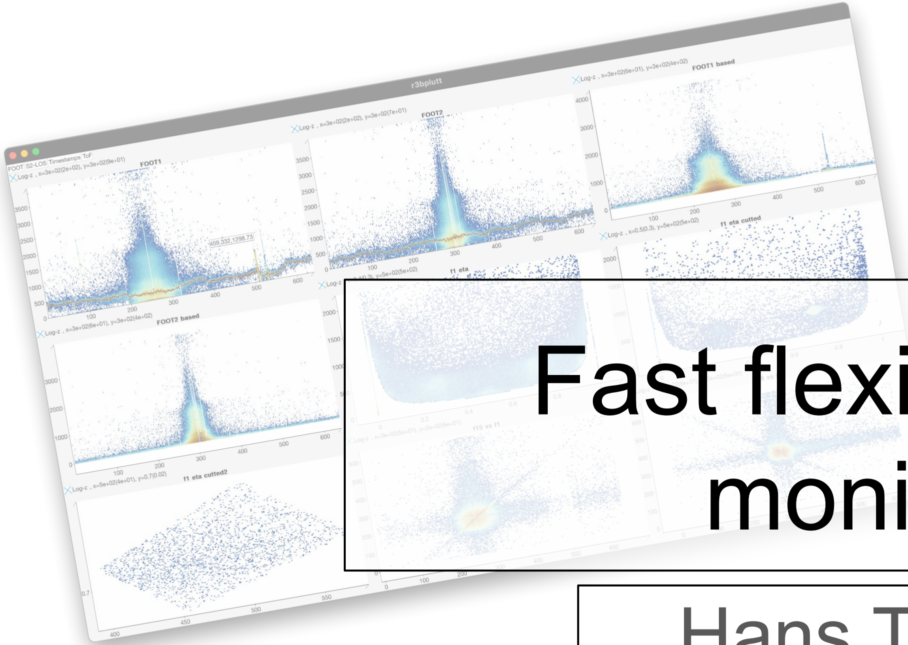
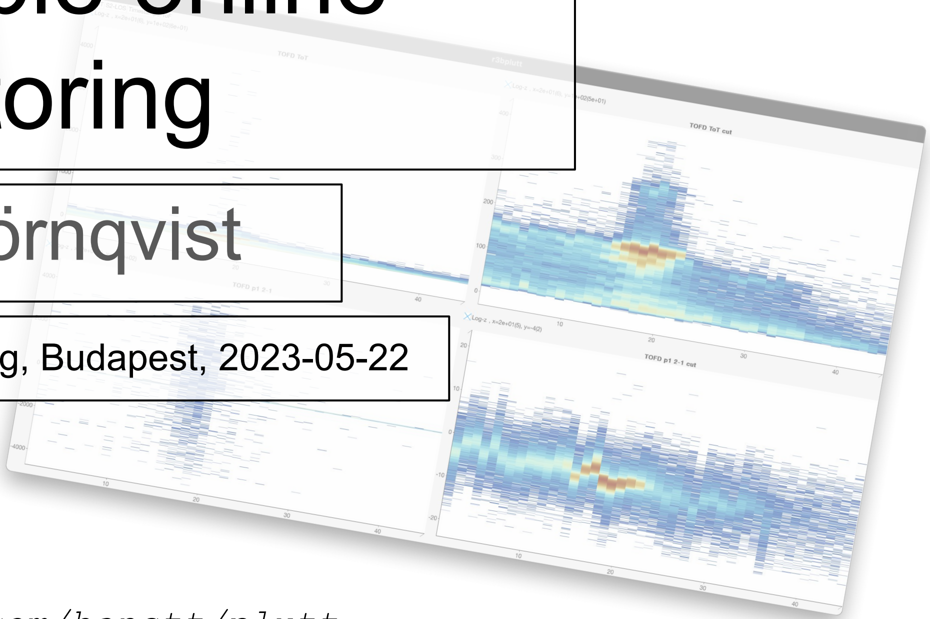




Fast flexible online monitoring

Hans Törnqvist

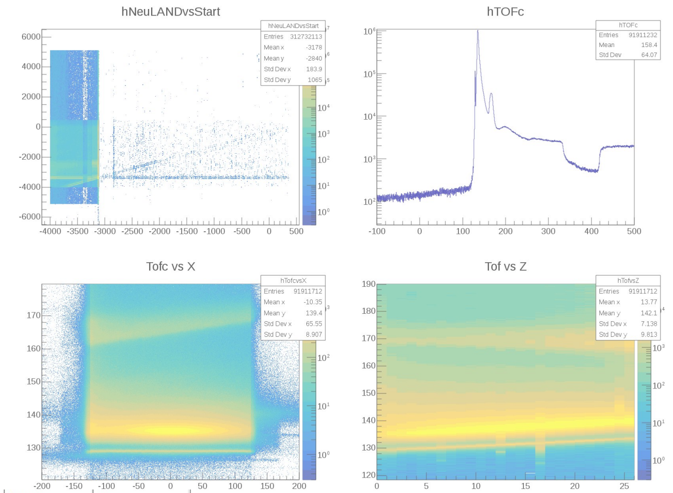
R3B Collaboration Meeting, Budapest, 2023-05-22

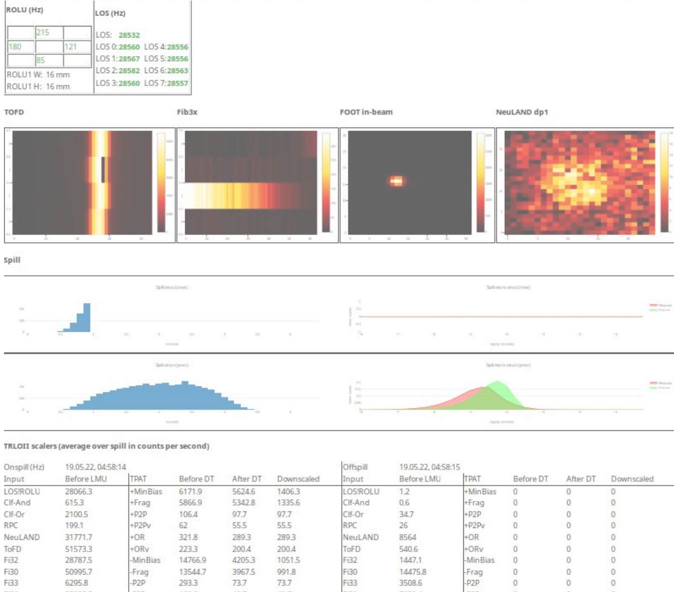
https://github.com/hanstt/plutt

Current state

- **R3BRoot**
  - Calibrated data, simulations, online plots, the **ROOT** kit at hand, ie the business
  - **JSROOT** is pretty good, but doesn’t do online cutting?
  - There are always several versions when beam-time hits
- **R3BMon**
  - “100%” up-time hacked unpacker + **JS+WS** web-page
  - Unpackers should not analyse
  - **plotly** is cute, but does a lot less than **JSROOT**

- During my time at GSI, felt like something was missing…





ROLU (Hz)

	215	
180		121
	85	

ROLU1 W: 16 mm
ROLU1 H: 16 mm

LOS (Hz)

LOS: 28532
LOS 0: 28560 LOS 4: 28556
LOS 1: 28567 LOS 5: 28556
LOS 2: 28582 LOS 6: 28563
LOS 3: 28560 LOS 7: 28557

TRLOII scalers (average over spill in counts per second)

Onspill (Hz)	19.05.22, 04:58:14				
Input	Before LMU	TPAT	Before DT	After DT	Downscaled
LOS!ROLU	28066.3	+MinBias	6171.9	5624.6	1406.3
Clf-And	615.3	+Frag	5866.9	5342.8	1335.6
Clf-Or	2100.5	+P2P	106.4	97.7	97.7
RPC	199.1	+P2Pv	62	55.5	55.5
NeuLAND	31771.7	+OR	321.8	289.3	289.3
ToFD	51573.3	+ORv	223.3	200.4	200.4
Fi32	28787.5	-MinBias	14766.9	4205.3	1051.5
Fi30	50995.7	-Frag	13544.7	3967.5	991.8
Fi33	6295.8	-P2P	293.3	73.7	73.7

Offspill	19.05.22, 04:58:15				
Input	Before LMU	TPAT	Before DT	After DT	Downscaled
LOS!ROLU	1.2	+MinBias	0	0	0
Clf-And	0.6	+Frag	0	0	0
Clf-Or	34.7	+P2P	0	0	0
RPC	26	+P2Pv	0	0	0
NeuLAND	8564	+OR	0	0	0
ToFD	540.6	+ORv	0	0	0
Fi32	1447.1	-MinBias	0	0	0
Fi30	14475.8	-Frag	0	0	0
Fi33	3508.6	-P2P	0	0	0

What would be nice to have online?

- **Scripting** sources, transformations, histogramming
- A “full” setup config that could be e-mailed
- Simple **on-the-fly calibration**
- Plotting **all data**, no histogram limits
- Plot detector *Grunka* **vs** detector *Mojäng* for the 1st time in 30s
- **Clicky-cut** on blob in one plot, see what another plot gives
- **Fast** zooming and projection
- **Light-weight** in code-size, prerequisites, startup
- Remote shifters able to do **remote online** work
- If all services are down, should work even with **ssh+X11**
- It should be “easy” to use at *4:00 in the morning!*

plutt – yet another thing to learn, sorry

- Started as some tests on top of **ucesb**
    - Is it possible to adapt in runtime to **ucesb** structs?
    - Match two sides of a detector and plot the matching indices with scripting
    - Is **X11 over ssh** completely useless? In case **VNC** or online services crash
- It grew… Current state:
    - Good old **C++11**
    - Auto **ucesb** struct parsing and **TTree *h101**
    - Histograms auto-adjust to all data
    - Zooming, projecting, polygonal cutting
    - On-the-fly fine-time calibration, tpat-aware pedestals
    - Auto multi-peak finding & fitting with simplified **SNIP** + **nlopt** (super WIP)
    - Trigger maps for **FPGA-style TDC:s**
    - Linear fit calibration
    - **SDL** (easily replaced…), user input buffering for slow clients

Example

- Invocation: `./plutt -f my_config.r3bp -r h101 my_file.root`
- FOOT energy vs channel:
  - `hist2d("FOOT1", FOOT1)` OR `hist2d("FOOT1", FOOT1.v, FOOT1.I)`
- Off-spill pedestal subtraction:

  -
    ```
    offspill = tpat(TPAT, 12--15)
    f1, f1std = pedestal(FOOT1, 6, tpat=offspill)
    hist2d("FOOT1", f1, logz, binsx=640)
    hist2d(“FOOT1 std”, f1std, logz, binsx=640)
    ```
- Apply cut on FOOT clusters:

  -
    ```
    f1x, f1e, f1eta = cluster(f1)
    hist2d("FOOT1 eta cut", f1e, f1eta, cut("cut1.txt"))
    ```
- Fine-time calibrated TOFD:
  - `tofd_p1t1l = coarse_fine(TOFD_P1T1TCL, TOFD_P1T1TFL, tamex3)`
- And so on, but now, we’ll do it live!

Some thoughts

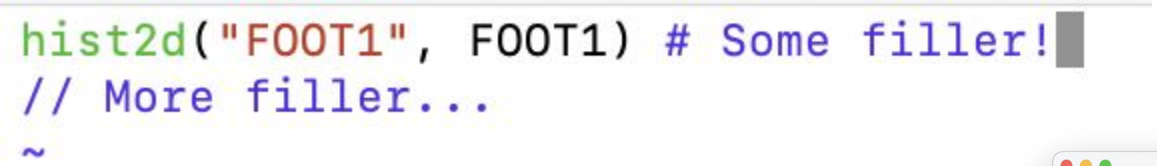
- Ucesb input interesting for R3BRoot?
  - Config file parsed, signals resolved, we have a list of missing signals **SIG1**,**SIG2**,…, expect them to come from the unpacker
  - Ask unpacker to generate struct containing **SIG1**,**SIG2**,...
  - Parse macro blob to fetch signal type and variable array sizes
  - Allocate event storage, and bind with **ext_data_struct*** calls
  - Ie move lots of work **from** setting up readers and control-macros **into** core input
- Config file
  - **flex**+**bison** parser
  - Each data transformation is a **node** in a graph, there are currently **21 types**
  - Graph “starts” with bindings to input stage, “ends” with histograms
  - One node executes at most once per event
  - Consolidate with Bastii’s **nupeline**?
- Data representation
  - **STL**-ified ucesb signals **mi[]**, **me[]**, **v[]**, both from **ucesb** and **ROOT** inputs
  - Used as **input** and **output** of node which pipe values
  - Fast compact code to process data, *data-oriented over object-oriented!*
- Plotting
  - Histogramming, rebinning etc all custom, super-specialized and unreliable… On top on of **SDL** primitives

TODO

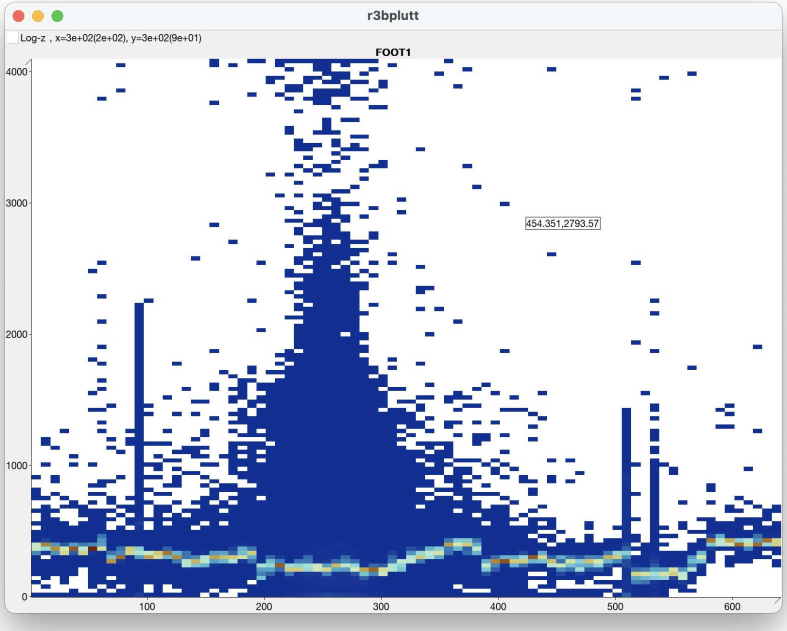
- This will explode once anybody else gets their hands on it…
- Not so hard to do:
  - Slap **LGPL** on it so it can go public
  - Config-file watcher for live plotting updates (probably harder than I think)
  - Input watcher, eg auto-load new **ROOT** files
  - More tests, always
  - **README** is there, but it's getting old…
- Longer term:
  - Introspection/reflection of **TTree** + arrays of objects
  - More robust peak finding
  - Replace **R3BMon**?
  - Make it beautiful
  - Make it go even faster, lots of "safe" slow code
  - Abandon immediate GUI style for more flexible widgeting
- BUT, this is a small part of the job, it won't get much more powerful

Finished!

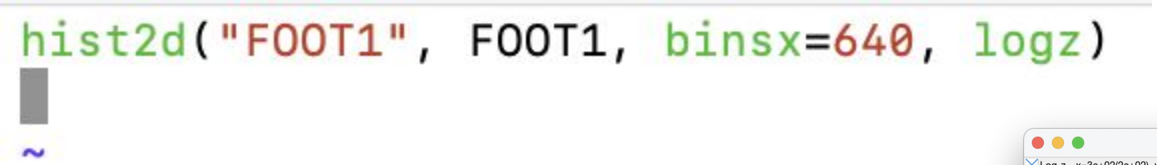
demo1



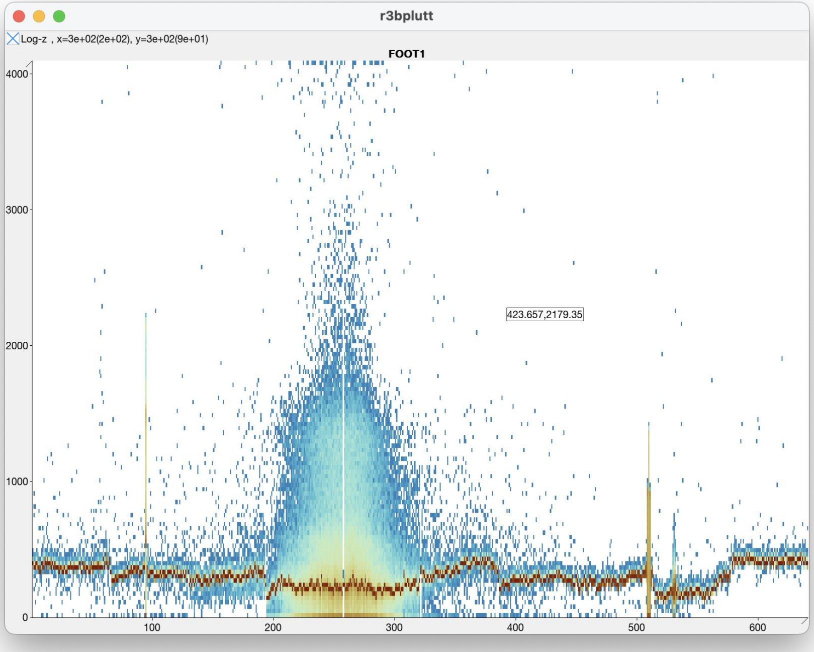
```
hist2d("FOOT1", FOOT1) # Some filler!
// More filler...
~
```



demo2

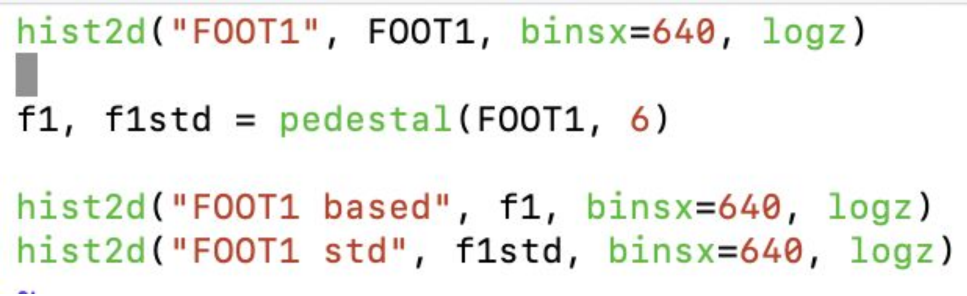


```
hist2d("FOOT1", FOOT1, binsx=640, logz)

~
```

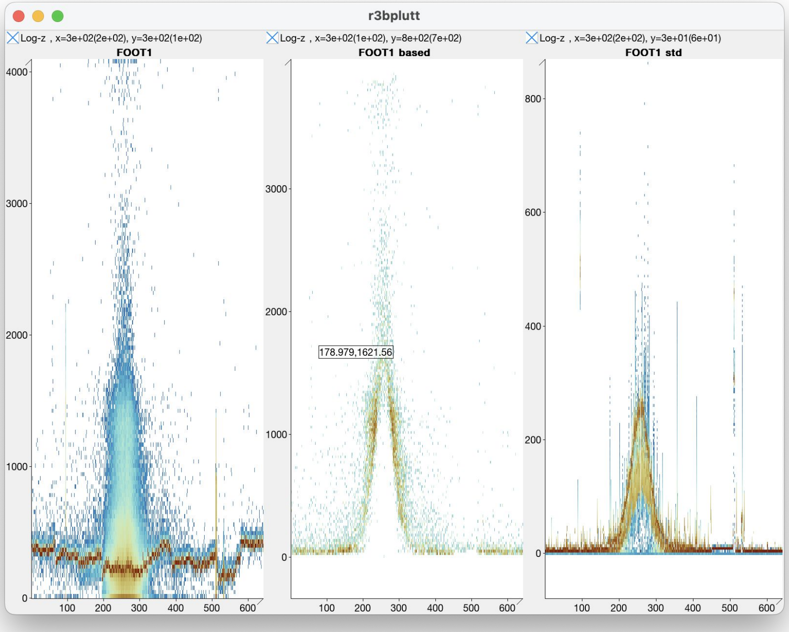


```
Log-z , x=3e+02(2e+02), y=3e+02(9e+01)
```

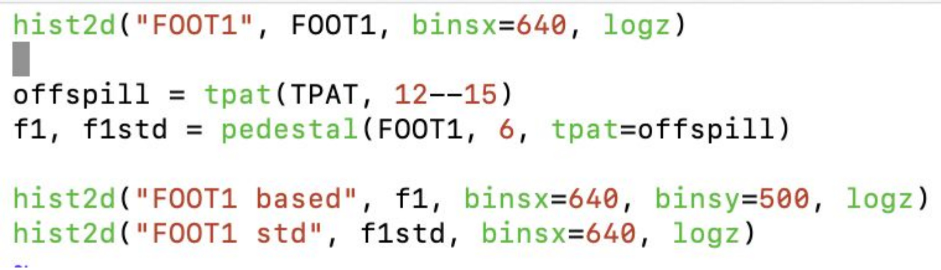
demo3



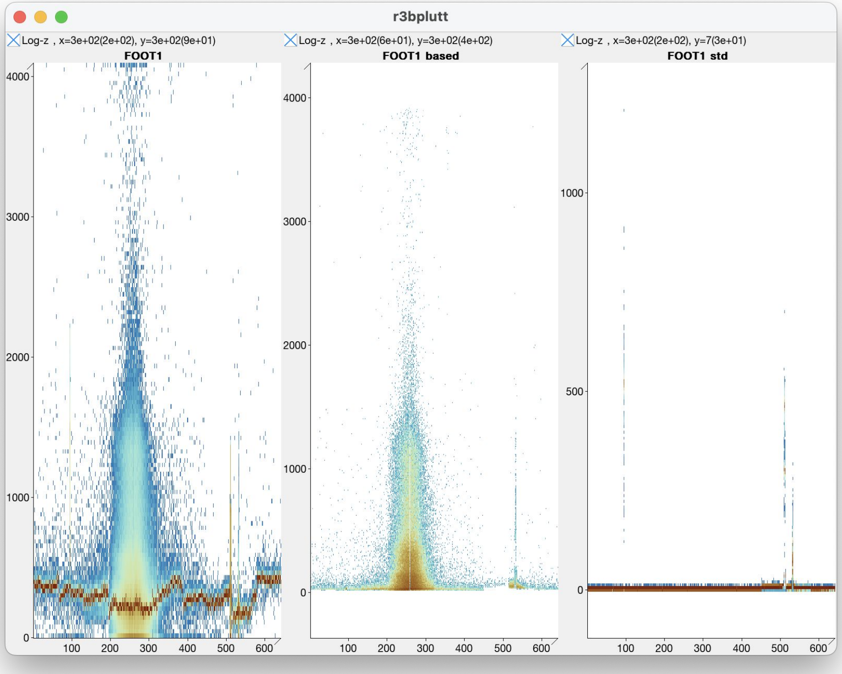
```
hist2d("FOOT1", FOOT1, binsx=640, logz)

f1, f1std = pedestal(FOOT1, 6)

hist2d("FOOT1 based", f1, binsx=640, logz)
hist2d("FOOT1 std", f1std, binsx=640, logz)
```



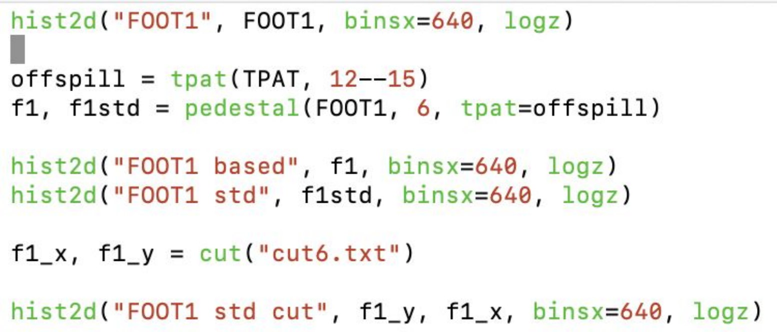
demo4



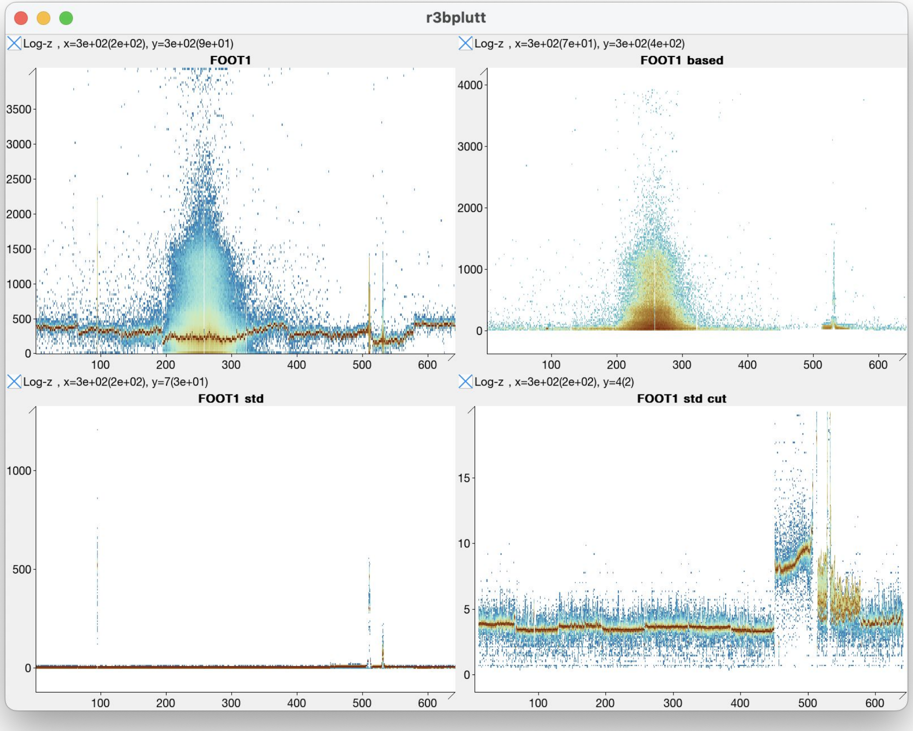
```
hist2d("FOOT1", FOOT1, binsx=640, logz)

offspill = tpat(TPAT, 12--15)
f1, f1std = pedestal(FOOT1, 6, tpat=offspill)

hist2d("FOOT1 based", f1, binsx=640, binsy=500, logz)
hist2d("FOOT1 std", f1std, binsx=640, logz)
```



demo5



```
hist2d("FOOT1", FOOT1, binsx=640, logz)

offspill = tpat(TPAT, 12--15)
f1, f1std = pedestal(FOOT1, 6, tpat=offspill)

hist2d("FOOT1 based", f1, binsx=640, logz)
hist2d("FOOT1 std", f1std, binsx=640, logz)

f1_x, f1_y = cut("cut6.txt")

hist2d("FOOT1 std cut", f1_y, f1_x, binsx=640, logz)
```



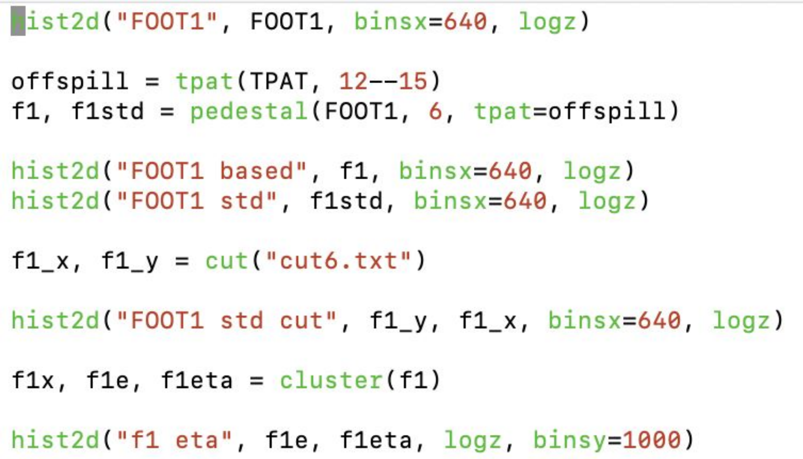
Log-z , x=3e+02(2e+02), y=3e+02(9e+01)

Log-z , x=3e+02(7e+01), y=3e+02(4e+02)

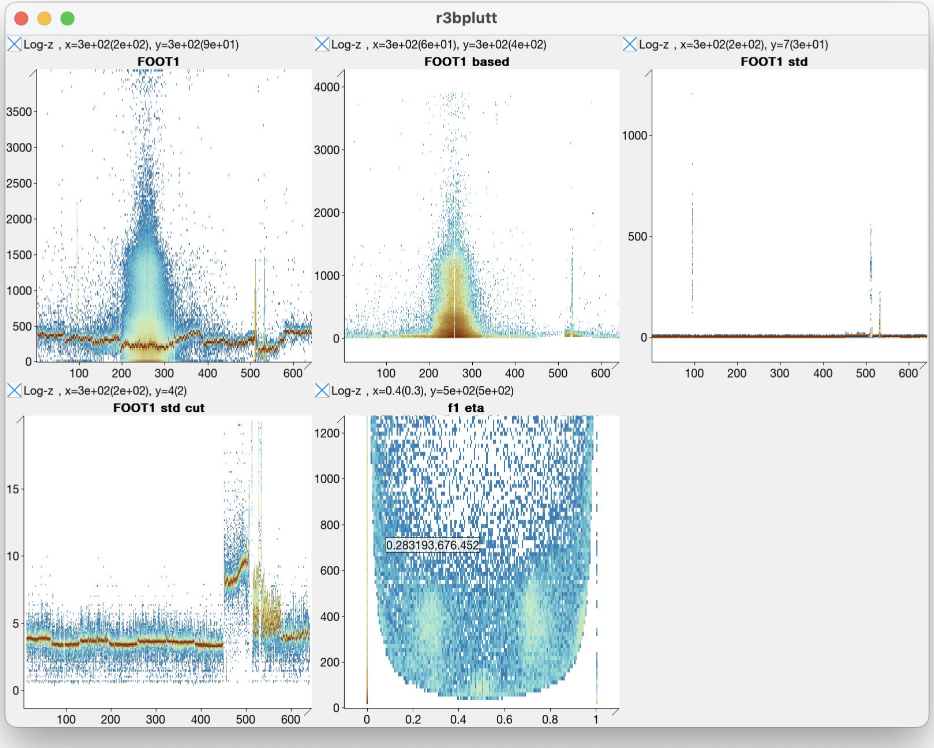
Log-z , x=3e+02(2e+02), y=7(3e+01)

Log-z , x=3e+02(2e+02), y=4(2)

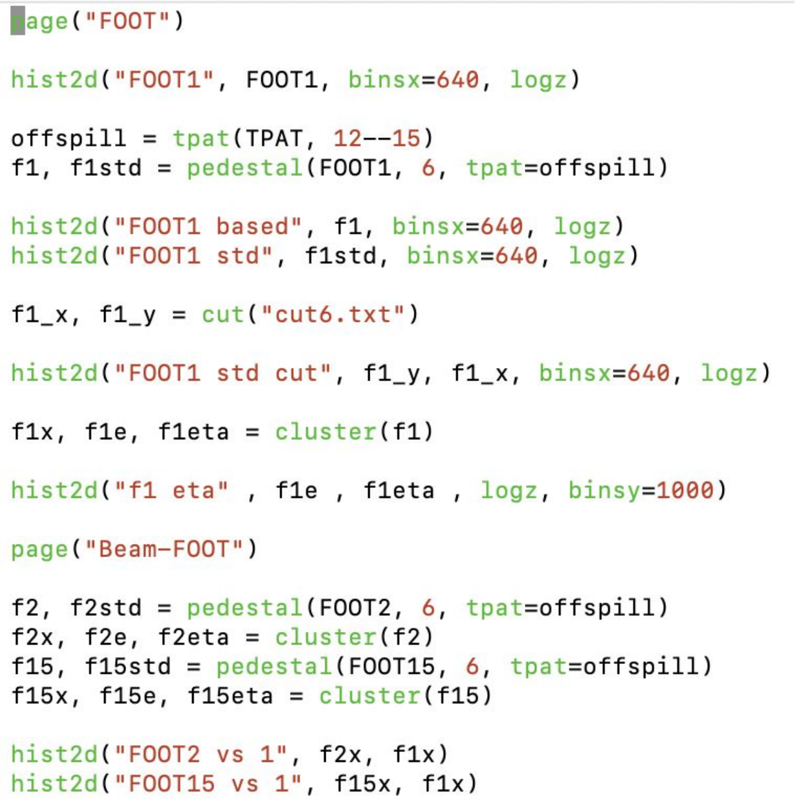
demo6



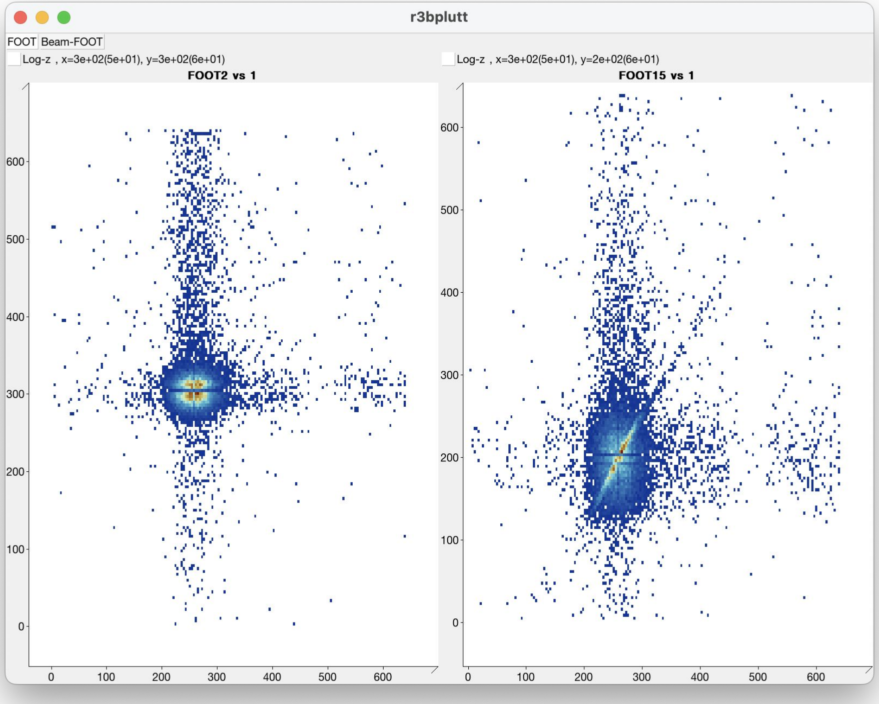
```
hist2d("FOOT1", FOOT1, binsx=640, logz)

offspill = tpat(TPAT, 12--15)
f1, f1std = pedestal(FOOT1, 6, tpat=offspill)

hist2d("FOOT1 based", f1, binsx=640, logz)
hist2d("FOOT1 std", f1std, binsx=640, logz)

f1_x, f1_y = cut("cut6.txt")

hist2d("FOOT1 std cut", f1_y, f1_x, binsx=640, logz)

f1x, f1e, f1eta = cluster(f1)

hist2d("f1 eta", f1e, f1eta, logz, binsy=1000)
```



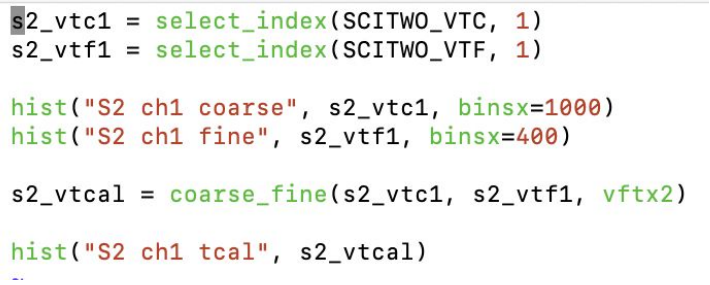
demo7



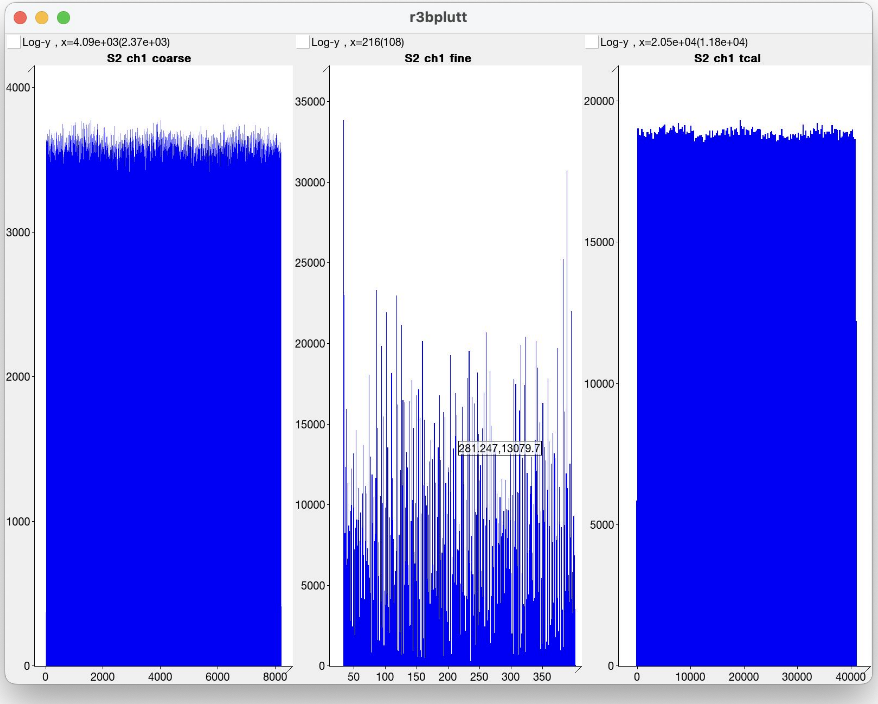
```
page("FOOT")

hist2d("FOOT1", FOOT1, binsx=640, logz)

offspill = tpat(TPAT, 12--15)
f1, f1std = pedestal(FOOT1, 6, tpat=offspill)

hist2d("FOOT1 based", f1, binsx=640, logz)
hist2d("FOOT1 std", f1std, binsx=640, logz)

f1_x, f1_y = cut("cut6.txt")

hist2d("FOOT1 std cut", f1_y, f1_x, binsx=640, logz)

f1x, f1e, f1eta = cluster(f1)

hist2d("f1 eta" , f1e , f1eta , logz, binsy=1000)

page("Beam-FOOT")

f2, f2std = pedestal(FOOT2, 6, tpat=offspill)
f2x, f2e, f2eta = cluster(f2)
f15, f15std = pedestal(FOOT15, 6, tpat=offspill)
f15x, f15e, f15eta = cluster(f15)

hist2d("FOOT2 vs 1", f2x, f1x)
hist2d("FOOT15 vs 1", f15x, f1x)
```



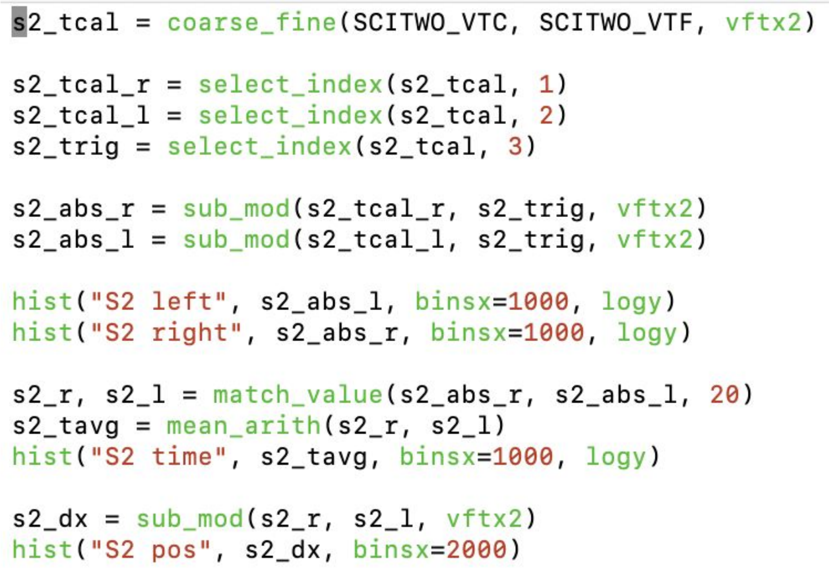
demo8



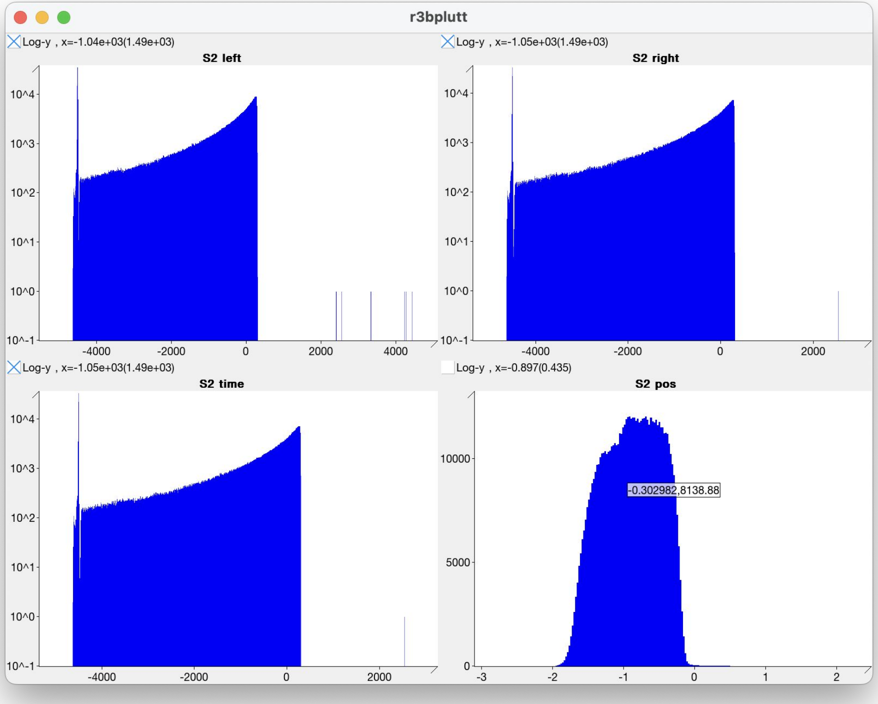
```
s2_vtc1 = select_index(SCITWO_VTC, 1)
s2_vtf1 = select_index(SCITWO_VTF, 1)

hist("S2 ch1 coarse", s2_vtc1, binsx=1000)
hist("S2 ch1 fine", s2_vtf1, binsx=400)

s2_vtcal = coarse_fine(s2_vtc1, s2_vtf1, vftx2)

hist("S2 ch1 tcal", s2_vtcal)
```



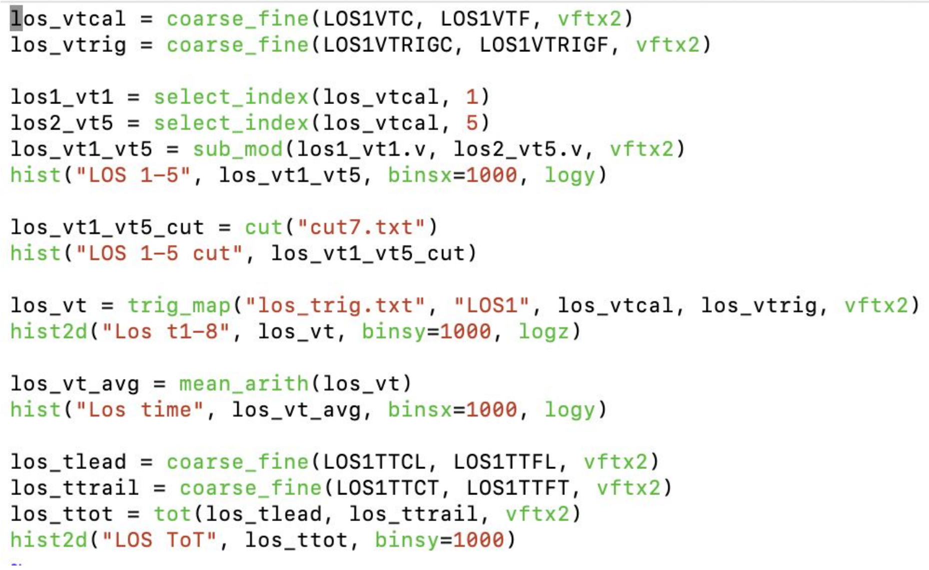
demo9



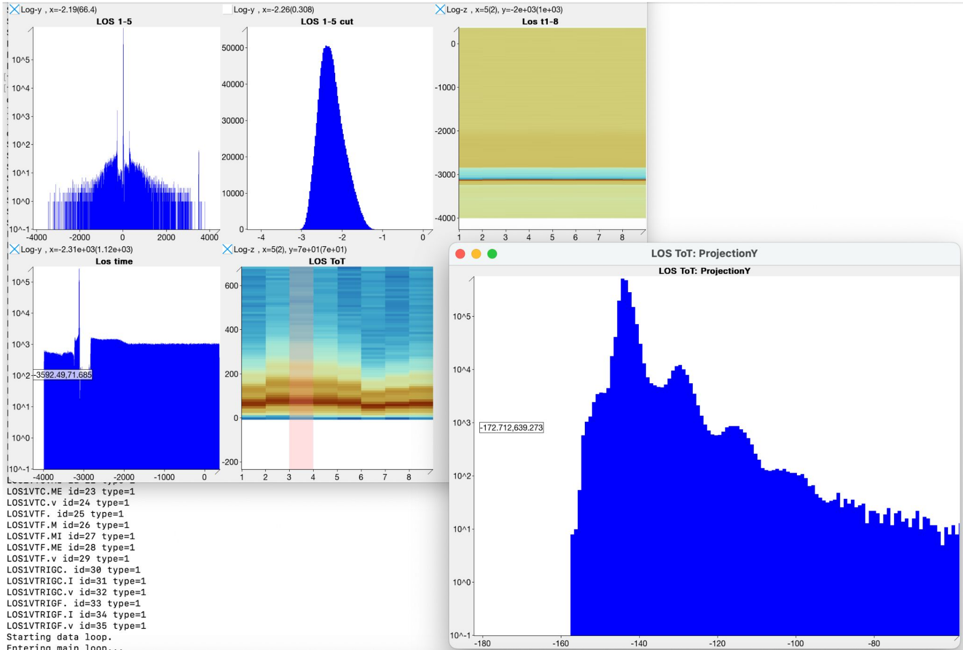
```
s2_tcal = coarse_fine(SCITWO_VTC, SCITWO_VTF, vftx2)

s2_tcal_r = select_index(s2_tcal, 1)
s2_tcal_l = select_index(s2_tcal, 2)
s2_trig = select_index(s2_tcal, 3)

s2_abs_r = sub_mod(s2_tcal_r, s2_trig, vftx2)
s2_abs_l = sub_mod(s2_tcal_l, s2_trig, vftx2)

hist("S2 left", s2_abs_l, binsx=1000, logy)
hist("S2 right", s2_abs_r, binsx=1000, logy)

s2_r, s2_l = match_value(s2_abs_r, s2_abs_l, 20)
s2_tavg = mean_arith(s2_r, s2_l)
hist("S2 time", s2_tavg, binsx=1000, logy)

s2_dx = sub_mod(s2_r, s2_l, vftx2)
hist("S2 pos", s2_dx, binsx=2000)
```



demo10

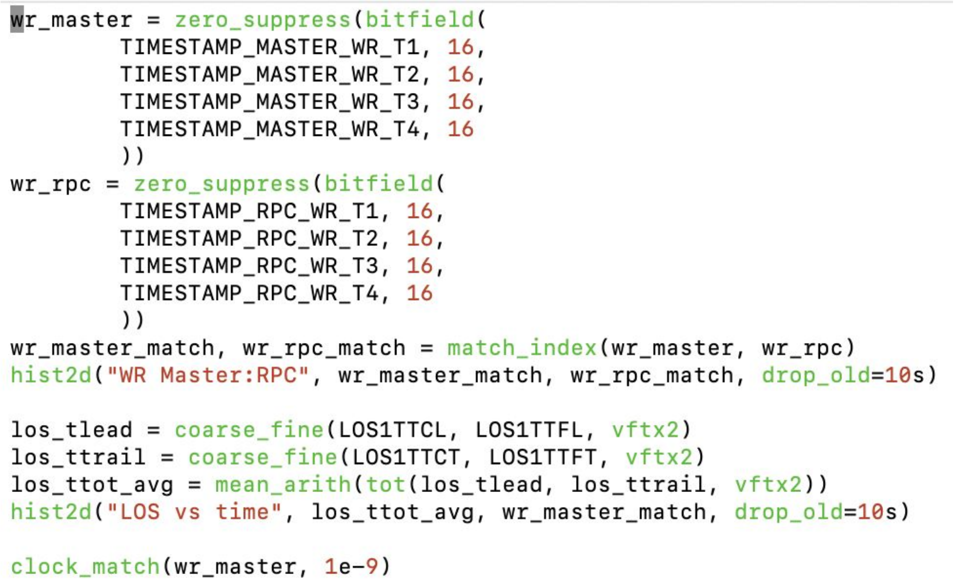


```
los_vtcal = coarse_fine(LOS1VTC, LOS1VTF, vftx2)
los_vtrig = coarse_fine(LOS1VTRIGC, LOS1VTRIGF, vftx2)

los1_vt1 = select_index(los_vtcal, 1)
los2_vt5 = select_index(los_vtcal, 5)
los_vt1_vt5 = sub_mod(los1_vt1.v, los2_vt5.v, vftx2)
hist("LOS 1-5", los_vt1_vt5, binsx=1000, logy)

los_vt1_vt5_cut = cut("cut7.txt")
hist("LOS 1-5 cut", los_vt1_vt5_cut)

los_vt = trig_map("los_trig.txt", "LOS1", los_vtcal, los_vtrig, vftx2)
hist2d("Los t1-8", los_vt, binsy=1000, logz)

los_vt_avg = mean_arith(los_vt)
hist("Los time", los_vt_avg, binsx=1000, logy)

los_tlead = coarse_fine(LOS1TTCL, LOS1TTFL, vftx2)
los_ttrail = coarse_fine(LOS1TTCT, LOS1TTFT, vftx2)
los_ttot = tot(los_tlead, los_ttrail, vftx2)
hist2d("LOS ToT", los_ttot, binsy=1000)
```

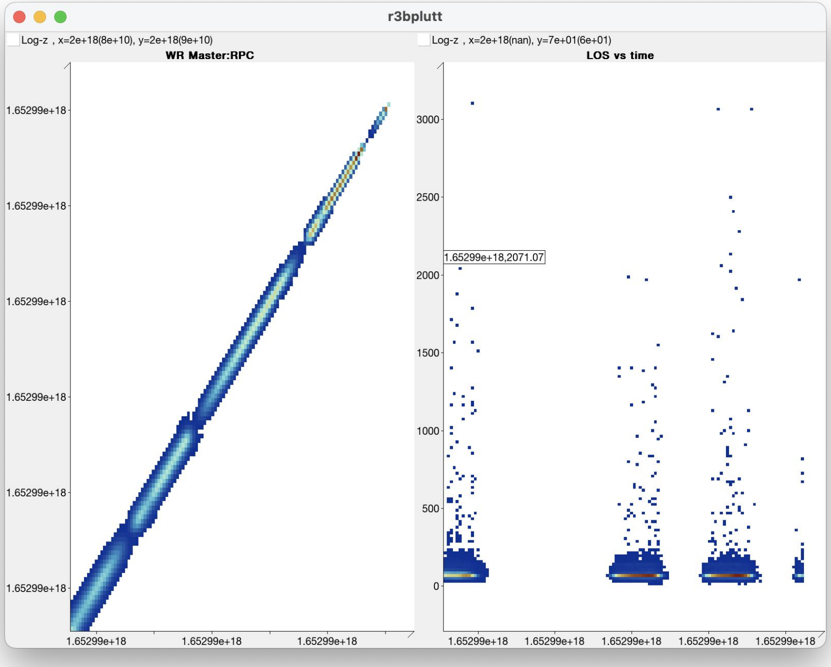


```
LOS1VTC.ME id=23 type=1
LOS1VTC.v id=24 type=1
LOS1VTF. id=25 type=1
LOS1VTF.M id=26 type=1
LOS1VTF.MI id=27 type=1
LOS1VTF.ME id=28 type=1
LOS1VTF.v id=29 type=1
LOS1VTRIGC. id=30 type=1
LOS1VTRIGC.I id=31 type=1
LOS1VTRIGC.v id=32 type=1
LOS1VTRIGF. id=33 type=1
LOS1VTRIGF.I id=34 type=1
LOS1VTRIGF.v id=35 type=1
Starting data loop.
Entering main loop...
```

demo11



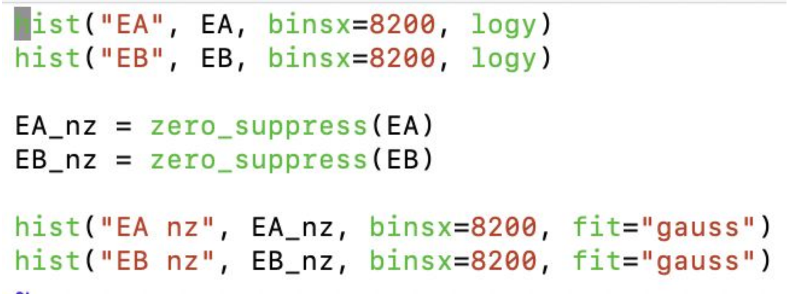
```
wr_master = zero_suppress(bitfield(
        TIMESTAMP_MASTER_WR_T1, 16,
        TIMESTAMP_MASTER_WR_T2, 16,
        TIMESTAMP_MASTER_WR_T3, 16,
        TIMESTAMP_MASTER_WR_T4, 16
        ))
wr_rpc = zero_suppress(bitfield(
        TIMESTAMP_RPC_WR_T1, 16,
        TIMESTAMP_RPC_WR_T2, 16,
        TIMESTAMP_RPC_WR_T3, 16,
        TIMESTAMP_RPC_WR_T4, 16
        ))
wr_master_match, wr_rpc_match = match_index(wr_master, wr_rpc)
hist2d("WR Master:RPC", wr_master_match, wr_rpc_match, drop_old=10s)

los_tlead = coarse_fine(LOS1TTCL, LOS1TTFL, vftx2)
los_ttrail = coarse_fine(LOS1TTCT, LOS1TTFT, vftx2)
los_ttot_avg = mean_arith(tot(los_tlead, los_ttrail, vftx2))
hist2d("LOS vs time", los_ttot_avg, wr_master_match, drop_old=10s)

clock_match(wr_master, 1e-9)
```



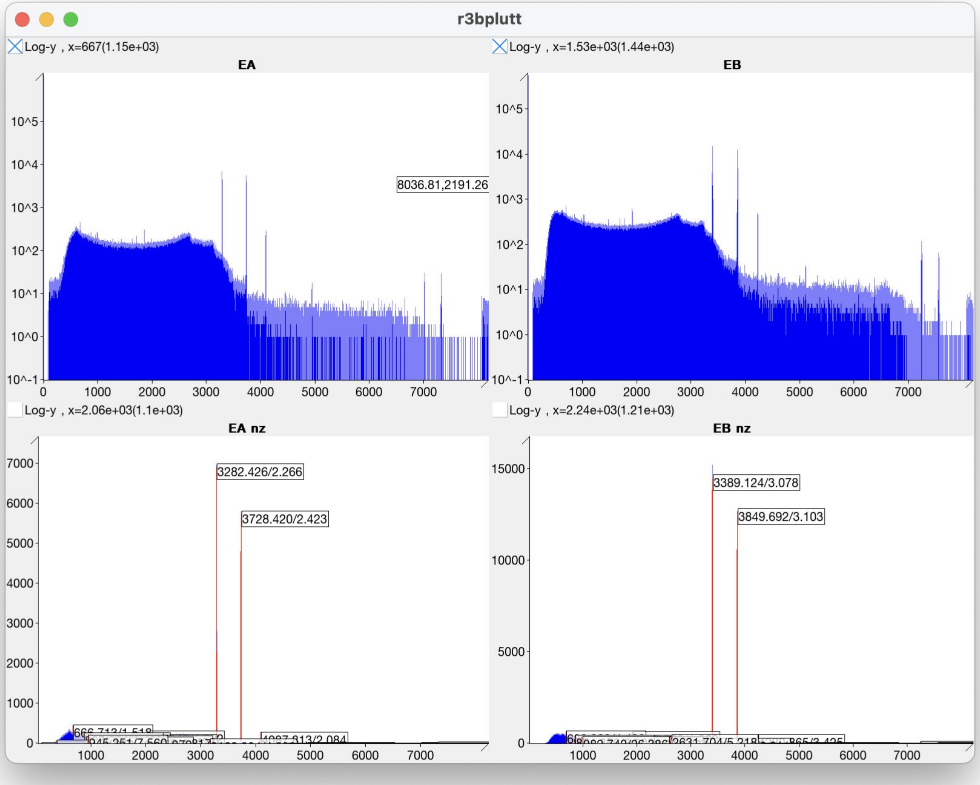
□ Log-z , x=2e+18(8e+10), y=2e+18(9e+10)

□ Log-z , x=2e+18(nan), y=7e+01(6e+01)

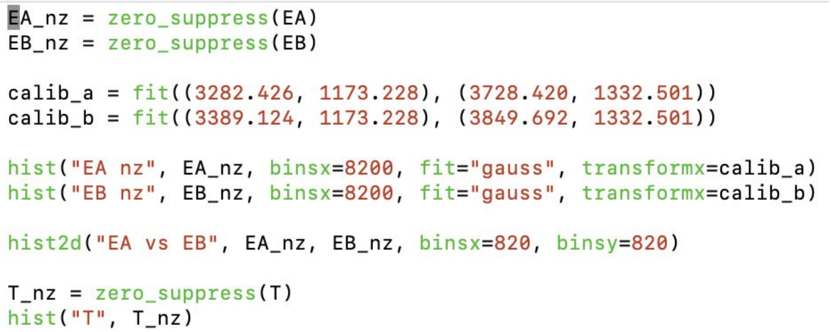
demo12



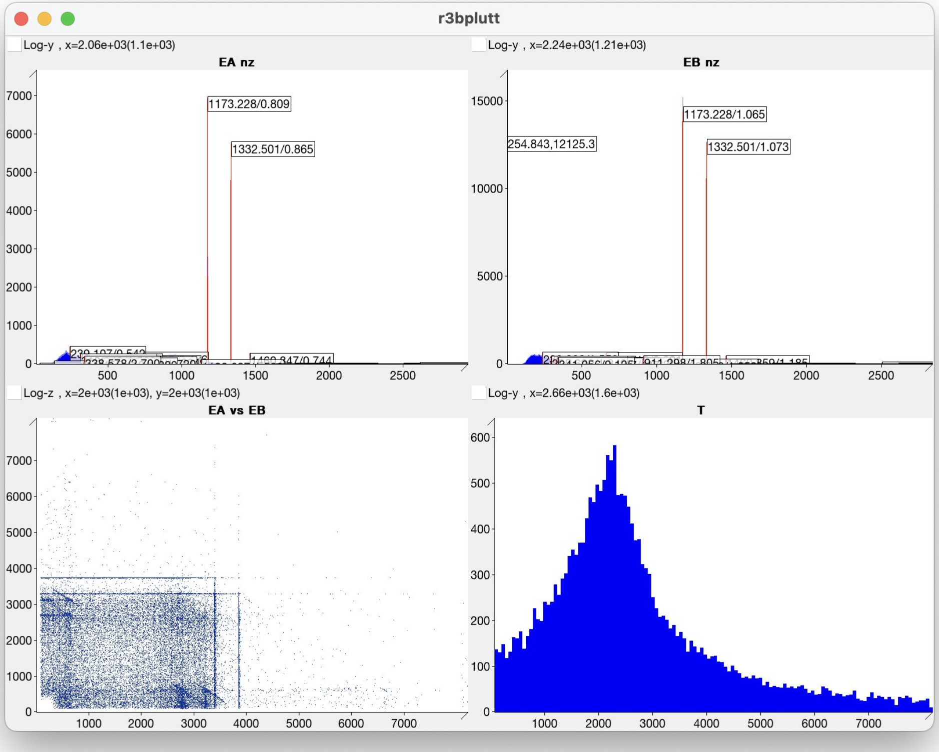
```
hist("EA", EA, binsx=8200, logy)
hist("EB", EB, binsx=8200, logy)

EA_nz = zero_suppress(EA)
EB_nz = zero_suppress(EB)

hist("EA nz", EA_nz, binsx=8200, fit="gauss")
hist("EB nz", EB_nz, binsx=8200, fit="gauss")
```



demo13



```
EA_nz = zero_suppress(EA)
EB_nz = zero_suppress(EB)

calib_a = fit((3282.426, 1173.228), (3728.420, 1332.501))
calib_b = fit((3389.124, 1173.228), (3849.692, 1332.501))

hist("EA nz", EA_nz, binsx=8200, fit="gauss", transformx=calib_a)
hist("EB nz", EB_nz, binsx=8200, fit="gauss", transformx=calib_b)

hist2d("EA vs EB", EA_nz, EB_nz, binsx=820, binsy=820)

T_nz = zero_suppress(T)
hist("T", T_nz)
```


r3bplutt
Log-y , x=2.06e+03(1.1e+03)
EA nz
1173.228/0.809
1332.501/0.865
1460.847/0.744
7000
6000
5000
4000
3000
2000
1000
0
500
1000
1500
2000
2500
Log-y , x=2.24e+03(1.21e+03)
EB nz
1173.228/1.065
254.843,12125.3
1332.501/1.073
15000
10000
5000
0
500
1000
1500
2000
2500
Log-z , x=2e+03(1e+03), y=2e+03(1e+03)
EA vs EB
7000
6000
5000
4000
3000
2000
1000
0
1000
2000
3000
4000
5000
6000
7000
Log-y , x=2.66e+03(1.6e+03)
T
600
500
400
300
200
100
0
1000
2000
3000
4000
5000
6000
7000