Diana bootcamp: tools

Berkeley CUORE general meeting - Fall 2019
L. Marini

Continuous data

QRaw_RDCF_302210_R_C1_p001.root

- Run number: 1st digit - experiment specifier: 3 = CUORE, 2 = CUORE-0,
  9 = Unknown; 6 = CUPID-0; 8 = CUPID-Mo
  2nd digit >=5 are retriggered runs
- Run type: C = Calibration, B = Background, T = Test, S = Setup, N = NPulser, R = Retriggered
- DAQ crate: 6 in CUORE
- Partial: depends on the length of the run (each partial is 500MB)
- Let’s inspect it in ROOT

```
cd /global/homes/l/lmarini/cuoresw/ContinuousData/run302210/
root QRaw_RDCF_302210_R_C1_p001.root
f.ls();
f.cd(“Global”);
```

Continuous files on CORI: /global/homes/l/lmarini/cuoresw/ContinuousData/
Continuous files on CNAF:/storage/gpfs_data/cuore/data/cuore3/data/CUORE/ContinuousData/

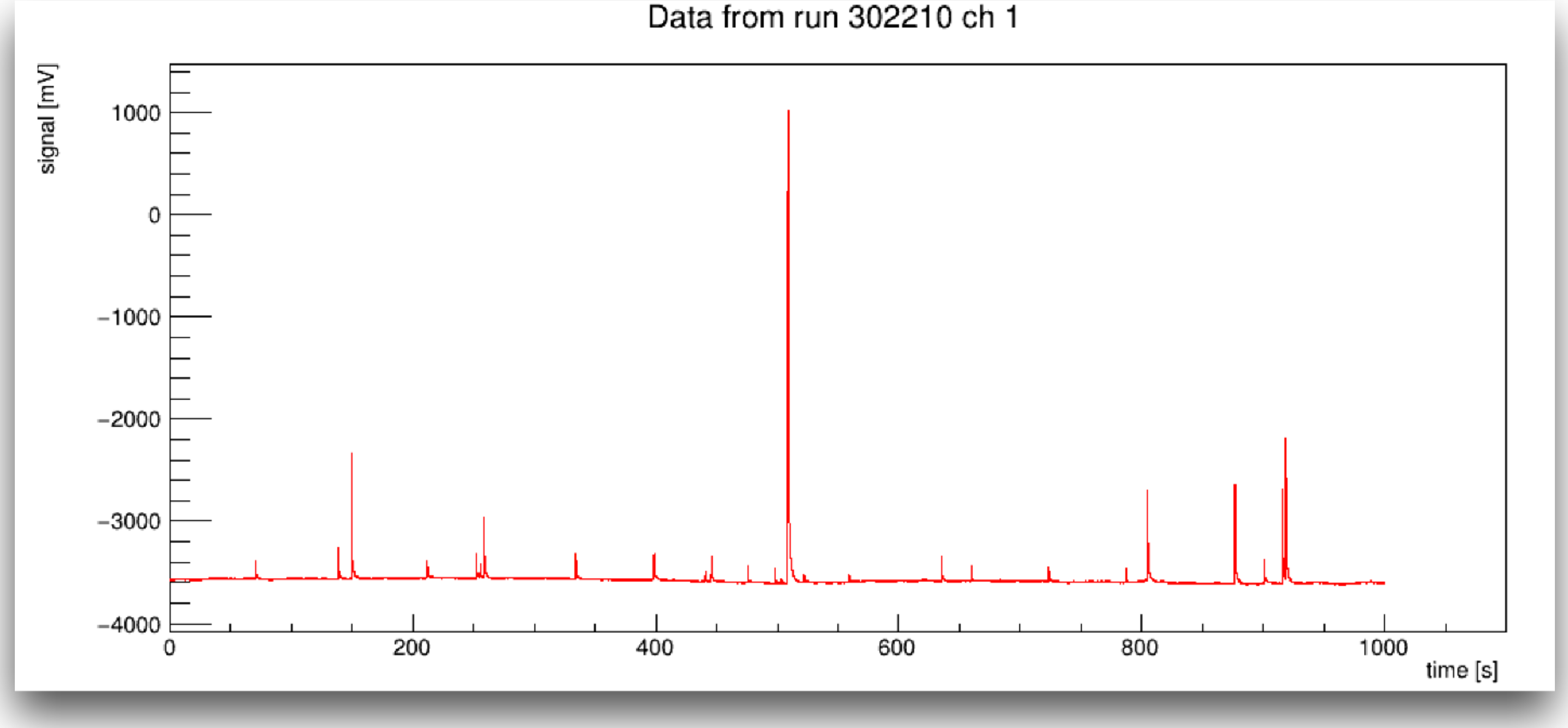
Continuous files on ULITE:/nfs/cuore3/data/CUORE/ContinuousData/

qfile executable

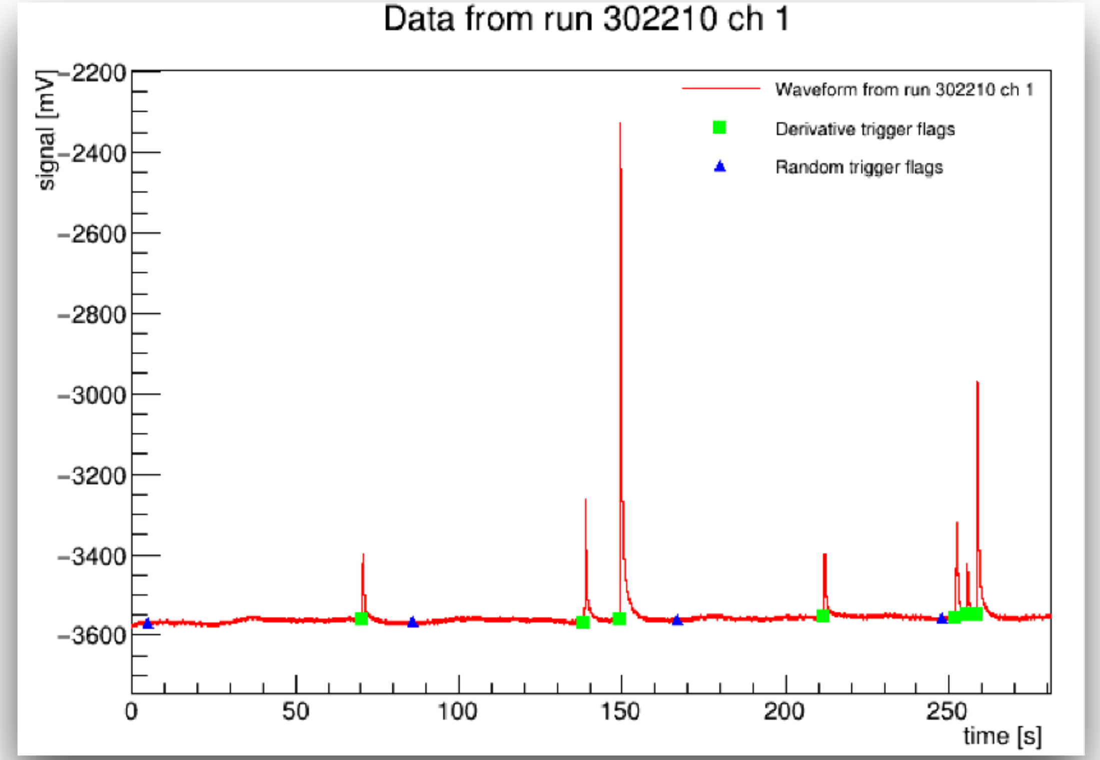
Load Diana software: cd cuoresw; source setup;
qfile code is in cuoresw/pat/qfile/

```
(virtual) [lmarini@cori10 cuoresw]$ qfile -h

RooFit v3.60 -- Developed by Wouter Verkerke and David Kirkby
                Copyright (C) 2000-2013 NIKHEF, University of California & Stanford University
                All rights reserved, please read http://roofit.sourceforge.net/license.txt


        Usage: qfile [OPTION]...

        Description: display signal of selected channel within a time interval

        Options:
        -d, --directory      specify the continuous file directory
        -r, --run            specify the run number
        -p, --partial        specify the partial file: in this case start and stop are automatically set
        -t, --start          specify start time from the beginning of run (seconds)
        -T, --stop           specify stop time from the beginning of run (seconds)
        -S, --stride         specify a sample stride (skip N samples)
        -c, --channel        specify channel id
        -R, --random         display random trigger flags
        -D, --derivative     display derivative trigger flags
        -s, --savePDF        save image to pdf file
        -l, --log            write output file with samples values
        -h, --help           Display this help and exit.
        -v, --version        Output version information and exit.
```

```
qfile -d ContinuousData/run302210/ -r 302210 -c 1 -S 10 -t 0 -T 1000
```



```
qfile -d ContinuousData/run302210/ -r 302210 -c 1 -t 0 -T 500 -R -D
```



qfile library

- Never load libqfile 1st, load sth else first!!!

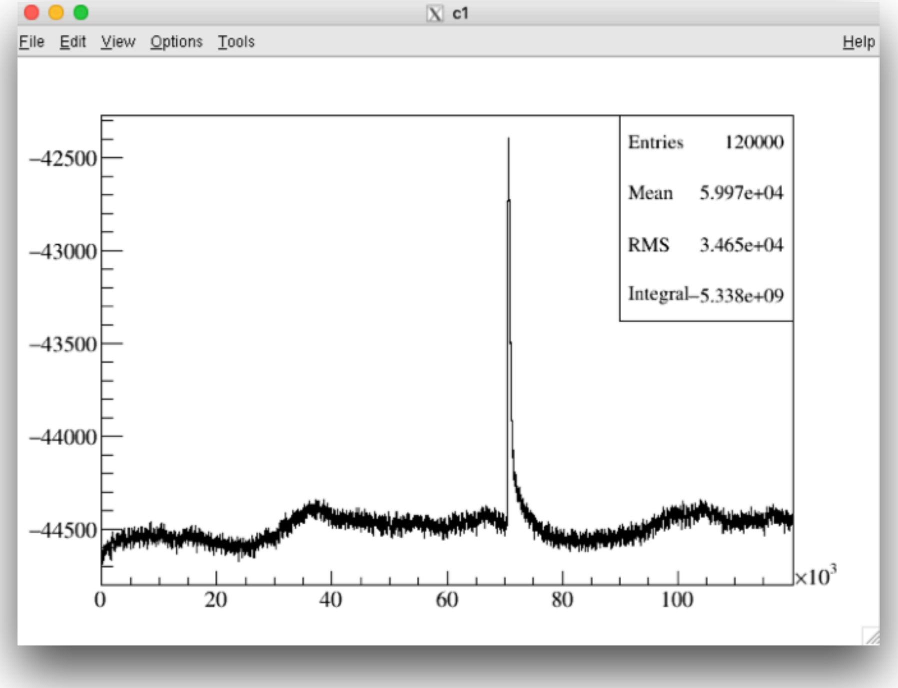
```
gSystem->Load(“libqroot");
gSystem->Load(“libqfile”);
```

- Main function to use:
  Cuore::QVector GetRawWaveform(int run, int channel, double start_time, double stop_time, const string& source = "DB", int stride = 1);
  - times can be either from run_start or UnixTime (seconds since 1/1/1970)
  - source: “DB” by default, but can also be a directory

Now you can use libqroot also in your own software!

qfile library

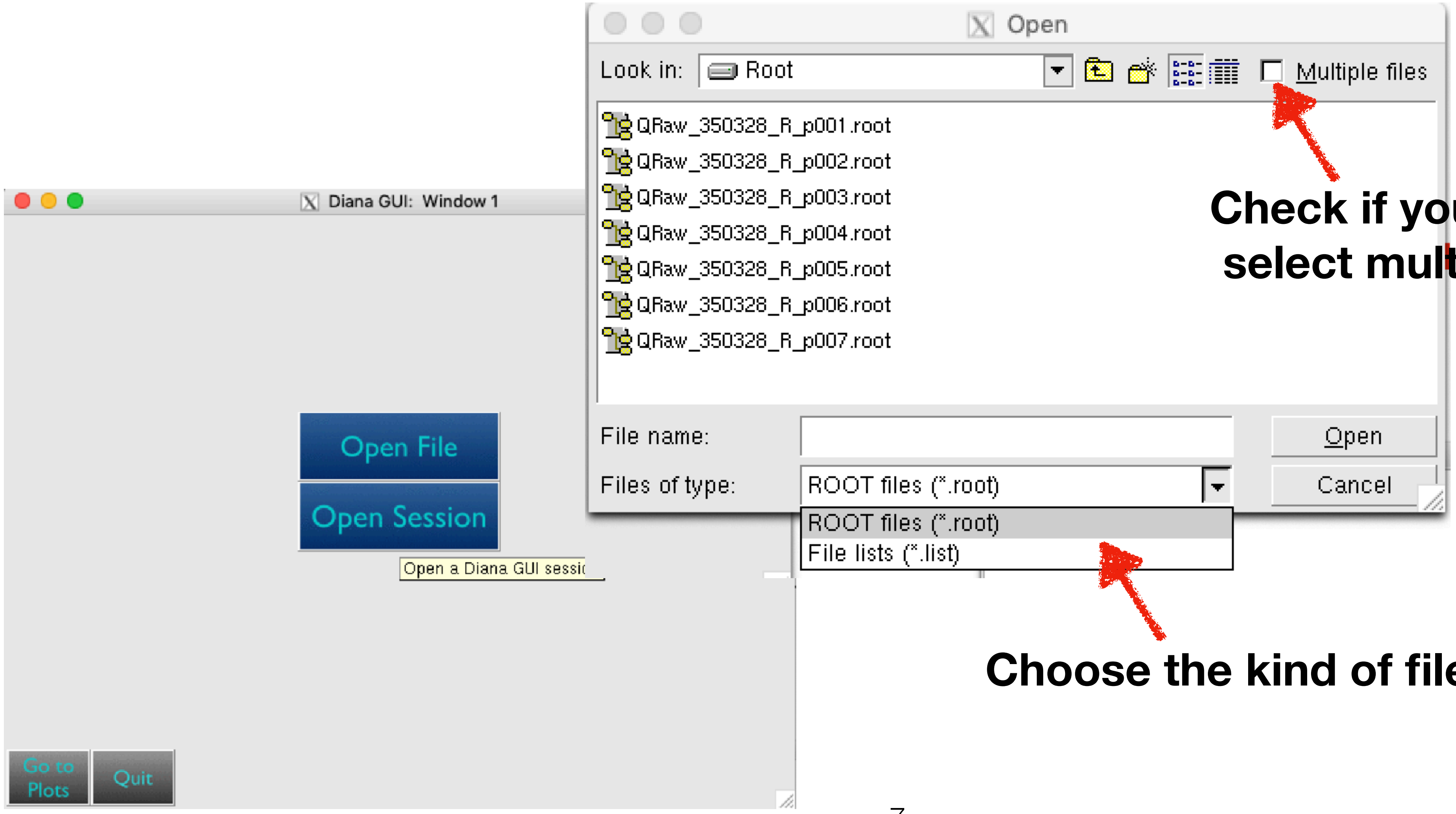
```
[lmarini@cuore-login1 analysis]$ root -l
QStyles: Style"qprod" has been set
***********************************************
* Diana root library and include paths loaded *
***********************************************
root [0] gSystem->Load("libqroot")
(int)1
root [1] gSystem->Load("libqfile")
(int)0
root [2] Cuore::QVector wf = GetRawWaveform(302210, 1, 0, 120, "DB", 1)
QGlobalReaderDispatcher:       loaded default g-reader LRootGlobalReader
* filename is:      /nfs/cuore3/data/CUORE/ContinuousData/run302210/
QRaw_RDCF_302210_C_C2_p001.root
* file version:     0
* vector length is: 10000000000 ns
* run start is:     0
* run stop (from last partial) is:  2230000000000 ns
* filename is:      /nfs/cuore3/data/CUORE/ContinuousData/run302210/
QRaw_RDCF_302210_C_C2_p041.root
* file version:     0
* vector length is: 10000000000 ns
* run start is:     0
* run stop (from last partial) is:  90695920000000 ns
root [3] wf.Draw()
root [4] TGraph* g=wf.GetGraph()
```



DianaGui

Note: if connecting remotely, use X2Go or NoMachine to use graphical tools

```
[lmarini@cuore-login1 analysis]$ dianagui
```



Look in: Root

QRaw_350328_R_p001.root
QRaw_350328_R_p002.root
QRaw_350328_R_p003.root
QRaw_350328_R_p004.root
QRaw_350328_R_p005.root
QRaw_350328_R_p006.root
QRaw_350328_R_p007.root

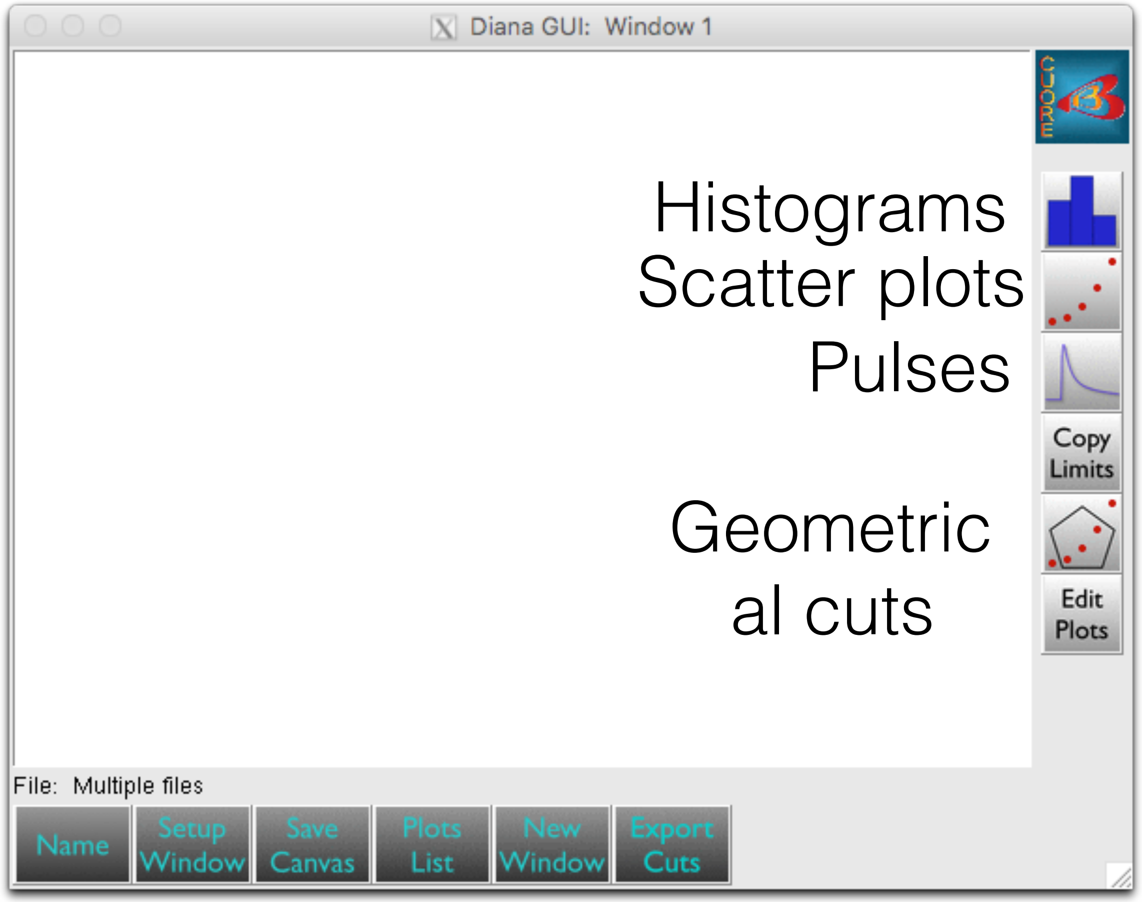
File name:

Files of type: ROOT files (*.root)

Check if you want to select multiple files

Choose the kind of file

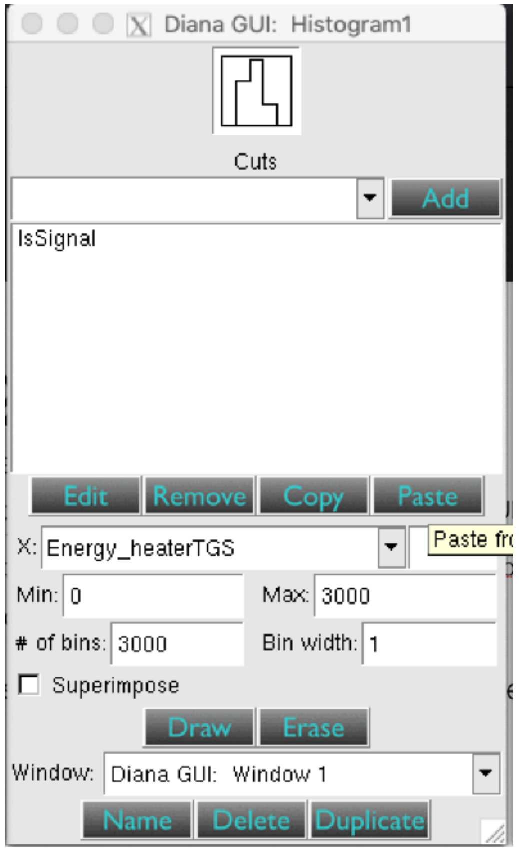
Dianagui has many options to visualise data: Features similar to root GUI


Diana GUI: Window 1

Histograms
Scatter plots
Pulses

Geometric
al cuts

File: Multiple files


Diana GUI: Histogram1

Cuts

IsSignal

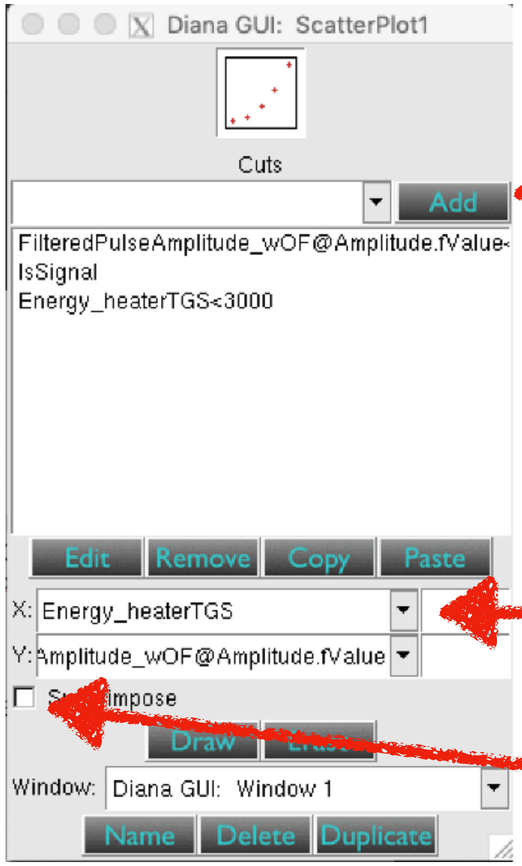
X: Energy_heaterTGS

Min: 0 Max: 3000

# of bins: 3000 Bin width: 1

□ Superimpose

Window: Diana GUI: Window 1


Diana GUI: ScatterPlot1

Cuts

FilteredPulseAmplitude_wOF@Amplitude.fValue<
IsSignal
Energy_heaterTGS<3000

X: Energy_heaterTGS

Y: Amplitude_wOF@Amplitude.fValue

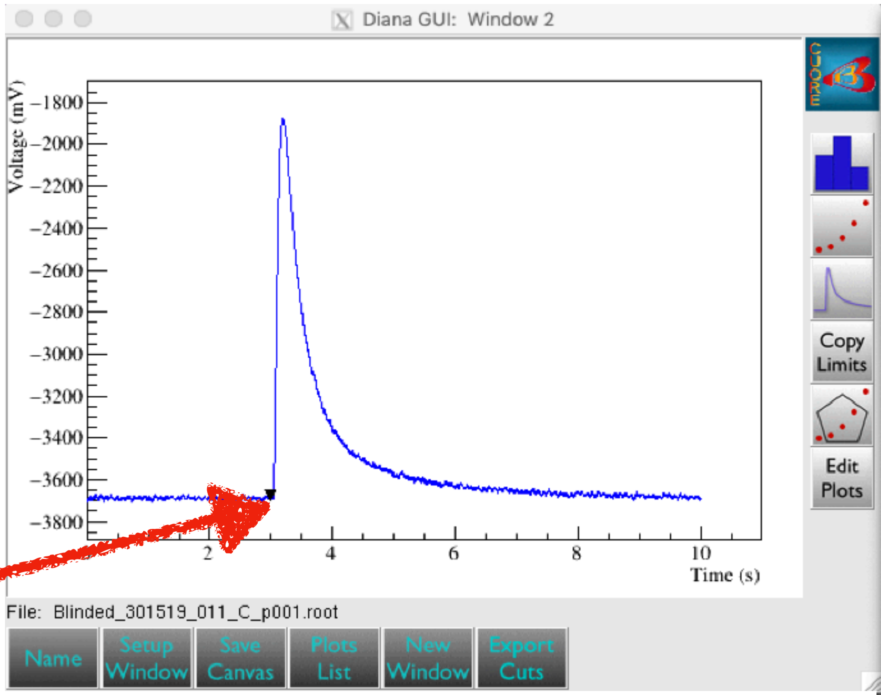
□ Superimpose

Window: Diana GUI: Window 1

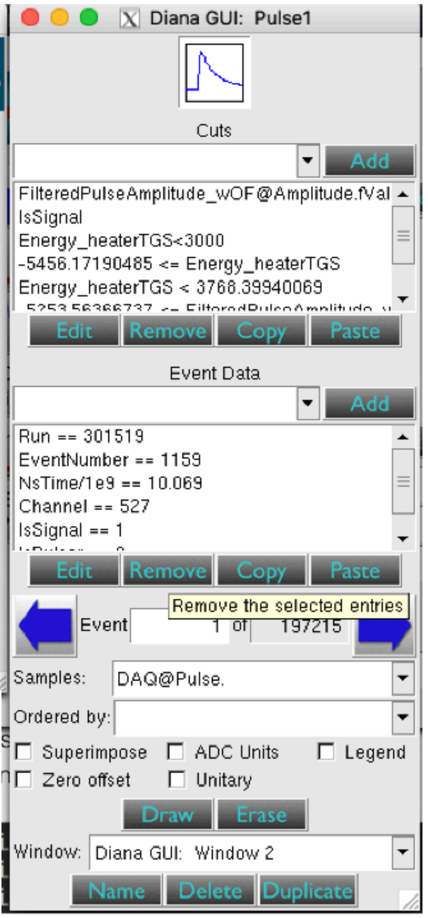
Cuts

Variables

“same” option


Diana GUI: Window 2

File: Blinded_301519_011_C_p001.root


Diana GUI: Pulse1

Cuts

FilteredPulseAmplitude_wOF@Amplitude.fVal
IsSignal
Energy_heaterTGS<3000
-5456.17190485 <= Energy_heaterTGS
Energy_heaterTGS < 3768.39940069

Event Data

Run == 301519
EventNumber == 1159
NsTime/1e9 == 10.069
Channel == 527
IsSignal == 1

Remove the selected entries

Event 1 of 197215

Samples: DAQ@Pulse.

Ordered by:

□ Superimpose □ ADC Units □ Legend
□ Zero offset □ Unitary

Window: Diana GUI: Window 2

▲ **Current trigger**

▲ **Noise event in same window**

▲ **Other trigger event in same window**

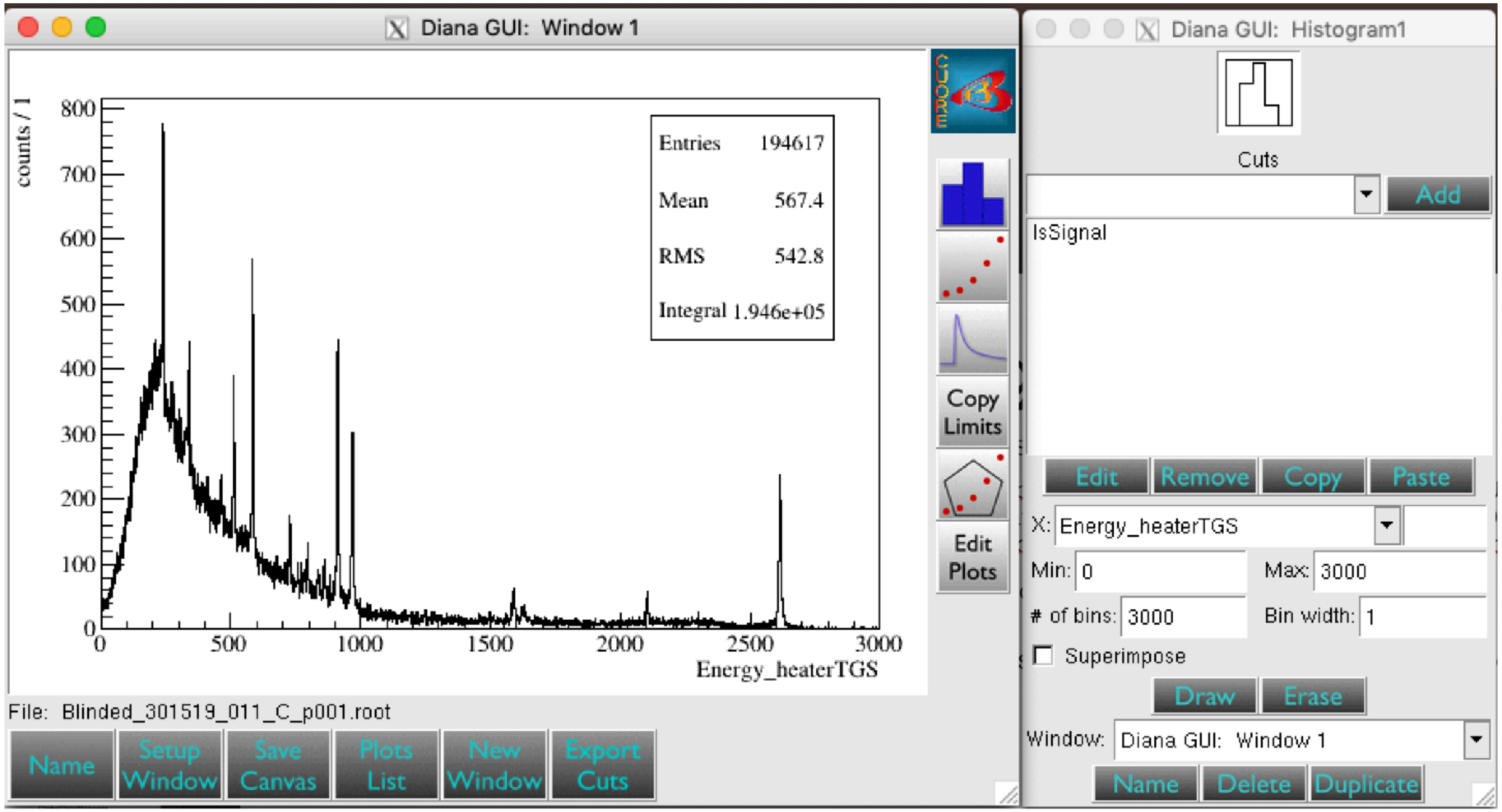
Explore processed data with dianagui:

CORI

```
/global/projecta/projectdirs/cuore/syncData/CUORE/
OfficialProcessed/TwoNu_DataRelease_Jan2019/output/ds3021/
Blinded_301519_011_C_p001.root
```

ULITE

```
/nfs/cuore3/data/CUORE/OfficialProcessed/TwoNu_DataRelease_Jan2019/
output/ds3021/Blinded_301519_011_C_p001.root
```



Entries	194617
Mean	567.4
RMS	542.8
Integral	1.946e+05

File: Blinded_301519_011_C_p001.root

IsSignal

X: Energy_heaterTGS

Min: 0

Max: 3000

# of bins: 3000

Bin width: 1

□ Superimpose

Window: Diana GUI: Window 1

RunsManager

RunsManager package for calculating live times, run times, exposure, make official exposure plots, select channels, etc.

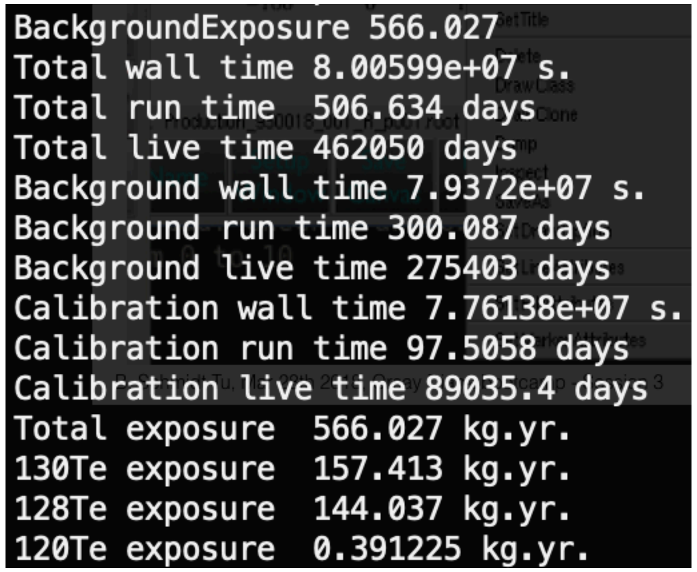
Load Diana software: cd cuoresw; source setup;
RunsManager code is in cuoresw/pat/runsmanager/
Example is in cuoresw/pat/runsmanager/MyRunsManagerExample.C
Run as: `root MyRunsManagerExample.C`

```
Cuore::QGlobalDataManager dm;
dm.SetOwner("CuoreMetaData");

QRunsManagerHandle aHan("RunsManager");
aHan.SetBadAnalysisType("Shifter");
aHan.AddDataset(3021);

aHan.GetOnlyGoodRuns(true);
dm.Get(&aHan,"DB"); // Build from DB

// Write to file
dm.Set(&aHan,"ATestFile.root");

RunsManager Runs = aHan.Get();
Runs.SetAnalysisMask(202);
```

Now you have a RunsManager container with all the info from ds3021

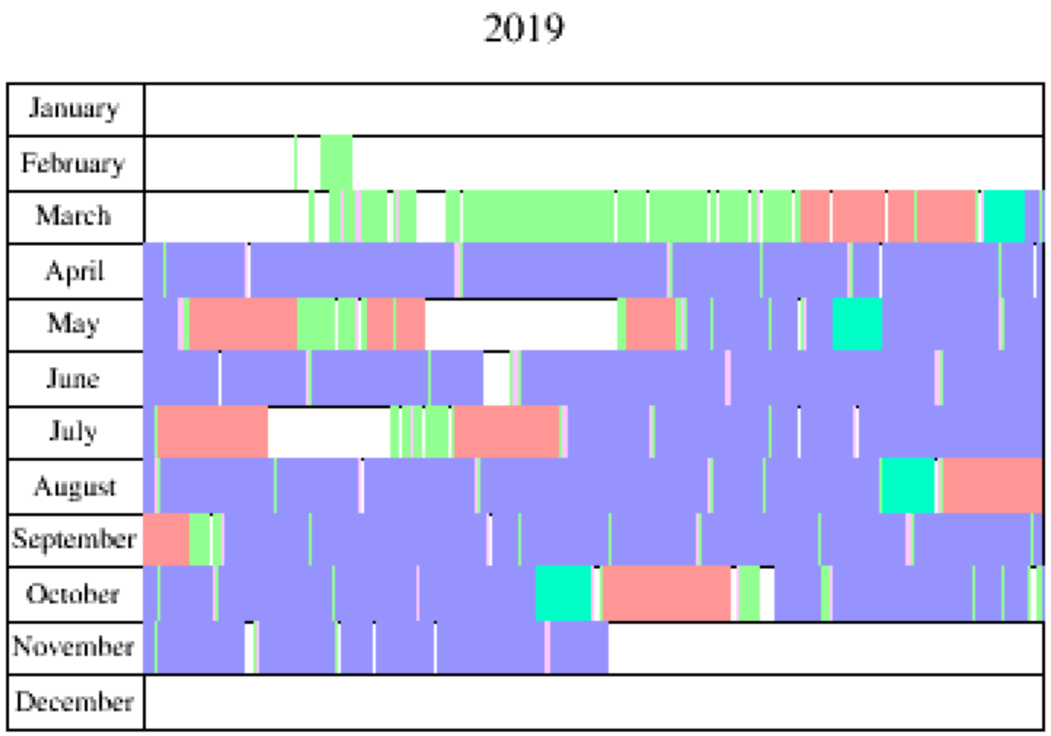
There are functions to get pretty much every information that you want:

```
  std::cout << "Total wall time " << Runs.GetLastRunEndTime()-Runs.GetFirstRunStartTime() << " s." << std::endl;
  std::cout << "Total run time  " << Runs.GetRunTime() << " s." << std::endl;
  std::cout << "Total live time " << Runs.GetLiveTime() << " s." << std::endl;

  std::cout << "Background wall time" << Runs.GetLastBackgroundRunEndTime()-Runs.GetFirstBackgroundRunStartTime()<<"s."
<< std::endl;
  std::cout << "Background run time" << Runs.GetBackgroundRunTime() << " s." << std::endl;
  std::cout << "Background live time " << Runs.GetBackgroundLiveTime() << " s." << std::endl;

  std::cout << "Calibration wall time " << Runs.GetLastCalibrationRunEndTime()-Runs.GetFirstCalibrationRunStartTime() <<
" s." << std::endl;
  std::cout << "Calibration run time " << Runs.GetCalibrationRunTime() << " s." << std::endl;
  std::cout << "Calibration live time " << Runs.GetCalibrationLiveTime() << " s." << std::endl;

  std::cout << "Total exposure  " << Runs.GetBackgroundExposure() << " kg.yr." <<  std::endl;
  std::cout << "130Te exposure  " << Runs.GetBackground130TeIsotopicExposure() << " kg.yr." << std::endl;
  std::cout << "128Te exposure  " << Runs.GetBackground128TeIsotopicExposure() << " kg.yr." << std::endl;
  std::cout << "120Te exposure  " << Runs.GetBackground120TeIsotopicExposure() << " kg.yr." << std::endl;

  std::cout << "Background exposure (channel 365): " << Runs.GetChannelBackgroundExposure(365) << " kg.yr." <<
std::endl;
  std::cout << "Background exposure (tower 13): " << Runs.GetTowerBackgroundExposure(13) << " kg.yr." << std::endl;
```



```
BackgroundExposure 566.027
Total wall time 8.00599e+07 s.
Total run time  506.634 days
Total live time 462050 days
Background wall time 7.9372e+07 s.
Background run time 300.087 days
Background live time 275403 days
Calibration wall time 7.76138e+07 s.
Calibration run time 97.5058 days
Calibration live time 89035.4 days
Total exposure  566.027 kg.yr.
130Te exposure  157.413 kg.yr.
128Te exposure  144.037 kg.yr.
120Te exposure  0.391225 kg.yr.
```

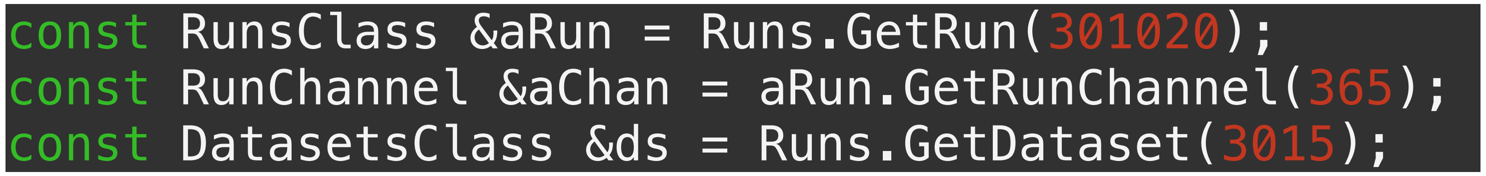
```
Runs.DrawDataTakingTimeline("all"); // <- Draw the data taking calendar
```



Now try to create your own RunsManager and run:

```
Runs.DrawExposureAccumulation("cumulative"); // <- Draw the accumulated TeO2 exposure vs time
DrawRunTypeBreakdown(Runs); // <- Draw the breakdown of run types
```

Other useful classes of the RunsManager:



```
const RunsClass &aRun = Runs.GetRun(301020);
const RunChannel &aChan = aRun.GetRunChannel(365);
const DatasetsClass &ds = Runs.GetDataset(3015);
```