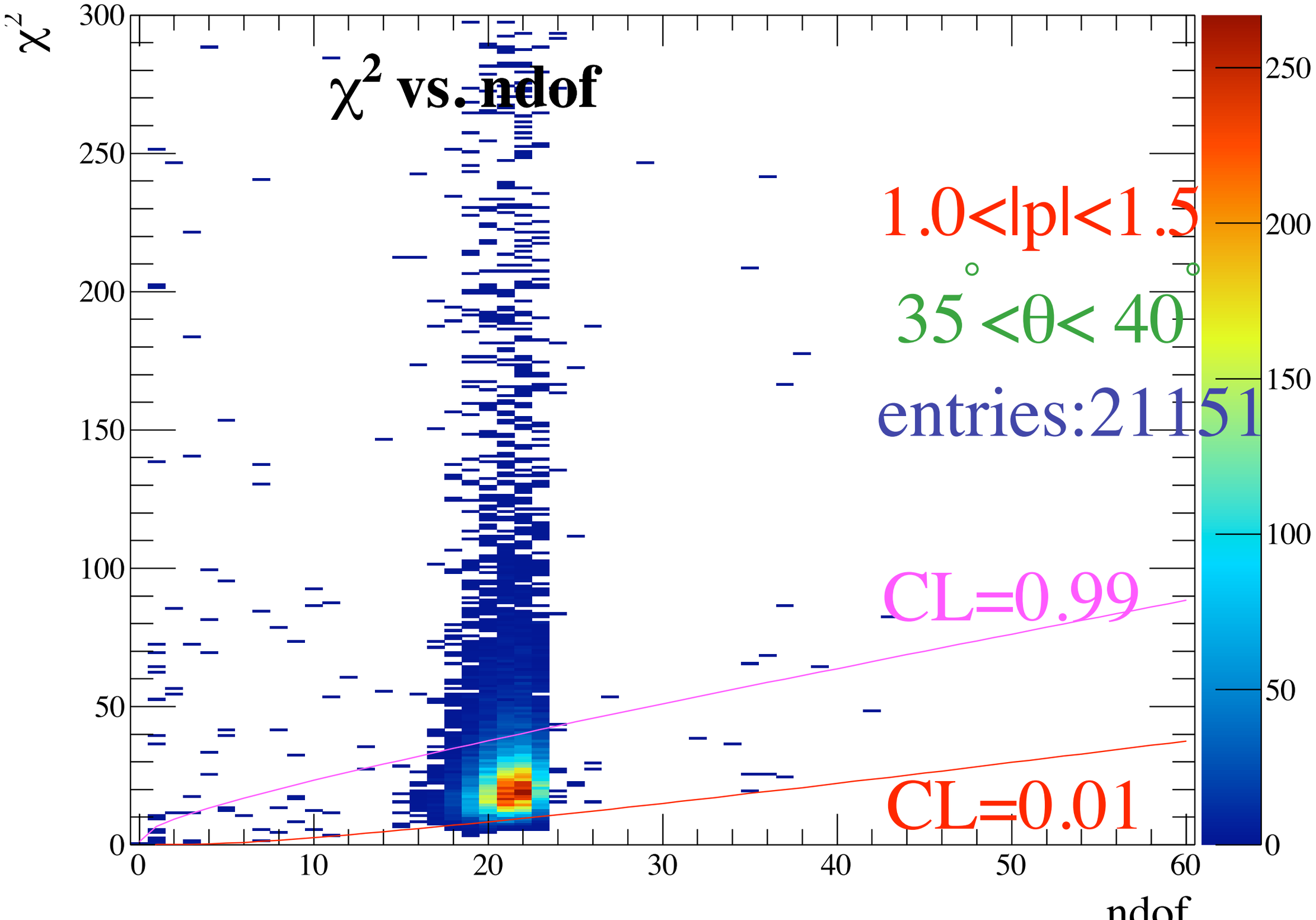
# Distributions

- $\chi^2$ vs. ndof for each track



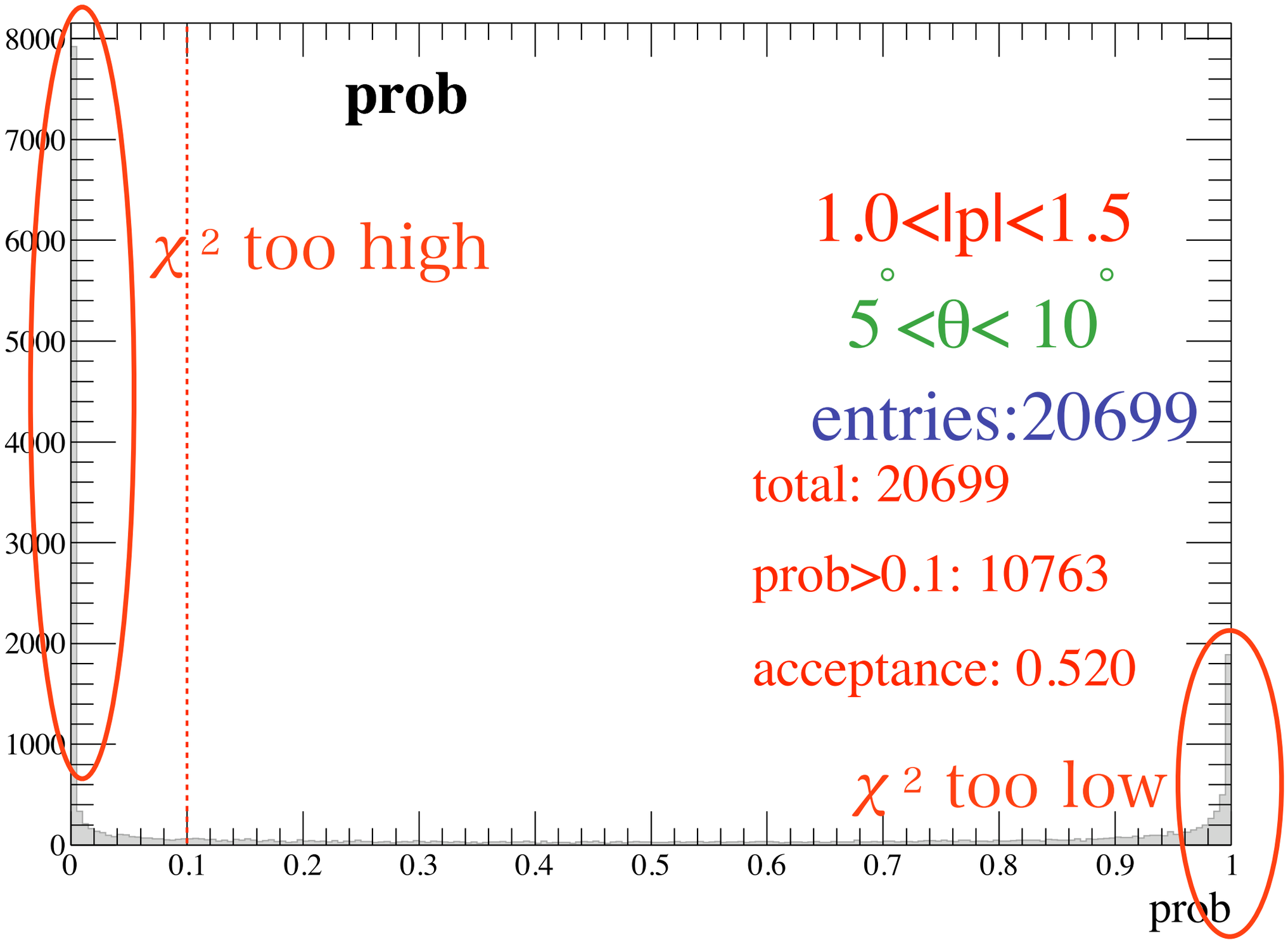
# Distributions

- $\chi^2$ vs. ndof for each track



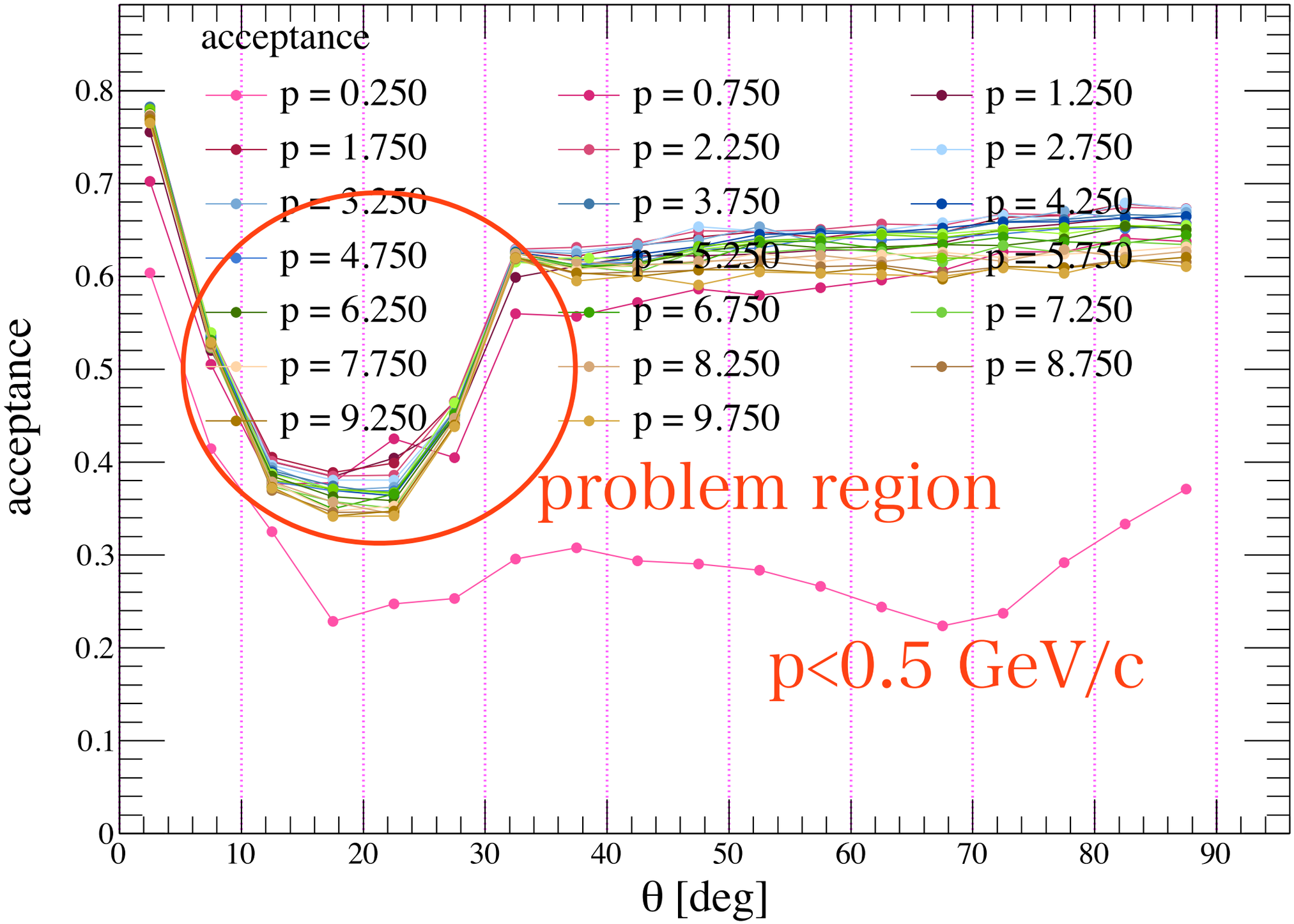
# Turn Into prob

- prob = TMath::Prob( $\chi^2$, ndof)



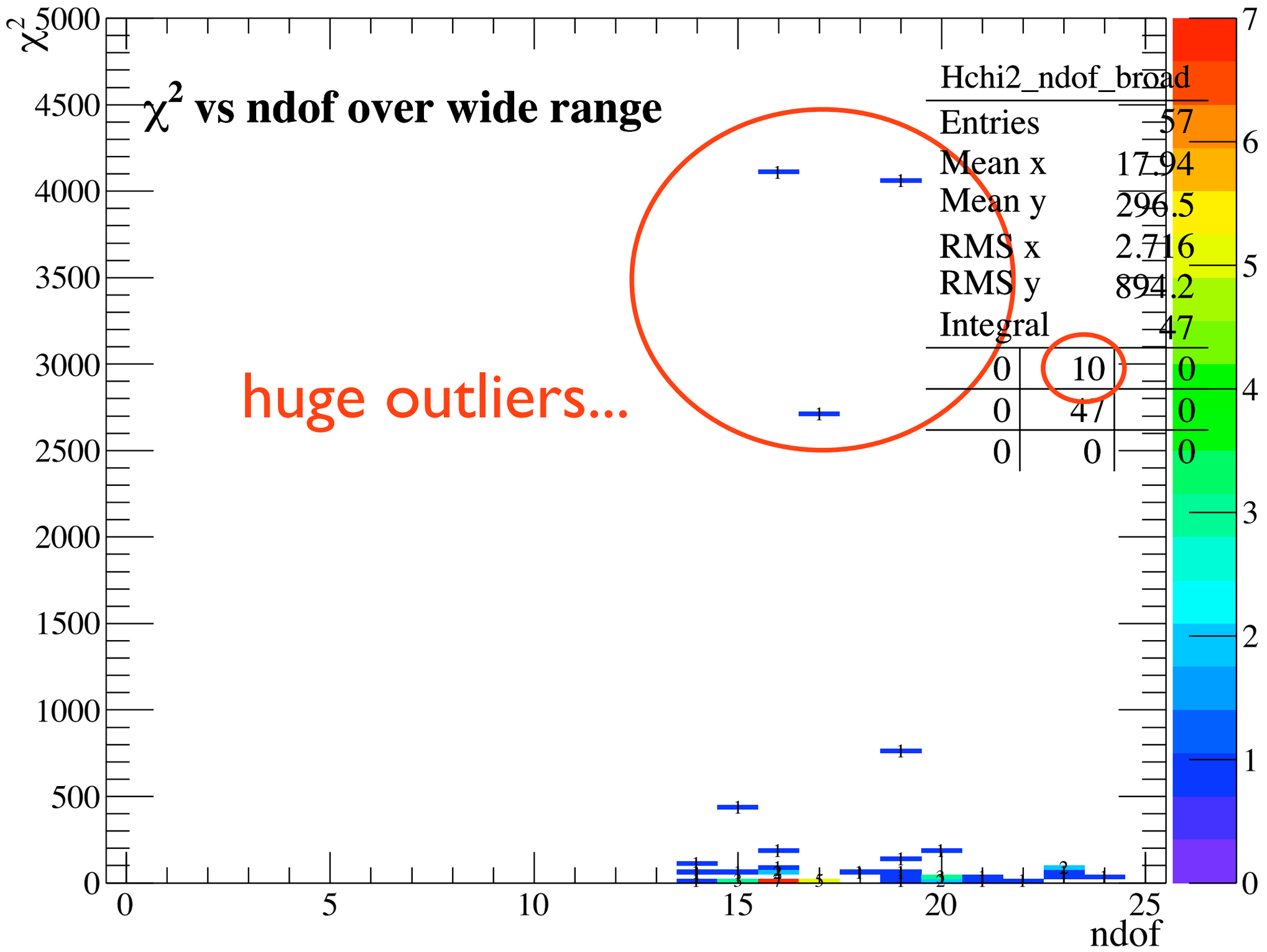
# Calculate Acceptance

- define acceptance as prob >0.10



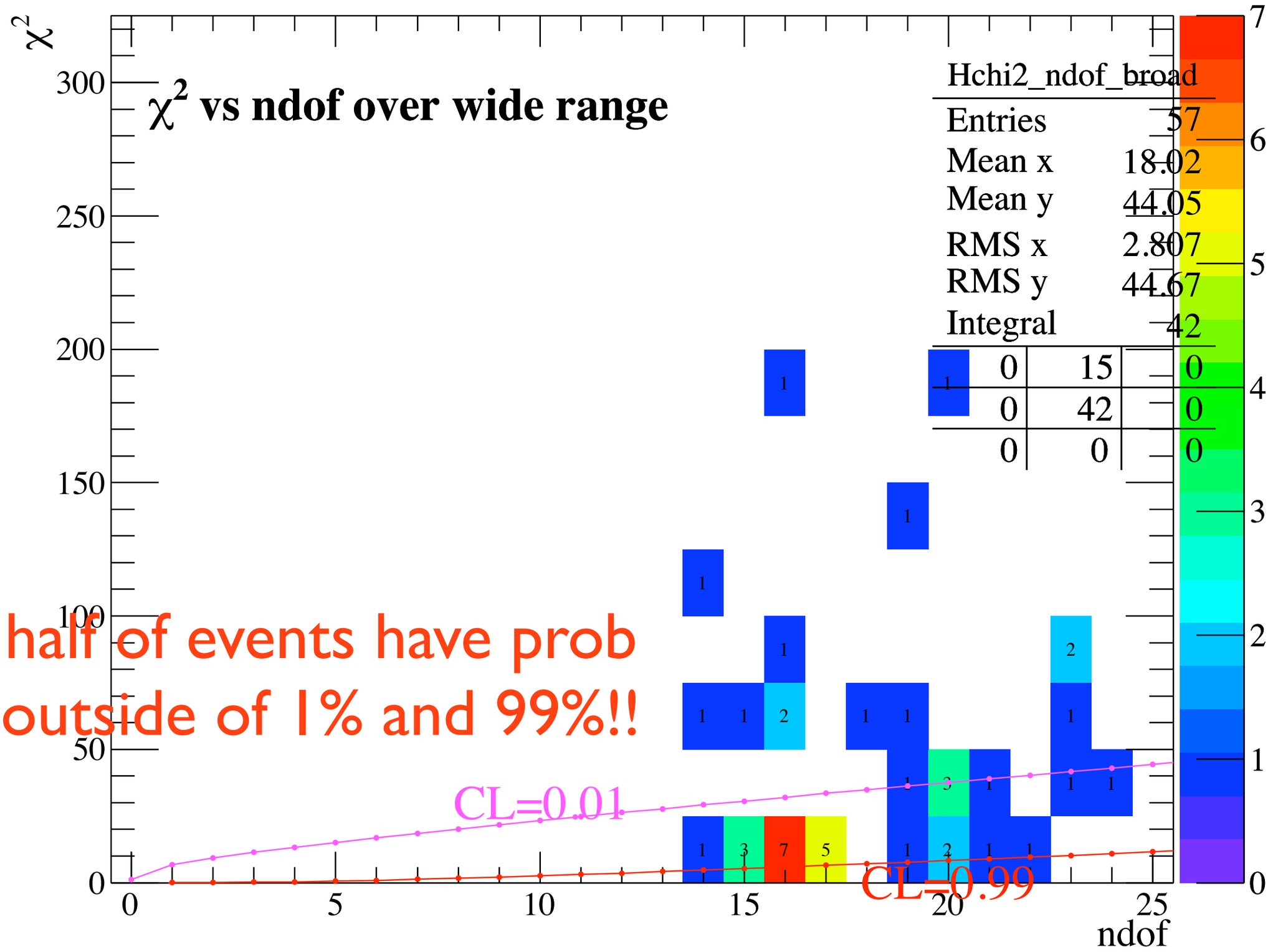
# Shockingly High $\chi^2$

- for $20^\circ<\theta<25^\circ$,



# Shockingly High $\chi^2$

- for 20°<θ<25°, closer look



# Improving Results

- want to improve acceptance
- want better $\chi^2$ values
- where does $\chi^2$ come from?
- have spent past few days understanding the Kalman filter, and the actual code that implements it

# Origin of $\chi^2$

- within analysis code:

```
vector< const DChargedTrack* > tracks;
loop->Get(tracks);
vector< const DTrackTimeBased* > hypo
        = tracks[i]->hypotheses;
Double_t chisq = hypo[j]->chisq;
Double_t ndof  = hypo[j]->Ndof;
```

# Origin of $\chi^2$

- within DTrackTimeBased_factory::evnt

DTrackFitter::fit_status_t status

= fitter->FindHitsAndFitTrack(*track, rt, loop,
track->mass(),mStartTime);

DTrackFitter*

fit_status = FitTrack(pos, mom,q, mass,t0);

DTrackFitter::fit_status_t status = FitTrack();

virtual function

# Origin of $\chi^2$

- within DTrackFitter_factory.h

  vector<const DTrackFitter*> fitters;

  loop->Get(fitters, "KalmanSIMD");

use DTrackFitterKalmanSIMD

- within DTrackFitterKalmanSIMD:: FitTrack()

  error = KalmanLoop();

almost there....

# Origin of $\chi^2$

- within DTrackFitterKalmanSIMD:: FitTrack()

error = KalmanLoop();

1.if there are FDC hits

error=KalmanForward(anneal_factor,S,C,

chisq,my_ndf);

vector of params

cov. matrix

2.if there are CDC hits and $\theta$ <60°

error=KalmanForwardCDC(anneal_factor,S,C,

chisq,my_ndf);

3.if there are CDC hits

error=KalmanCentral(anneal_factor,Sc,Cc,pos,

chisq_central,my_ndf);

# Origin of $\chi^2$

- within DTrackFitterKalmanSIMD:: KalmanForwardCDC()
- loop over CDC hits{
  - get J (Jacobian), Q (noise cov. matrix), S0 from kth CDC hit

set within PropagateForwardCDC

called within SetCDCForwardReferenceTrajectory

GetProcessNoise(...,Q)

Lynch-Dahl algorithm

StepJacobian(...,J)

CalcJacobian(...,$J_1$)

$J=\Delta z * J_1$

  - $S = S0 + J\ (S-S0)$
  - $C = Q + J\ C\ J^T$
  - $dx = S.x\text{-}wire.x,\ dy = S.y\text{-}wire.y$
  - $doca = sqrt(dx^2+dy^2)$

# Origin of $\chi^2$

- within DTrackFitterKalmanSIMD:: KalmanForwardCDC()
- loop over CDC hits{
  - if(doca doesn't keep decreasing){
    - calculate new dz
    - Step(z,newz=z+dz,dedx,S)
    - StepJacobian(z,newz,S0,dedx,J)

J=J+Δz * $J_1$, $J_1$ set in CalcJacobian

    - C = J C $J^T$
    - define H (track projection matrix) 1x5
    - $H_x$=dx $\cos^2\theta_{stereo}$/sqrt($dx^2$+$dy^2$), same for $H_y$

# Origin of $\chi^2$

- within DTrackFitterKalmanSIMD:: KalmanForwardCDC()
- loop over CDC hits{
  - if(doca doesn't keep decreasing){
    - dm = $v_{CDC}$ [(cdchit->tdrift)-(vertex time)
      -(t of kth hit)]
    - V = $\sigma^2_{CDC}(dm) + v_{CDC}^2\ \sigma_{t0}^2$
    - InvV = 1 /(V + $H^T$CH)
    - K (Kalman gain) = InvV * (C H)
    - S = S + (dm-d) K
    - C = C - K H C

# Origin of $\chi^2$

- within DTrackFitterKalmanSIMD:: KalmanForwardCDC()
- loop over CDC hits{
  - if(doca doesn't keep decreasing){
    - res_scale = 1 - H K = V/(V+HCH$^T$)
    - res = (dm-d) * res_scale
    - $\Delta$chisq = anneal * res$^2$ / V / res_scale
      = anneal * (dm-d)$^2$ /(V+HCH$^T$)
    - ndof++
    - anneal is a factor currently set to 1

# Origin of $\chi^2$

- within DTrackFitterKalmanSIMD:: KalmanForwardCDC()
- loop over CDC hits{
  - if(doca doesn't keep decreasing){
    - StepJacobian(newz,z,S0,dedx,J)
    - $C = J\ C\ J^T$
    - Step(newz,z,dedx,S)
    - calculate new doca
  - }
  - set kth CDC state S and cov. matrix C
- }

# Conclusion

- looking into Kalman filter code
- looking for where $\chi^2$ values are too small/too large
- may need some help in understanding Kalman filtering in general