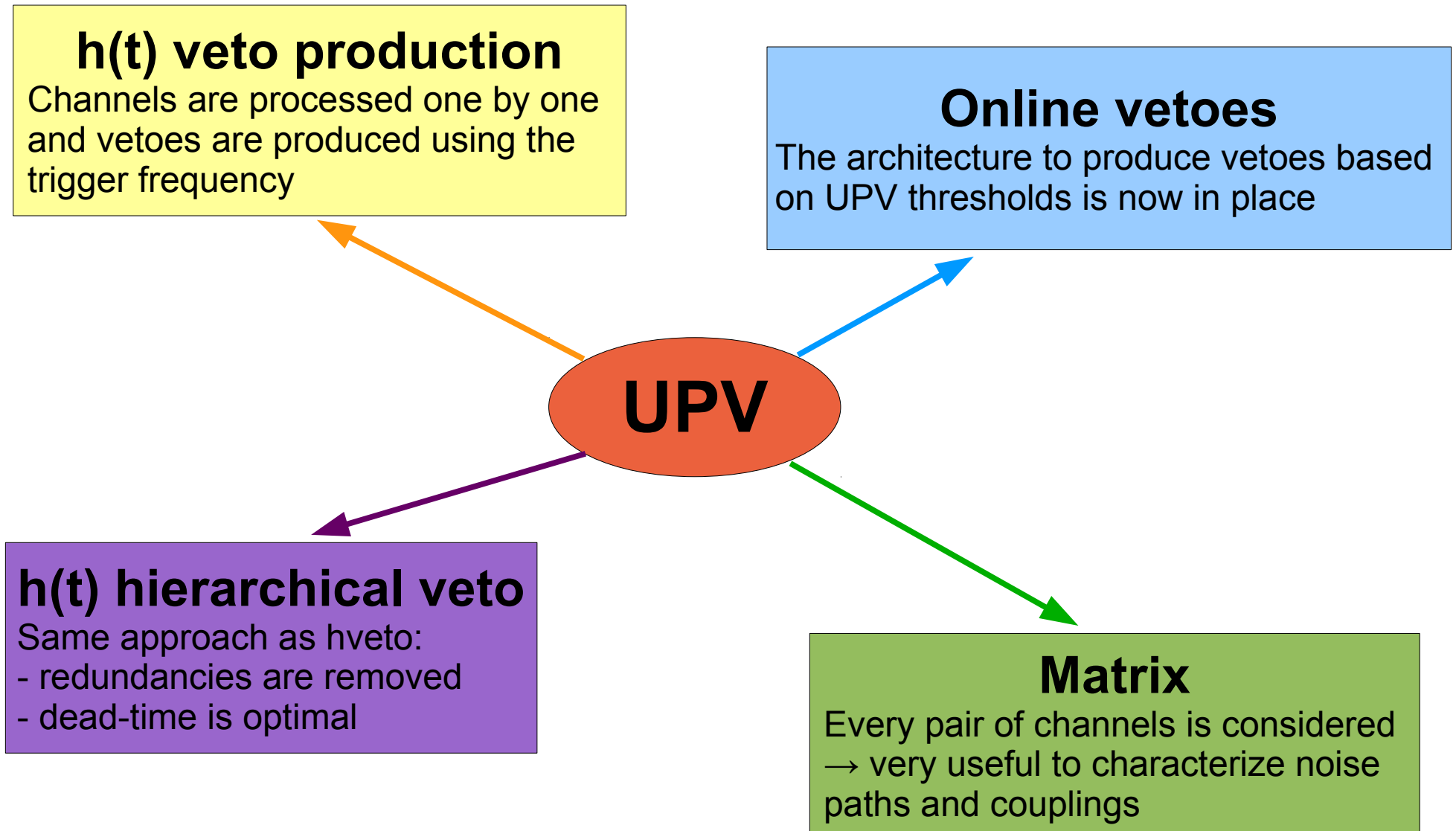
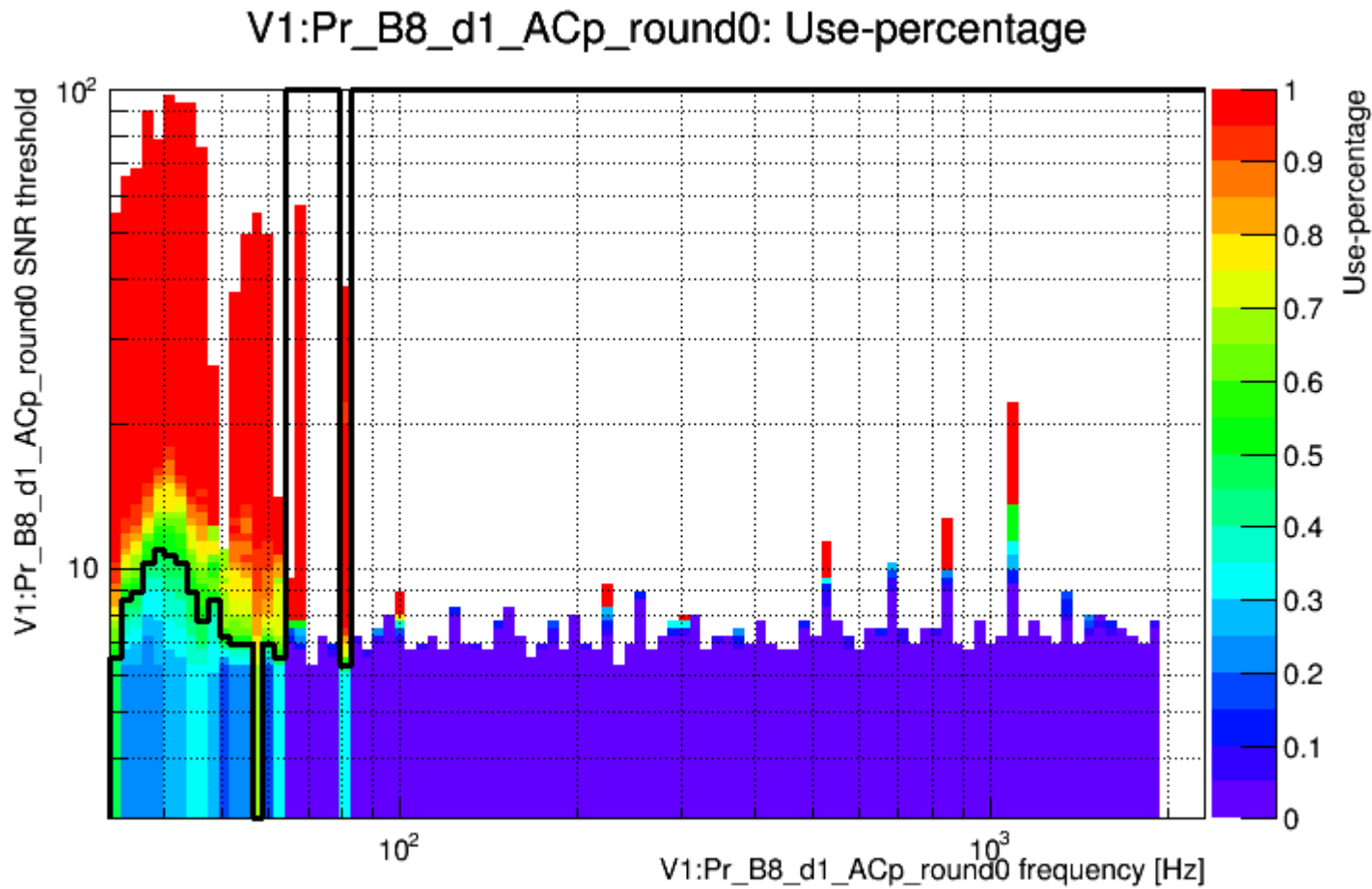


UPV now comes with many utilities





A threshold is adjusted as a function of the trigger frequency to provide optimal veto performance

Why do we need a hierarchical approach?

UPV is run on several hundreds of channels. Many of them provide redundant data quality information.

We cannot blindly combine hundreds of vetoes. The channel-by-channel performance is very good but the overall performance is poor.

Triggers selected by UPV

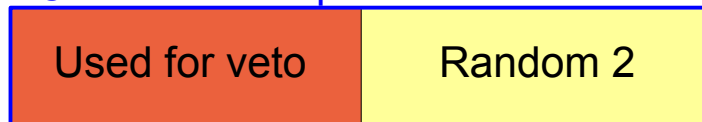
Channel 1

up = 50%

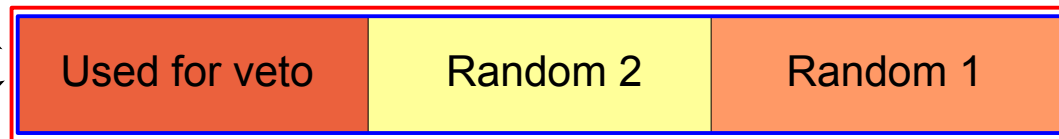


Channel 2

up = 50%



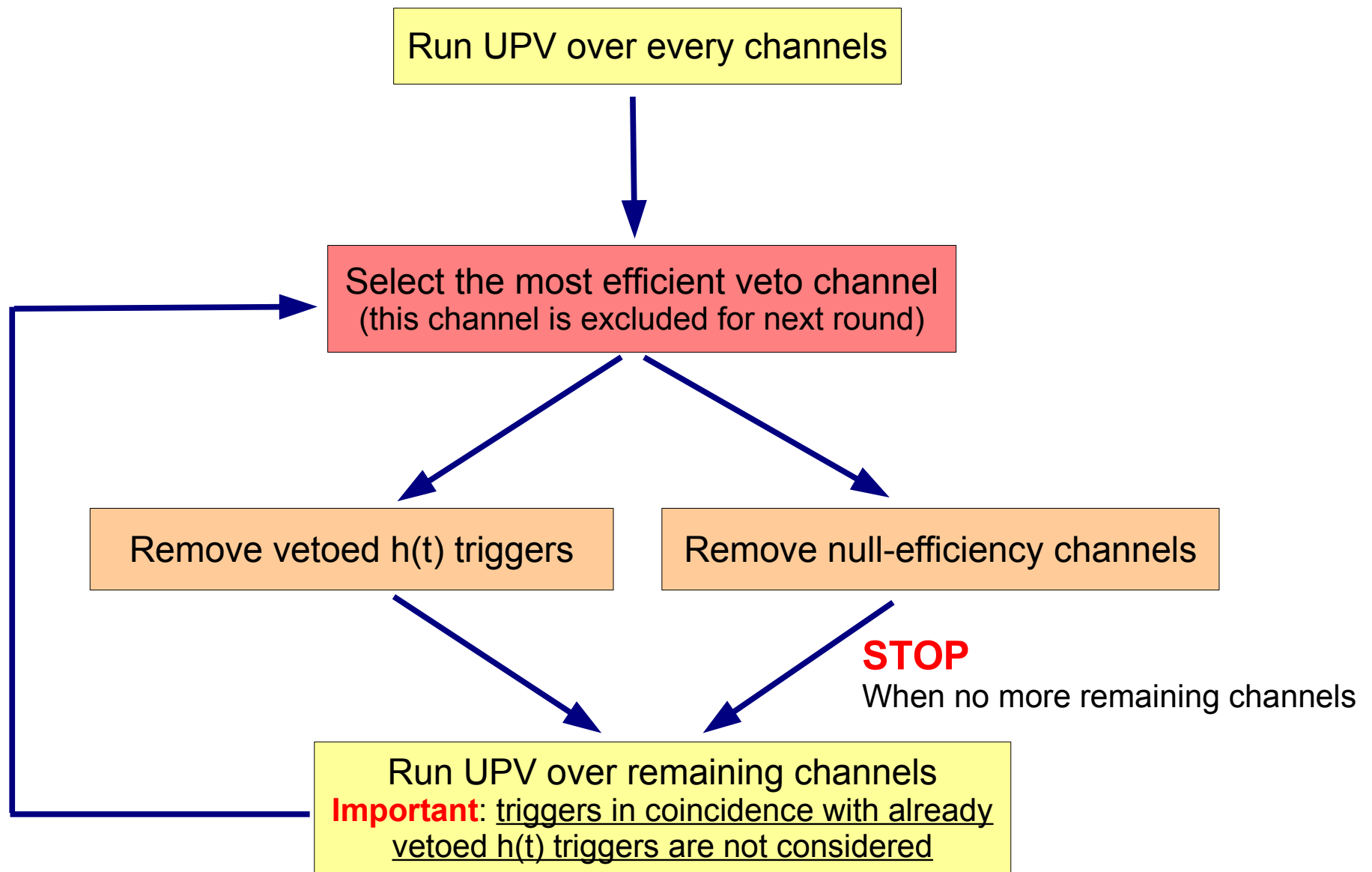
Worst case combination (100% veto overlap / 0% random overlap)



up = 33%

→ We need to combine a minimal number of channels

UPV hierarchical algorithm



UPV report round by round

- The hierarchical approach of UPV has many advantages:
- A minimal set of channels are selected for vetoing
 - The dead-time is minimal
 - Channels are naturally ranked → helpful for noise investigations

Preliminary tests on VSR2 data show good performance:

- For $h(t)$ triggers with $SNR > 8$:
- number of channels = 13
 - dead-time = 1.2%
 - efficiency = 48%

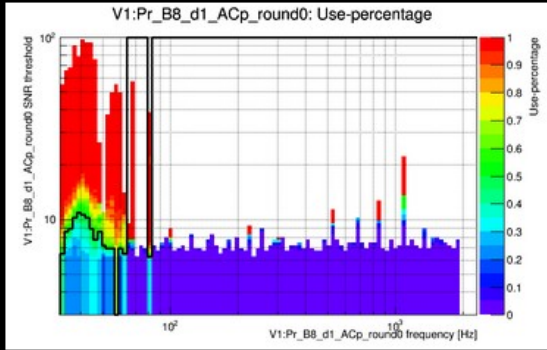
Auxiliary channel = V1:Pr_B8_d1_ACp_round0

[list of segments](#) — [list of segments up to this round](#) — [UPV output](#)

Veto Efficiency = 0.12489 (0.12489)
 Veto Dead-time = 0.01284 (0.01275)
 Veto Efficiency / Dead-time = 9.73 (9.89)
 (Up to this round)
 Veto most useful f-band = 32.3-33.7Hz
 Veto most targeted f-band = 32.3-33.5Hz

TUNING PLOTS:
[Statistics](#)
[Number of used clusters](#)
[Number of vetoed main clusters](#)
[Use-percentage](#)
[Efficiency](#)
[Efficiency / dead-time](#)

FINAL VETO PLOTS:
[Veto efficiency](#)
[Veto used triggers](#)
[SNR-SNR](#)
[Frequency-Frequency](#)
[Frequency-propagation](#)
[Veto flagging rate](#)



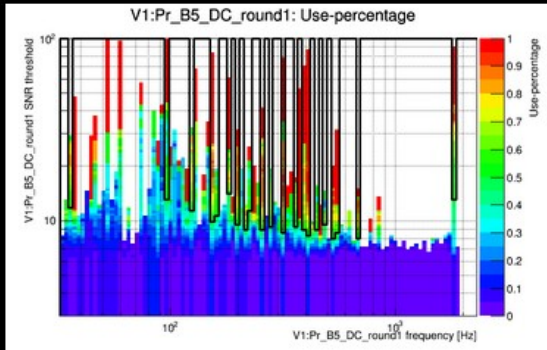
Auxiliary channel = V1:Pr_B5_DC_round1

[list of segments](#) — [list of segments up to this round](#) — [UPV output](#)

Veto Efficiency = 0.12026 (0.24515)
 Veto Dead-time = 0.00184 (0.01455)
 Veto Efficiency / Dead-time = 65.50 (16.84)
 (Up to this round)
 Veto most useful f-band = 310.9-324.5Hz
 Veto most targeted f-band = 280.3-286.3Hz

TUNING PLOTS:
[Statistics](#)
[Number of used clusters](#)
[Number of vetoed main clusters](#)
[Use-percentage](#)
[Efficiency](#)
[Efficiency / dead-time](#)

FINAL VETO PLOTS:
[Veto efficiency](#)
[Veto used triggers](#)
[SNR-SNR](#)
[Frequency-Frequency](#)
[Frequency-propagation](#)
[Veto flagging rate](#)



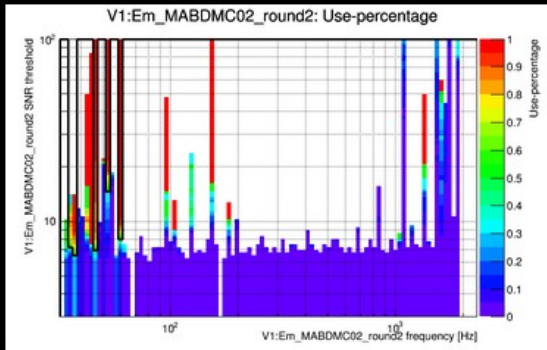
Auxiliary channel = V1:Em_MABDMC02_round2

[list of segments](#) — [list of segments up to this round](#) — [UPV output](#)

Veto Efficiency = 0.02940 (0.27455)
 Veto Dead-time = 0.00416 (0.01867)
 Veto Efficiency / Dead-time = 7.07 (14.71)
 (Up to this round)
 Veto most useful f-band = 45.5-47.2Hz
 Veto most targeted f-band = 43.0-47.2Hz

TUNING PLOTS:
[Statistics](#)
[Number of used clusters](#)
[Number of vetoed main clusters](#)
[Use-percentage](#)
[Efficiency](#)
[Efficiency / dead-time](#)

FINAL VETO PLOTS:
[Veto efficiency](#)
[Veto used triggers](#)
[SNR-SNR](#)
[Frequency-Frequency](#)
[Frequency-propagation](#)
[Veto flagging rate](#)

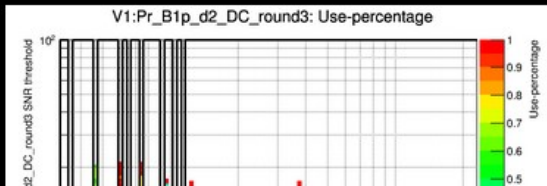


Auxiliary channel = V1:Pr_B1p_d2_DC_round3

[list of segments](#) — [list of segments up to this round](#) — [UPV output](#)

Veto Efficiency = 0.01036 (0.28491)
 Veto Dead-time = 0.00073 (0.01940)
 Veto Efficiency / Dead-time = 13.74 (14.69)
 (Up to this round)
 Veto most useful f-band = 58.8-61.4Hz
 Veto most targeted f-band = 615.2-696.4Hz

TUNING PLOTS:
[Statistics](#)



The architecture to produce online vetoes based on UPV optimized thresholds is now in place

- Thresholds are produced by UPV and saved on disk (ROOT files). We need to determine how often this tuning should be performed
- Thresholds are loaded and veto segments are produced by Omicron when the triggers are produced (latency~20s)
- Veto segments are saved in a FrEvent structure and send to SegOnline
- Threshold files can be updated anytime without stopping any process

→ Some preliminary tests were successful. More tests (greater scale) are needed.

→ Some threshold history needs to be implemented

→ We need to think about about a way to save these vetoes (I'm pessimistic about DQSEGDB)

Click on a given cell to access the page characterizing the coupling (RMS) - (RMS)

For each cell, 2 values are given:
The first one is the resulting value
The second one is the frequency conversion factor: $\omega = \omega_{\text{conversion}}$

16 x 16 conversion

UPV matrix

SOURCES →	H1:ALS- Y_QPD_B_NSUM_OUT_DQ	H1:IMC- DOF_1_P_IN1_DQ	H1:IMC- DOF_1_Y_IN1_DQ	H1:IMC- DOF_2_P_IN1_DQ	H1:IMC- DOF_2_Y_IN1_DQ
TARGETS ↓					
H1:ALS- Y_QPD_B_NSUM_OUT_DQ	0.998 / 0.00	0.000 / 0.00	0.000 / 0.00	0.000 / 0.00	
H1:IMC-DOF 1 P IN1 DQ	0.000 / 0.00	0.992 / 0.00	0.110 / 0.01	0.300 / 0.01	
H1:IMC-DOF 1 Y IN1 DQ	0.000 / 0.00	0.086 / 0.09 ↑	0.995 / 0.00	0.064 / 0.07 ↑	
H1:IMC-DOF 2 P IN1 DQ	0.000 / 0.00	0.720 / -0.00	0.092 / -0.01	0.993 / 0.00	
H1:IMC-DOF 2 Y IN1 DQ	0.000 / 0.00	0.106 / 0.07 ↑	0.807 / -0.01	0.093 / 0.06 ↑	
H1:IMC-DOF 3 P IN1 DQ	0.000 / 0.00	0.250 / -0.11 ↓	0.044 / -0.12 ↓	0.393 / -0.11 ↓	
H1:IMC-DOF 4 P IN1 DQ	0.000 / 0.00	0.078 / -0.07 ↓	0.019 / -0.06 ↓	0.143 / -0.08 ↓	
H1:IMC-DOF 4 Y IN1 DQ	0.000 / 0.00	0.129 / -0.05 ↓	0.134 / -0.04	0.162 / -0.06 ↓	
H1:IMC-F OUT DQ	0.000 / 0.00	0.012 / -0.09 ↓	0.017 / -0.01	0.058 / -0.07 ↓	

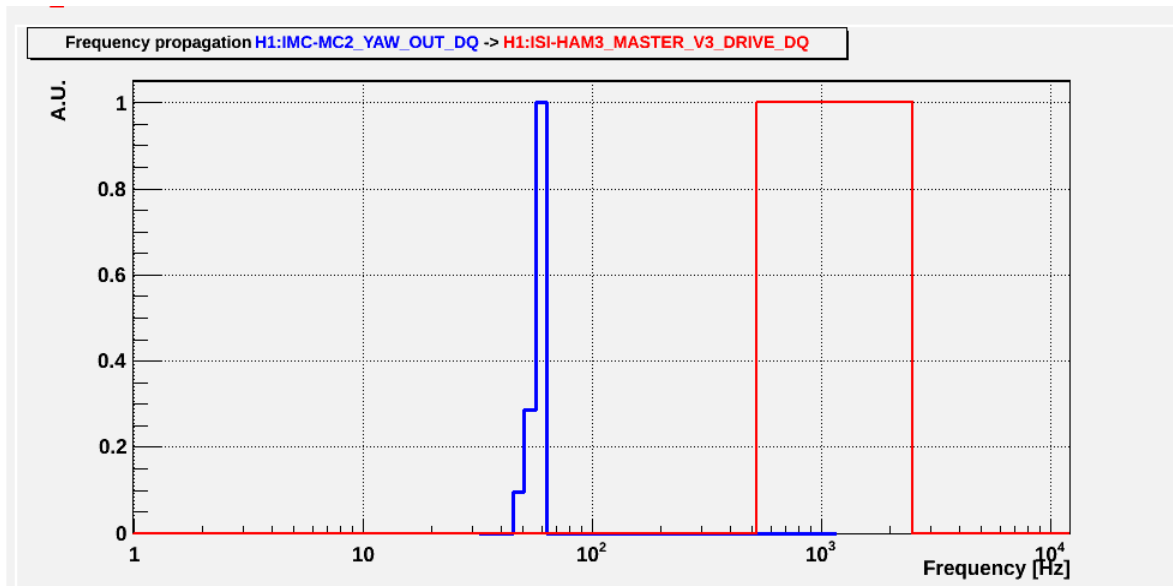
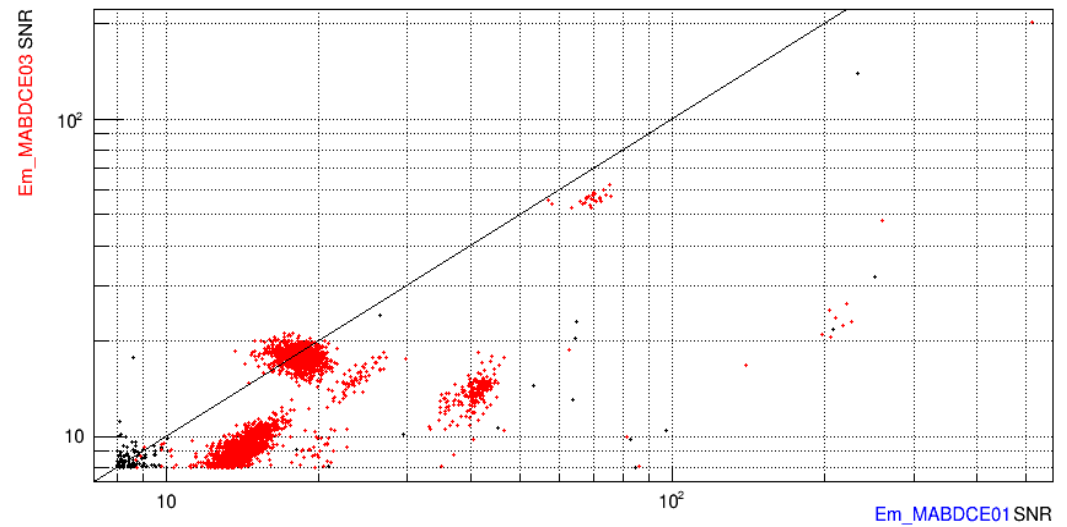
Coupling scale
(~efficiency)

Conversion factor
>0 → up conversion
<0 → down conversion

up/down conversion indicator

Diagnostic plots are provided for each identified coupling

Coincidence SNR vs. SNR (red points = identified coupling)



- The UPV algorithm can be used for many purposes (veto, online, detchar...)
- UPV is being used in LIGO (N. Christensen) → more and more stable/reliable
- UPV will be a useful tool for Virgo detchar
- UPV is user-friendly: one command line → analysis + web report
- A paper is in preparation (co-written with Nelson)