PLR Code

SIRun4.cxx

```
RooMsgService::instance().setGlobalKillBelow(RooFit::WARNING) ;
TStopwatch timer;
timer.Start();
RooRandom::randomGenerator()->SetSeed(0); //use clock, get different seed each run
```

- Pretty self explanatory
- Sets up a timer to time the process
- kill warning messages
- Make sure each run uses a different random seed

```
RooWorkspace * ws = MakeWorkspace();
```

- Creates the workspace using the MakeWorkspace method from "MakeWorkspace.h"
  - Declares the RooRealVars r, S1, log10S2, phi, drift, mWimp, and xsec and imports into workspace.
  - Creates timeBin categories (whatever that means) and defines exposure for each.
  - Returns the workspace

```
ImportSignalModel_5D(ws,thisWimpMass, useAnalyticIntegration, useFactorizedPDF, op, nucleon, useNuisParams);
```

- Uses “ImportSignalModel_5D” from “ImportSignalModel.h” to import the signal PDFs nrPop_z[z-slice]_t[time bin] (RooProdPDF) into the workspace.

```
double poimin;
double poimax;
//double events_per_zb;
double events_per_zb_par0;
double events_per_zb_par1;
RooArgList kLindCoeff;
if( useXSec ) {
   ifstream LimitsPars("LimitParameters_XSec.txt");
   int LineIter=0;
   string StrIn;
   if(!LimitsPars){cout<<"LimitParameters_XSec.txt not found!\n";}
   else{
          double DoubleInMass,DoubleIn0,DoubleIn1,PoiMinIn,PoiMaxIn;
          //double DoubleInMass,PoiMinIn,PoiMaxIn;
          while(!LimitsPars.eof()){
                 LineIter++;
                 if(LineIter == 1){
                        getline(LimitsPars,StrIn,'\n');
                 }
                 else{
                        //LimitsPars>>DoubleInMass>>events_per_zb>>PoiMinIn>>PoiMaxIn;
                        LimitsPars >> DoubleInMass >> DoubleIn0 >> DoubleIn1 >> PoiMinIn >> PoiMaxIn;
                        getline(LimitsPars,StrIn,'\n');
                        if(fabs(thisWimpMass-DoubleInMass) < 1e-6){
                               poimin=PoiMinIn;
                               poimax=PoiMaxIn;
                               events_per_zb_par0=DoubleIn0;
                               events_per_zb_par1=DoubleIn1;
                               break;
                        }
                 }
          }
   }
}
```

LimitParameters_XSec.txt contents

```
mWIMP[GeV] (evts/zb)_par0 (evts/zb)_par1 poimin poimax
5 -0.0169158 0.0229533     200 6000
7 -0.255225 0.413313       0.1 70
10 -0.892262 2.21958       0.1 10
12 -0.91426 3.72913        0.1 6
14 -0.433867 5.10018       0.05 3.5
17 1.11058 6.56744         0.05 2.5
21 3.82891 7.61035         0.02 2
33 10.8276 7.36094         0.02 1
50 14.8909 4.63995         0.03 1.2
100 12.8643 1.12056        0.05 2
200 7.69796 0.181083       0.07 4
400 4.11083 0.00358812     0.1  6
1000 1.69626 -0.0162692    0.2  20
4000 0.429713 -0.00594765  0.5  90
20000 0.0865611 -0.00163092        .7  600
100000 0.0172476 -0.000251011      1.0 2100
```

- Reads the line corresponding to the current WIMP-mass from the “LimitParameters_XSec.txt” text file and sets the poi variables poimin, poimax, etc. accordingly

```
ImportBkgModel(ws,"./BackgroundModel/bgws.root",useAnalyticIntegration,useFactorizedPDF);
ImportWallModel(ws);
ImportAccidentalModel(ws);
Import8BModel(ws);
```

- Imports the various background PDFs into the workspace. (I think, haven't looked too much into this chunk)

```
if(useXSec == 1){
        cout<<"COUT:: Creating nSig as a product of xsec and evt_per_zb\n";
        cout<<"COUT:: POI Range = "<<poimin<<" : "<<poimax<<endl;
        //RooRealVar roo_events_per_zb("events_per_zb","events_per_zb",events_per_zb);
        //ws->import(roo_events_per_zb);
        //ws->factory("prod::nSig(xsec,events_per_zb)");
        kLindCoeff.add(RooConst(events_per_zb_par0));
        kLindCoeff.add(RooConst(events_per_zb_par1));
        RooPolynomial EvtPerZb_kLind("EvtPerZb_kLind","EvtPerZb_kLind",*ws->var("kLindVar"),kLindCoeff,0);
        ws->import(EvtPerZb_kLind);
        ws->factory("prod::nSig(xsec,EvtPerZb_kLind)");
}
else{
        cout<<"COUT:: Creating nSig with just a range of 0 to 100\n";
        ws->factory("nSig[0.0,100]");
        poimin = 0;
        poimax = nPoints - 1;
}
```

- Creates a new RooRealVar “nSig”
  - If useXSec is true, the poi variables from earlier are used to calculate the value of nSig
  - If false, nSig is just something between 0 and 100

```
        TString str_simulModel("SIMUL::simulModel( timeBin");
        for (int tt=1; tt<=NTIMEBIN; tt++) {
                TString factory_cmd = TString::Format("RooFormulaVar::sigCoeff%d('(timeBin%d_sig_exposure_frac*nSig)',{timeBin%d_sig
_exposure_frac,nSig})",tt,tt,tt);
                ws->factory(factory_cmd.Data()); //signal coeff
                factory_cmd = TString::Format("RooFormulaVar::B8Coeff%d('(timeBin%d_sig_exposure_frac*nB8)',{timeBin%d_sig_exposure_
frac,nB8})",tt,tt,tt);
                ws->factory(factory_cmd.Data()); //Boron-8 coeff
                factory_cmd = TString::Format("RooFormulaVar::accidentalCoeff%d('(timeBin%d_sig_exposure_frac*nAccidental)',{timeBin
%d_sig_exposure_frac,nAccidental})",tt,tt,tt);
                ws->factory(factory_cmd.Data()); //accidental coeff
                factory_cmd = TString::Format("RooFormulaVar::comptonBottomCoeff%d('comptonBottom%d_scale*nComptonBottom',{comptonBo
ttom%d_scale,nComptonBottom})",tt,tt,tt);
                ws->factory(factory_cmd.Data()); //Compt bottom coeff
                factory_cmd = TString::Format("RooFormulaVar::comptonRestCoeff%d('comptonRest%d_scale*nComptonRest',{comptonRest%d_s
cale,nComptonRest})",tt,tt,tt);
                ws->factory(factory_cmd.Data()); //Compt rest coeff
                factory_cmd = TString::Format("RooFormulaVar::rnkrCoeff%d('rnKr%d_scale*nRnKr',{rnKr%d_scale,nRnKr})",tt,tt,tt);
                ws->factory(factory_cmd.Data()); //Rn and Kr coeff
                factory_cmd = TString::Format("RooFormulaVar::wallCoeff%d('(timeBin%d_sig_exposure_frac*nWall)',{timeBin%d_sig_expos
ure_frac,nWall})",tt,tt,tt);
                ws->factory(factory_cmd.Data()); //wall coeff
                factory_cmd =
 TString::Format("SUM::fullModel_timeBin%d(sigCoeff%d*nrPop%d,B8Coeff%d*B8Pop%d,wallCoeff%d*wallPop%d,accidentalCoeff%d*AccidentalPo
p%d,comptonBottomCoeff%d*ComptonBottomPop%d,comptonRestCoeff%d*ComptonRestPop%d,rnkrCoeff%d*RnKrPop%d)", tt,tt,tt,tt,tt,tt,tt,tt,tt,
tt,tt,tt,tt,tt,tt);
          ws->factory(factory_cmd.Data()); // sum signal and bkg model for this time bin
```

- Generating coefficients for each model and combining them together to form a jumbo PDF (I think) called “fullModel_timeBin[time bin]”.

```
   str_simulModel += TString::Format(", timeBin%d=fullModel_timeBin%d",tt,tt);
}
str_simulModel += " )";
// Combing models of the Time bins.
ws->factory(str_simulModel.Data());
```

- Together with the top line from the last slide creates a new SIMUL (whatever that is) from the four time bin models called “simulModel”

```
        //Make constraint terms:
        //ws->factory("PROD::constraints(comptonBottomConstraint,comptonRestConstraint,rnKrConstraint)");
        if (useNuisParams) {
           ws->factory("PROD::constraints(B8Constraint,WallConstraint,accidentalConstraint,comptonRestConstr
aint,comptonBottomConstraint,rnKrConstraint,G2VarConstraint,kLindVarConstraint)");
        } else {
           ws->factory("PROD::constraints(B8Constraint,WallConstraint,accidentalConstraint,comptonRestConstr
aint,comptonBottomConstraint,rnKrConstraint,kLindVarConstraint)");;
        }

        //-------------------
        //Multiply whole thing by constraint terms
        ws->factory("PROD::modelWithConstraints(simulModel,constraints)");
```

- I don’t understand this part.
- End up with a new PDF? “modelWithConstraints”

```
ConfigureModel(ws,useNuisParams,useXSec);
```

- Creates a RooStats::ModelConfig object so HypoTestInv demo knows what to do and imports it into the workspace.
- Defines sets of variables into observables, global observables, and nuisance parameters.
- Sets parameter of interest

```
float nBGTotal = ws->var("nComptonBottom0")->getVal()
               + ws->var("nComptonRest0")->getVal()
               + ws->var("nRnKr0")->getVal()
               + ws->var("nWall0")->getVal()
               + ws->var("nAccidental0")->getVal()
               + ws->var("nB80")->getVal();
cout<<endl;
cout<<"===== EXPECTED BKGS====="<<endl;
cout<<"ComptonBottom: "<< ws->var("nComptonBottom0")->getVal() <<endl;
cout<<"ComptonRest:   "<< ws->var("nComptonRest0")->getVal() <<endl;
cout<<"RnKr:          "<< ws->var("nRnKr0")->getVal() <<endl;
cout<<"Wall:          "<< ws->var("nWall0")->getVal() <<endl;
cout<<"Accidental:    "<< ws->var("nAccidental0")->getVal() <<endl;
cout<<"Boron-8:       "<< ws->var("nB80")->getVal() <<endl;
cout<<"Total:         "<< nBGTotal <<endl;
cout<<endl;
```

- Calculates the total number of background events and prints the contributions.

```
ModelConfig *bModel=(ModelConfig*)ws->obj("bModel");
```

- Creates a ModelConfig object “bModel” from the one imported earlier into the workspace.

```
// Read in WIMP search data
RooDataSet* searchData;
std::string data_string (dataFile);
unsigned extention = data_string.find_last_of(".");
unsigned data_is_root_file = data_string.substr(extention+1) == "root";
if(dataFile && data_is_root_file) {
        // if the input file is valid, read in the search data
        TFile* f_in = TFile::Open(dataFile);
        TTree *tree = (TTree*)f_in->Get("platinum");
        //TTree *tree = (TTree*)f_in->Get("t1");
        searchData = new RooDataSet("obsData","searchData",*ws->set("obs"),Import(*tree));
        cout<<"COUT:: searchData has "<<searchData->numEntries()<<" entries\n";
        f_in->Close();
}
else {  // if not, generate one from the background model, and store
        cout<<"COUT:: The input file is not valid. Generating a background only dataset and";
        cout<<" using it for limit setting."<<endl;
        searchData = bModel->GetPdf()->generate(*ws->set("obs"),nBGTotal);
        searchData->SetName("obsData");
}
```

- Creates a RooDataSet “searchData” from data in a provided root file.
- If the provided file doesn’t have the .root extension a data set is generated from the background model. (Though I know from testing that this doesn’t work correctly)

```
ws->import(*searchData);
```

- … Imports the data set that was just created into the workspace.

```
if(useXSec == 1){
        ws->var("xsec")->setVal((poimin+poimax)/2.0);
} else {
        ws->var("nSig")->setVal(1.0);
}
```

- Sets the poi to a value (look for yourself to see what)

```
if( runHypoTest ){

        //ws->Print();
        //StandardHypoTestInvDemo(ws,thisWimpMass,filenum,"ws","model","bModel",
        //              "obsData",0,2,0,10,1,10,ntrials,0);

        cout<<"Running HypoTestInverter with"<<endl;
        cout<<"WIMP mass = "<<thisWimpMass<<endl;
        cout<<"poimin:poimax = "<<poimin<<" : "<<poimax<<endl;
        cout<<"nPoints = "<<nPoints<<endl;
        StandardHypoTestInvDemo(ws,thisWimpMass,filenum,"ws","model","bModel",
                        //"obsData",0,2,0,10,2,15,ntrials,0);
                        "obsData",0,2,0,nPoints,poimin,poimax,ntrials,0);
}
```

- Runs the HypoTestInvDemo (the actual PLR)

```
cout << "------------------------------------------------------------------" << endl;
cout << "Finished running in " << timer.RealTime() << " seconds." << endl;
cout << "------------------------------------------------------------------" << endl;
```

- Tells you how long it took