EXCELENCIA SEVERO OCHOA
BIST Barcelona Institute of Science and Technology

gLike

Javier Rico

MAGIC Dark Matter Workshop, January 16-18, 2019

What is gLike?

- gLike is an open source, ROOT-based code framework for the numerical maximization of general-purpose joint likelihood functions.
- The joint likelihood function has **one** *free* parameter (named *g*) and as many nuisance parameters as wanted, which will be profiled in the maximization process
- Non-exhaustive list of examples where gLike is useful (in order of increasingly complexity):
  - Estimating the number of signal events (with uncertainties) and significance in a dataset whose background content is in turn estimated from an independent measurement in a signal-free control-region (Li&Ma).
  - Same as before, but considering in addition a systematic uncertainty in the estimation of the background or efficiency of signal detection (Rolke).
  - Estimating the intensity of a steady source of signal particles in the presence of background particles from datasets obtained under different experimental conditions.
  - Same as before, but each dataset actually comes from a different instrument and in different data format.
  - Estimating the dark matter annihilation cross-section combining observations of dwarf spheroidal galaxies by different ground-based gamma-ray telescopes, satellite gamma-ray detectors, neutrino telescopes, ….
  - Estimating the energy scale of quantum gravity by combining observations of recently measured GRBs, maybe observed by different ground-based gamma-ray telescopes.

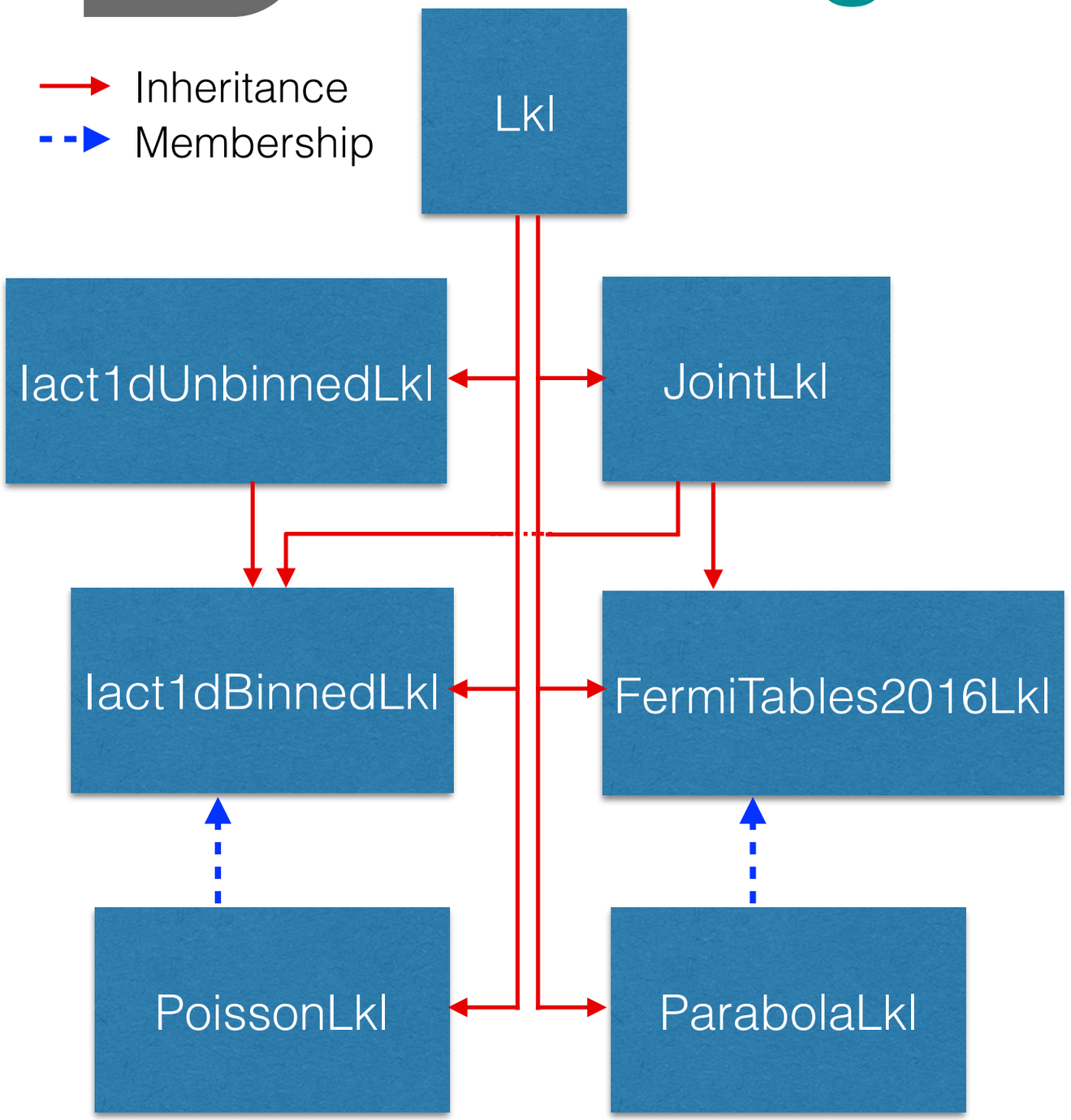
Installation

- gLike is open source, hosted at GitHub (https://github.com/javierrico/gLike)
- At user level you are recommended to download, compile and run the latest stable release, latest one is v00.06.00.
- Once you become an expert you will want to develop your own classes, for which you will need to check out the *master* (for new developments) or *release* (for bug fixes) branches. Check the release wiki entry for more information about the gLike repository branch and release structure.
- Define the environment variable GLIKESYS pointing to your main gLike directory, e.g., for bash:
  - `export GLIKESYS="/Users/rico/gLike/"`
- and include it in the library path, eg. for Mac:
  - `export DYLD_LIBRARY_PATH="${GLIKESYS}:${DYLD_LIBRARY_PATH}"`
- Create the gLike library by typing *make* in the main gLike directory
- [Optional] Create the gLike online documentation with *make doc*

gLike distribution structure

- In gLike you find the following directories:
  - `src`: source files (*.cc) with definition of every class,
  - `include`: include files (*.h) with declaration of every class,
  - `scripts`: root macros and scripts with some gLike example applications,
  - `rcfiles`: examples of rcfiles (in principle the only thing a non-expert user should edit and modify),
  - `data`: examples of input data files, e.g. events and corresponding IRFs mimicking a generic IACT telescope of the 2nd generation like MAGIC,
  - `DM`: files for DM-related analysis, e.g. the dN/dE functions for different masses and annihilation channels,
  - `logo`: the gLike logo
- Other directories are created when compiling/using gLike:
  - `out`: .o files for creating the librar
  - `lib`: contains library libgLike.so
  - `plots`: for storing plots created by macros

gLike code structure



- `Lkl`: abstract class to link with TMinuit, perform the minimization (it assumes 1 pure free parameter + any number of nuisance parameters) and cope with several common technicalities.
- `JointLkl`: Combine `Lkl` objects into a joint likelihood (works also recursively)
- Almost all the rest implement a particular likelihood function
  - `Iact1dUnbinnedLkl`: unbinned likelihood for energy spectra measured with IACTs
  - `Iact1dBinnedLkl`: binned version of `Iact1dUnbinnedLkl`
  - `FermiTables2016Lkl`: total lkl values according to Fermi tables (lkl vs flux for bins in Energy)
  - `PoissonLkl`: On/Off double Poisson lkl (LiMa, Rolke,…)
  - `ParabolaLkl`: holds values of lkl vs *g*
  - `TemplateLkl`: template for new classes
- `IactEventListIRF`: simple data+IRF format for the IACT lkl classes

User levels

- ★ Non-expert user: uses only high-level scripts or executables for solving specific problem (e.g. DM combination of results from different instruments and targets), with virtually no direct access to gLike classes, just a configuration file.
- ★ Expert user: creates gLike classes and directly accesses their methods and functions to solve problems for which there is no existing script yet.
- ★ Developer: develops new gLike classes with new likelihood functions and/or scripts

gLike expert user

Basic expert usage

- Instantiate and configure (e.g. read input files) preferred Lkl-based object
- Configure the minimization algorithm:
  - `SetErrorDef` → set the level of -2logL over its minimum that defines the error in *g*
  - `SetUnitsOfG` → results are provided as -2logL vs *g**fUnitsOfG, called “user” units (more about why this is extremely useful later)
  - `SetGIsPositive` → if we want to restrict *g* to be definite positive
- Call the minimization method with `ComputeLklVsG`
- Access the results, e.g.:
  - `PrintOverview`: print results from the minimization process
  - `GetLklVsG`: TGraph with values of the -2logL function vs *g*
  - `GetGLklMin`: value of *g* (g_{min}) for which -2logL is minimum ($-2\log L_{min}$)
  - `GetGLklMinErr`: value of Δg for which $-2\log L(g_{min}+\Delta g) = -2\log L_{min}+$fErrorDef

Example, using PoissonLkl

```
void testPoissonLkl()
{
  // create and configure the simplest possible PoissonLkl object
  const Int_t Non = 130;
  const Int_t Noff = 90;
  const Double_t tau = 1.;
  PoissonLkl* p = new PoissonLkl(Non,Noff,tau);

  // configure the minimization algorithm
  const Double_t errorDef = 4;
  const Double_t unitsOfG = 2;
  p->SetErrorDef(errorDef);
  p->SetUnitsOfG(unitsOfG);

  // call the minimization
  p->ComputeLklVsG();

  // print fit results
  cout << endl << "PrintOverview:" << endl;
  p->PrintOverview();     // print the details from the fit
  p->GetLklVsG()->Draw(); // plot the -2logL vs g curve

  // access to the gLike results
  const Double_t gmin = p->GetGLklMin();               // get the value of g that minimizes -2logL
  const Double_t gerr = p->GetGLklMinErr();            // get the value of gerr such that -2logL(gmin+/-gerr)=-2logLmin+fErrorDef
  const Double_t sig  = sqrt(p->GetLklVsG()->Eval(0)) ; // compute significance of detection

  // compute significance with LiMa formula for comparison
  const Double_t LiMasig = sqrt(2.*(Non*log((tau+1.)*(1.*Non/(Non+Noff)))+Noff*log((tau+1)/tau*(1.*Noff/(Non+Noff)))));

  // compare confidence interval with Rolke method for comparison
  TRolke* r = new TRolke(0.9544);
  r->SetPoissonBkgKnownEff(Non,Noff,tau,1);
  Double_t low,upp;
  r->GetLimits(low,upp);
  Double_t grolke    = unitsOfG*(upp+low)/2;
  Double_t grolkeerr = unitsOfG*(upp-low)/2;

  // print and compare results
  cout << endl;
  cout << "The max lkl  value of g = " << gmin << ",\t with "<< sqrt(errorDef) << "-sigma error bar = " << gerr << endl;
  cout << "The Rolke    value of g = " << grolke << ",\t with 2-sigma error bar = " << grolkeerr << endl;
  cout << "The gLike significance of the detection of signal is " << sig     << " sigma" << endl;
  cout << "The Li&Ma significande of the detection of signal is " << LiMasig << " sigma" << endl;
}
```

Define the following lkl function:

$$\mathcal{L}(g;b|N_{\rm on},N_{\rm off}) = \frac{(g+b)^{N_{\rm on}}e^{-(g+b)}}{N_{\rm on}!}\,\frac{(\tau b)^{N_{\rm off}}e^{-\tau b}}{N_{\rm off}!}$$

The error is defined for the value of *g* with -2logL = min + 4
Units of g is 2

Call the minimization of the -2logL and scan it around minimum

Print fit results and draw the curve -2logL vs *g*

Compute best fit for *g*, with error bars and significance, using gLike and standard (Li&Ma, Rolke) methods, compare results

gLike output

ComputeLklvsG

```
root [10]   p->ComputeLklVsG();
Lkl::ComputeLklVsG (PoissonLkl) Message: Finding minimum of -2logL...
Lkl::MinimizeLkl (PoissonLkl) Message: minimizing -2logL
Lkl::CallMinimization (PoissonLkl) Results: Trial #1, g: 4.00e+01 +/- 2.97e+01 (8.00e+01 +/- 5.93e+01); b: 9.00e+01 +/- 1.90e+01;
tau: 1.00e+00 +/- 0.00e+00; eff: 1.00e+00 +/- 0.00e+00; -2logL = 13.0462; iflag = 0 (converged)
Lkl::ComputeLklVsG (PoissonLkl) Message: computing -2logL in 200 points between g=-1.93973(-3.87946), and g=81.9447(163.889), this
 could take a while
..................... Completed 200 points
Lkl::ComputeLklVsG (PoissonLkl) Message: g_min = 39.7917 +/- 30.242 (79.5834 +/- 60.4839), -2logLmin = 13.0464
```

1: Name of class and function issuing the log message

2: Name of class issuing the message

3: The message

4: Migrad method in TMinuit is used to find the value of *g* minimizing the -2logL function (g_{min}) and estimating the error for the free and nuisance parameters, which are reported

5: Values of *g* always shown in “intrinsic” and “user” (between brackets) units, configurable when one of them must be chosen, like e.g. in axis of plots

6: Status of TMinuit after minimization finished (will try several times looking for convergence, changing e.g. step)

7: The -2logL function is scanned around g_{min}, for a configurable number of points in a range that should contain $-2\log L_{min}$+fErrorDef at both sides of the minimum (several trials might be needed if the -2logL function is not parabolic)

8: From the result of point 5, the values of best fit for *g* (g_{min}) and the confidence interval corresponding to fErrorDef are reported

gLike fit results

PrintOverview

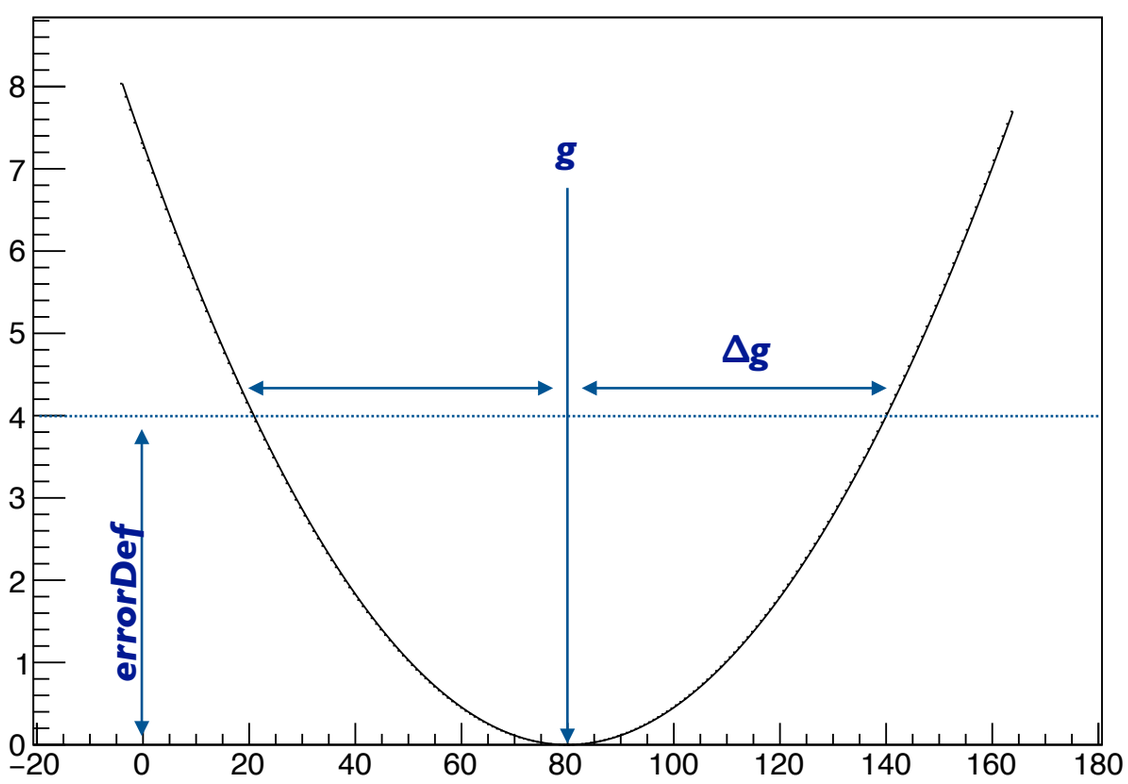
```
root [11]   p->PrintOverview();     // print the details from the fit
 *  Name                = PoissonLkl
 *  Status              = 0 (converged)
 *  Delta(2logL)        = 4
 *  # of parameters     = 4 (2 free):
 *                  g = 39.7917 +/- 30.242 (79.5834 +/- 60.4839)
 *                  b = 90.0852 +/- 0.0948683
 *                tau = 1 (fixed)
 *                eff = 1 (fixed)
 *  Units of G          = 2
 *  -2logL_min          = 13.0464
```

Object-by-object result of the minimization process

GetLklVsG()->Draw()

Draw the -2logL function vs *g* (in “user” units).

Always try to have a look to this plot, specially if you see strange things, most of the times you will get a clue of what is going on



Comparing gLike and "standard" results

```
// access to the gLike results
const Double_t gmin = p->GetGLklMin();                  // get the value of g that minimizes -2logL
const Double_t gerr = p->GetGLklMinErr();               // get the value of gerr such that -2logL(gmin+/-gerr)=-2logLmin+fErrorDef
const Double_t sig  = sqrt(p->GetLklVsG()->Eval(0)) ; // compute significance of detection
```

```
The max lkl  value of g = 79.5834,       with 2-sigma error bar = 60.4839
The Rolke    value of g = 80.4773,       with 2-sigma error bar = 59.5622
The gLike significance of the detection of signal is 2.70428 sigma
The Li&Ma significande of the detection of signal is 2.70432 sigma
```

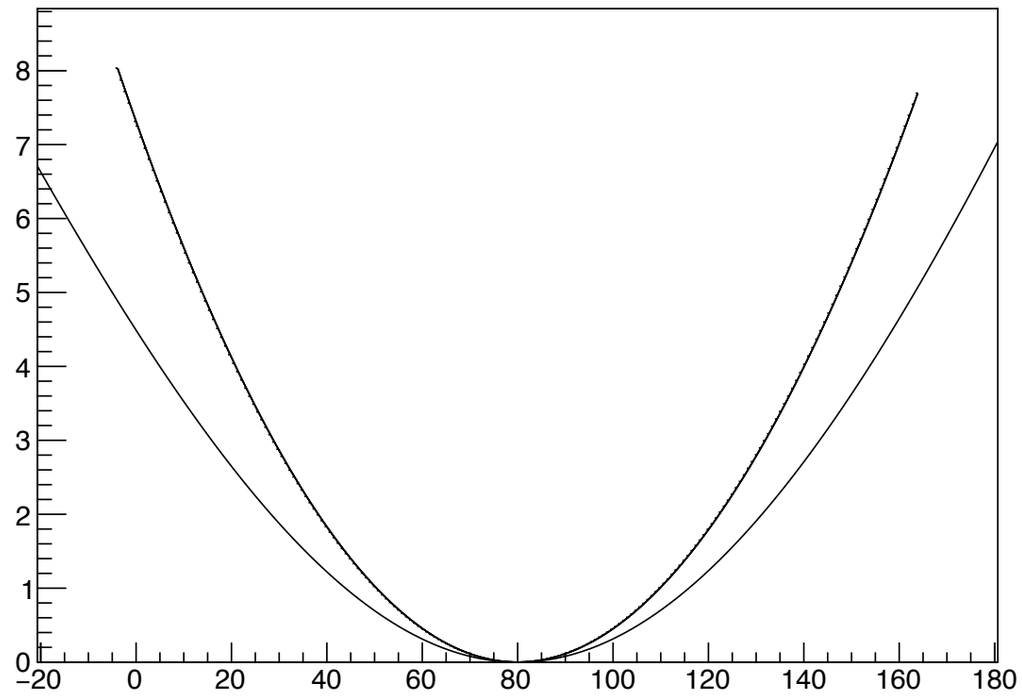
Extra nuisance parameters

Total implementation of ParabolaLkl likelihood function:

$$\mathcal{L}(g; b, \tau, \epsilon | N_{\rm on}, N_{\rm off}, \mu_\tau, \sigma_\tau, \sigma_\epsilon) = \frac{(\epsilon g + b)^{N_{\rm on}} e^{-(\epsilon g + b)}}{N_{\rm on}!} \frac{(\tau b)^{N_{\rm on}} e^{-\tau b}}{N_{\rm off}!} \frac{1}{\sqrt{2\pi}\sigma_\tau} e^{\frac{(\tau-\mu_\tau)^2}{2\sigma_\tau^2}} \frac{1}{\sqrt{2\pi}\sigma_\epsilon} e^{\frac{(\epsilon-1)^2}{2\sigma_\epsilon^2}}$$

```
Double_t Deff  = 0.1;
Double_t Dtau  = 0.1;
p->SetDEff(Deff);
p->SetDTau(Dtau);
p->ComputeLklVsG();
p->PrintOverview();     // print the details from the fit
p->GetLklVsG()->Draw("same"); // plot the -2logL vs g curve
```

```
*  Name                = PoissonLkl
*  Status              = 0 (converged)
*  Delta(2logL)        = 4
*  # of parameters     = 4 (4 free):
*                   g = 39.7475 +/- 37.1452 (79.4951 +/- 74.2905)
*                   b = 90.1232 +/- 0.0948683
*                 tau = 0.999306 +/- 0.01
*                 eff = 1.00056 +/- 0.01
*  Units of G          = 2
*  -2logL_min          = 7.51185
```



Building a joint likelihood

```
void testJointLkl()
{
  // define number of PoissonLkl objects that we will use in the JointLkl and their configuration parameters
  const Int_t nsamples = 2;
  const Int_t Non[nsamples]  = {130,90};
  const Int_t Noff[nsamples] = {105,110};

  // create the JointLkl object that will collect the PoissonLkl objects
  const Double_t errorDef = 4;
  JointLkl* j = new JointLkl;
  j->SetErrorDef(errorDef);

  // configure samples and introduce them into the JointLkl
  PoissonLkl* p[nsamples];
  for(Int_t isample=0;isample<nsamples;isample++)
    {
      p[isample] = new PoissonLkl(Non[isample],Noff[isample]);
      p[isample]->SetName(Form("PoissonLkl_%1d",isample));
      j->AddSample(p[isample]);
    }

// print how JointLkl looks like
j->PrintData();

// call minimization and print/plot results
j->ComputeLklVsG();
j->PrintOverview();     // print the details from the fit
j->GetLklVsG()->Draw(); // plot the -2logL vs g curve

// compare it to each PoissonLkl object separately
for(Int_t isample=0;isample<nsamples;isample++)
  {
    p[isample]->SetErrorDef(errorDef);
    p[isample]->ComputeLklVsG();
    p[isample]->PrintOverview();     // print the details from the fit
    TGraph* lklvsg = p[isample]->GetLklVsG();
    lklvsg->SetLineColor(isample+2); // plot the -2logL vs g curve
    lklvsg->Draw("same");            // plot the -2logL vs g curve
  }
```

Configuration parameters for the PoissonLkl objects

Create and configure the JointLkl object

Create and configure the JointLkl object and input the PoissonLkl objects within

Print data, call minimization, print results

Compare result with those from individual minimization of the two PoissonLkl terms

Results from `JointLkl` minimisation

In testJointLkl.C we have built the following Joint likelihood:

$$\mathcal{L}(g; b_1, b_2, |N_{\mathrm{on}1}, N_{\mathrm{on}2}, N_{\mathrm{off}1}, N_{\mathrm{off}2}) = \underbrace{\mathrm{Pois}(N_{\mathrm{on}1}|g+b_1) \times \mathrm{Pois}(N_{\mathrm{off}1}|\tau_1 b_1)}_{\mathrm{PoissonLkl\_1}} \times \underbrace{\mathrm{Pois}(N_{\mathrm{on}2}|g+b_2) \times \mathrm{Pois}(N_{\mathrm{off}2}|\tau_2 b_2)}_{\mathrm{PoissonLkl\_2}}$$

(note that *g* is common to both terms, there is always only 1 free parameter!)

PrintData

```
*            Object Name : JointLkl
*           # of samples = 2
*****************************************
*              Object Name : PoissonLkl_0
*                    Non  = 130
*                    Noff = 105
*             Measured tau  = 1 +/- 0
*             Measured Deff = 1 +/- 0
*     Fraction of G in Off = 0
*  Foreground events in On = 0
*    Background in On (b) is NUISANCE
*****************************************
*              Object Name : PoissonLkl_1
*                    Non  = 90
*                    Noff = 110
*             Measured tau  = 1 +/- 0
*             Measured Deff = 1 +/- 0
*     Fraction of G in Off = 0
*  Foreground events in On = 0
*    Background in On (b) is NUISANCE
*****************************************
```

PrintOverview

```
*  Name              = JointLkl
*  Status            = 0 (converged)
*  Delta(2logL)      = 4
*  # of parameters   = 1 (1 free):
*                  g = 0.845592 +/- 20.8252 (0.845592 +/- 20.8252)
*  Units of G        = 1
*  -2logL_min        = 30.7432
*  # of samples      = 2
*****************************************
                                        *  Name              = PoissonLkl_0
                                        *  Status            = 0 (converged)
                                        *  Delta(2logL)      = 1
                                        *  # of parameters   = 4 (1 free):
                                        *                  g = 0.845592 (0.845592) (fixed)
                                        *                  b = 117.033 +/- 0.10247
                                        *                tau = 1 (fixed)
                                        *                eff = 1 (fixed)
                                        *  Units of G        = 1
                                        *  -2logL_min        = 15.6878
                                        *****************************************
                                        *  Name              = PoissonLkl_1
                                        *  Status            = 0 (converged)
                                        *  Delta(2logL)      = 1
                                        *  # of parameters   = 4 (1 free):
                                        *                  g = 0.845592 (0.845592) (fixed)
                                        *                  b = 99.621 +/- 0.104881
                                        *                tau = 1 (fixed)
                                        *                eff = 1 (fixed)
                                        *  Units of G        = 1
                                        *  -2logL_min        = 15.0554
                                        *****************************************
```

Why user units are so important?

- Example in `testJointLkl.C` is only useful if experiments represented by the PoissonLkl objects have the same expectation value for *g*
  - e.g. we observe a steady gamma-ray source with the same instrument in exactly the same experimental conditions (i.e. IRF) and for exactly the same amount of time, in that case *g* is the expected number of signal events in **both** On regions. This is a rare case because:

$$g = T_{\rm obs} \int_{E'_{\rm min}}^{E'_{\rm max}} dE' \int dE \, \frac{d\Phi}{dE} \, A_{\rm eff}(E) \, G(E'|E)$$

  - Following this example: in many practical cases the spectral shape (*df/dE*) can be assumed and the spectral normalization *k* is unknown, i.e.:

$$g = k \, T_{\rm obs} \int_{E'_{\rm min}}^{E'_{\rm max}} dE' \int dE \, \frac{df}{dE} \, A_{\rm eff}(E) \, G(E'|E)$$

  - In such a case (ie. whenever we can refer *g* to some global common parameter) we can refer all sample-dependent g values to common *k* parameter by setting units for each sample as:

$$\texttt{SetUnitsOfG}\left(\frac{1}{T_{\rm obs} \int_{E'_{\rm min}}^{E'_{\rm max}} dE' \int dE \, \frac{df}{dE} \, A_{\rm eff}(E) \, G(E'|E)} \right)$$

  - The units of each sample are used by `JointLkl` to compute the value of *g* that it passes to it `Lkl` objects, in such a way that they all refer to the same value of *g**fUnitsOfG

JointLkl minimisation with units of g

- Example: we assume g is the number of gamma rays measured by some Cherenkov telescope from a given steady source in the On region, during a given known observation time, with known IRF and threshold. The spectrum of the source is a power law with index -2, and unknown normalization k = (dPhi/dE)(E=100GeV)
- We consider the following simplifications:
  - constant effective area with energy
  - perfect energy resolution
- In such a case $\texttt{fUnitsOfG} = \frac{1}{T_{\rm obs} A_{\rm eff} \int_{E_{\rm min}}^{E_{\rm max}} dE \, \frac{df}{dE}}$ with $\int_{E_{\rm min}}^{E_{\rm max}} dE \, \frac{df}{dE} = (100{\rm GeV})^2 (\frac{1}{E_{\rm min}} - \frac{1}{E_{\rm max}})$

```
  // these are the values of observation time, IRF, etc...
  const Double_t Emin[nsamples]  = {100,200};           // [GeV] energy threshold
  const Double_t Emax[nsamples]  = {10000,10000};       // [GeV] maximum measurable energy
  const Double_t Tobs[nsamples]  = {2*60*60,1.8*60*60}; // [s] observation times
  const Double_t Aeff[nsamples]  = {1e9,0.98e9};        // [cm^2] effective area (considered constant)
  Double_t IntegraldFdE[nsamples];                      // [GeV] integral of the spectral shape function betwen Emin and Emax
  Double_t unitsOfG[nsamples];                          // [s-1 cm-2 GeV-1] 1/(Tobs*Aeff*IntegraldFdE)
  for(Int_t isample=0;isample<nsamples;isample++)
    {
      IntegraldFdE[isample] = 100.*100.*(1./Emin[isample]-1./Emax[isample]);
      unitsOfG[isample]     = 1./(Tobs[isample]*Aeff[isample]*IntegraldFdE[isample]);
      p[isample]->SetUnitsOfG(unitsOfG[isample]);
    }

  // print how JointLkl looks like
  cout << endl << "################################################" << endl;
  j->PrintData();

  // call minimization and print/plot results
  j->ComputeLklVsG();
  j->PrintOverview();     // print the details from the fit
  j->GetLklVsG()->Draw(); // plot the -2logL vs g curve
}
```

→ Define observation time, threshold, Aeff, etc, for each sample

→ Compute and set the units of *g* for each sample

→ Compute -2logL vs ***k***

-2logL vs g with units of g

PrintOverview

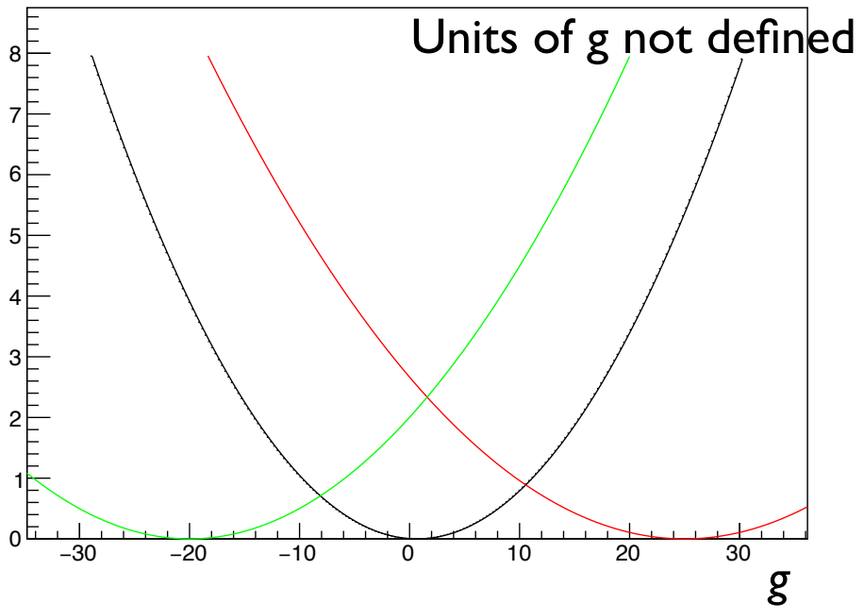
```
*  Name              = JointLkl
*  Status            = 0 (converged)
*  Delta(2logL)      = 4
*  # of parameters   = 1 (1 free):
*                  g = 11.982 +/- 27.9261 (1.68097e-14 +/- 3.9178e-14)
*  Units of G        = 1.40292e-15
*  -2logL_min        = 29.9868
*  # of samples      = 2
****************************************
                                    *  Name              = PoissonLkl_0
                                    *  Status            = 0 (converged)
                                    *  Delta(2logL)      = 4
                                    *  # of parameters   = 4 (1 free):
                                    *                  g = 11.982 (1.68097e-14) (fixed)
                                    *                  b = 111.176 +/- 0.10247
                                    *                tau = 1 (fixed)
                                    *                eff = 1 (fixed)
                                    *  Units of G        = 1.40292e-15
                                    *  -2logL_min        = 13.923
                                    ****************************************
                                    *  Name              = PoissonLkl_1
                                    *  Status            = 0 (converged)
                                    *  Delta(2logL)      = 4
                                    *  # of parameters   = 4 (1 free):
                                    *                  g = 5.23068 (1.68097e-14) (fixed)
                                    *                  b = 97.7132 +/- 0.104881
                                    *                tau = 1 (fixed)
                                    *                eff = 1 (fixed)
                                    *  Units of G        = 3.21368e-15
                                    *  -2logL_min        = 16.0638
                                    ****************************************
```

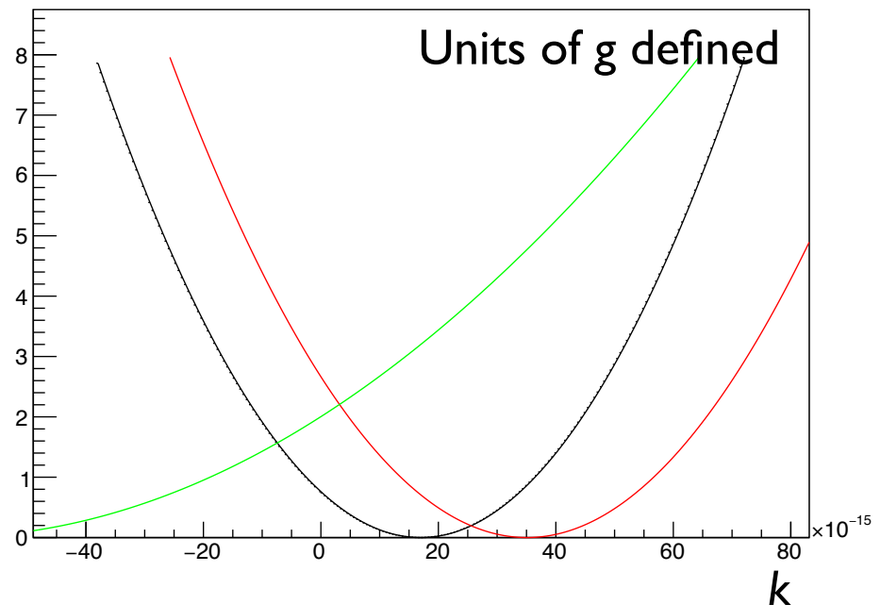
By construction, *g* in `JointLkl` is identical to *g* of the first `Lkl`-based element of the list (`PoissonLkl_0` in this case)

For the rest of the elements of the list, *g* is computed so that *g**`fUnitsOfG` has the same value for all elements

In other words, the actual common free parameter is *g**`fUnitsOfG` rather than *g*

GetLklVsG()->Draw()





IactldUnbinnedLkl

- `IactldUnbinnedLkl` implements the following likelihood function*:

$$\mathcal{L}(g;b,\tau|\{E'_i\}_{i=1,\dots,N_{\rm on}},N_{\rm on},N_{\rm off},\mu_\tau,\sigma_\tau) = {\rm Pois}(N_{\rm on}|g+b)\times{\rm Pois}(N_{\rm off}|\tau b)\times{\rm Gaus}(\tau|\mu_\tau,\sigma_\tau)\times\prod_{i=1}^{N_{\rm on}} f(E'_i)$$

  - with $f(E) = \dfrac{g\frac{dN_g}{dE'dt}(E') + b\frac{dN_b}{dE'dt}(E')}{g+b}$ and $dN_{g,b}/dE'dt$ the signal and background rates vs measured E, respectively
  - (*) actually the total likelihood is a bit more complex, but in most practical cases reduces to this expression
- As in the `PoissonLkl` case, $dN_g/dE'dt$ can be computed convoluting the spectrum with IRF and dividing by the observation time, and $dN_b/dE'dt$ modelled from Off data
- `IactldUnbinnedLkl` is declared with a string containing the name of the input file containing the event list and IRF in a single file, in the `IactEventListIRF` format (see more later).
- The class is optimised for DM searches:
  - One needed input is the average dN_g/dE of reactions producing gamma rays at the source (which appears in the formula of the DM flux).
  - It provides member functions to set the `fUnitsOfG` for annihilation and decay processes

IactldBinnedLkl

- ★ `IactldBinnedLkl` is the binned version of `IactldUnbinnedLkl`, it implements either of these two likelihood functions:

$$\mathcal{L}(g;\{b_i,\tau_i\}_{i=1,\ldots,N_{\rm bins}}|\{N_{{\rm on}i},N_{{\rm off}\,i}\}_{i=1,\ldots,N_{\rm bins}},\mu_\tau,\sigma_\tau)=\prod_{i=1}^{N_{\rm bins}}{\rm Pois}(N_{\rm on}|g_i(g)+b_i)\times{\rm Pois}(N_{\rm off}|\tau_i b_i)\times{\rm Gaus}(\tau_i|\mu_\tau,\sigma_\tau)$$

(for <u>energy-dependent</u> systematic uncertainties on background determination), or

$$\mathcal{L}(g;\{b_i\}_{i=1,\ldots,N_{\rm bins}},\tau|\{N_{{\rm on}i},N_{{\rm off}\,i}\}_{i=1,\ldots,N_{\rm bins}},\mu_\tau,\sigma_\tau)={\rm Gaus}(\tau|\mu_\tau,\sigma_\tau)\times\prod_{i=1}^{N_{\rm bins}}{\rm Pois}(N_{\rm on}|g_i(g)+b_i)\times{\rm Pois}(N_{\rm off}|\tau_i b_i)$$

(for <u>energy-independent</u> systematic uncertainties on background determination)

- ★ Which one is used (and other details) depends on how we configure the instantiated `IactldBinnedLkl` objects (with constructor and/or Setters)

Simple DM analysis

```
void testDMSearches()
{
  // input data
  const Double_t logJ         = 19.;   // [GeV^2 cm^-5] log_10 of J-factor of the assumed DM source
  const Double_t DlogJ        = 0;     // [GeV^2 cm^-5] statistical error in log_10 of J-factor of the assumed DM source
  const Double_t mass         = 1000.; // [GeV] mass of the DM particle
  const TString  dNdEFileName = TString(Form("./DM/dNdE/Cirelli/dNdESignal_bb_%.1fmass.root",mass)); // dN/dE input file
  const TString  inputFile1   = "./data/genericIact_dataIRF_01.root";  // input file with event list and their associated IRF
  const Double_t errorDef     = 4;

  // create and configure an Iact1dUnbinnedLkl object for 1D unbinned likelihood analysis
  Iact1dUnbinnedLkl* unbn = new Iact1dUnbinnedLkl(Form("logJ=%.2f DlogJ=0 inputfile=%s",logJ,inputFile1.Data()));
  unbn->ReaddNdESignal(dNdEFileName);
  unbn->SetDMAnnihilationUnitsForG(mass);    // set units for DM annihilation <sv>

  // create and configure an Iact1dBinnedLkl object for 1D unbinned likelihood analysis
  Iact1dBinnedLkl* bn = new Iact1dBinnedLkl(Form("logJ=%.2f DlogJ=0 inputfile=%s",logJ,inputFile2.Data()));
  bn->ReaddNdESignal(dNdEFileName);
  bn->SetDMAnnihilationUnitsForG(mass);    // set units for DM annihilation <sv>

  // create and fill a JointLkl object for the combined analysis of both datasets
  JointLkl* jointLkl = new JointLkl(Form("DlogJ=%.2f",DlogJ));
  jointLkl->SetErrorDef(errorDef);
  jointLkl->AddSample(unbn);
  jointLkl->AddSample(bn);

   // print how JointLkl looks like
  jointLkl->PrintData();

  // call minimization and print/plot results
  jointLkl->ComputeLklVsG();
  jointLkl->PrintOverview();     // print the details from the fit
  TCanvas* c1 = new TCanvas("c1","",700,500);
  jointLkl->GetLklVsG()->Draw(); // plot the -2logL vs g curve
}
```

Define J-factor, m_{DM}, input files for data and dN/dE…

Create Iact1dUnbinned object and configure it for DM annihilation searches

Create Iact1dBinned object and configure it for DM annihilation searches

Create JointLkl object, fill it with lkl objects and call minimisation

Show results

Doing a full DM analysis is repeating this for more masses, allow some options…

Output of simple DM analysis

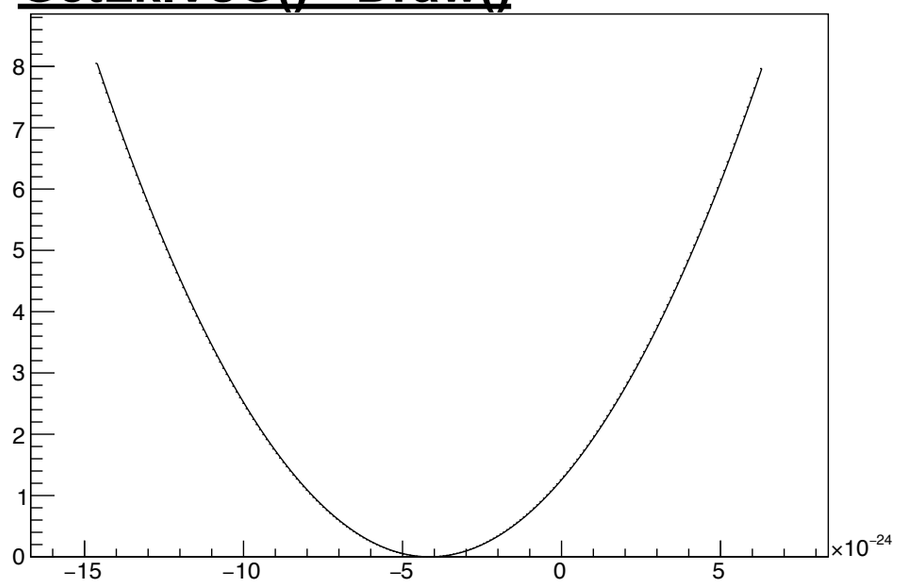
PrintData

```
*             Object Name : JointLkl
*            # of samples = 2
****************************************
                                    *             Object Name : Iact1dUnbinnedLkl
                                    *               E' range  = [59.2,10000] GeV
                                    *                Non/Noff = 11468/11526
                                    *            Measured tau = 1.00251 +/- 0.00574
                                    *         Observation time = 50 h
                                    *                  log(J) = (19 +/- 0) GeV^2 cm^-5  or  GeV cm^-2
                                    *              Aeff histo : YES
                                    *           Off Aeff histo : NO
                                    *       E Reso/Bias graphs : YES
                                    *         Migration matrix : NO
                                    *    Background rate histo : YES
                                    *    Foreground rate histo : NO
                                    *       Signal dN/dE histo : YES
                                    *      Signal dN/dE' histo : YES
                                    *  Signal dN/dE' Off histo : NO
                                    ****************************************
                                    *             Object Name : Iact1dBinnedLkl
                                    *               E' range  = [59.2,10000] GeV
                                    *                Non/Noff = 11576/11616
                                    *            Measured tau = 1.00402 +/- 0.00599
                                    *         Observation time = 45 h
                                    *                  log(J) = (19 +/- 0) GeV^2 cm^-5  or  GeV cm^-2
                                    *              Aeff histo : YES
                                    *           Off Aeff histo : NO
                                    *       E Reso/Bias graphs : YES
                                    *         Migration matrix : NO
                                    *    Background rate histo : YES
                                    *    Foreground rate histo : NO
                                    *       Signal dN/dE histo : YES
                                    *      Signal dN/dE' histo : YES
                                    *  Signal dN/dE' Off histo : NO
                                    *                # of bins = 10
                                    *                Rebinned? = YES (to 1 min entries)
                                    ****************************************
```

PrintOverview

```
*  Name               = JointLkl
*  Status             = 0 (converged)
*  Delta(2logL)       = 4
*  # of parameters    = 1 (1 free):
*                   g = -83.6586 +/- 150.017 (-4.10051e-24 +/- 7.35306e-24)
*  Units of G         = 4.90148e-26
*  -2logL_min         = 276367
*  # of samples       = 2
****************************************
                                    *  Name               = Iact1dUnbinnedLkl
                                    *  Status             = 0 (converged)
                                    *  Delta(2logL)       = 1
                                    *  # of parameters    = 3 (2 free):
                                    *                   g = -83.6586 (-4.10051e-24) (fixed)
                                    *                   b = 11531.8 +/- 10.7359
                                    *                 tau = 1.00121 +/- 0.000574
                                    *  Units of G         = 4.90148e-26
                                    *  -2logL_min         = 276249
                                    ****************************************
                                    *  Name               = Iact1dBinnedLkl
                                    *  Status             = 0 (converged)
                                    *  Delta(2logL)       = 1
                                    *  # of parameters    = 2 (1 free):
                                    *                   g = -75.2553 (-4.10051e-24) (fixed)
                                    *                 tau = 1.00283 +/- 0.00599
                                    *  Units of G         = 5.4488e-26
                                    *  -2logL_min         = 117.862
                                    ****************************************
```

GetLklVsG()->Draw()



IactEventListIRF data format

Data members:

```
TNtuple* fOnSample;        //-> ON  event list ("Energy:Point_RA:Point_DEC:Delta_RA:Delta_DEC:Arr_time:Hadronness")
TNtuple* fOffSample;       //-> OFF event list ("Energy:Point_RA:Point_DEC:Delta_RA:Delta_DEC:Arr_time:Hadronness")

Float_t  fEpmin;           // [GeV] minimum estimated energy
Float_t  fEpmax;           // [GeV] maximum estimated energy
Float_t  fTau;             // normalization Noff/Non (e.g # of off regions)
Float_t  fDTau;            // statistical error in fTau
Float_t  fTauPValue;       // Probability value of agreement between On/Off
Float_t  fObsTime;         // [s] observation time

TH1F*    fHAeff       ;    //-> Effective area vs E histogram for signal events
TH1F*    fHAeffOff    ;    //-> Effective area vs E histogram for signal events IN THE OFF REGION
TGraph*  fGEreso      ;    //-> Graph with energy resolution vs energy values
TGraph*  fGEbias      ;    //-> Graph with energy bias vs energy values
TH2F*    fMigMatrix   ;    //-> Migration matrix (overrides fGEreso and fGEbias)

TH1F*    fHdNdEpBkg   ;    //-> dN/dE'dt vs E' for background events (normalized)
TH1F*    fHdNdEpFrg   ;    //-> dN/dE'dt vs E' for foreground (gamma-events from a different source) events (normalized)
```

With fOnSample and fOffSample n-tuples with events formed by structure:

```
// structure to read the events from the NTuple
typedef struct {
  Float_t  E;         // [GeV] measured energy of event
  Float_t  pointRA;   // [deg] RA of telescope pointing direction
  Float_t  pointDEC; // [deg] DEC of telescope pointing direction
  Float_t  dRA;       // [deg] distance to pointing direction in the RA axis
  Float_t  dDEC;      // [deg] distance to pointing direction in the DEC axis
  Float_t  t;         // [MJD] arrival time
  Float_t  had;       // [1]   hadronness
} IactEvent_t;
```

Data with this format is produced, e.g., by gLikeInputs in MARS

gLike non-expert user

jointLklDM

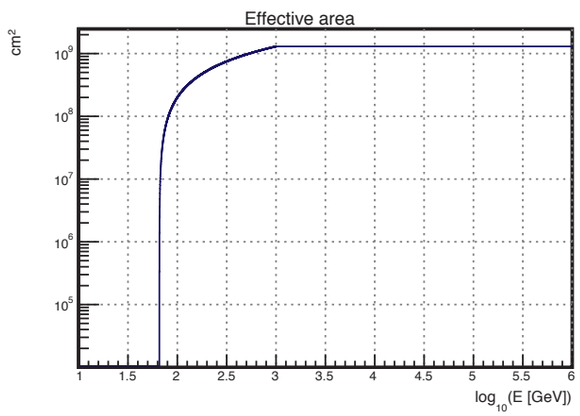
- Experts are expected to run gLike executables, that read a configuration (rc) file
- So far, no executables, but just one macro: jointLklDM.C, for DM analyses
- scripts/jointLklDM.C macro to be run with rcfiles/jointLklDM.rc
- As other macros, jointLklDM.C **<u>should always be run in compiled mode</u>**
- jointLklDM.C is nothing but a sophisticated version of testDMSearches.C, where all possible configuration options are available through the rc file
- jointLklDM.C computes bounds to the DM annihilation <sv> or decay rate 1/tau, for a given channel, and a list of masses
- Usage (in root command line): `jointLklDM.C+(<rcfile>,<rndseed>)`
  - If random seed is provided, it will produce MonteCarlo mock data according to the observation time and IRFs of all input files. Otherwise it will use the event lists in those input files. This is useful for producing confidence bands.

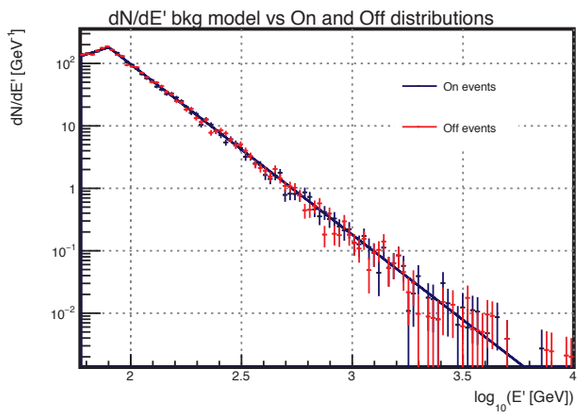
the rc file

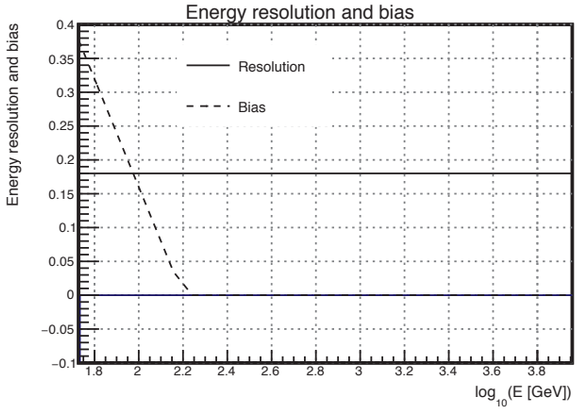
```
#################
I/O CONFIGURATION
#################

# Label to indentify this analysis (for naming plots and output files)
jointLklDM.Label: GenericIACT_Annihilation_bb_Unbinned_GisFree_JisFixed

# Path (will be added to all specified files and directories in this rcfile)
jointLklDM.path: ./

# Directory with dN/dE files
jointLklDM.dNdEDir: DM/dNdE/Cirelli/

# Directory for output plots
jointLklDM.plotsDir: plots
#####################
ANALYSIS CONFIGURATION
#####################

# channel to be analyzed (bb, tautau, mumu, WW, gammagamma, pi0pi0 or pi0gamma)
jointLklDM.Channel: bb

# process to be analyzed ("ann" or "dec" for annhilation and decay, respectively, default is annihilation
# (remember to change consistently the list of masses and J-factor value and uncertainty)
jointLklDM.Process: ann

# list of DM masses to be considered (in GeV)
jointLklDM.MassList:  100  200 500  1000  2000  5000  10000  20000  50000  100000

##############################
JOINT LIKELIHOOD CONFIGURATION
##############################

# Now include one line per likelihood function term, with the following general syntaxis:
#    jointLklDM.lklTerm<XXX>: <termType> <parentTerm> [<options>]
#
jointLklDM.lklTerm000: JointLkl -   DlogJ=0
jointLklDM.lklTerm001: lact1dUnbinnedLkl    0   logJ=19.  DlogJ=0  inputfile=data/genericlact_dataIRF_01.root
jointLklDM.lklTerm002: lact1dBinnedLkl      0   logJ=19.  DlogJ=0  inputfile=data/genericlact_dataIRF_02.root  nbins=20
jointLklDM.lklTerm003: FermiTables2016Lkl   0   logJ=19.  DlogJ=0  inputfile=fermi_p302_dsphs_like_all/like_segue_1.txt
```

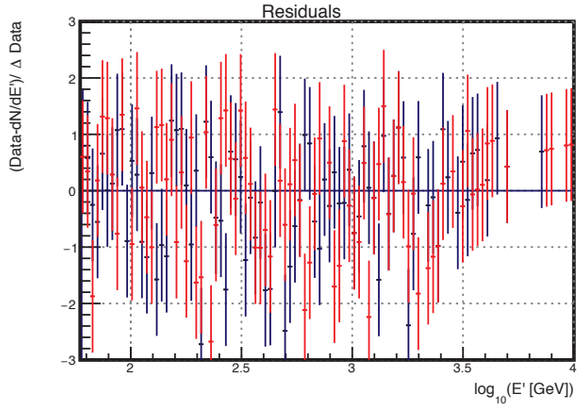
jointLklDM.C output (1/3)

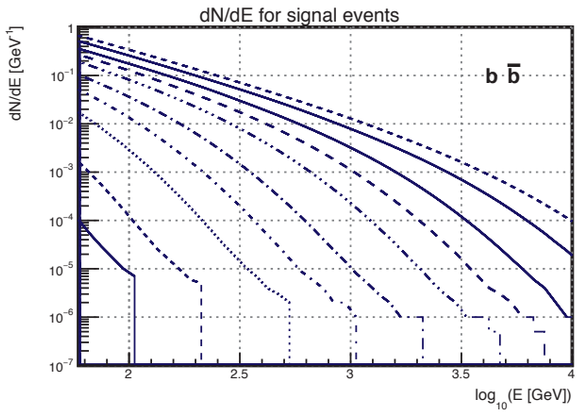
IactdUnbinnedLkl

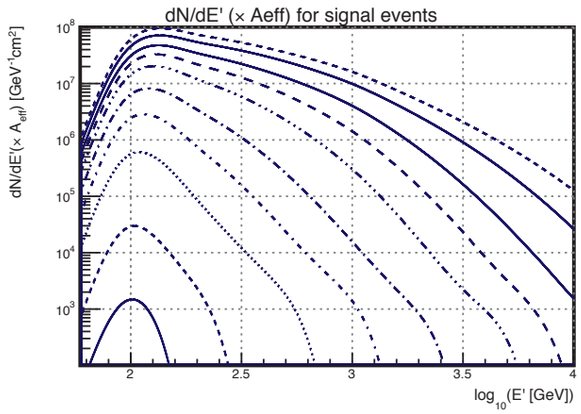




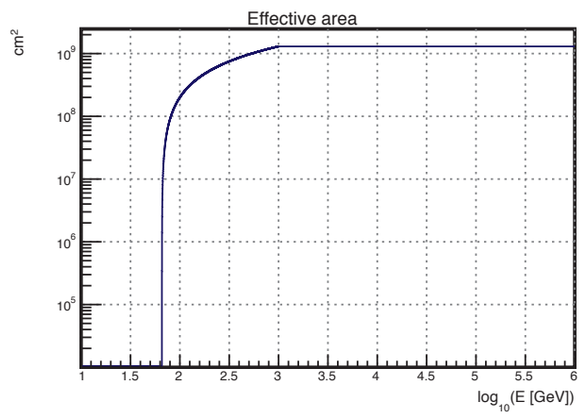


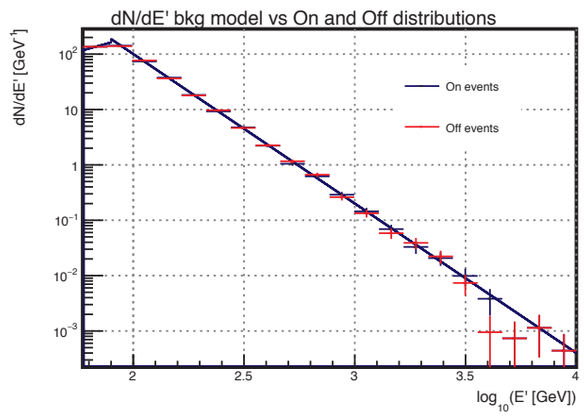


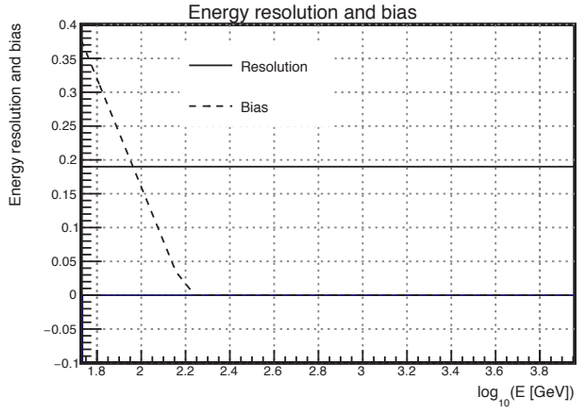


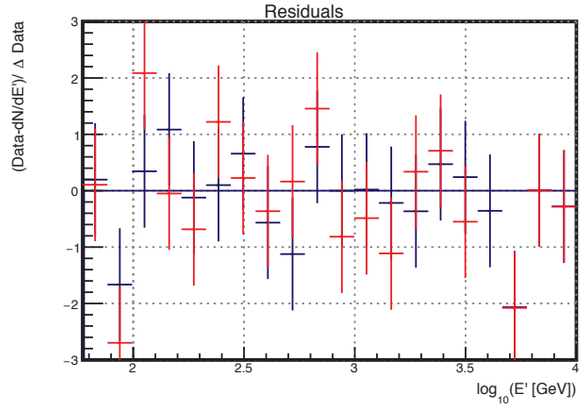


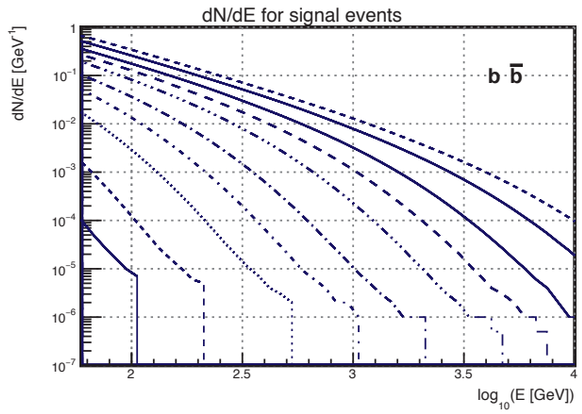
IactdBinnedLkl

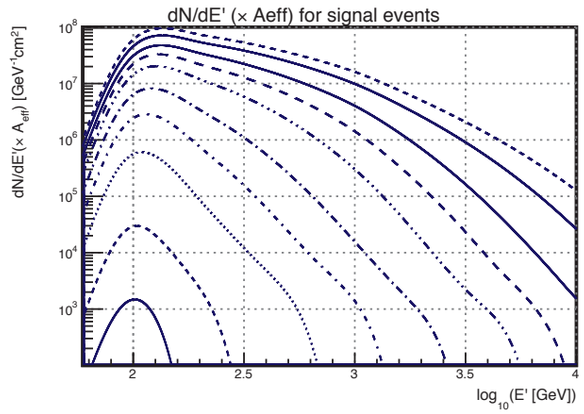




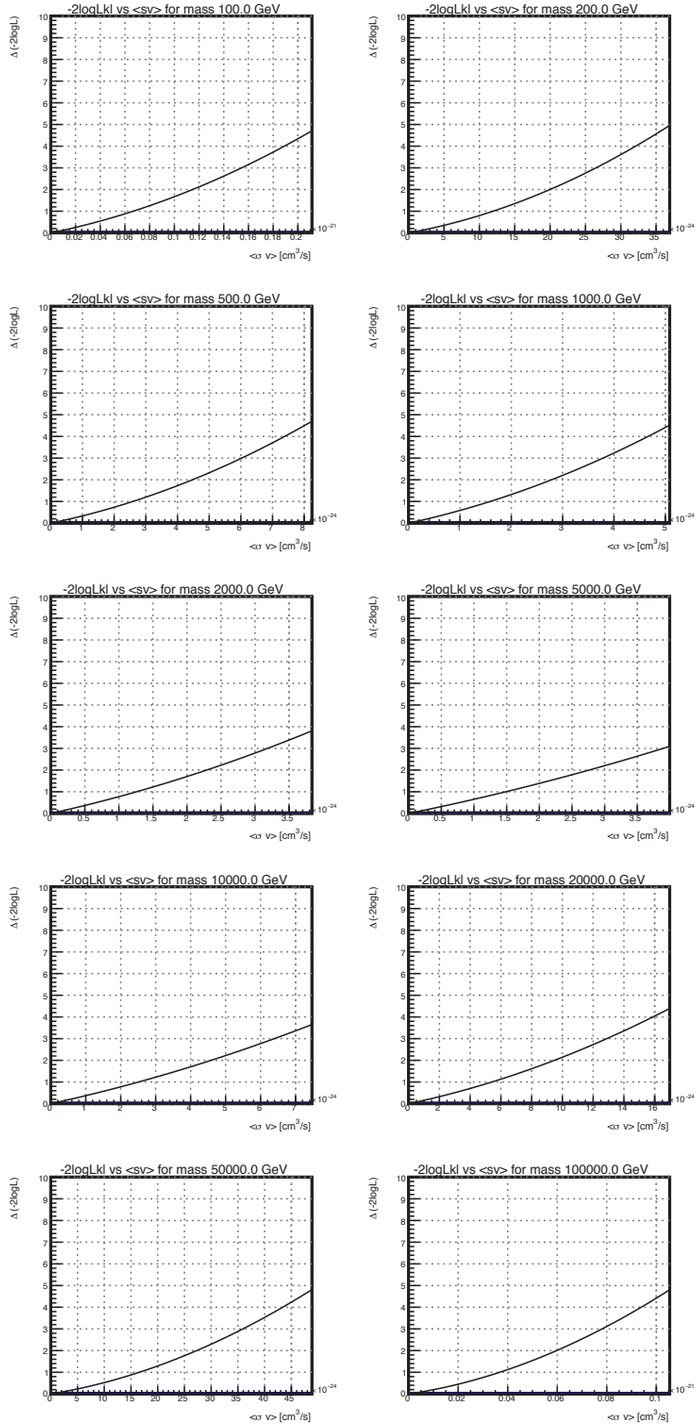




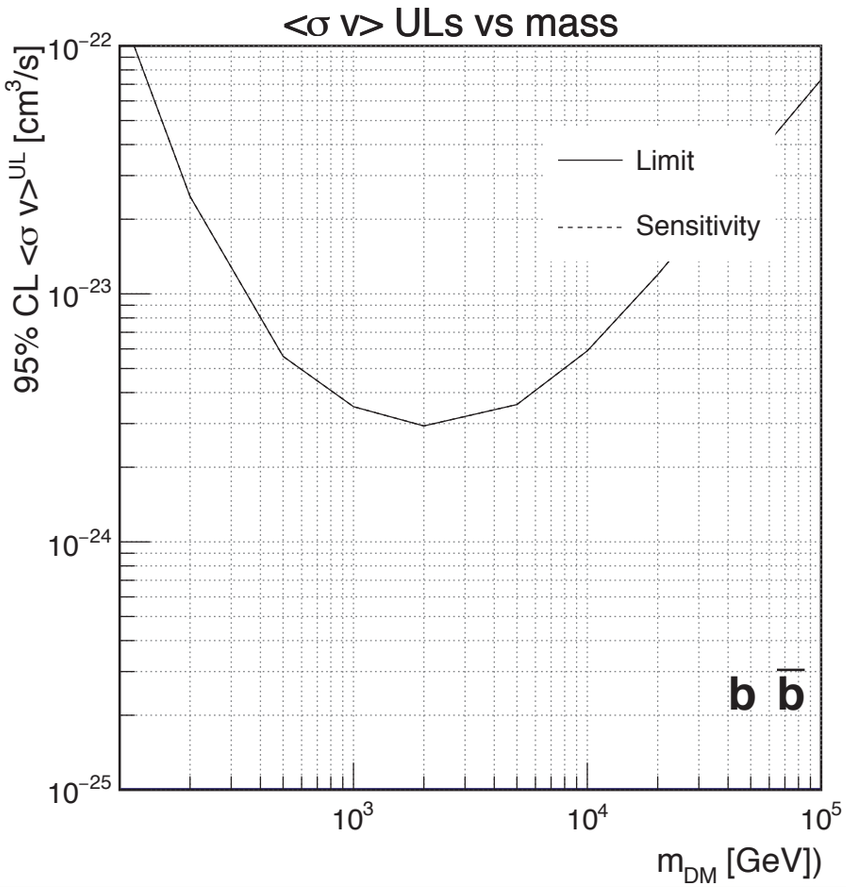




jointLklDM.C output (2/3)



jointLklDM.C output (3/3)




```
READY-TO-COPY results:

Double_t mass[nmass]  = {100,200,500,1000,2000,5000,10000,20000,50000,100000};
Double_t limit[nmass]  = {1.43531e-22,2.47085e-23,5.59858e-24,3.5092e-24,2.9348e-24,3.57842e-24,5.89093e-24,1.19476e-23,3.35726e-23,7.30914e-23};
Double_t snstvt[nmass]  = {1.43531e-22,2.47085e-23,5.59858e-24,3.5092e-24,2.9348e-24,3.57842e-24,5.89093e-24,1.19476e-23,3.35726e-23,7.30914e-23};

********************************
Joint likelihood results
********************************

<sv> limit vs mass
****************************************
mass = 100 GeV, <sv>^UL = 1.43531e-22, <sv>_snstvty = 1.43531e-22 cm^3s-1
mass = 200 GeV, <sv>^UL = 2.47085e-23, <sv>_snstvty = 2.47085e-23 cm^3s-1
mass = 500 GeV, <sv>^UL = 5.59858e-24, <sv>_snstvty = 5.59858e-24 cm^3s-1
mass = 1000 GeV, <sv>^UL = 3.5092e-24, <sv>_snstvty = 3.5092e-24 cm^3s-1
mass = 2000 GeV, <sv>^UL = 2.9348e-24, <sv>_snstvty = 2.9348e-24 cm^3s-1
mass = 5000 GeV, <sv>^UL = 3.57842e-24, <sv>_snstvty = 3.57842e-24 cm^3s-1
mass = 10000 GeV, <sv>^UL = 5.89093e-24, <sv>_snstvty = 5.89093e-24 cm^3s-1
mass = 20000 GeV, <sv>^UL = 1.19476e-23, <sv>_snstvty = 1.19476e-23 cm^3s-1
mass = 50000 GeV, <sv>^UL = 3.35726e-23, <sv>_snstvty = 3.35726e-23 cm^3s-1
mass = 100000 GeV, <sv>^UL = 7.30914e-23, <sv>_snstvty = 7.30914e-23 cm^3s-1
```

Conversion of files into v00.05 and beyond

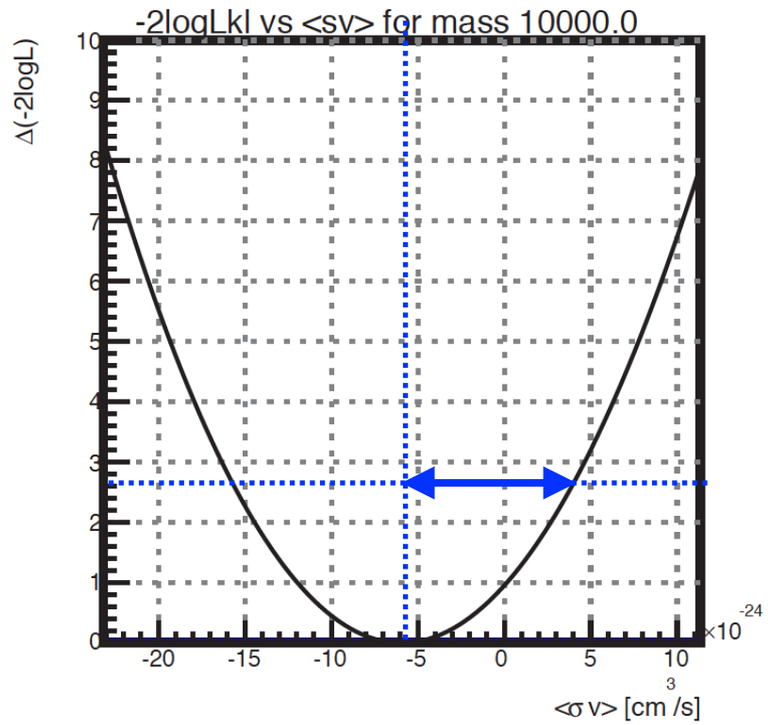
- To allow for gLike future expansion, several actions have been taken:
  - gLike classes removed from the dark matter oriented mdm library and moved to gitHub
  - Naming convention of classes has changed:

Old name	New name
MLkl	Lkl
MJointLkl	JointLkl
MFullLkl	Iact1dUnbinnedLkl
MBinnedFullLkl	Iact1dBinnedLkl
MBinnedFluxLkl	FermiTables2016L
MPoissonLkl	PoissonLkl
MParabolaLkl	ParabolaLkl
MIACTEventListI	IactEventListIRF

- These changes should be transparent to the user except for two aspects:
  - Class names are used in the rcfile, and the new naming convention should be used from now on.
  - Data+IRF input files produced in the format defined by `MIACTEventListIRF` will not work anymore. New files use format defined in `IactEventListIRF`. Convert your old files into new ones with the macro `convertGLikeInputFiles.C`

computing sensitivity

- The sensitivity (for a given confidence level, CL) is the average limit (with that CL) we would obtain on the free parameter, under the null hypothesis.
- There are two ways to estimate the sensitivity:
  - More accurate but slower: produce as many independent Monte Carlo simulated samples as needed by the CL and compute for each the limit to *g*. The estimated sensitivity is the average of the obtained limits.
  - Less accurate but faster: take the difference between the value of *g* for which -2logL takes the value corresponding to the CL and the for which -2logL minimises. This approach is acceptable if we have a good description of the -2logL parabolic shape (i.e. jointLkIDM.isGpositive: FALSE and DlogJ=0).



gLike developer

The TemplateLkl class

- ★ The class TemplateLkl, provided with gLike distribution, provides the minimal skeleton for including your favourite likelihood function

```
class TemplateLkl : public Lkl
{
 public:

  // constructors
  TemplateLkl(TString inputString="");

  // destructor
  virtual ~TemplateLkl();

 protected:
          Int_t    InterpretInputString(TString inputString);
  virtual void     SetFunctionAndPars(Double_t ginit=0);
  virtual Int_t    MakeChecks();
  virtual void     SetMinuitLink();


 private:

  ClassDef(TemplateLkl,1) // Full Likelihood (unbinned)
};
```