

Multiple Subrun MC POT Counting Bug Fix

Lauren Yates

LArSoft Coordination Meeting

September 25, 2018



Description of the Bug



- MicroBooNE has recently started generating MC files with multiple subruns in a file
 - ▶ Why? Briefly — MicroBooNE is moving towards using cosmic data overlay on neutrino MC, instead of simulated cosmic rays; MC events inherit the (run, subrun, event) from the overlaid cosmic data event; some data files have more than one subrun per file
- For MC files with multiple subruns, the POT value reported by the `generator sumdata::POTSummary` data product is incorrect
- For the n^{th} subrun in the file, the POT reported is actually the POT for all subruns 1 through n

Description of the Bug

- At the end of each subrun in the file, the `endSubRun` function in larsim's `GENIEGen_module.cc` is called
- This fills the `sumdata::POTSummary` data product based on the `TotalExposure` variable from the `GENIEHelper` object

```
void GENIEGen::endSubRun(art::SubRun& sr)
{
    std::unique_ptr<sumdata::POTSummary> p(new sumdata::POTSummary());

    p->totpot = fGENIEHelp->TotalExposure();
    p->totgoodpot = fGENIEHelp->TotalExposure();

    sr.put(std::move(p));

    return;
}
```

Description of the Bug

- The GENIEHelper class defined in nutools's GENIEHelper.cxx
- The TotalExposure variable is set to 0 when GENIEHelper object is initialized, but is never reset after that
 - ▶ Only ever updated in Stop function, which adds POT from each event

```
void GENIEHelper::Initialize()
{
    // set the pot/event counters to zero
    fSpillEvents = 0;
    fSpillExposure = 0.;
    fTotalExposure = 0.;
}
```

```
bool GENIEHelper::Stop()
{
    // determine if we should keep throwing neutrinos for
    // this spill or move on
    // ...
    if(fFluxType.compare("atmo_FLUKA") == 0 || fFluxType.compare("atmo_BARTOL") == 0 ||
        fFluxType.compare("atmo_MINIBALL") == 0 || fFluxType.compare("atmo_HAKKM") == 0 ){
        //the exposure for atmo is in SECONDS. In order to get seconds, it needs to
        //be normalized by 1e4 to take into account the units discrepancy between
    }
    else{
        fTotalExposure += fSpillExposure;
    }

    fSpillEvents = 0;
    fSpillExposure = 0.;
    fHistEventsPerSpill = fHelperRandom->Poisson(fXSecMassPOT*fTotalHistFlux);
    return true;
}
```

Not applicable to MicroBooNE, but similar...

Description of the Fix

- Bug is essentially that larsim's `GENIEGen EndSubRun` function implicitly assumes that nutools's `GENIEHelper TotalExposure` variable will contain only POT relevant to that subrun
- For MC files with multiple subruns, this assumption is false
- Fix is to update the way larsim uses `TotalExposure` variable to fill each subrun's POT
- Code now gets `TotalExposure` at beginning of the subrun, stores its value, gets it again at the end of the subrun, and fills the `generator sumdata::POTSummary` data product based on the difference between the two
- POT reported for that subrun is then just the POT associated with events in that subrun

Description of the Fix

- In definition of GENIEGen class in larsim's GENIEGen_module.cc, add public beginSubRun function and private variables for storing POT from previous subruns
- Maintain existing functionality for distinguishing between total POT and total good POT, although this seems not used for MC

```
class GENIEGen : public art::EDProducer {
public:
    explicit GENIEGen(fhicl::ParameterSet const &pset);
    virtual ~GENIEGen();

    void produce(art::Event& evt);
    void beginJob();
    void beginRun(art::Run& run);
    void beginSubRun(art::SubRun& sr);
    void endSubRun(art::SubRun& sr);

private:
    :
    double fRandomTimeOffset;          /// The width of a simulated "beam gate".
    ::sim::BeamType_t fBeamType;       /// The type of beam


    double fPrevTotPOT;                ///< Total POT from subruns previous to current subrun
    double fPrevTotGoodPOT;            ///< Total good POT from subruns previous to current subrun

    TH1F* fGenerated[6];               ///< Spectra as generated

    TH1F* fVertexX;                   ///< vertex location of generated events in x
```

Description of the Fix

- In the GENIEGen beginJob function, just after GENIEHelper object is initialized, set these new variables to zero



```
void GENIEGen::beginJob(){  
    fGENIEHelp->Initialize();  
  
    fPrevTotPOT = 0.;  
    fPrevTotGoodPOT = 0.;  
  
    // Get access to the TFile service.  
    art::ServiceHandle<art::TFileService> tfs;
```

Description of the Fix

- In the new GENIEGen beginSubRun function, fill variables with the value reported by GENIEHelper TotalExposure
- This records the POT associated with any previous subruns

```
void GENIEGen::beginSubRun(art::SubRun& sr)
{
    fPrevTotPOT = fGENIEHelp->TotalExposure();
    fPrevTotGoodPOT = fGENIEHelp->TotalExposure();

    return;
}
```


Description of the Fix

- Finally, subtract these values off from the `TotalExposure` before filling the `sumdata::POTSummary` data product
- Ensures that the POT reported for that subrun is then just the POT associated with events in that subrun

```
void GENIEGen::endSubRun(art::SubRun& sr)
{
    std::unique_ptr<sumdata::POTSummary> p(new sumdata::POTSummary());

    p->totpot = fGENIEHelp->TotalExposure() - fPrevTotPOT;
    p->totgoodpot = fGENIEHelp->TotalExposure() - fPrevTotGoodPOT;

    sr.put(std::move(p));

    return;
}
```

- Validated by generating MC file with multiple subruns, using the same set of subruns (same input data file) with and without fix
- After fix, see subrun POT in good agreement with expectations

```
[root [1] SubRuns->Scan("sumdata::POTSummary_generator__OverlayGen.obj.totpot")
*****
*      Row      * sumdata:: *
*****
*          0 * 9.609e+15 *
*          1 * 1.943e+16 *
*          2 * 2.986e+16 *
*          3 * 3.731e+16 *
*          4 * 6.913e+16 *
*          5 * 1.110e+17 *
*          6 * 1.336e+17 *
*          7 * 1.379e+17 *
*          8 * 1.624e+17 *
*****
```

Before fix, subrun POT increasing

Typical *difference* $\sim 6e15$

```
[root [1] SubRuns->Scan("sumdata::POTSummary_generator__OverlayGen.obj.totpot")
*****
*      Row      * sumdata:: *
*****
*          0 * 8.051e+15 *
*          1 * 4.050e+15 *
*          2 * 1.122e+16 *
*          3 * 4.048e+15 *
*          4 * 1.093e+16 *
*          5 * 7.964e+15 *
*          6 * 4.686e+15 *
*          7 * 3.486e+15 *
*          8 * 3.714e+15 *
*****
```

After fix, subrun POT are independent

Typical value $\sim 6e15$, in good agreement with expectations

- There is a bug in the MC POT counting for MC files with multiple subruns — unusual use case, but now used by MicroBooNE
- Bug is essentially that larsim's `GENIEGen` implicitly assumes that nutools's `GENIEHelper TotalExposure` is POT for the subrun, but it is actually POT for that instance of `GENIEHelper`
- Fixed storing the `TotalExposure` at the beginning of the subrun, subtracting it off from the `TotalExposure` at the end of the subrun to get the POT for just that subrun
- Based on my validation, fix works as expected

Backup Slides

- When counting POT for an overlay file, take the POT for the file to be the POT reported by the `generator sumdata::POTSummary` data product for the last subrun

run	subrun	pot
8315	18	1.318242e+18
8315	19	1.923767e+18
8315	26	3.263035e+18
8315	46	4.426622e+18
8315	47	5.452171e+18

To get POT for a typical file, would sum all of these

To get correct POT for an overlay file, just use this

- Caveat: Have to be extra careful when counting POT based on files produced by running over multiple input overlay files (i.e., the output of grid jobs run with multiple files per job)