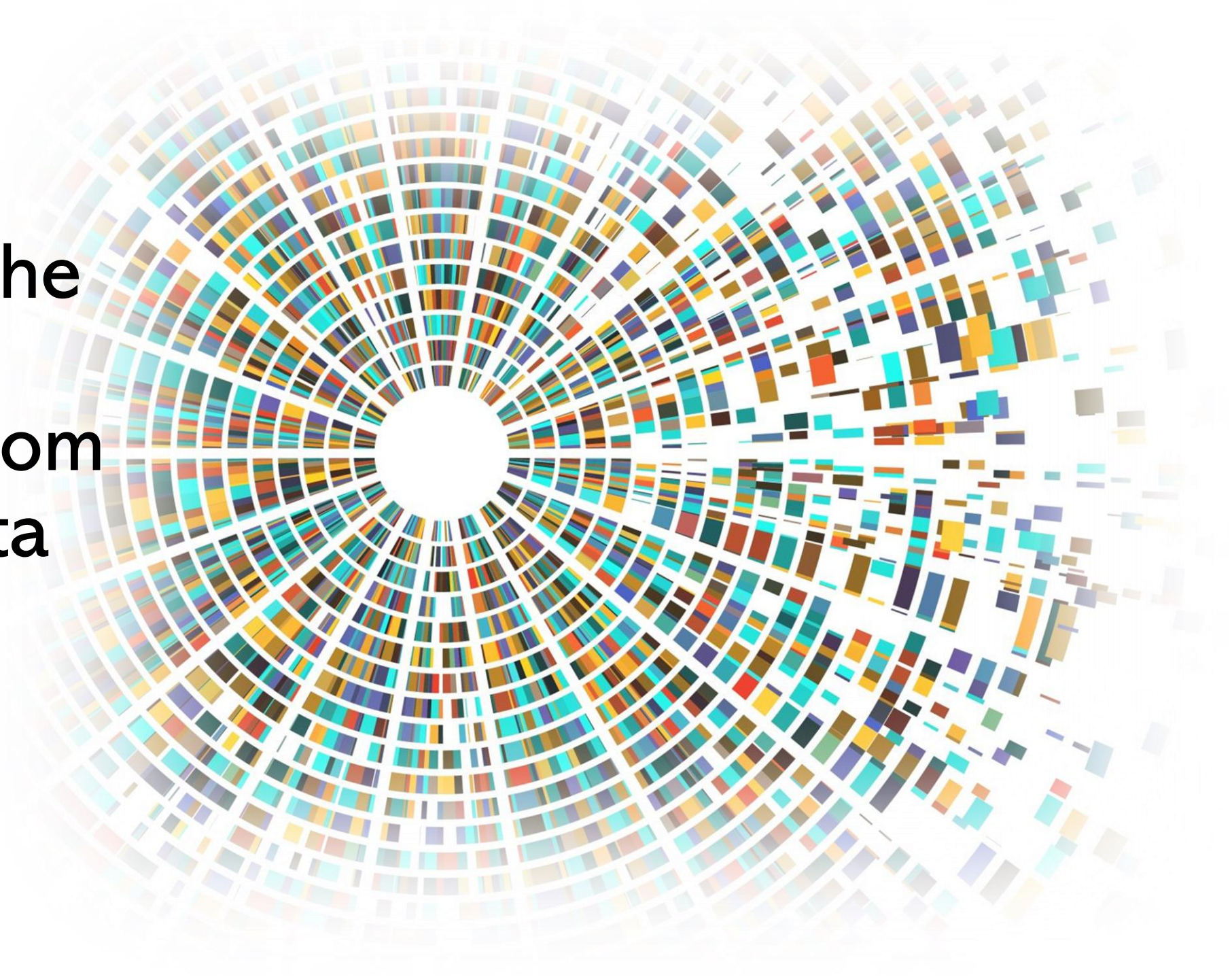
Calculating the Helicity Asymmetry from Cooked Data



By Derek Holmberg

Measuring the Helicity Asymmetry

- **Goal**: determine the minimum number of electrons needed to well-constrain the target polarization
- The number of electrons needed to do so is constrained by seeing how many electrons are needed to produce stable helicity asymmetry measurements

$$A = \frac{N^+ - N^-}{N^+ + N^-}$$

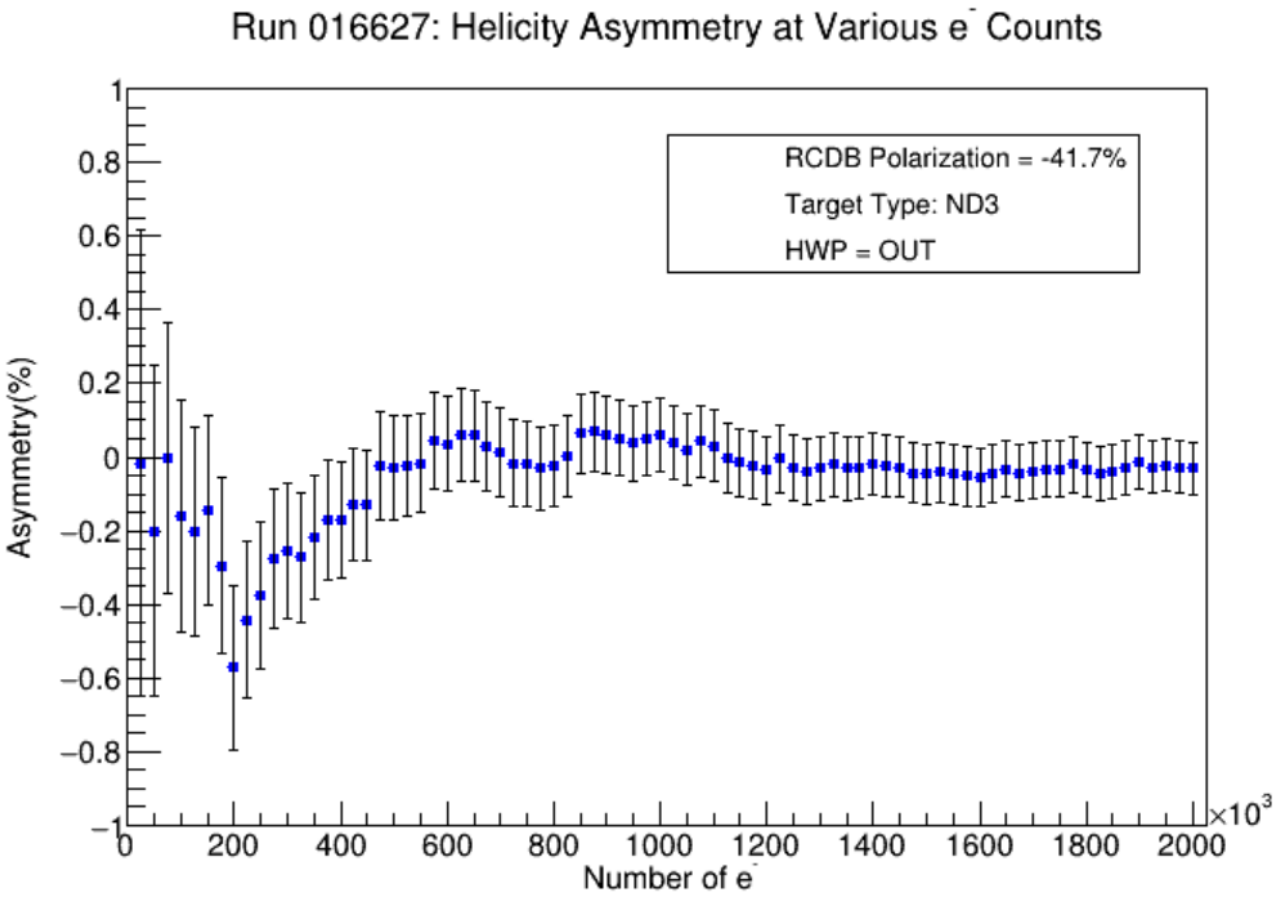
- where $N^{+,-}$ are the counts of the helicity up/down measurements and A is the calculated asymmetry
- Calculate and plot the asymmetry for different numbers of electrons measured

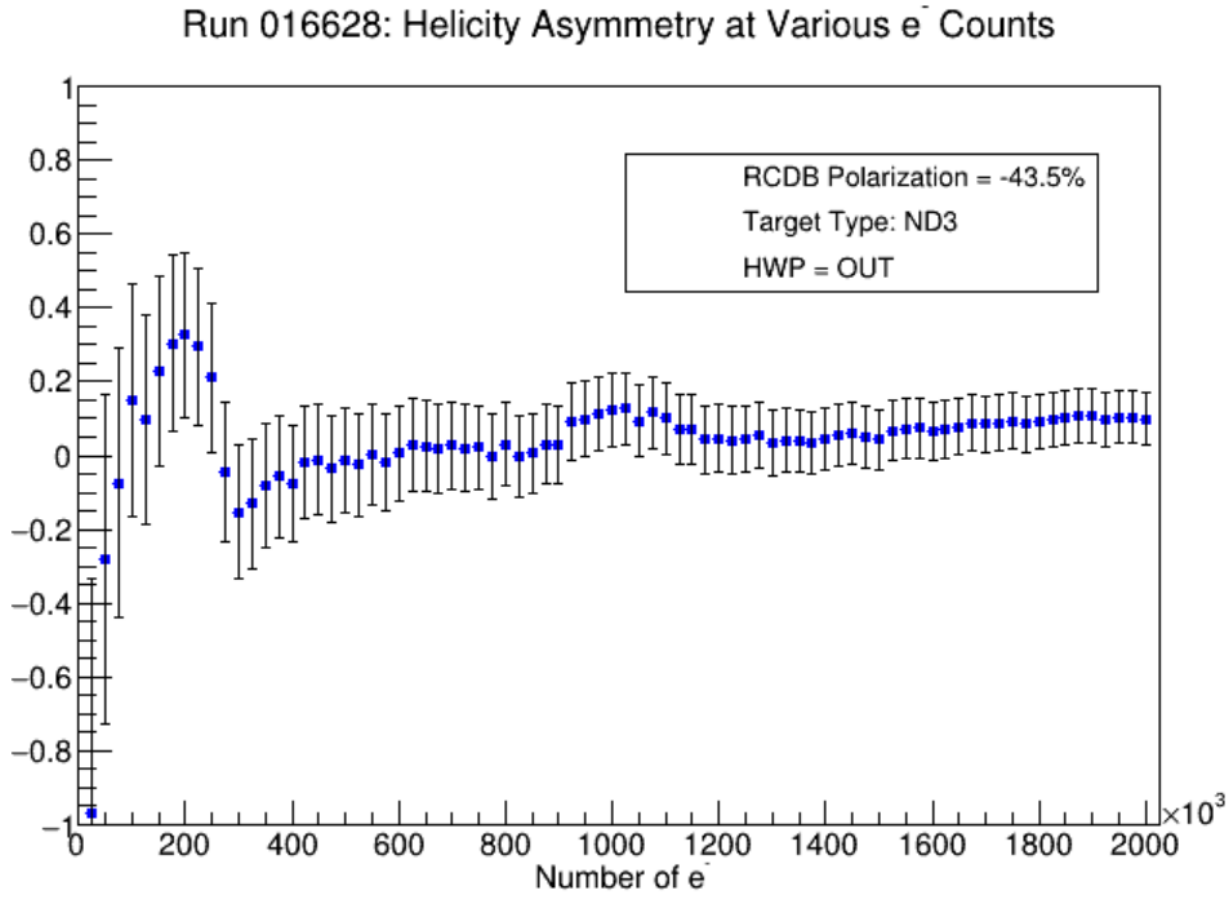
Description of the Plotting Program

1. Choose a run that's already been cooked and set how many HIPO files to loop through (500 for large runs; all files if less than 500)
2. Loop through the file event by event, and if the first particle listed in the REC::Particle list **is an electron**, and the **helicity is defined** (i.e. hel = +/-1)
    1. Log the helicity and add to the e^-count
    2. If the e^- count is a multiple of 25,000 calculate and log asymmetry
3. Once 2,000,000 electrons (with a defined helicity) have been measured, finish the loop (or finish if all files have been looped through)
4. Plot all calculated asymmetries vs. electron count

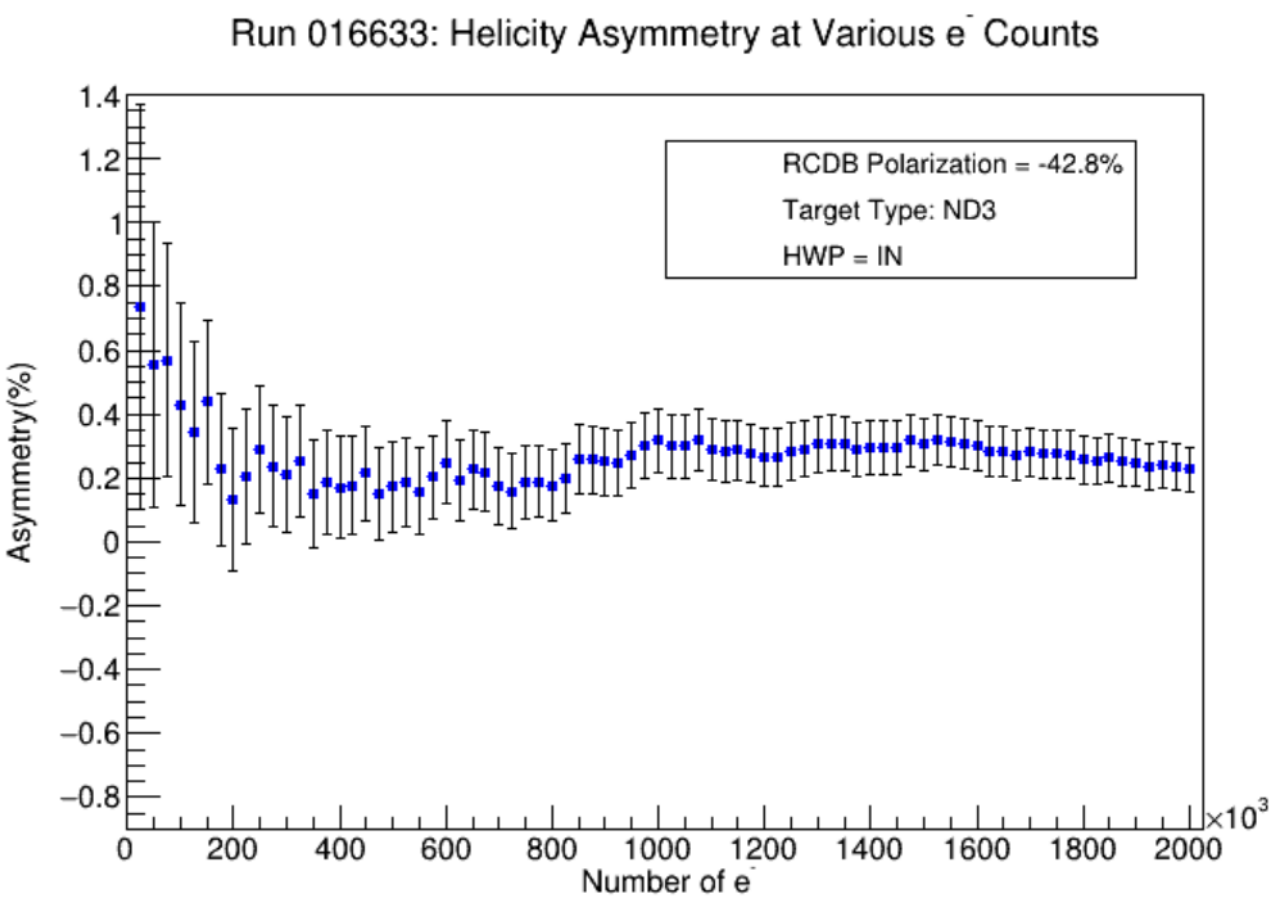
Note: helicity measurements are obtained from the REC::Event bank, specific code in backup slides

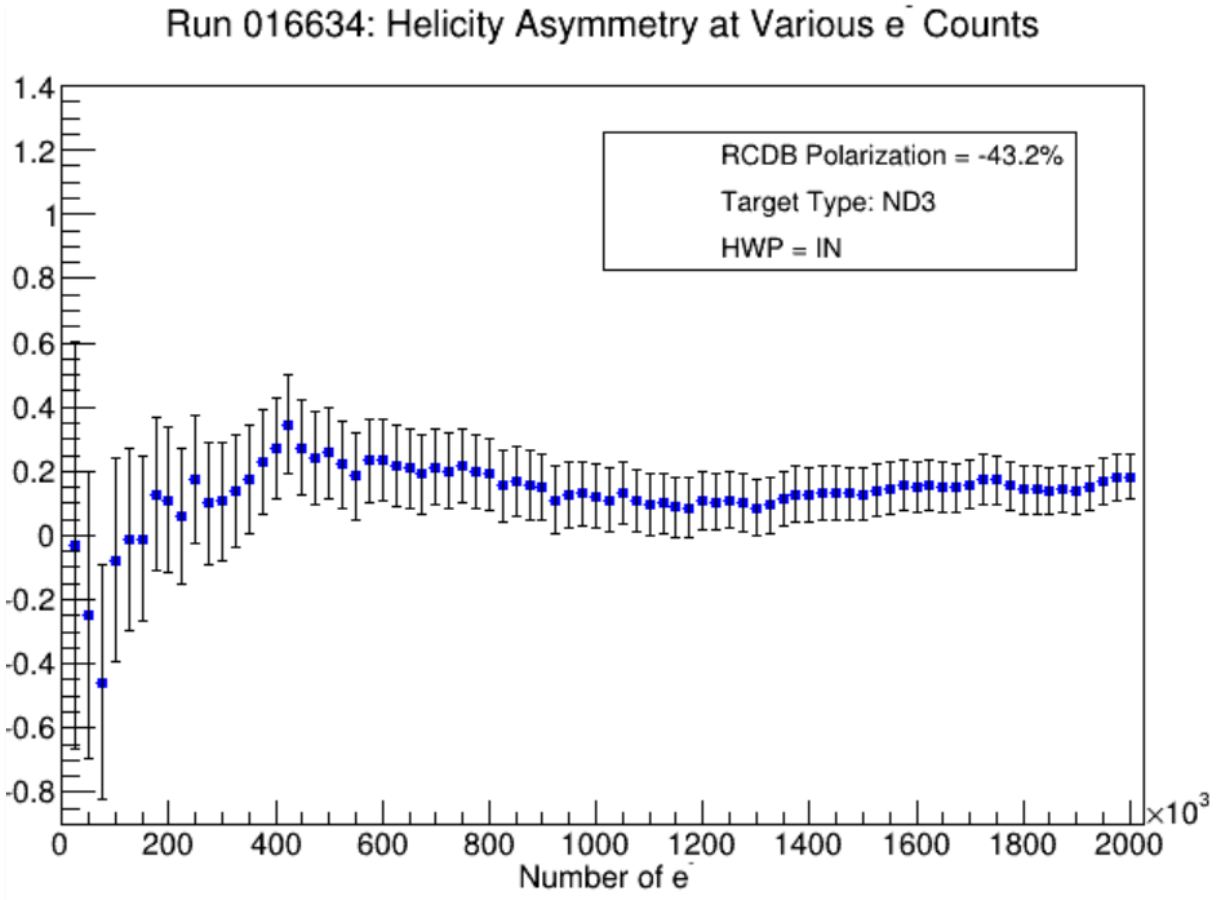
Run 16627 & 16628, HWP out, ND3



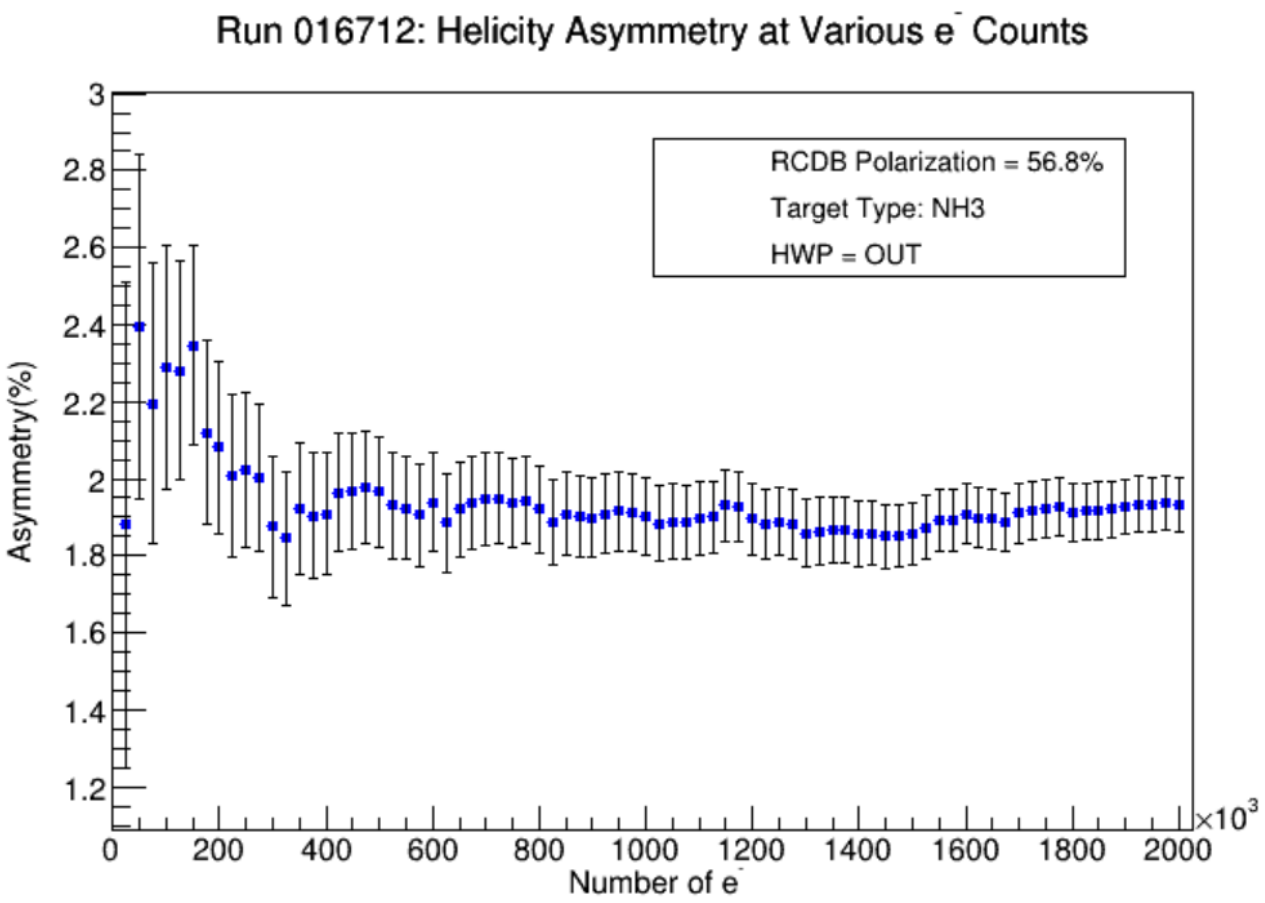


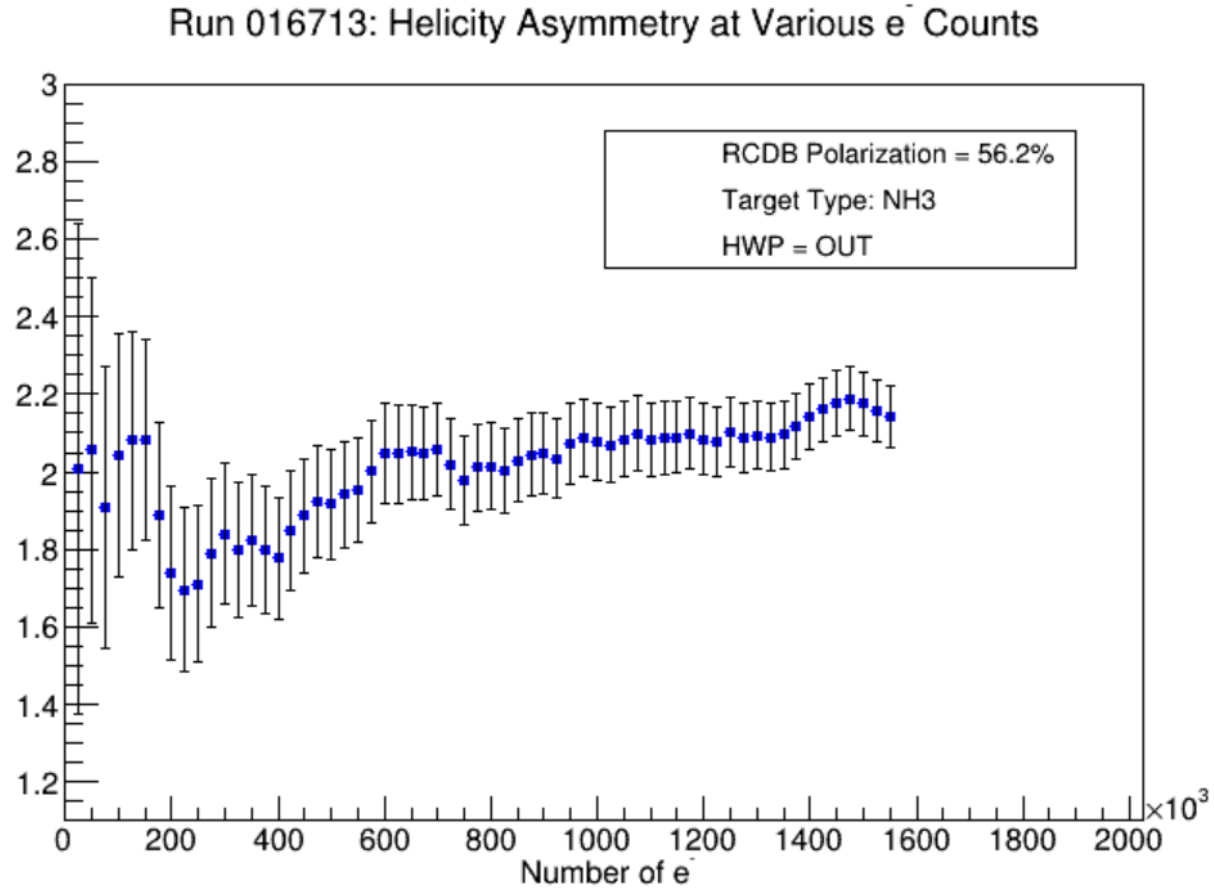
Run 16633 & 16634, HWP in, ND3



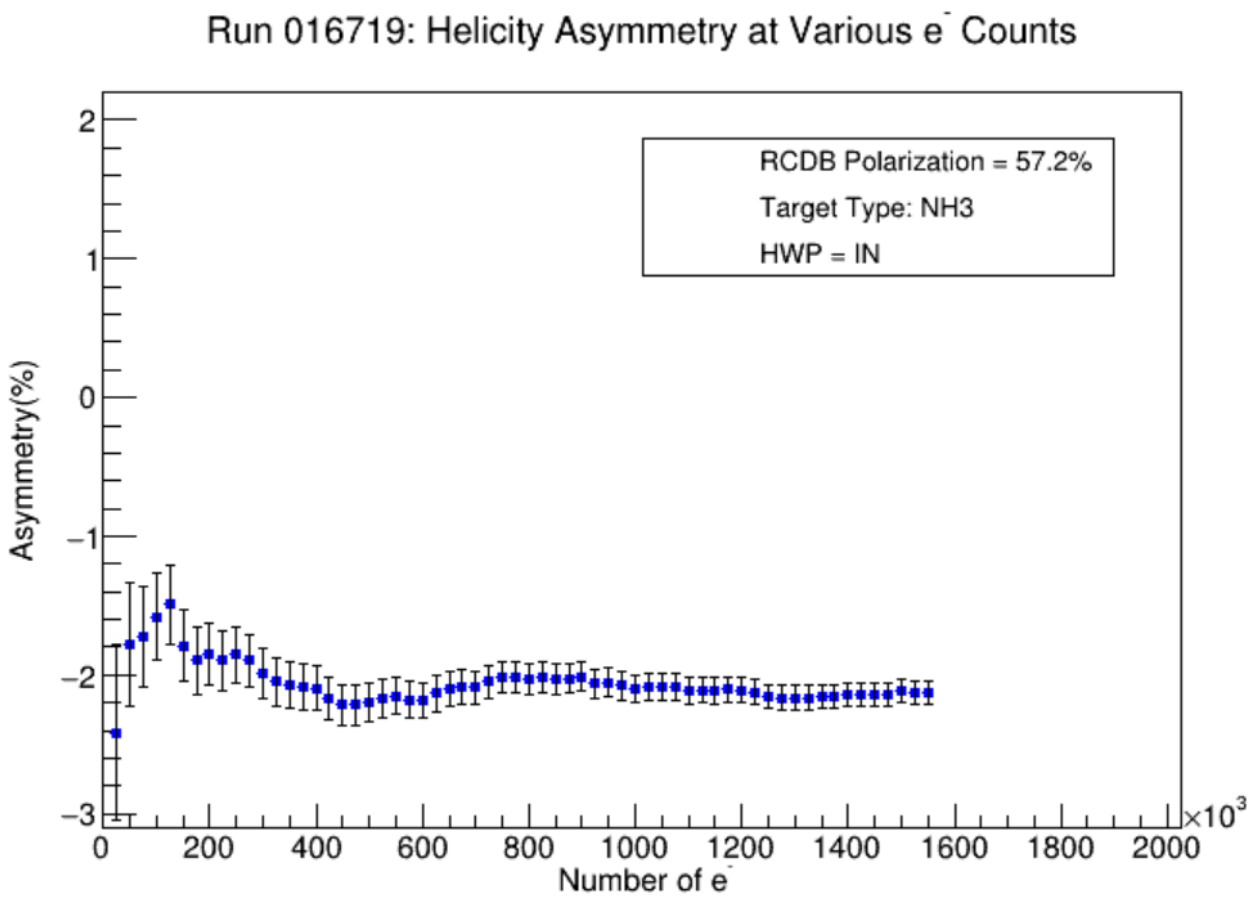


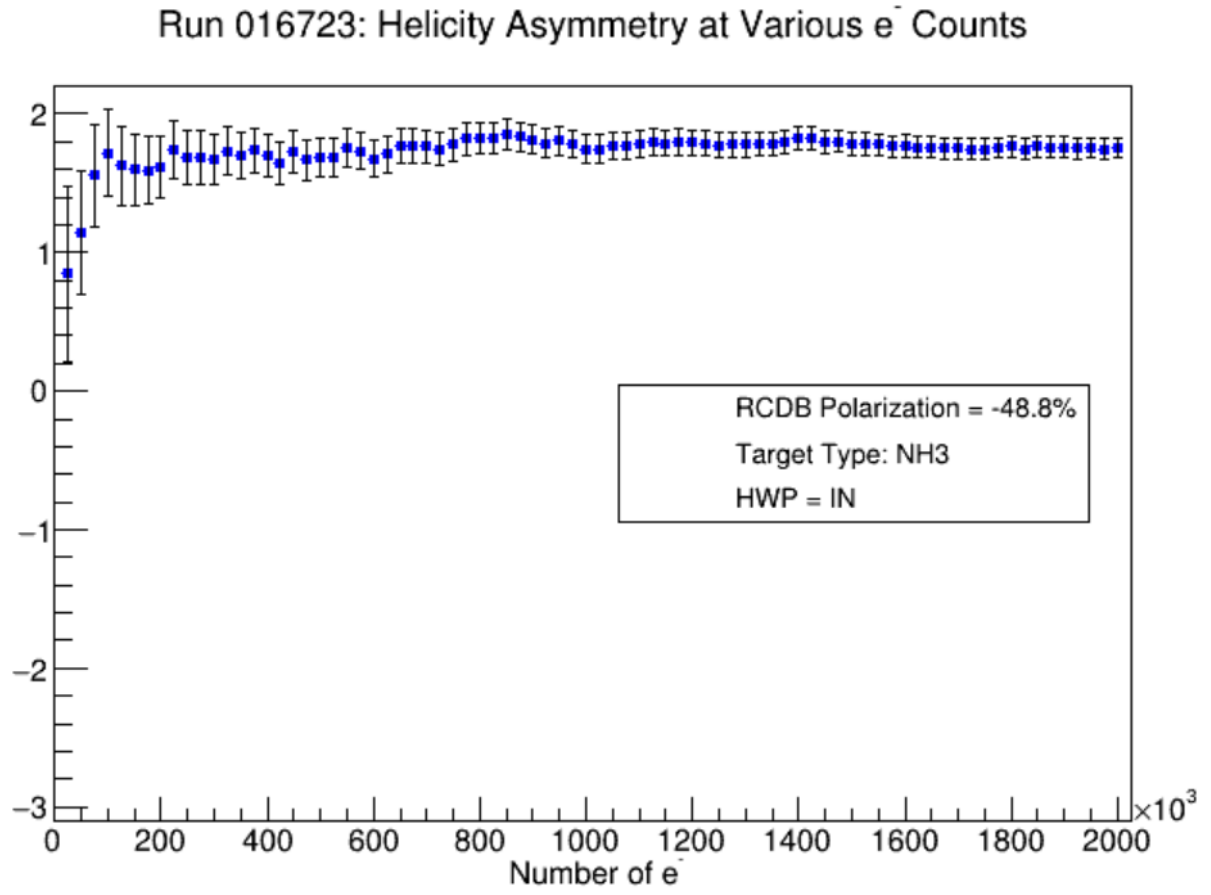
Run 16712 & 16713, HWP out, NH3





Run 16719 & 16723, HWP in, NH3





Outlook

1. Follow Rafaella's suggestion to plot the calculated asymmetries as functions of the electron momentum or the W of the interaction
2. Calculate the target polarization from the kinematics of the files to see how closely it compares to the RCDB value

Double-Checking the Code (Backup Slides)

```
// This stuff just declares all the HIPO stuff to read the HIPO files
hipo::reader  reader;
reader.open(file_name.c_str());
hipo::dictionary  factory;
reader.readDictionary(factory);
hipo::structure  particles;
hipo::structure  detectors;
hipo::event      event;
int counter = 0;
hipo::bank  dataHEL;
hipo::bank  dataPART;
hipo::bank  dataCALO;

hipo::bank PART(factory.getSchema("REC::Particle"));
hipo::bank EVENT(factory.getSchema("REC::Event"));
```

```
while(reader.next()==true){
        // This stuff loads in the events?
        reader.read(event);
        event.getStructure(PART);
        event.getStructure(EVENT);

        int nrows = PART.getRows();
                        for(int i = 0; i < 1; i++){

                int   pid = PART.getInt("pid",i); // This grabs the particle ID
                int   hel = EVENT.getInt("helicity",0); // This gets the event helicity
                // This loop only bins the helicity if the first particle in the REC::Particle
                // bank is an electron
                if(pid == 11){
                        // This makes sure only defined helicites of +/-1 are binned
                        if(abs(hel) == 1){
                                hHEL->Fill(hel);
                                e_count++;
                                if(hel == 1){hel_up++;}
                                else if(hel == -1){hel_down++;}
                                if(e_count == A_counts[iterator_A_count]){
                                        double a = Polarization(hel_up,hel_down);
                                        Pol_Vals.push_back(a);
                                        cout << "Polarization at "<<e_count<<" electrons: ";
                                        cout << a << endl;
                                        iterator_A_count++;
                                        Used_A_counts.push_back(e_count);
                                }
                        }
```