TELLIE PCA:
Processing automation

Michal Rigan

February 2023

This document describes what the TELLIE PCA automation does; why, and how.

Contents

1 Why automation

The process of extracting and validating PCA constants from TELLIE data is complex. This piece of software was developed to streamline the process of obtaining the PCA constants, at reasonably high speed.
Additionally, it was designed to:

- be independent of the method used for Data-taking
- be modular, easily modifiable and configurable
- require minimal human input
- provide monitoring
- be mostly standalone

2 Overview

The TELLIE PCA Automation overview is shown in Figure 1.
There are two main parts: TELLIE `Data-taking` and `Data-processing`.

Data-taking is done independently of the processing (as the exact method was not yet finalised before developing processing). More information can be found in TELLIE Data-taking automation document. It should be noted that `Validation #1` is taken care of by Data-processing.

Data-processing is everything that is done with TELLIE PCA data once it is stored. This includes performing checks on the data, making fits required for further processing, generating tables (both local and online), extracting PCA constants, benchmarking these constants, and a suite of monitoring for these steps. These will be described below.

3 Data-taking

3.1 Validation #1

As mentioned above, even though `Validation #1` is logistically part of Data-taking, it is performed by Data-processing, and is also independent of

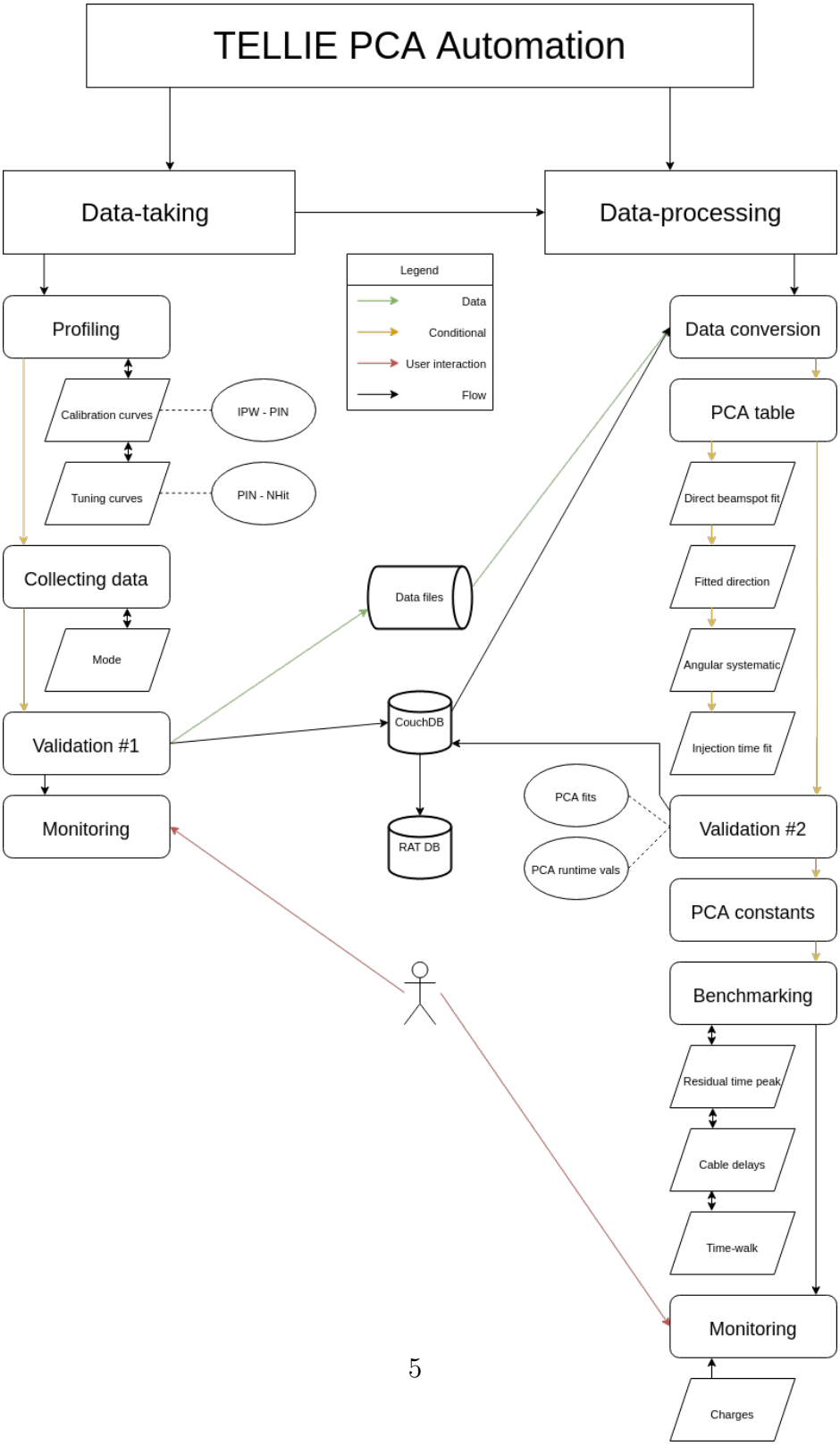


Figure 1: Overview of the TELLIE PCA Automation.

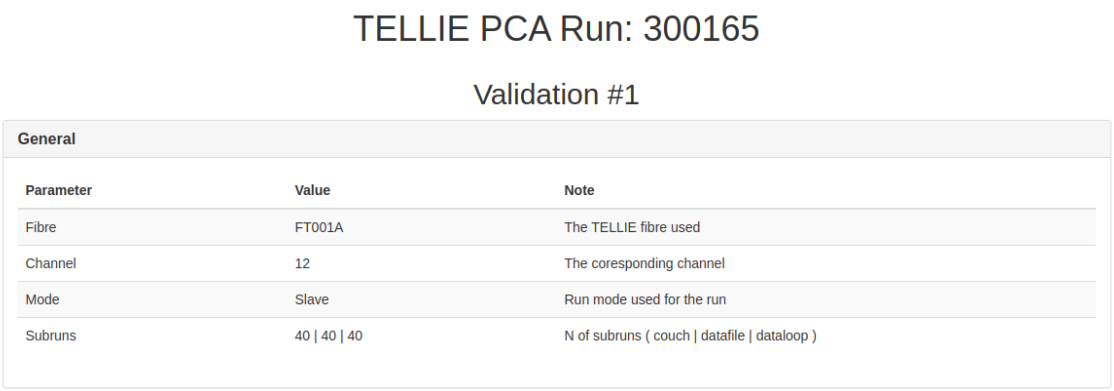


Figure 2: `Validation #1`: General information.
This is useful to confirm that the expected fibre is being used, the mode of firing is correct, and that the number of subruns is consistent.

the method used to take data.
Goal: Validate that the data is of required quality for PCA.

Some of the checks performed are: correct fibre, number of events, number of EXTA events, passed hits, cuts on PMTs, checks on LPC, run lenght, frequency, NHit distribution, NHit over time, delays, time of hits over time, number of peaks, PMTs in beamspot, PMT occupancy, PIN, PIN vs NHit, events over subrruns, and more.

The corresponding monitoring page will show basic information for this run; data for events, hits, and times; NHit information; PMTs information; Cuts; Flags; and associated plots. The flags are of special importance, as they form a 'bitword'. This bitword is shown in the list page for each fibre for particular dataset.
Examples from the monitoring are shown in Figures 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, and 16.

The code running `Validation #1` is located at:

```
Automation/Processing/validate/TestUser.cc
```

This includes the logic for the checks, and generates the bitword. Plots are also made here.

Events / Hits / Times

Parameter	Value	Note
Total events	218061	All events in data file
CouchDB events	200000	All events from couchdb (sum over subruns)
EXTA events	199993	EXTA events in data file
Passed events	199895	Events that pass cuts
Total hits	8322581	Total PMT hits
Passed hits	2100650	PMT hits that pass cuts
Hits - peaks	1	N of found peaks in dirct time distribution
Peak time	277.350874 ns	The peak of the time distribution
Direct hit time	274.661957 ns	The peak of the direct hits time distribution
Time over event - mean	275.692921 ns	Mean of the time over event distribution

Figure 3: `Validation #1`: Information regarding events, hits, and times. Important features: the number of total events should be equal to or more than CouchDB events. CouchDB events should correspond to the number of requested TELLIE events. The number of EXTA events should be close to CouchDB events (some will be lost due to stolen triggers). Most events should pass checks. The percentage of passed hits will be always low ($\sim$25%) mostly due to the angular cut, defining the beamspot. The peak times should all be very similar (within few ns). Finally, the direct hit time should not change (much) over datasets, as long as we are not chaning the environment (scintillator) anymore.

NHIT

Parameter	Value	Note
Rolling NHit min	35.328500	Minimum of the rolling NHit
Rolling NHit max	38.243194	Maximum of the rolling NHit
Rolling NHit mean	37.004809	Mean of the rolling NHit
NHit distribution - mean	37.004883	Mean of the NHit distribution (hist)
NHit distribution - rms	11.790064	RMS of the NHit distribution (hist)

Figure 4: `Validation #1`: NHit information.
The NHit should be close to the expected value (this was 40-42 for old datasets). There should only be a small variation.

PMTs

Parameter	Value	Note
Good PMTs	67.175573 %	Ratio of PMTs with good occupancy in the beamspot
PMTs in beamspot	262 | 2.693257 %	N of PMTs in the beamspot | % to all PMTs
PMTs in BS, good occupancy	176 | 1.809211 %	N of PMTs in the beamspot with good occupancy | % to all PMTs
Occupancy 1-5 to all	33.909507 %	Ratio of PMTs with good occupancy to all PMTs (integral from hist)
Occupancy 1-5 to all - beamspot	70.610687 %	Ratio of PMTs with good occupancy to beamspot PMTs (integral from hist)

Figure 5: `Validation #1`: PMT statistics.
We should make sure we are hitting PMTs in the beamspot, and that these PMTs have good occupancy (between 1-5%, as to have high enough statistics above noise to extract the constants, but to avoid being contamined by multiple PE hits).

Other

Parameter	Value	Note
EXTA length	209.000000 s	The length of the run obtained as Tlast - Tfirst EXTA event
Frequency (from length)	956.244019 Hz	Calculated frequency from the EXTA length
PIN RMS	1	The RMS of the PIN distribution (hist)
Correlation factor	0.113082	The value of the correlation factor (NHit to PIN distribution)
Covariance	0.206720	The coviariance (NHit to PIN distribution)

Figure 6: `Validation #1`: Other information, including calculated run length from EXTA events, calculated frequency, and correlation between PIN and NHit.
We should check the real run length and frequency are as expected (they will be slightly lower than expected, due to missing EXTA triggers).

Cuts

Parameter	Value	Note
CDAQ	0 | 0.00 %	N not-DAQ-enabled hits
CECA	519481 | 6.24 %	N hits with bad ECA bit
CANG	2903647 | 34.89 %	N hits not passing angular cut
CCO	0 | 0.00 %	N hits on offline channel
CPV	0 | 0.00 %	N hits with not valid path (LPC)
CPCA	373503 | 4.49 %	N hits with bad ECA bit
CXT	6961 | 0.08 %	N hits caused by cross-talk
COFF	0 | 0.00 %	N hits on offline PMT
CTIR	600645 | 7.22 %	N hits causing total internal reflection (LPC)
CRH	0 | 0.00 %	N of hits not within locality (LPC)
CMAG	0 | 0.00 %	N of hits on PMT with bad position
CNEXTA	18068 | 8.29 %	N of not EXTA events
CE	0 | 0.00 %	N of hits on not enabled PMT
CDIST	1792853 | 21.54 %	N of near reflection hits (given by LPC endpoint)
CNN	0 | 0.00 %	N of hits on not-normal PMT
CCHS	23992 | 0.29 %	N of hits on bad channel
CDAV	849 | 0.01 %	N of hits with weird path (not completely through AV)

Figure 7: `Validation #1`: Hit statistics, showing the number of hits that were cut, by category.
Check cuts on hits and PMTs. Should especially look for outliers. The angular cut will always be big, and that is ok, as we only want to use PMTs in the beamspot (other fibres will cover other PMTs, to get the preferred occupancy).

Flags

Parameter	Value	Note
Runtime	1	Is the real length close to expected?
Frequency	1	Is the real frequency close to expected?
Subruns	1	Is the number of subruns as expected?
Check on evens in subruns	1	Are there enough events in each subrun?
Event	1	Are there enough events that passed cuts?
EXTA	1	Are there enough EXTA events?
NHit distribution	0	NHit distribution: mean and RMS within limits?
NHit distribution over subruns	1	Is the NHit avg over subruns stable?
Stable NHit	1	NHit fluctuations (rolling avg to global avg) within limits?
PIN RMS	0	Is the PIN distribution RMS within limits?
PIN-NHit cov and corr	0	Is the PIN-NHit relation sensible?
Hit	1	Are there enough valid hits?
Angular cut	1	N of hits cut due to angular cut within limits?
Offline channel	1	N of offline channel hits within limits?
Offline PMT	1	N of offline PMT hits within limits?
Not DAQ enabled	1	N of not DAQ enabled hits within limits?
Not enabled flag	1	N of not enabled channel hits within limits?
Bad PMT position	1	N of bad PMT position hits within limits?
Bad channel	1	N of bad channel hits within limits?
Bad ECA	1	N of bad ECA hits within limits?
Bad PCA	1	N of bad PCA hits within limits?
X-talk	1	N of cross-talk removed hits within limits?
Not normal PMT	1	N of not normal PMT hits within limits?
Hit peak	1	Is the peak of the time distribution as expected?
Peak check flag	1	Direct hits peak value within limits?
Time over event	1	Time fluctuations (rolling avg to global avg) within limits?
Trigger delay dev	1	Is the trigger delay constant over subruns?
Fibre delay dev	1	Is the fibre delay constant over subruns?
Beamspot PMTs	1	Are there enough PMTs in the beamspot?
Occupancy all	1	Are there enough PMTs with good occupancy?
Check on PMTs (beamspot / occupancy)	0	Are there enough good PMTs in beamspot (as number)?
Occupancy beamspot	1	Are there enough PMTs with good occupancy in the beamspot (integral to beamspot PMTs)?
LPC locality	1	N of hits outside of LPC locality within limits?
Near reflection	0	N of near reflection hits within limits?
LPC TIR	1	N of hits with total internal reflection (LPC) within limits?
LPC invalid path	1	N of hits with invalid path (LPC) within limits?
LPC weird path (not AV)	1	N of hits with weird LPC path within limits?

Figure 8: `Validation #1`: Flags. These are the results of checks/tests on a variety of run/hits atributes. The comments should be a reasonable hint to what each check does.

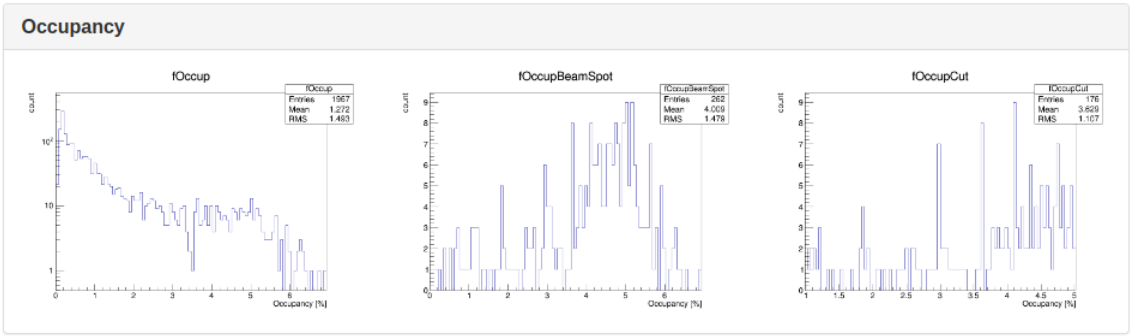


Figure 9: `Validation #1`: Plots showing occupancy information.
We should make sure that there are enough PMTs in the beamspot with the required occupancy (>100).

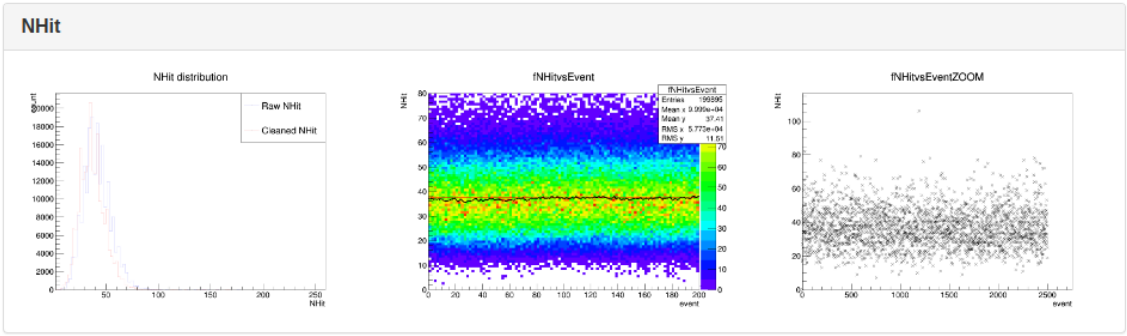


Figure 10: `Validation #1`: Plots showing NHit information.
The NHit distribution should be approximately gaussian, with the peak around the expected value (40-42). The NHit should be stable over time, and there should be no noticeable drop at the start of the run.

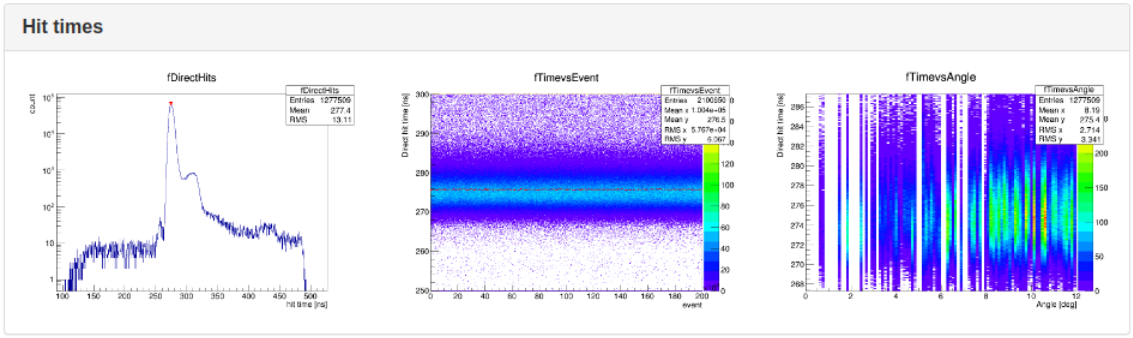


Figure 11: `Validation #1`: Plots showing hit times information. Similar to NHit, we should confirm the peak is around expected value, the times are consistend over events, and there is no high variation with angle.

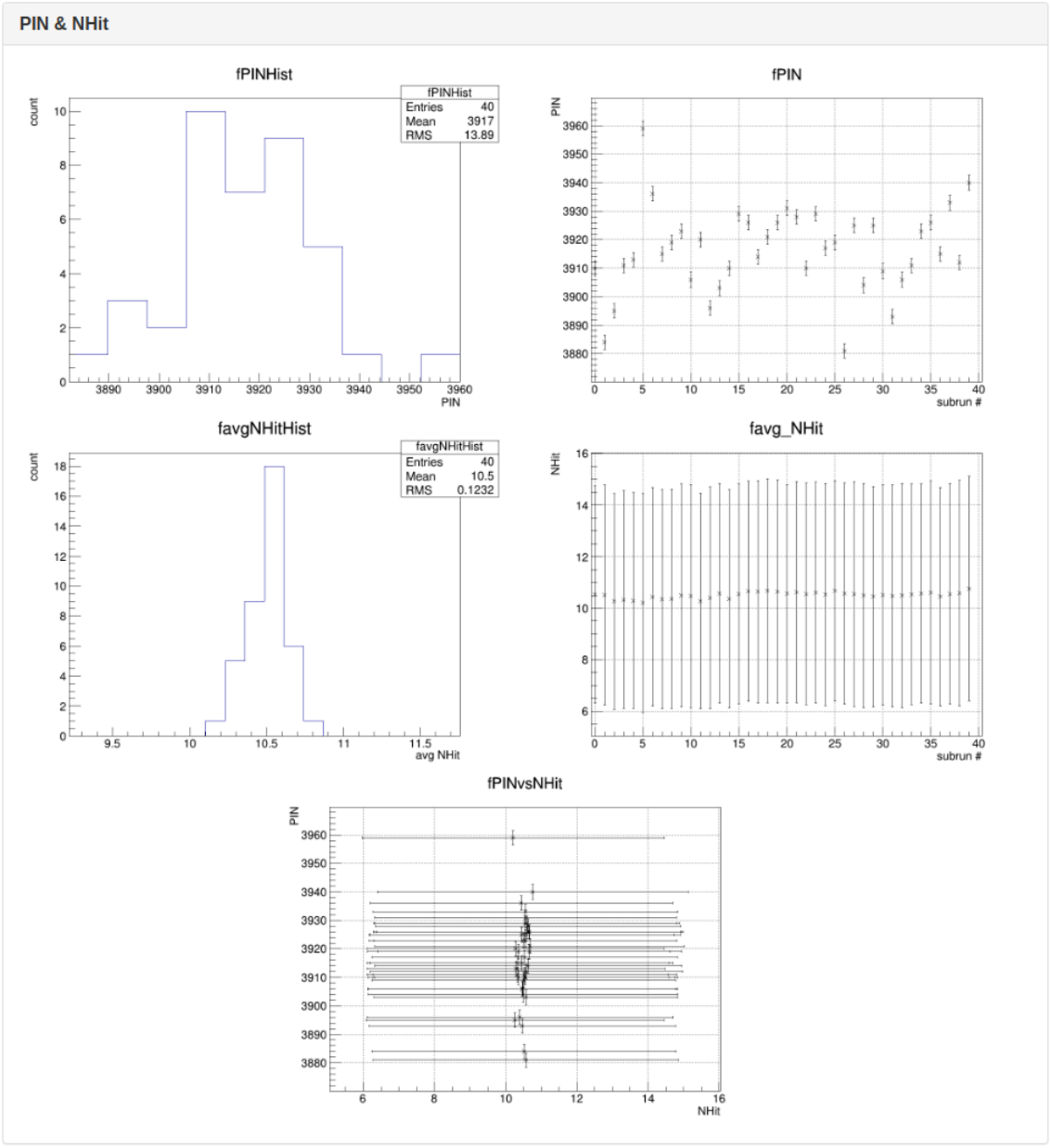


Figure 12: `Validation #1`: Plots showing PIN and NHit information. The PIN values should have low variation (within 100s) and there should be no clear trend over subruns. The NHits here are per subrun, therefore the low value. It should be stable over subruns. Finally, there should be positive correlation between the NHit and PIN (mind the range of erros).

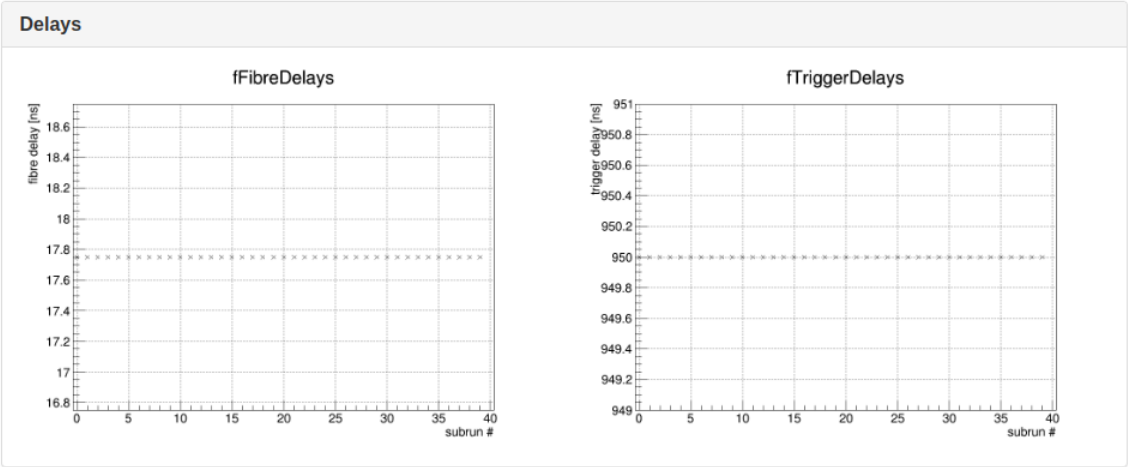


Figure 13: `Validation #1`: Plots showing delays information.
Both fibre and trigger delay have to be constant over subruns! Additionally, the trigger delay should be constant across fibres for a dataset. This is essential for the calibration.

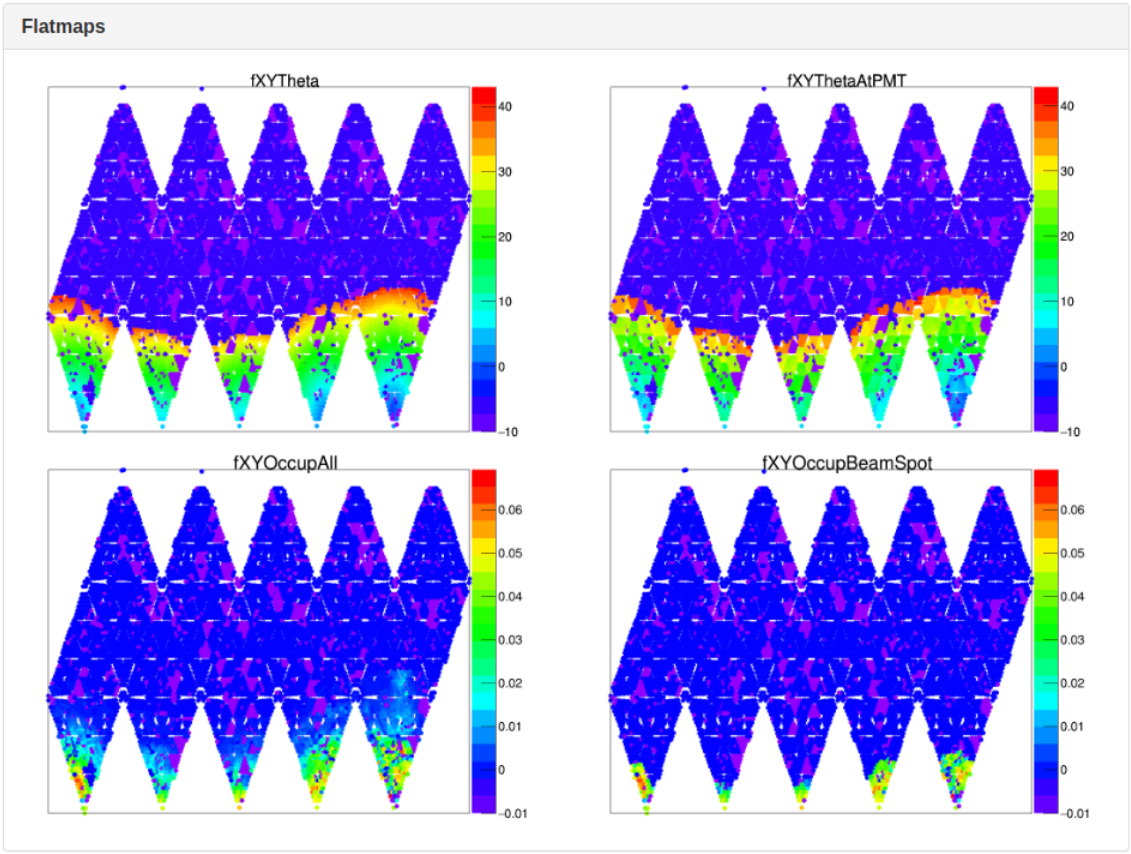


Figure 14: `Validation #1`: Plots showing data using flatmaps.
These are to confirm we are getting a beamspot, that the light is there, is not spread out too much, and the occupancy is correct in the beamspot.

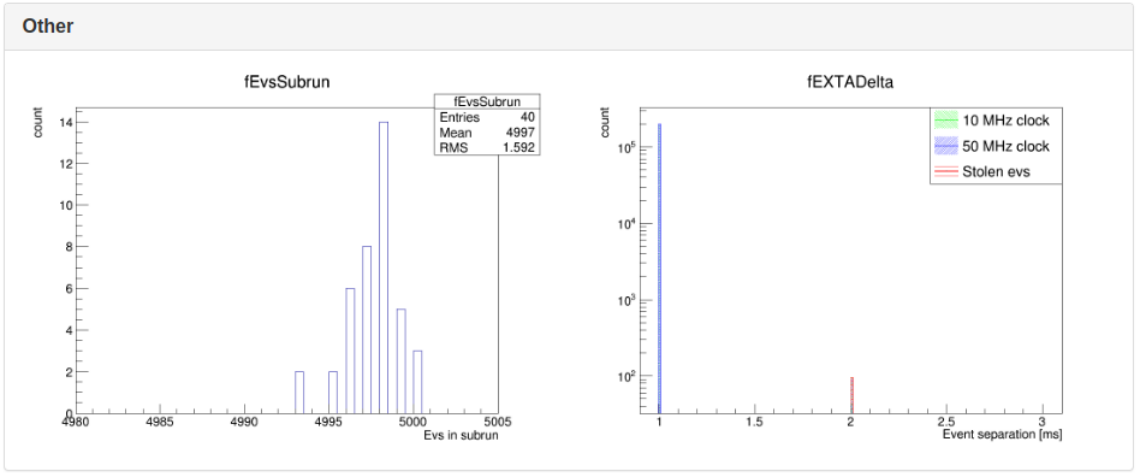


Figure 15: `Validation #1`: Other information.
The number of events should be approximately constant over subruns (again, we lose some due to stolen triggers.

TELLIE PCA dataset:

Fibre	Run	Validation 1	Fitting	Validation 2
FT001A	300165	11111111111111111101011111011111100	43.86 ± 0.093 | 190.72 ± 1.378	111111111110
FT002A	300167	11111101111111111111111111111011110110	30.42 ± 0.080 | 185.21 ± 1.373	111111111000
FT003A	300372	11011101111111111111000111111011110101	7.96 ± 0.083 | 185.64 ± 1.367	111111011111

Figure 16: `Validation #1`: The bitword from `Validation #1` shown on the dataset list page for selection of runs.
The bitword here is made up from the values of flags, corresponding to the checks, for each fibre. Clicking this bitword leads to the `Validation #1` page for this run.

4 Data-processing

4.1 PCA table generation

There are several corrections that need to be fitted for, which are later used for the extraction of PCA constants. The fits need to happen in succession, as the output of one feeds into the next. Between steps, these are stored as text files. After all fits are made, a local table is produced, combining the corrections. This table is loaded by the PCA Processor.
Goal: Make fits, obtain corrections required for the extraction of PCA constants. Produce final PCA table.

These fits are: beamspot fir, fibre direction, angular systematic, injection time. These are discussed below.

4.1.1 Fit: beamspot

First fit to data is the beamspot fit. The idea is to fit PMTs depending on the NHit. PMTs directly opposite the fibre will have the highest NHit, which will drop as the angle (PMT wrt fibre) increases. The method splits PMTs into faces (triangles). It should be mentioned that this could be improved by a dynamic NHit (correcting for offline PMTs and shadows from objects in the way).

Please look at Figure 17 for an example fit. For real data please look at the monitoring section, Section 4.6.

Code-wise, this is located at :

```
Automation/Processing/postition_fit/TestUser.cc
```

4.1.2 Fit: direction

Once the beamspot position/center is known, the fibre direction can be extracted. The fibre position as in ratDB is assumed correct (we don't expect this to change), however, the direction (or at least the effective direction) can change, depending on the status of the PMTs (offline, low rate) and objects in the way (belly plates, ropes, filling tubes, etc). Recalculating this direction

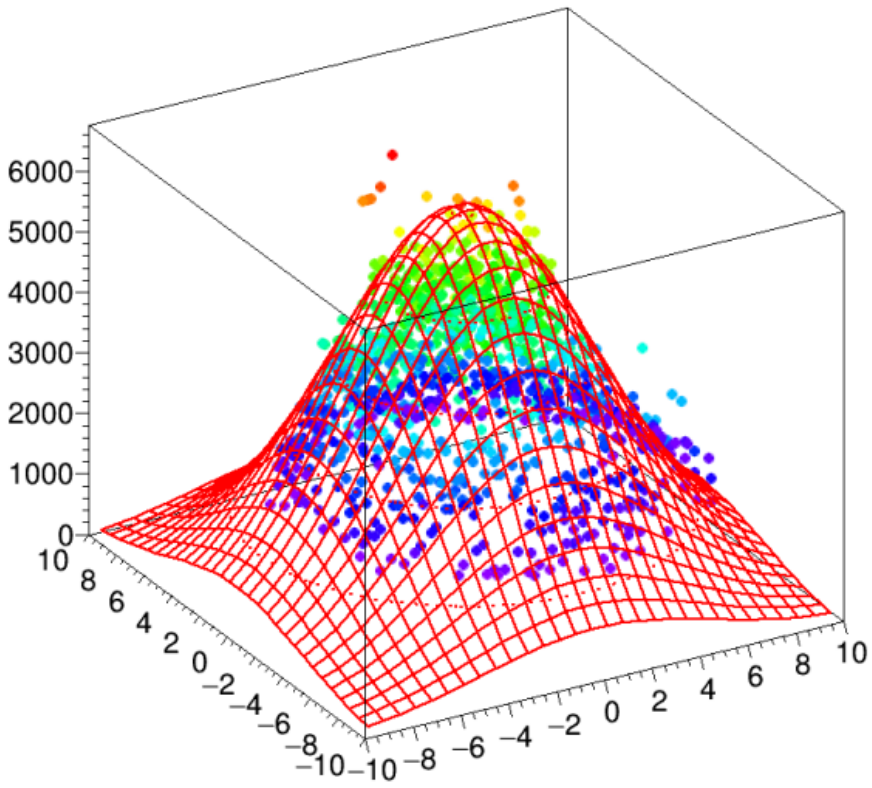


Figure 17: `Fit`: beamspot.

allows for a better definition of the beamspot, and this leads to better use of the PMTs during the calibration - there are angular cuts on the PMTs later on (as we only want to calibrate the PMTs mostly directly across the fibre - in straight line, without reflections, refractions, and scattering).

Please look at Figure 18 for an example fit. For real data please look at the monitoring section, Section 4.6.

Code-wise, this is located at the same as the beamspot fit (these two fits are done in single instance):

```
Automation/Processing/postition_fit/TestUser.cc
```

For more on the beamspot and fibre direction fit please look at TELLIE fibre validation.

4.1.3 Fit: angular systematic

The angular systematic is (in this case) a name encapsulating the correction of the effect of modal dispersion for an optical fibre. The light at an angle is injected into the detector at later times, increasing with the angle. This is shown in Figure 19.
For this purpose, the PMT residual hit times are fitted for, and the peak hit times for PMTs are plotted as a function of the angle for a fibre. Then, a fit is performed for the data. The fit function is based on a simplified geometric model of a fibre, Figure 20. There are two parameters describing the fit function, ang a and ang b. The ang b parameter is used for time correction in the PCA processor.

Small note: because angular systematic fit requires residual hit times, previous PCA calibration data is required.

Code-wise, this is located at:

```
Automation/Processing/angular_systematic/TestUser.cc
```

For more on the angular systematic effect and the fit please look at TELLIE fibre validation.

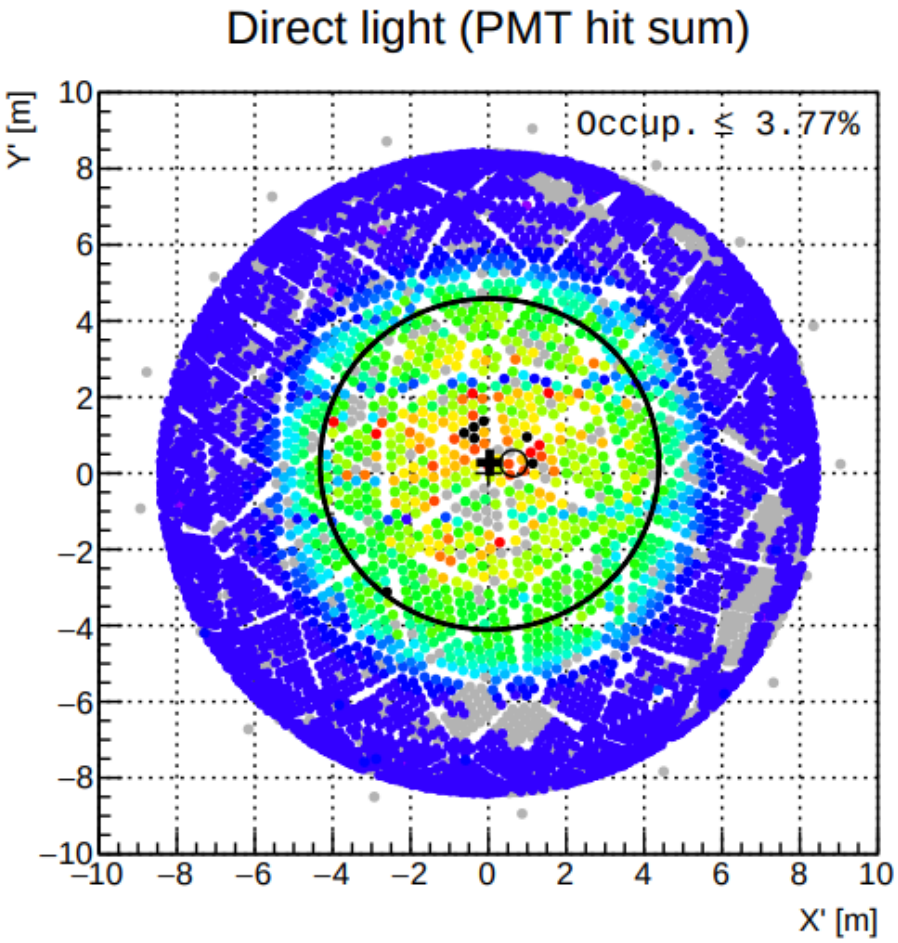


Figure 18: `Fit`: fibre direction.

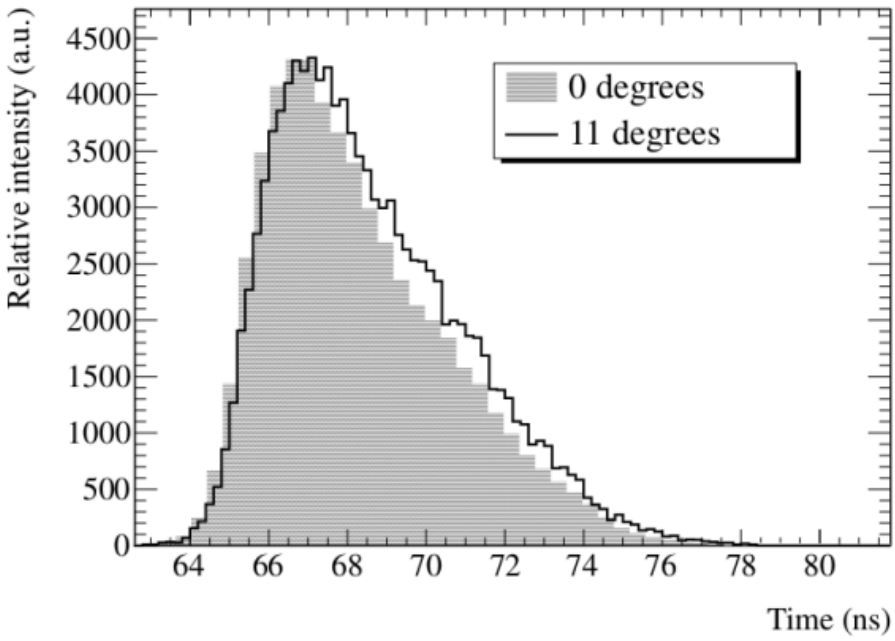


Figure 19: `Fit`: angular systematic.
The modal effects within the fibre. Light at an angle is injected later into the detector.

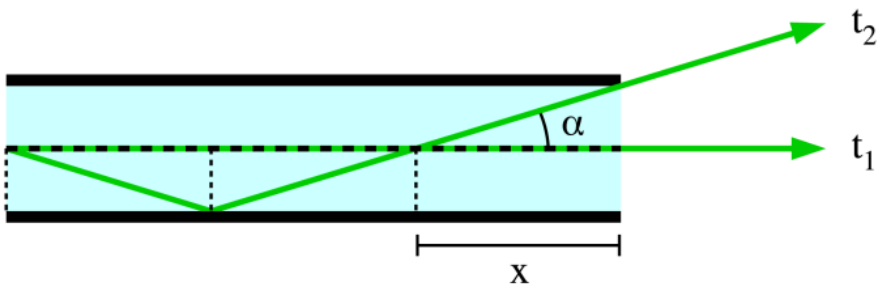


Figure 20: `Fit`: fibre direction.
The simplified geometric model for light modal effects in an optical fibre.

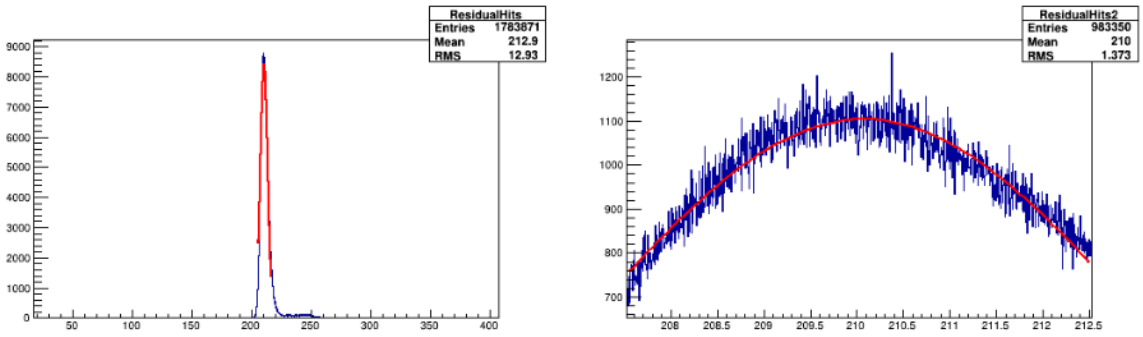


Figure 21: `Fit`: injetion time.
The residual hit times are fitted to obtain the injection time (pca offset).
Right plot is a zoom over the peak.

4.1.4 Fit: Injection time

The injection time is the time light leaves the fibre (and enters the detector). This is defined as:

$$Injection\,time = Hit\,time - Bucket\,time - Angular\,correction$$

This is sometimes called pca offest, and is needed for PCA extraction in the PCA processor. Theme method is to obtain the raw hit times, apply the corrections above, and fit the peak of that distribution, as shown in Figure 21.

Code-wise, this is located at:

```
Automation/Processing/pca_offset/TestUser.cc
```

For more details on bucket time, please look at PMT bucket time.

4.2 Comparing PCA tables

Goal: Compare the PCA tables (including corrections) for neighbouring datasets. This is useful to monitor stability of the system and to highlight outliers. Also a sense check that the values are in-line with previous sets (nothing went horribly wrong).

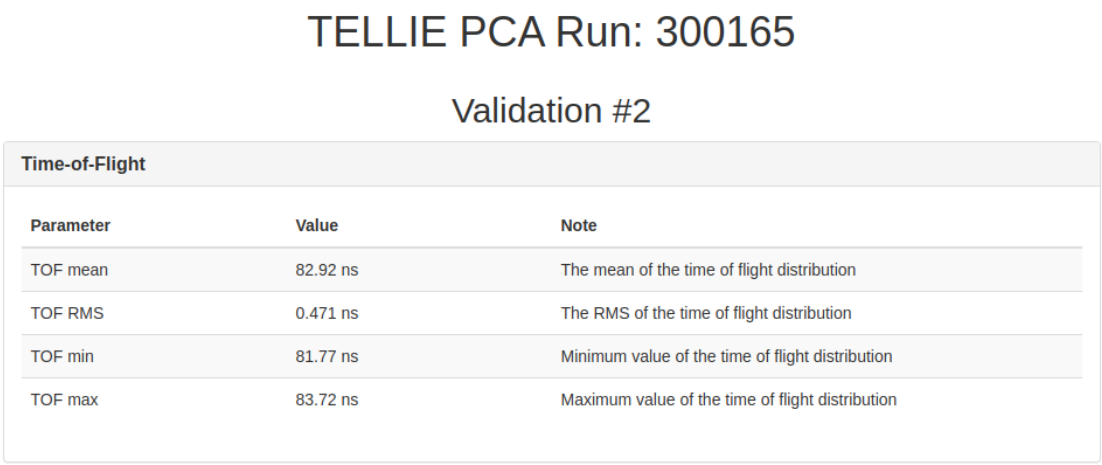


Figure 22: `Validation #2`: Information relating to Time-of-Flight. The mean should always be physical, and the RMS should not be very high (>0.5 ns).

The code is located at:

```
Automation/Processing/pca_tables/compare_two.py
Automation/Processing/pca_tables/compare_all.py
```

This creates PCA table plots shown in the monitoring section, Section 4.6.

4.3 Validation #2

Similar to `Validation #1`, `Validation #2` runs checks on data. In this case, it loads the corrections from the PCA table including the fits.
Goal: Check and confirm that the fits are sensible.

Some of the tests included here are: mean, rms, min, and max for each correction; distribution, peak(s), and angular dependence for the residual times; specific trends, and more.

The corresponding monitoring page will show the fits information for this run, including Flags; and associated plots. The flags are of special importance, as they form a 'bitword'. This bitword is shown in the list page for

Angular systematic effect

Parameter	Value	Note
Ang sys mean	0.47 ns	The mean of the AngSys distribution
Ang sys RMS	0.270 ns	The RMS of the AngSys distribution
Ang sys min	0.00 ns	Minimum value of the AngSys distribution
Ang sys max	0.98 ns	Maximum value of the AngSys distribution

Figure 23: `Validation #2`: Information relating to angular systematic fit. The mean should never be too high (>2 ns).

Bucket time

Parameter	Value	Note
Bucket mean	0.47 ns	The mean of the bucket time distribution
Bucket RMS	0.003 ns	The RMS of the bucket time distribution
Bucket min	0.46 ns	Minimum value of the bucket time distribution
Bucket max	0.47 ns	Maximum value of the bucket time distribution

Figure 24: `Validation #2`: Information relating to bucket time. The mean should always be (almost) constant (∼0.47 ns) with low RMS.

Hit times, angular dependence

Parameter	Value	Note
Direct peaktime	273.942780 ns	The peak of the direct time distribution
Direct peaks	1	The number of peaks in the direct time distribution
Residual peaktime	189.815674 ns	The peak of the residual time distribution
Residual peaks	1	The number of peaks in the residual time distribution
Direct vs Angle fit	274.97 ± 0.089 ns	Average of the direct time distribution as a function of angle
Residual vs Angle fit	190.67 ± 0.142 ns	Average of the residual time distribution as a function of angle
Dir-Resid fit	-82.97 ± 0.997	Straight line fit to the direct vs residual time distributions

Figure 25: `Validation #2`: Direct and residual hit times. There should only be 1 peak for each distribution, and the peak times should be where expected. A straigh line is expected.

each fibre for particular dataset.
Examples from the monitoring are shown in Figures 22, 23, 24, 25, 26, and 27.

The code running `Validation #2` is located at:

```
Automation/Processing/validate2/TestUser.cc
```

This includes the logic for the checks, and generates the bitword. Plots are also made here.

Flags

Parameter	Value	Note
TOF flag	1	Is the TOF distribution sensible (mean and RMS)?
TOF min-max flag	1	Are the min and max of TOF distribution ok?
TOF progression flag	1	Is the TOF descreasing with angle?
Ang sys flag	1	Is the AngSys distribution sensible (mean and RMS)?
Ang sys min-max flag	1	Are the min and max of AngSys distribution ok?
Ang sys progression flag	1	Is the AngSys increasing with angle?
Bucket flag	1	Is the bucket distribution sensible (mean and RMS)?
Bucket min-max flag	1	Are the min and max of bucket distribution ok?
Resid peak flag	1	Is there only one peak in the residual time distribution? Is it close to expected value?
Direct vs Angle fit flag	1	Is the average of direct light reasonable? Is the distribution flat?
Residual vs Angle fit flag	0	Is the average of residual light reasonable? Is the distribution flat?
Dir-Resid fit flag	1	Is the straight line fit within limits?

Figure 26: `Validation #2`: Flags. These are the results of checks/tests on a variety of run/hits atributes. The comments should be a reasonable hint to what each check does.

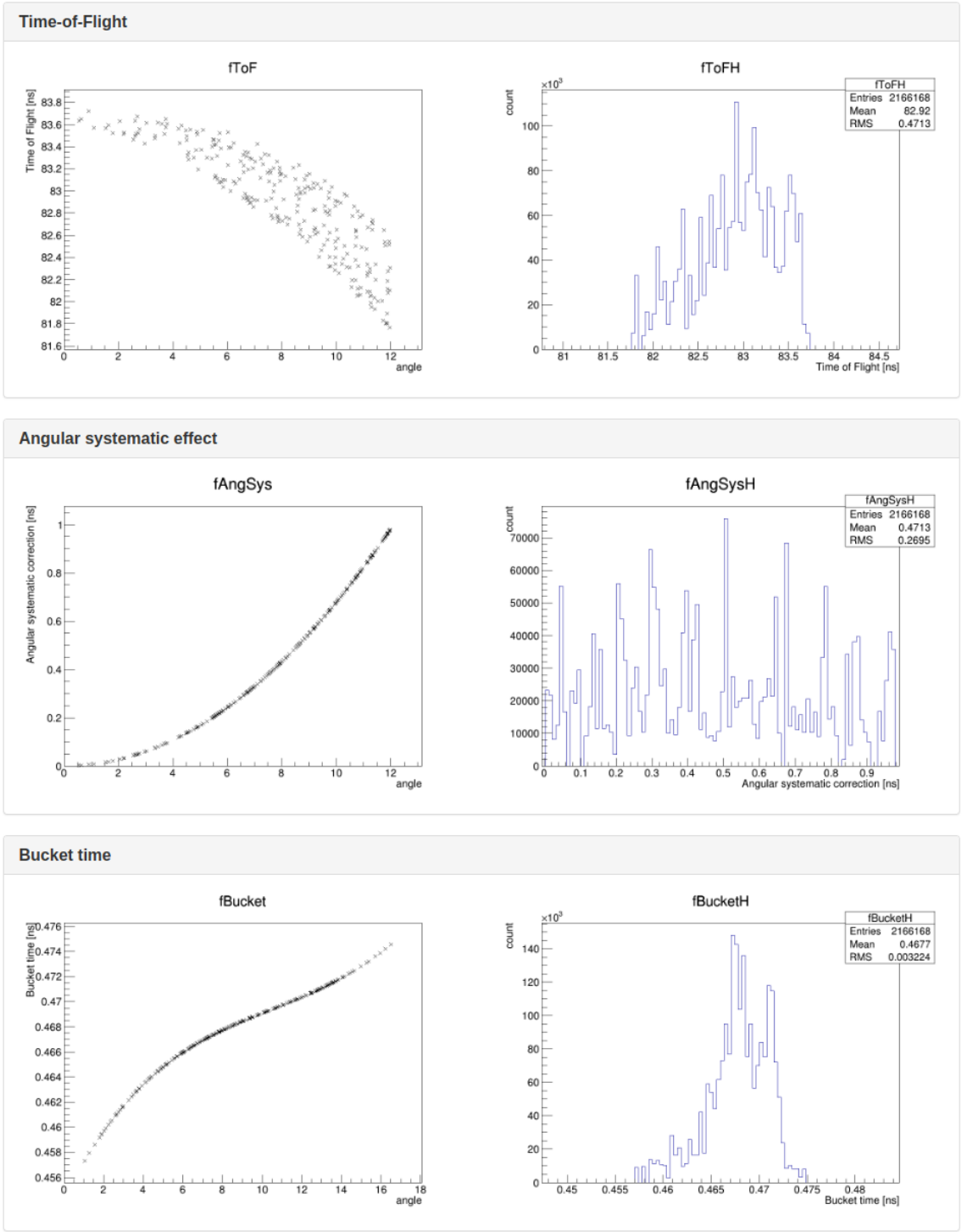


Figure 27: `Validation #2`: Plots showing the distribution of the corrections.

ToF should be decreasing over angle, peaking around 83 ns and can never be more than 84 ns. The angular systematic correction has to be increasing with the angle. Bucket time should be a narrow distribution around 0.47 ns.

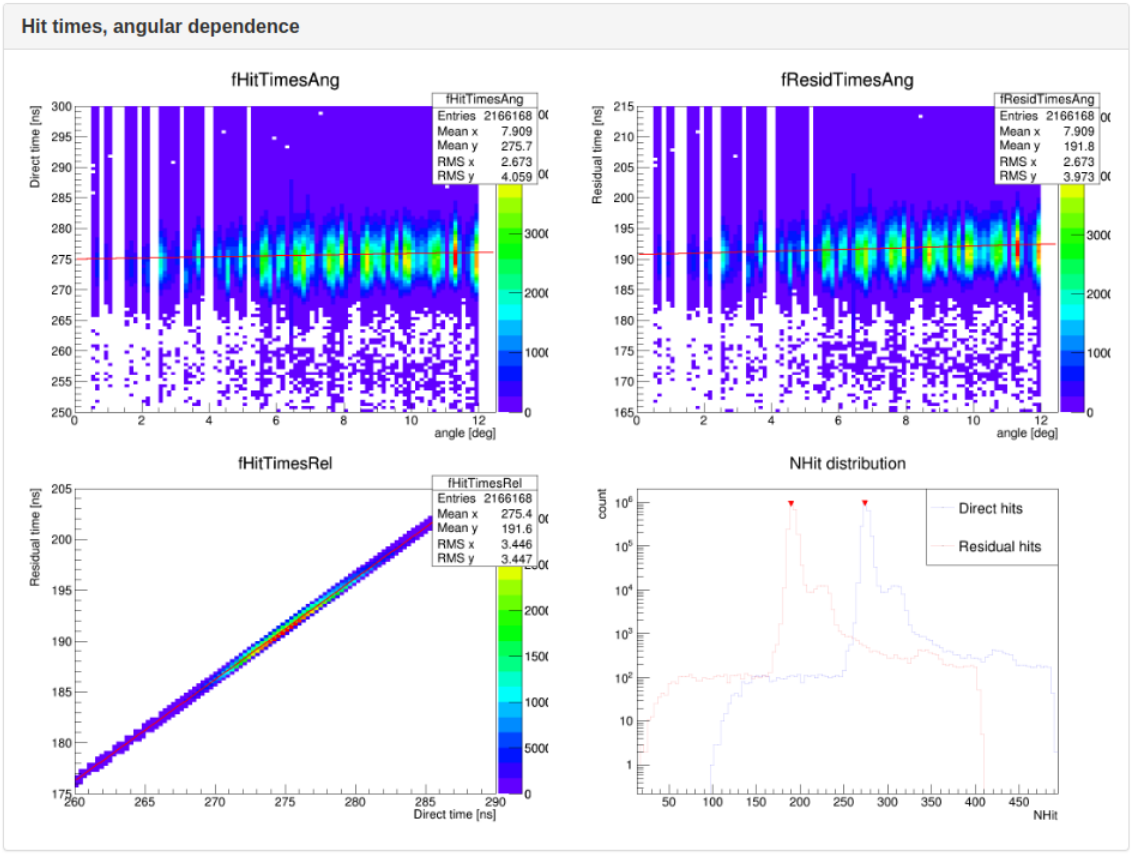


Figure 28: `Validation #2`: Plots showing the angular dependence of hit times, both raw and residual.
After corrections, the hit times should be mostly independent of the angle.

4.4 PCA constants

Goal: Run the PCA Processor that extract the PCA constants (both timing and charge).

This is the main step of the loop. The RAT PCA processor is run over the dataset. It loads the PCA table that cointains the corrections that was created in previous steps. The PCA constants are extracted here and stored as ratdb tables. PCA log files are also created here.

For details on PCA calibration, please read PCA calibration with SNO+ RAT.

4.5 Benchmarking

This process is used to actually load, use, and compare the PCA constants - cable delays, time-walk fit, and charge fits: threshold, peak, hhp for QHS and QHL.
There are two ways to achieve this comparison. One is to purely look at the values of the constants (time and charge) and compare them directly. The other one is to apply the new constants to a run: ie calibrate PMTs with these new constants, and analyse hit times in a well understood run. Usually the latest well understood laserball run is used (117567). Then, a residual hit times distribution is obtained from data.
Goal: Compare the set of constants against the closest (previous) set. Useful to see overall stability and for highlighting outliers.

As usual, look at the corresponding subsection of the Monitoring section, Section 4.6.

4.6 Monitoring

This is the most important part for the user. Every step of the system creates logs and plots that are available online on Minard. There are many pages, and everything important (and more) is presented. **Goal:** Provide monitoring of each step of the chain. Also compares fibres and PMTs between datasets.

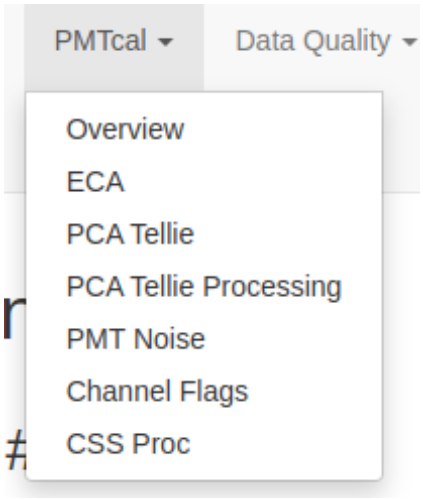


Figure 29: `Minard`: Location of PCA Tellie Processing button.
This is inside the PMTcal tab.

Most parts of the system will be highlighted here. A good rule to using minard portion of the TELLIE PCA processing is that most things are clickable and will lead to detailed page with logs and plots.

4.6.1 Where

The minard page for TELLIE PCA Processing can be accessed using the main header bar, by slecting the **PMTcal** tab, and clicking the **PCA Tellie Processing**, as shown in Figure 29.

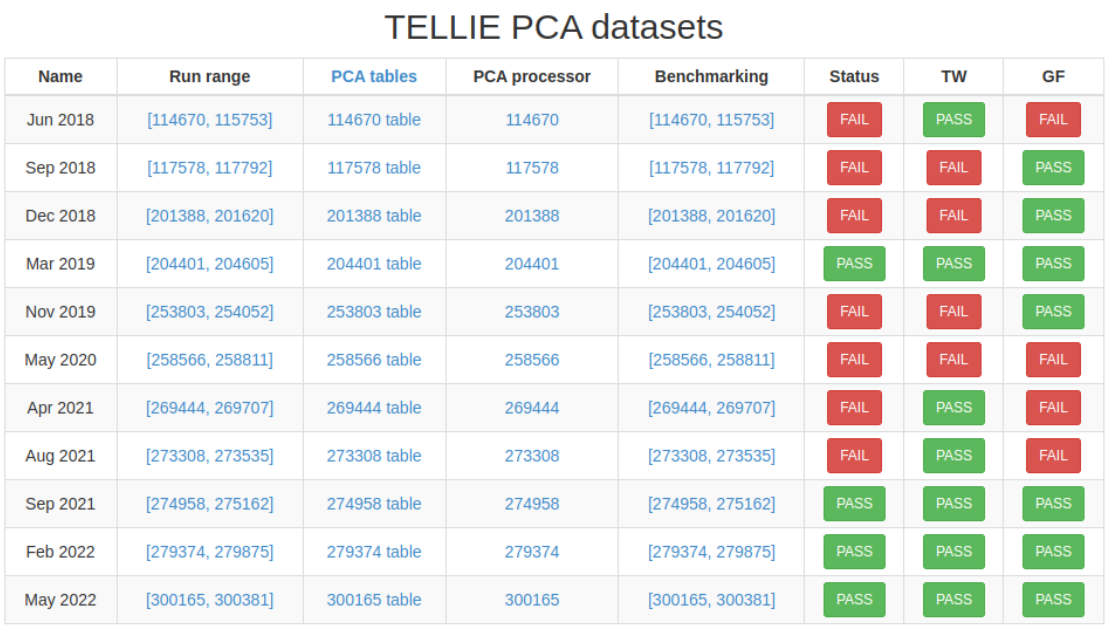
4.6.2 What

This leads to the **TELLIE PCA datasets page**. This is the main/overview page for the TELLIE PCA Processing. It shows the list of datasets (Figure 30), the PMT search box (Figure 31), and the list of thresholds (Figure 32).

The list of datasets (Figure 30) is the starting point. Again, most cells are clickable and will lead to more detailed page:

- Clicking on the cell in 'Run range' will bring an overview page for that particular dataset, Figure 33.

- The 'PCA table' cell will show a page comparing the PCA table associated with that dataset to previous PCA table. Figures 34, 35, 36, and 37.
- 'PCA processor' page shows the results of PCA Processor: Figures 38, 39, 40, 41, 42, and 43.
- 'Benchmarking' shows plot comparing the PCA constants obtained in this dataset to previous constant, shown in Figures 44, 45, 46, 47, 48, 49, and 50.
- 'Status' presents information from parsed log file from PCA Processor. It will list warnings or errors from the PCA processor, Figure 51.
- 'TW' shows the time-walk information for this set of constants. PMTs with issues such as too high RMS or high Q tail are listed here. See Figures 53, 54, 55, and 56.
- 'GF' shows the time-walk information for this set of constants. PMTs with issues such as QHS TH too high or Peakfinder found multiple peaks are listed here. Figures 52, 54, and 56.
- Additionally, clicking the 'PCA tables' header lead to page comparing PCA tables across all available datasets. Note Figures 57, 58, and 59.
- It is also possible to look at fibre data across datasets (this is perhaps the most interesting feature). One can get here by clicking the fibre name on the dataset page (Figure 33). A selection of plots are shown in Figure 60 and Figure 61.
- Finally, there are plots for PMTs for each dataset (Figure 62 and Figure 63) and over datasets (using PMT search, Figure 31).



TELLIE PCA datasets

Name	Run range	PCA tables	PCA processor	Benchmarking	Status	TW	GF
Jun 2018	[114670, 115753]	114670 table	114670	[114670, 115753]	FAIL	PASS	FAIL
Sep 2018	[117578, 117792]	117578 table	117578	[117578, 117792]	FAIL	FAIL	PASS
Dec 2018	[201388, 201620]	201388 table	201388	[201388, 201620]	FAIL	FAIL	PASS
Mar 2019	[204401, 204605]	204401 table	204401	[204401, 204605]	PASS	PASS	PASS
Nov 2019	[253803, 254052]	253803 table	253803	[253803, 254052]	FAIL	FAIL	PASS
May 2020	[258566, 258811]	258566 table	258566	[258566, 258811]	FAIL	FAIL	FAIL
Apr 2021	[269444, 269707]	269444 table	269444	[269444, 269707]	FAIL	PASS	FAIL
Aug 2021	[273308, 273535]	273308 table	273308	[273308, 273535]	FAIL	PASS	FAIL
Sep 2021	[274958, 275162]	274958 table	274958	[274958, 275162]	PASS	PASS	PASS
Feb 2022	[279374, 279875]	279374 table	279374	[279374, 279875]	PASS	PASS	PASS
May 2022	[300165, 300381]	300165 table	300165	[300165, 300381]	PASS	PASS	PASS

Figure 30: `Minard`: The main table of the main page.
This table holds a line for each processed dataset. Most cells in here are clickable, leading to a more detailed page or list. These get auto-populated with the processing suite.

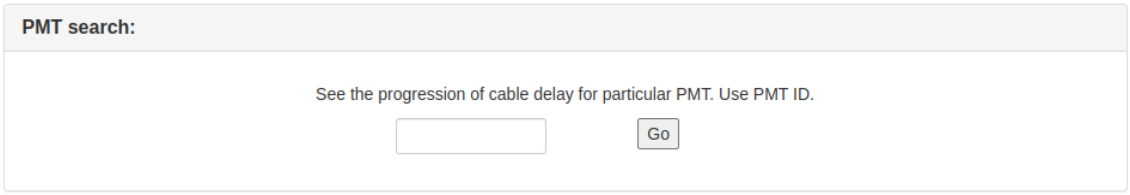


Figure 31: `Minard`: PMT search section.
One can input a PMT number and a detailed page showing the PMT cable delays over all datasets will be shown.

Limits

Parameter	Value	Note
RUNTIME	2500	[*s*]: allowed length of run
FREQ ± FREQ_DEV	1000 ± 7.5	[*Hz*]: expected frequency + dev
TOT_EVS ± DEV	200000 ± 1	[*N*]: expected number of events + dev
TH_EXTA	10	[%]: *threshold* = maximal allowed percentage of NON-EXTA events
TOT_HITS	20	[%]: *threshold* = minimal allowed percentage of passed hits
N_SUBS	40	[*N*]: number of allowed subruns
EV_SUB ± EV_SUB_DEV	5000 ± 0.5	[*N*]: expected events in subrun + dev
MIN_DIST	900	[*mm*]: allowed distance to evaluate the fibre that was firring (light fit to fibre db position)
DIR_LIGHT_ANG	48	[°]: angular limits for the cone of direct light
REF_LIGHT_ANG	20	[°]: angular limits for the cone of reflected light
NHIT_DIV	0.2	[*R*]: allowed deviation for the ratio of rolling average to global average of the NHit distribution (fibre stability)
NHIT_MEAN ± NHIT_MEAN_DEV	42 ± 12	[*NHit*]: expected mean of the NHit distribution + dev
NHIT_RMS	10	[*NHit*]: allowed RMS for the cleaned NHit dist
SUB_NHIT ± SUB_DEV	10 ± 1	[*NHit*]: expected mean of the cleaned NHit distribution (subrun) + dev
PIN_RMS	3	[*PIN*]: maximal allowed RMS of the pin distribution

Figure 32: `Minard`: Limits section.
This section lists the environment thresholds (used for validation checks).

TELLIE PCA dataset:

Fibre	Run	Validation 1	Fitting	Validation 2
FT001A	300165	11111111111111111101011111011111100	43.86 ± 0.093 | 190.72 ± 1.378	111111111110
FT002A	300167	11111101111111111111111111011110110	30.42 ± 0.080 | 185.21 ± 1.373	111111111000
FT003A	300372	11011101111111111110001111011110101	7.96 ± 0.083 | 185.64 ± 1.367	111111011111
FT004A	300171	11111101111111111111001111011100001	50.88 ± 0.085 | 183.27 ± 1.369	111111111111
FT005A	300173	11111101111111111111111111111111111	62.46 ± 0.090 | 184.10 ± 1.370	111111011111
FT006A	300175	11111111111111111111111111111111110	43.07 ± 0.095 | 185.43 ± 1.378	111111111111
FT007A	300177	11111111111111111111001111111111101	30.48 ± 0.073 | 185.46 ± 1.365	111111111111
FT008A	300179	11111111111111111111001111011100010	65.20 ± 0.085 | 184.67 ± 1.361	111111111111
FT009A	300181	11111111111111111101111111111111110	25.81 ± 0.081 | 184.77 ± 1.366	111111111111
FT010A	300183	11111101111111111111001111111111110	16.02 ± 0.081 | 185.72 ± 1.372	111111111000

Figure 33: `Minard`: A list for TELLIE PCA dataset.
. This table shows the bitwords for both validations (see Section 3.1 and Section 4.3) and results of main fitting parameters (angular b and injection time, see Section 4.1). Clicking a cell brings up more detailed page with data and plots. One can also click on the fibre name, which shows data for that fibre across all datasets.

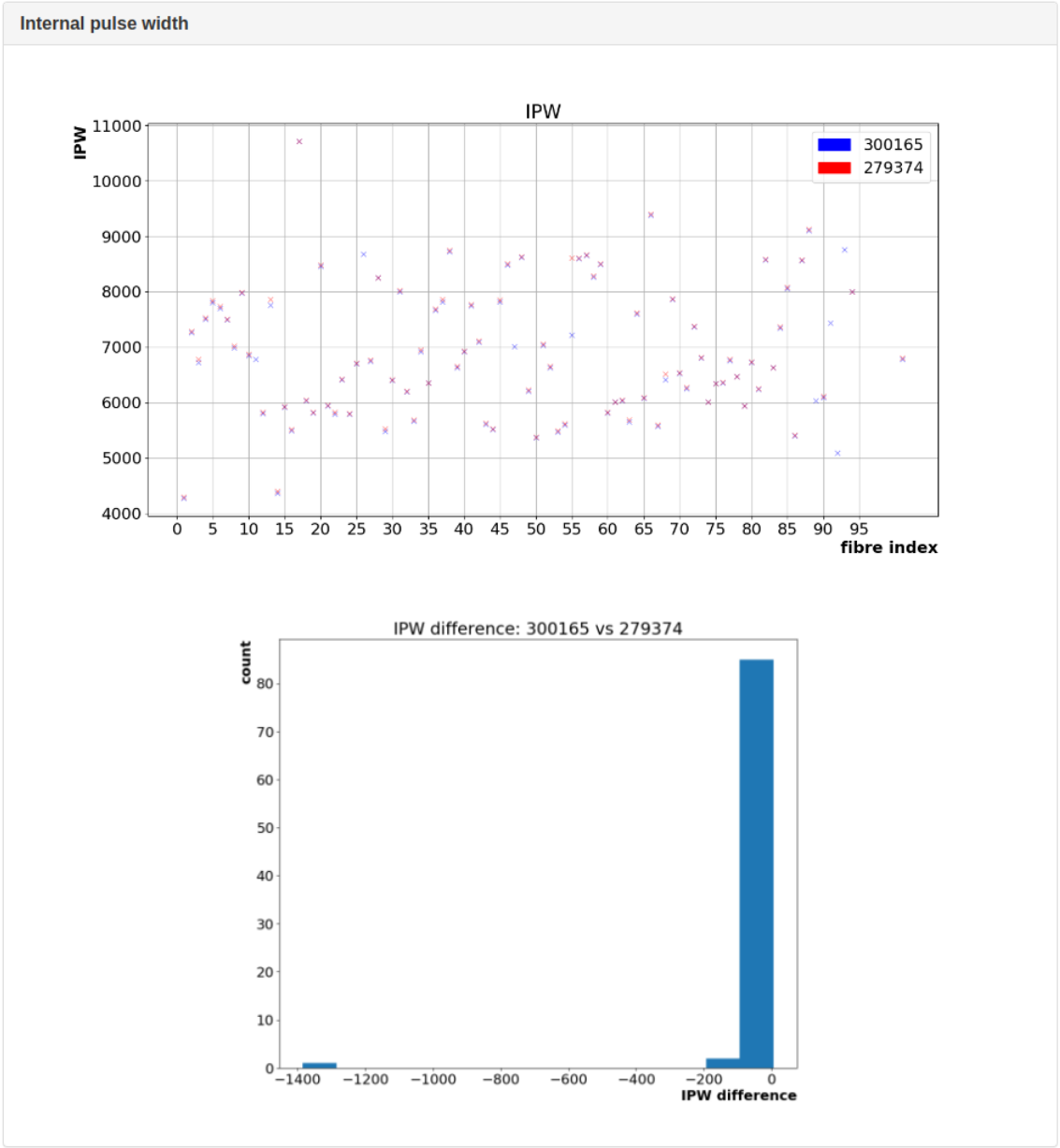


Figure 34: `Minard`: PCA table for one dataset.
This serves as a comparison of a single dataset's PCA table to the closest previous dataset's PCA table. These plots: IPW, IPW difference.

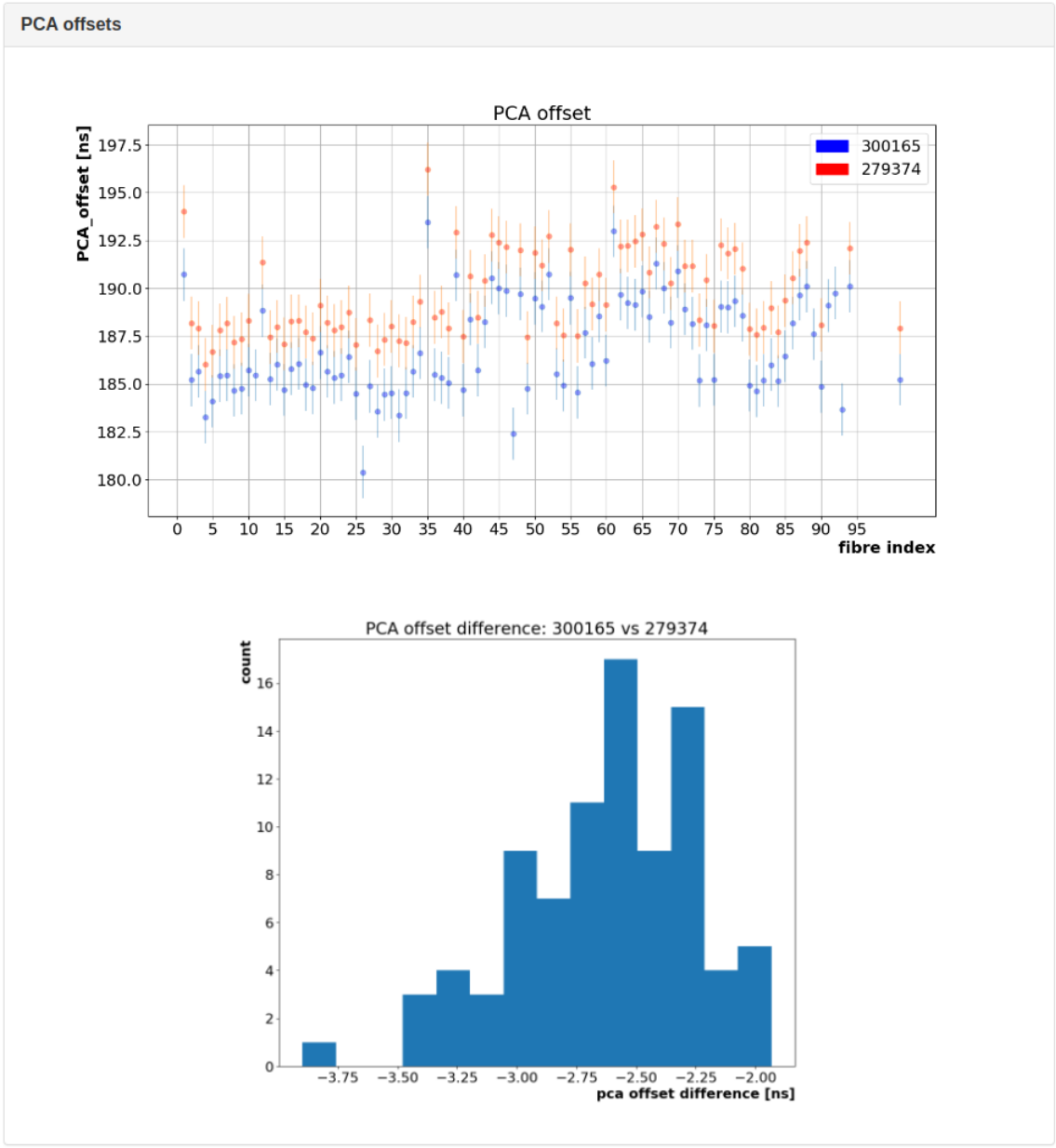


Figure 35: `Minard`: PCA table for one dataset.
This serves as a comparison of a single dataset's PCA table to the closest previous dataset's PCA table. These plots: PCA offset (also called injection time), PCA offset difference.

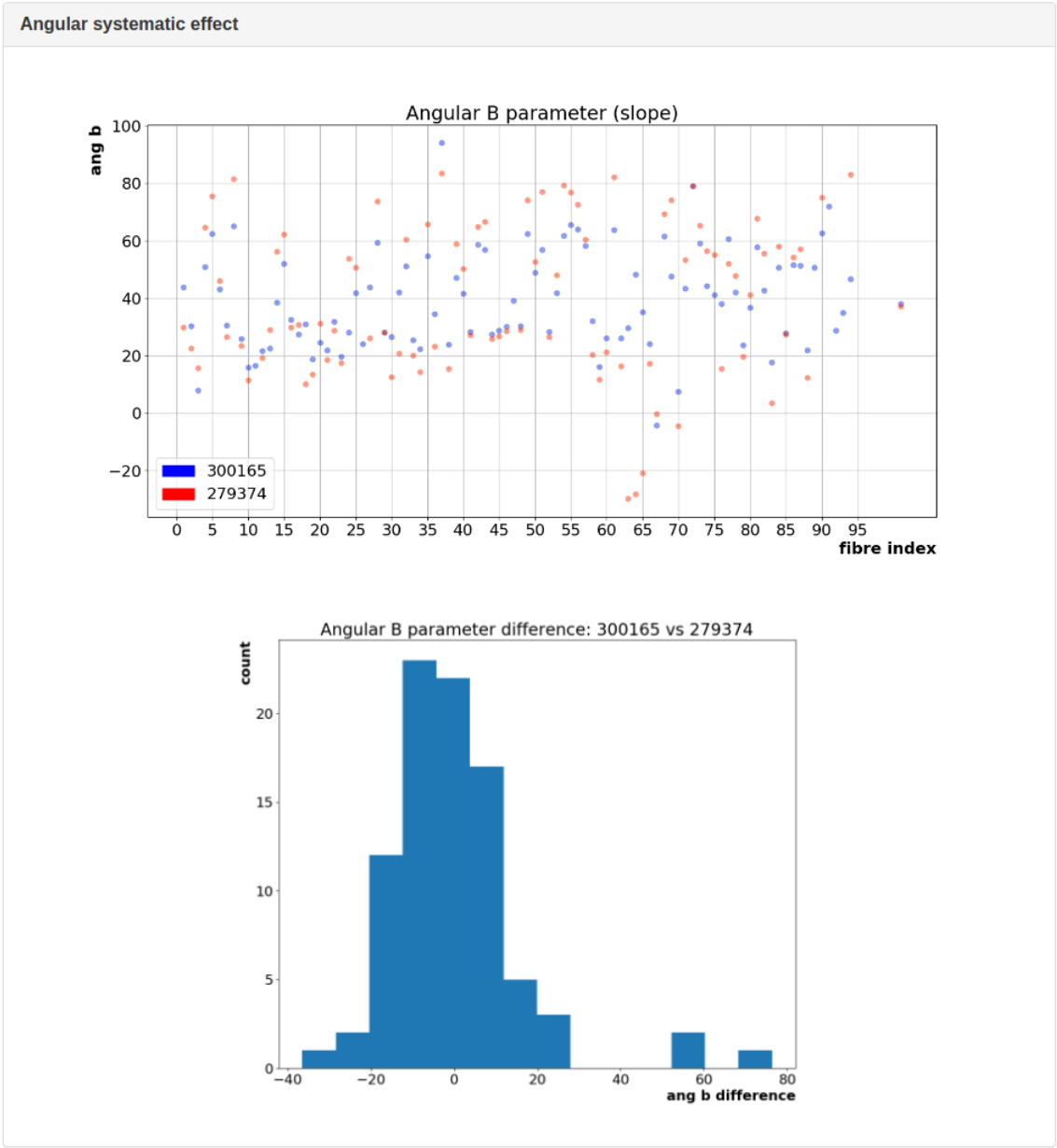


Figure 36: `Minard`: PCA table for one dataset.
This serves as a comparison of a single dataset's PCA table to the closest previous dataset's PCA table. These plots: Angular b parameter, angular b parameter difference.

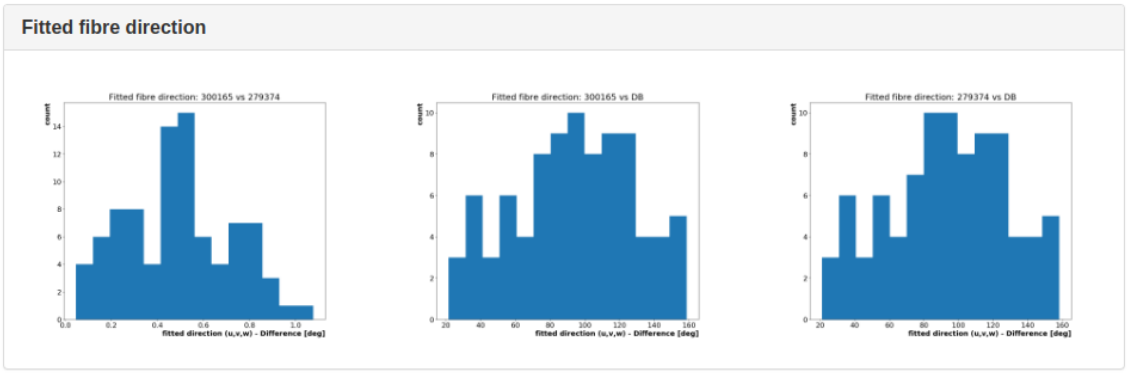


Figure 37: `Minard`: PCA table for one dataset.
This serves as a comparison of a single dataset's PCA table to the closest previous dataset's PCA table. These plots: fitted direction.

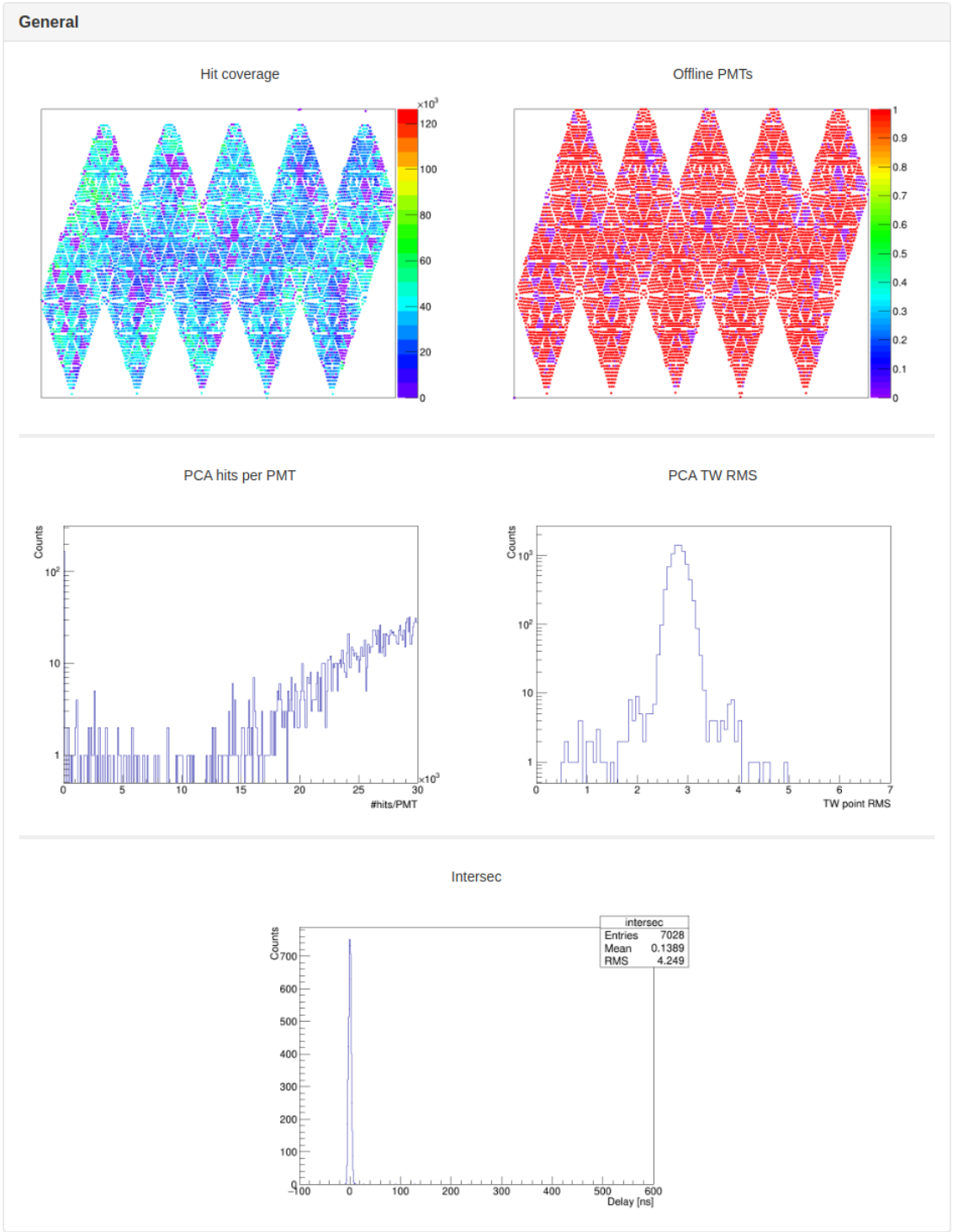


Figure 38: `PCA Processor`: Hit coverage (hits per PMT), Offline PMTs (purple means offline), PCA hits per PMT (hits passing cuts), PCA TW RMS, PCA intersec.

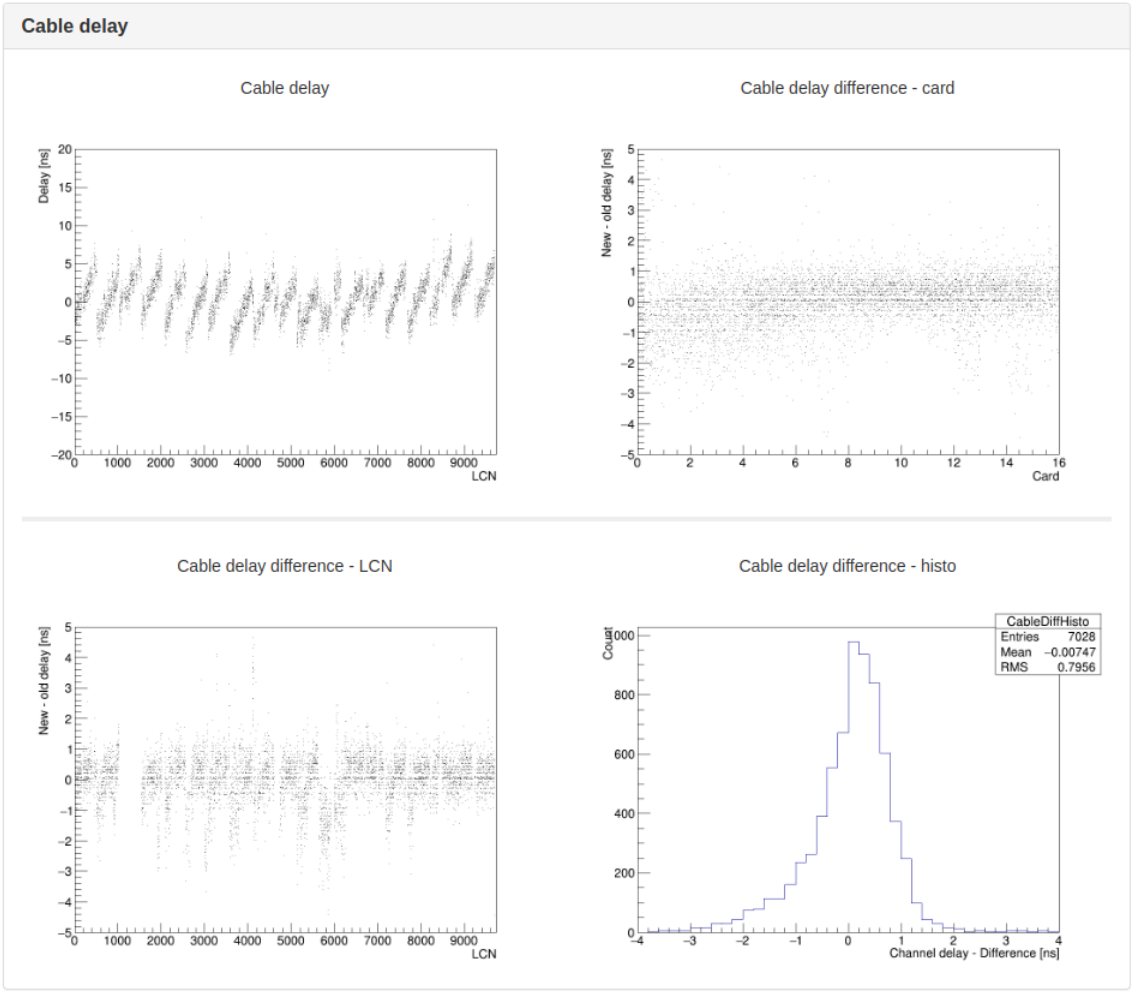


Figure 39: `PCA Processor`: Cable delays.
Cable delay per LCN (logical channel number), Cable delays across cards, the difference of this and previous cable delay per LCN, and the difference of this and previous cable delay - histogram.

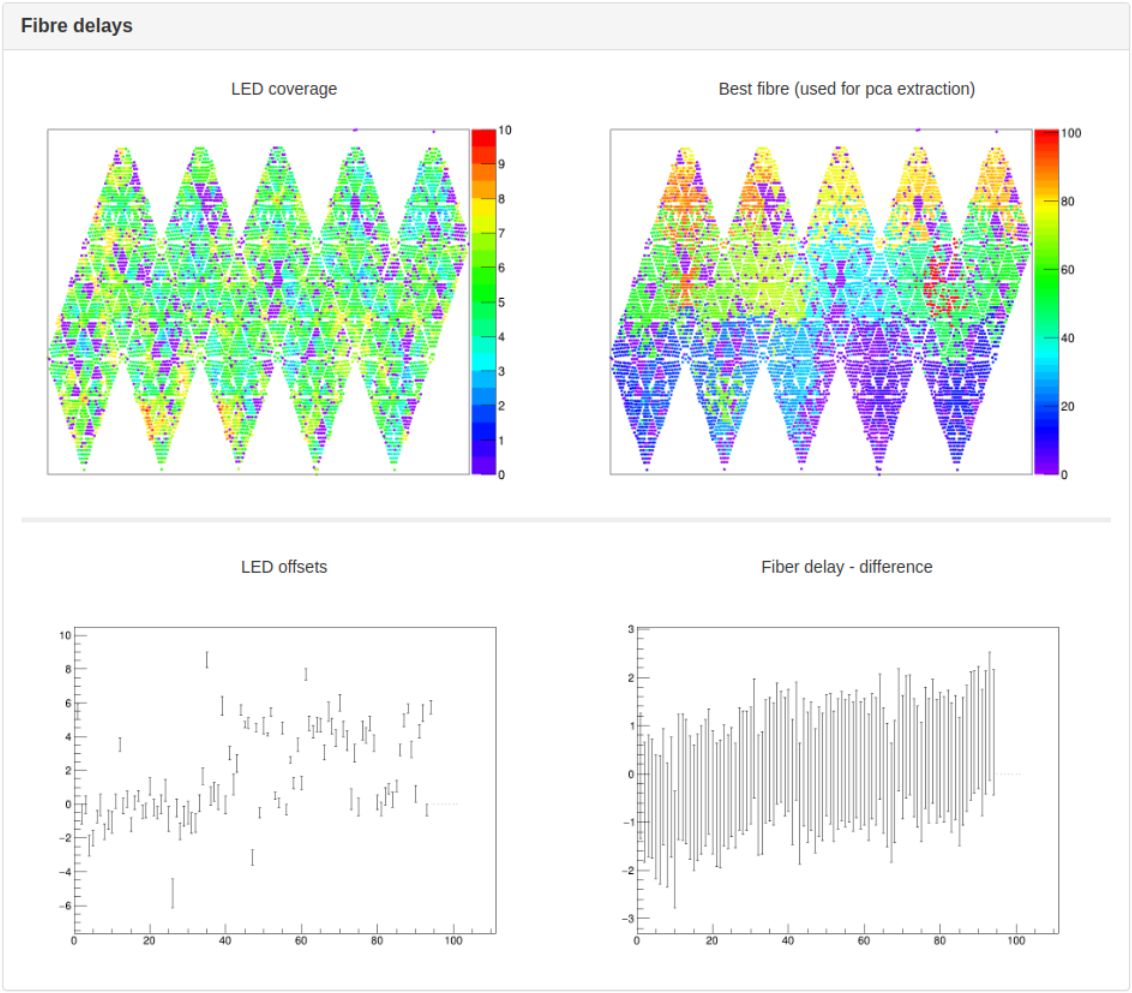


Figure 40: `PCA Processor`: LED data.
LED coverage (# LEDs hitting a PMT), Best fibre (the fibre with highest occupancy (within limits) used for calibration), LED offsets (respective time offset between LEDs), fibre delay (the difference of current and previous LED offset).

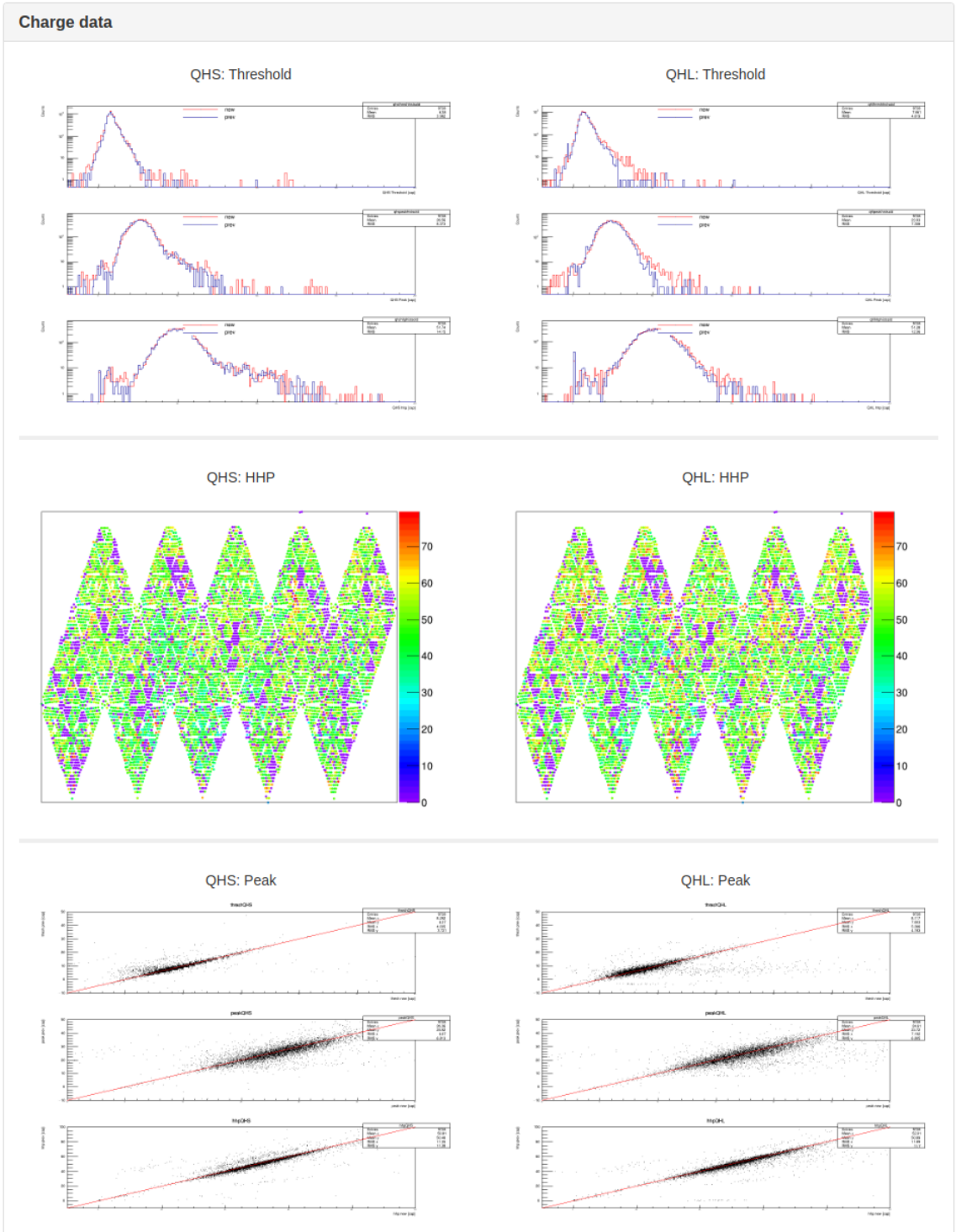


Figure 41: `PCA Processor`: Charge data.
The value of threshold, HHP (high half point), and peak for both QHS (left) and QHL (right).

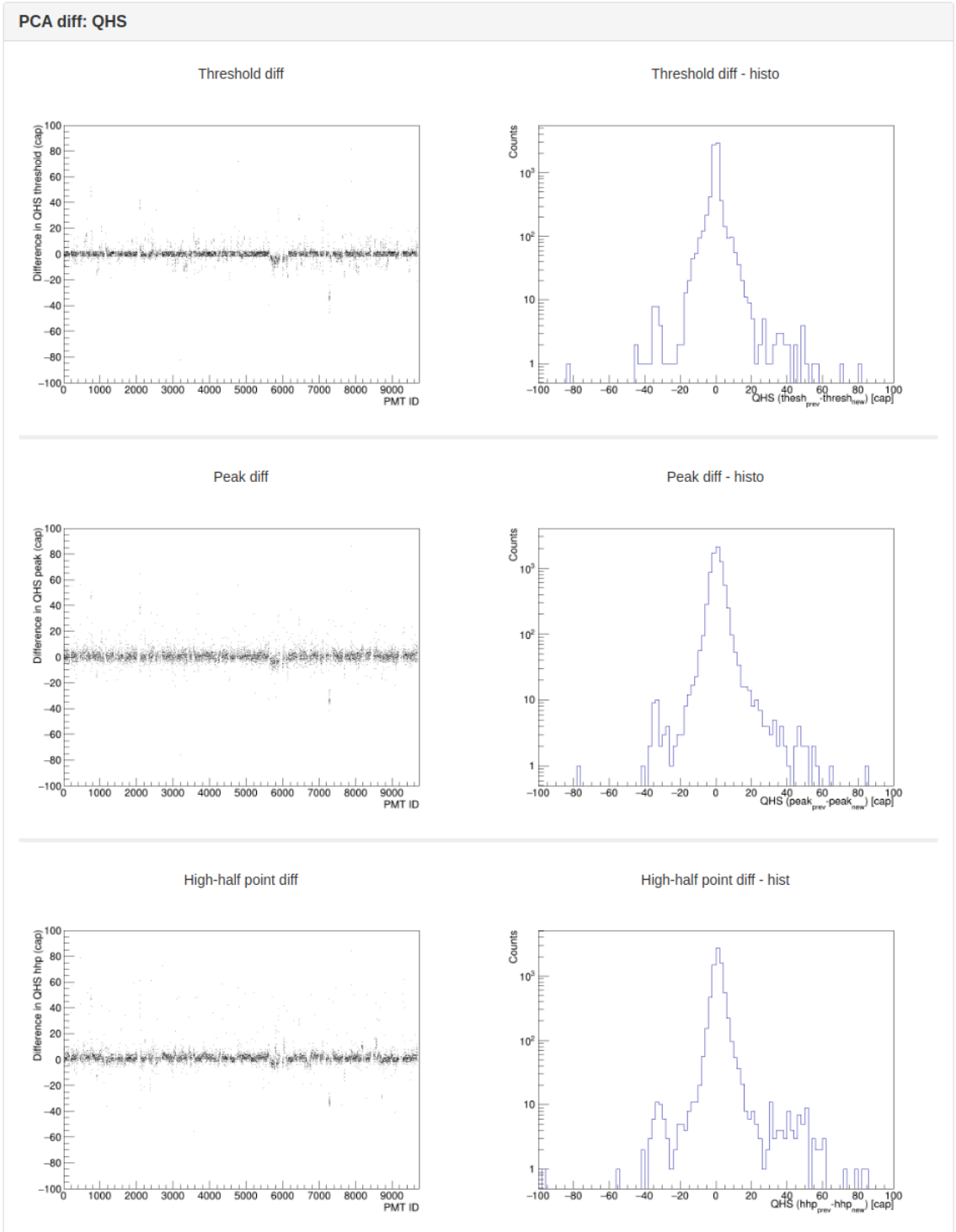


Figure 42: `PCA Processor`: QHS charge data.
This time showing the previously mentioned charge values as a difference of newly measured and latest value.

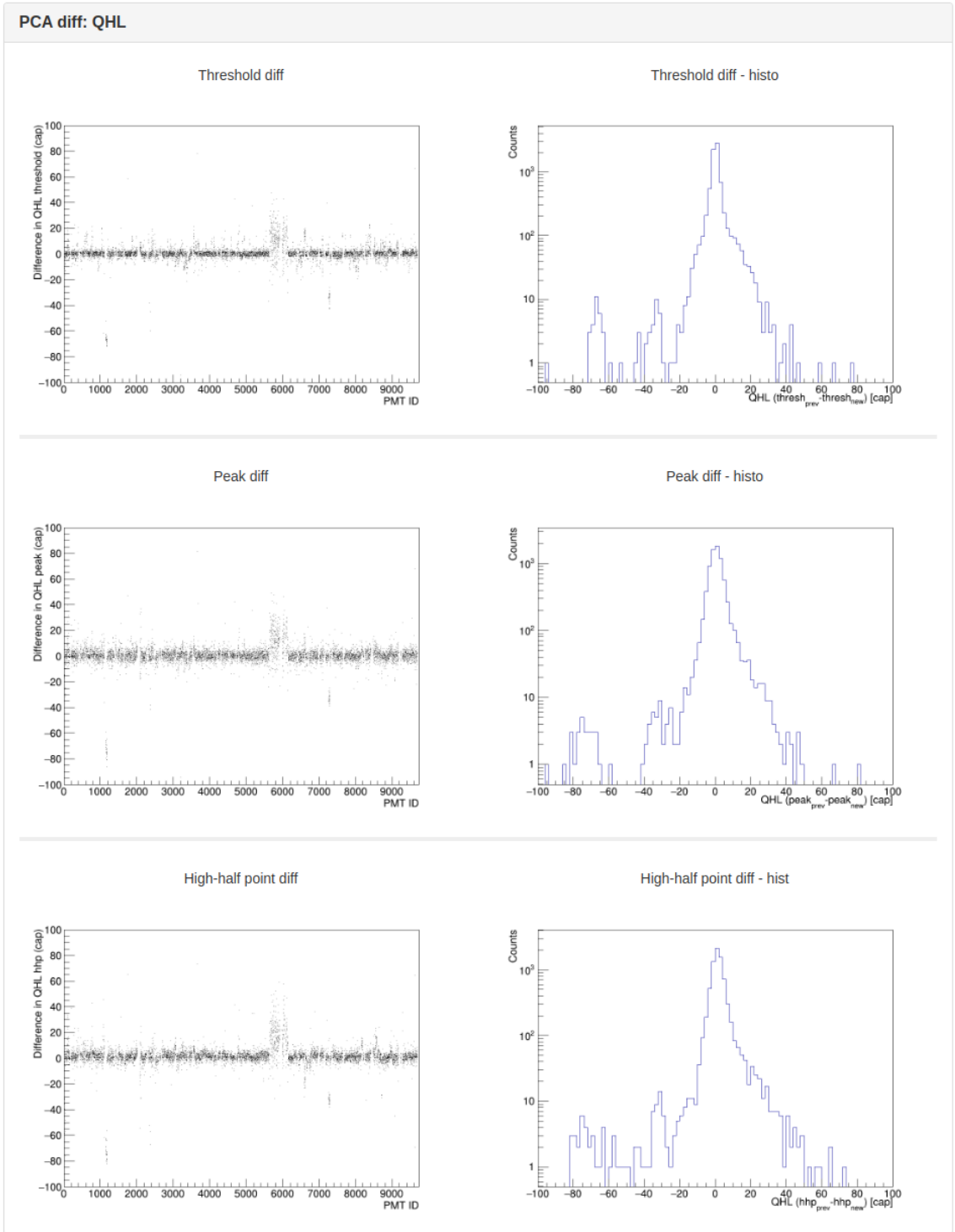
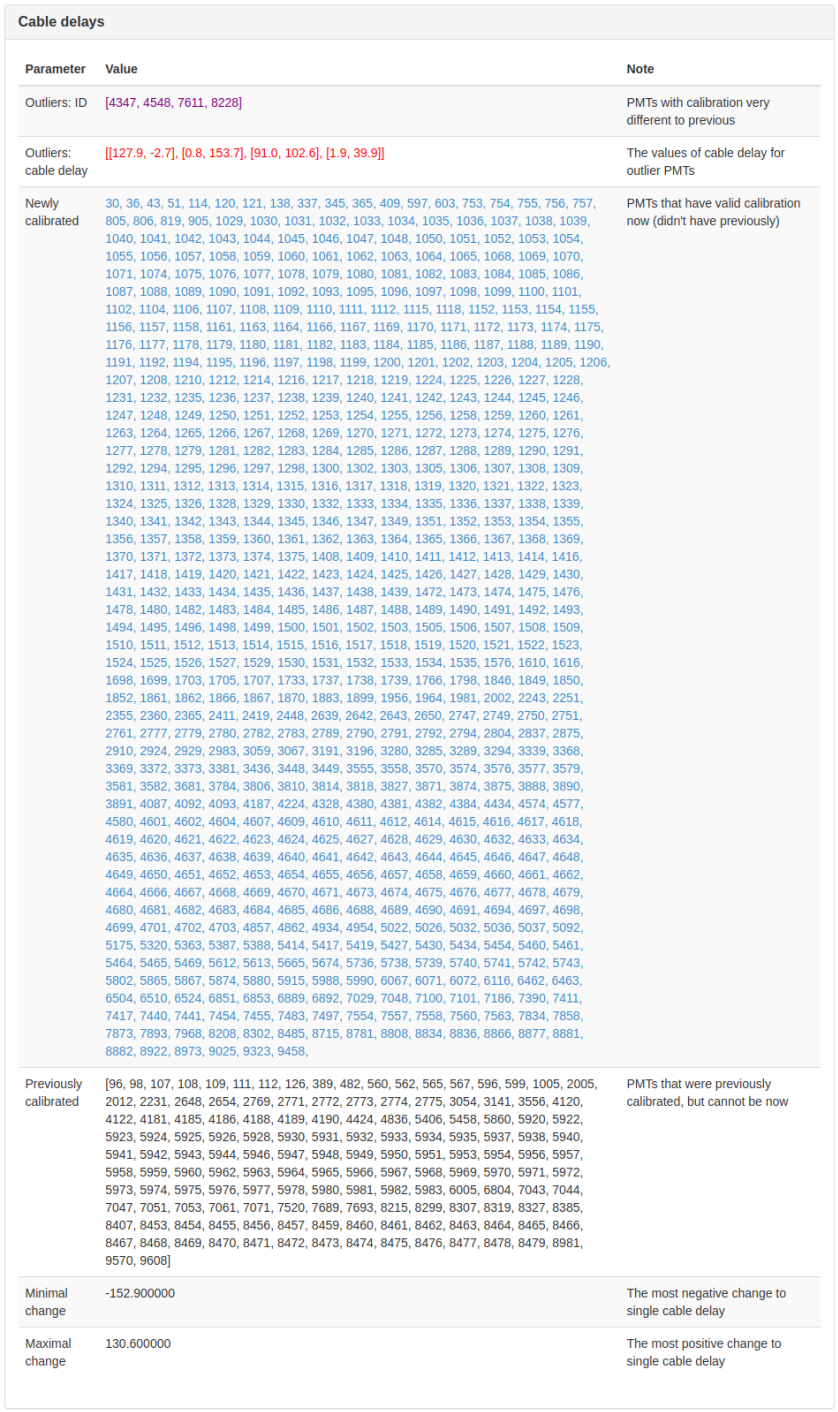


Figure 43: `PCA Processor`: QHL charge data #2.
This time showing the previously mentioned charge values as a difference of newly measured and latest value.

General

Parameter	Value	Note
Run1	300166	First run of the new PCA dataset
Run2	279375	First run of the previous PCA dataset
Offline PMTs	350	Number of offline PMTs (new dataset)
PMTs with ZERO occupancy	203	Number of PMTs reporting 0 valid hits
PMTs with LOW occupancy	175	Number of PMTs reporting low number of valid hits
Succesfully calibrated PMTs	7644	Number of PMTs with valid calibration

Figure 44: `Benchmarking`: General section.
This summarizes basic benchmarking data such as: run numbers (starting number of the set), # of offline PMTs for each set, PMTs with zero and low occupancy, and successfully calibrated PMTs.



Cable delays

Parameter	Value	Note
Outliers: ID	[4347, 4548, 7611, 8228]	PMTs with calibration very different to previous
Outliers: cable delay	[[127.9, -2.7], [0.8, 153.7], [91.0, 102.6], [1.9, 39.9]]	The values of cable delay for outlier PMTs
Newly calibrated	30, 36, 43, 51, 114, 120, 121, 138, 337, 345, 365, 409, 597, 603, 753, 754, 755, 756, 757, 805, 806, 819, 905, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1068, 1069, 1070, 1071, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1102, 1104, 1106, 1107, 1108, 1109, 1110, 1111, 1112, 1115, 1118, 1152, 1153, 1154, 1155, 1156, 1157, 1158, 1161, 1163, 1164, 1166, 1167, 1169, 1170, 1171, 1172, 1173, 1174, 1175, 1176, 1177, 1178, 1179, 1180, 1181, 1182, 1183, 1184, 1185, 1186, 1187, 1188, 1189, 1190, 1191, 1192, 1194, 1195, 1196, 1197, 1198, 1199, 1200, 1201, 1202, 1203, 1204, 1205, 1206, 1207, 1208, 1210, 1212, 1214, 1216, 1217, 1218, 1219, 1224, 1225, 1226, 1227, 1228, 1231, 1232, 1235, 1236, 1237, 1238, 1239, 1240, 1241, 1242, 1243, 1244, 1245, 1246, 1247, 1248, 1249, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1258, 1259, 1260, 1261, 1263, 1264, 1265, 1266, 1267, 1268, 1269, 1270, 1271, 1272, 1273, 1274, 1275, 1276, 1277, 1278, 1279, 1281, 1282, 1283, 1284, 1285, 1286, 1287, 1288, 1289, 1290, 1291, 1292, 1294, 1295, 1296, 1297, 1298, 1300, 1302, 1303, 1305, 1306, 1307, 1308, 1309, 1310, 1311, 1312, 1313, 1314, 1315, 1316, 1317, 1318, 1319, 1320, 1321, 1322, 1323, 1324, 1325, 1326, 1328, 1329, 1330, 1332, 1333, 1334, 1335, 1336, 1337, 1338, 1339, 1340, 1341, 1342, 1343, 1344, 1345, 1346, 1347, 1349, 1351, 1352, 1353, 1354, 1355, 1356, 1357, 1358, 1359, 1360, 1361, 1362, 1363, 1364, 1365, 1366, 1367, 1368, 1369, 1370, 1371, 1372, 1373, 1374, 1375, 1408, 1409, 1410, 1411, 1412, 1413, 1414, 1416, 1417, 1418, 1419, 1420, 1421, 1422, 1423, 1424, 1425, 1426, 1427, 1428, 1429, 1430, 1431, 1432, 1433, 1434, 1435, 1436, 1437, 1438, 1439, 1472, 1473, 1474, 1475, 1476, 1478, 1480, 1482, 1483, 1484, 1485, 1486, 1487, 1488, 1489, 1490, 1491, 1492, 1493, 1494, 1495, 1496, 1498, 1499, 1500, 1501, 1502, 1503, 1505, 1506, 1507, 1508, 1509, 1510, 1511, 1512, 1513, 1514, 1515, 1516, 1517, 1518, 1519, 1520, 1521, 1522, 1523, 1524, 1525, 1526, 1527, 1529, 1530, 1531, 1532, 1533, 1534, 1535, 1576, 1610, 1616, 1698, 1699, 1703, 1705, 1707, 1733, 1737, 1738, 1739, 1766, 1798, 1846, 1849, 1850, 1852, 1861, 1862, 1866, 1867, 1870, 1883, 1899, 1956, 1964, 1981, 2002, 2243, 2251, 2355, 2360, 2365, 2411, 2419, 2448, 2639, 2642, 2643, 2650, 2747, 2749, 2750, 2751, 2761, 2777, 2779, 2780, 2782, 2783, 2789, 2790, 2791, 2792, 2794, 2804, 2837, 2875, 2910, 2924, 2929, 2983, 3059, 3067, 3191, 3196, 3280, 3285, 3289, 3294, 3339, 3368, 3369, 3372, 3373, 3381, 3436, 3448, 3449, 3555, 3558, 3570, 3574, 3576, 3577, 3579, 3581, 3582, 3681, 3784, 3806, 3810, 3814, 3818, 3827, 3871, 3874, 3875, 3888, 3890, 3891, 4087, 4092, 4093, 4187, 4224, 4328, 4380, 4381, 4382, 4384, 4434, 4574, 4577, 4580, 4601, 4602, 4604, 4607, 4609, 4610, 4611, 4612, 4614, 4615, 4616, 4617, 4618, 4619, 4620, 4621, 4622, 4623, 4624, 4625, 4627, 4628, 4629, 4630, 4632, 4633, 4634, 4635, 4636, 4637, 4638, 4639, 4640, 4641, 4642, 4643, 4644, 4645, 4646, 4647, 4648, 4649, 4650, 4651, 4652, 4653, 4654, 4655, 4656, 4657, 4658, 4659, 4660, 4661, 4662, 4664, 4666, 4667, 4668, 4669, 4670, 4671, 4673, 4674, 4675, 4676, 4677, 4678, 4679, 4680, 4681, 4682, 4683, 4684, 4685, 4686, 4688, 4689, 4690, 4691, 4694, 4697, 4698, 4699, 4701, 4702, 4703, 4857, 4862, 4934, 4954, 5022, 5026, 5032, 5036, 5037, 5092, 5175, 5320, 5363, 5387, 5388, 5414, 5417, 5419, 5427, 5430, 5434, 5454, 5460, 5461, 5464, 5465, 5469, 5612, 5613, 5665, 5674, 5736, 5738, 5739, 5740, 5741, 5742, 5743, 5802, 5865, 5867, 5874, 5880, 5915, 5988, 5990, 6067, 6071, 6072, 6116, 6462, 6463, 6504, 6510, 6524, 6851, 6853, 6889, 6892, 7029, 7048, 7100, 7101, 7186, 7390, 7411, 7417, 7440, 7441, 7454, 7455, 7483, 7497, 7554, 7557, 7558, 7560, 7563, 7834, 7858, 7873, 7893, 7968, 8208, 8302, 8485, 8715, 8781, 8808, 8834, 8836, 8866, 8877, 8881, 8882, 8922, 8973, 9025, 9323, 9458,	PMTs that have valid calibration now (didn't have previously)
Previously calibrated	[96, 98, 107, 108, 109, 111, 112, 126, 389, 482, 560, 562, 565, 567, 596, 599, 1005, 2005, 2012, 2231, 2648, 2654, 2769, 2771, 2772, 2773, 2774, 2775, 3054, 3141, 3556, 4120, 4122, 4181, 4185, 4186, 4188, 4189, 4190, 4424, 4836, 5406, 5458, 5860, 5920, 5922, 5923, 5924, 5925, 5926, 5928, 5930, 5931, 5932, 5933, 5934, 5935, 5937, 5938, 5940, 5941, 5942, 5943, 5944, 5946, 5947, 5948, 5949, 5950, 5951, 5953, 5954, 5956, 5957, 5958, 5959, 5960, 5962, 5963, 5964, 5965, 5966, 5967, 5968, 5969, 5970, 5971, 5972, 5973, 5974, 5975, 5976, 5977, 5978, 5980, 5981, 5982, 5983, 6005, 6804, 7043, 7044, 7047, 7051, 7053, 7061, 7071, 7520, 7689, 7693, 8215, 8299, 8307, 8319, 8327, 8385, 8407, 8453, 8454, 8455, 8456, 8457, 8459, 8460, 8461, 8462, 8463, 8464, 8465, 8466, 8467, 8468, 8469, 8470, 8471, 8472, 8473, 8474, 8475, 8476, 8477, 8478, 8479, 8981, 9570, 9608]	PMTs that were previously calibrated, but cannot be now
Minimal change	-152.900000	The most negative change to single cable delay
Maximal change	130.600000	The most positive change to single cable delay

Figure 45: `Benchmarking`: Cable delays data.
This section shows cable delays data for PMTs. Outliers are PMTs with new cable delay value very different to previous. PMTs that are now calibrated that were not before are shown (can be clicked, showing the actual PMT data), as well as PMTs that were previously calibrated but couldn't be in this set.

Time Walk

Parameter	Value	Note
IDs1	7644	Number of valid PMTs for new dataset
IDs2	7051	Number of valid PMTs for previous dataset
off1	350	Number of offline PMTs for new dataset
off2	493	Number of offline PMTs for previous dataset
goodCount	6956	PMTs that were succesfully calibrated in both datasets
badCount	688	PMTs that were only calibrated in one dataset
min_grad	-0.015700	Minimum value of gradient (fit)
max_grad	0.018700	Maximum value of gradient (fit)
min_inter	-152.900000	Minimum value of intercept (fit)
max_inter	130.600000	Maximum value of intercept (fit)

Figure 46: `Benchmarking`: Time-walk data.
This is hopefully self-explanatory (see notes in the table).

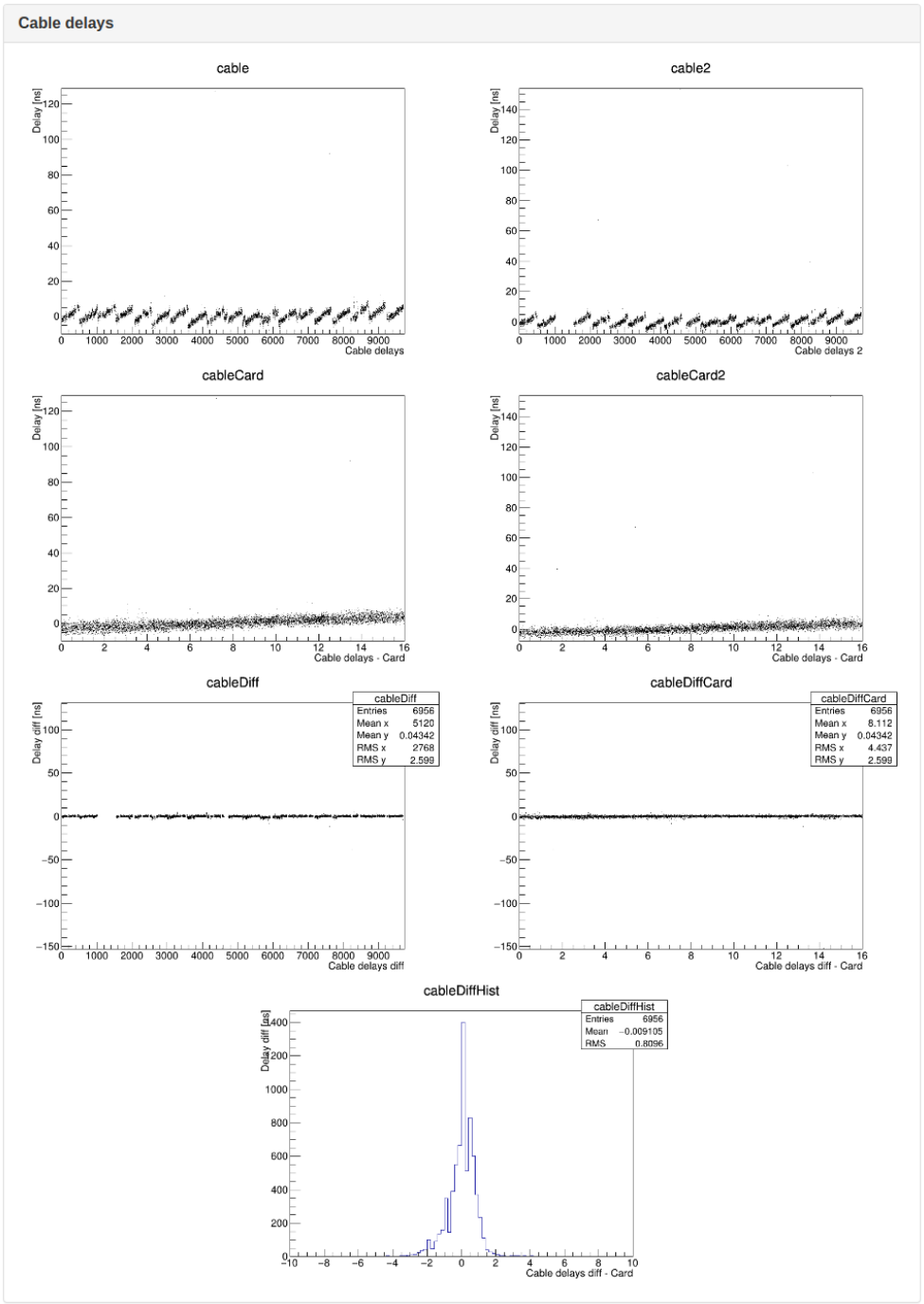


Figure 47: `Benchmarking`: Plots for cable delays.
Showing the cable delays for this (left) and previous (right) calibration.
Difference at the bottom.

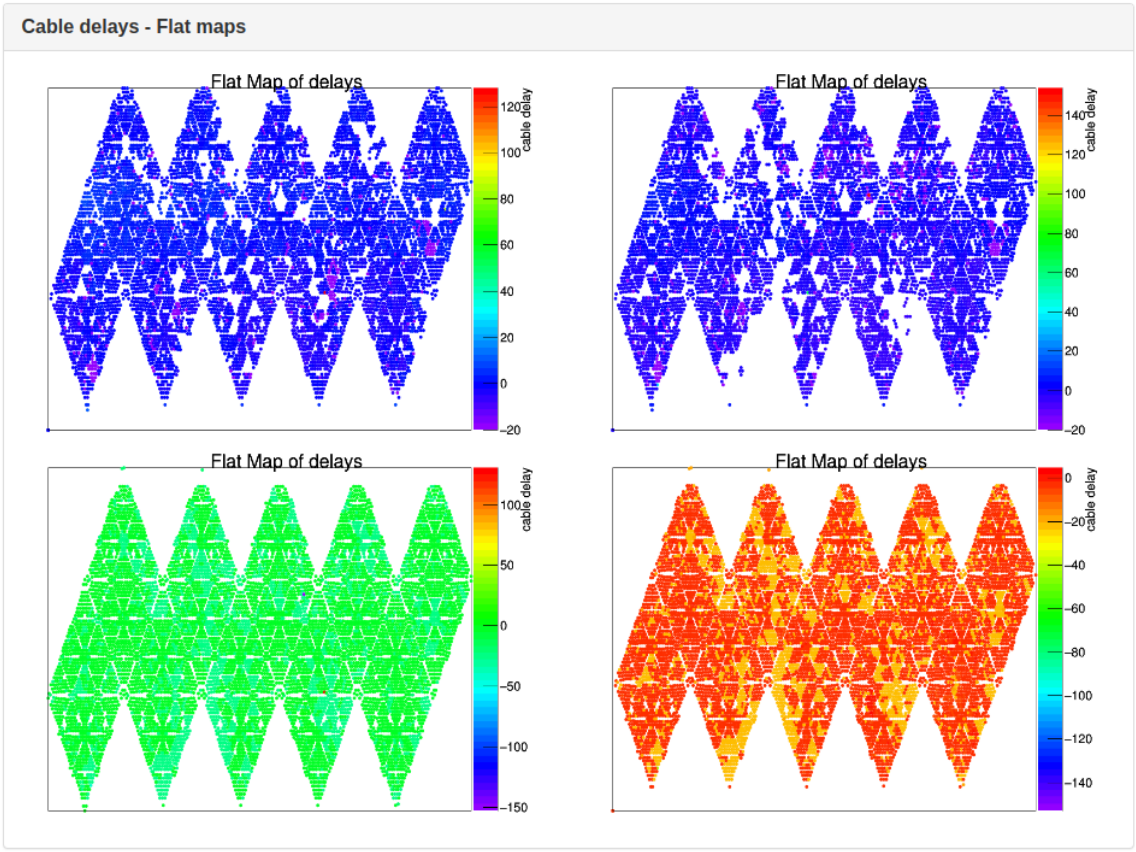


Figure 48: `Benchmarking`: Flat map plots.
Top: flat maps showing the cable delays for this (left) and previous (right) set. Bottom: the difference of the cable delays between this and previous set (left); zoomed view of left plot (right).

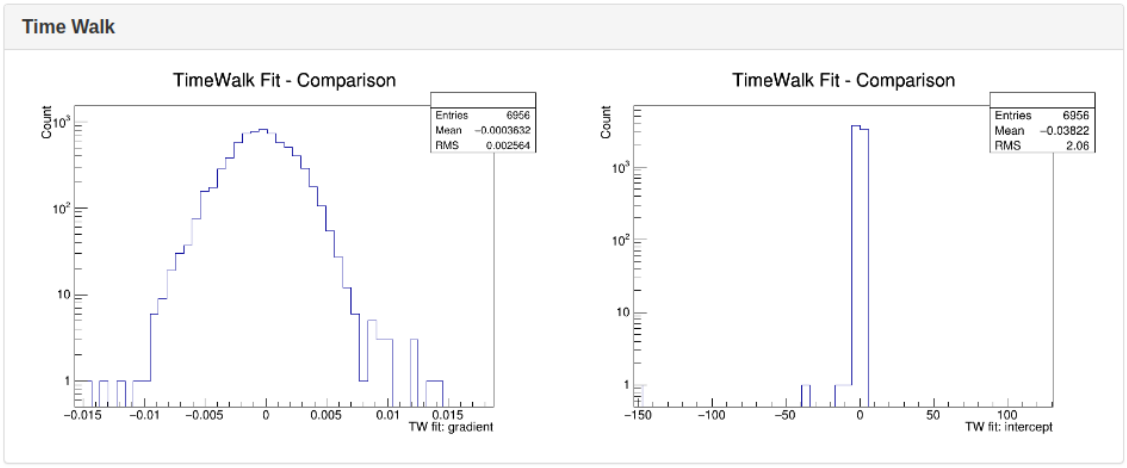


Figure 49: `Benchmarking`: Time-walk plots.
The difference between this and previous set: TW gradient (left) and TW intercept (right).

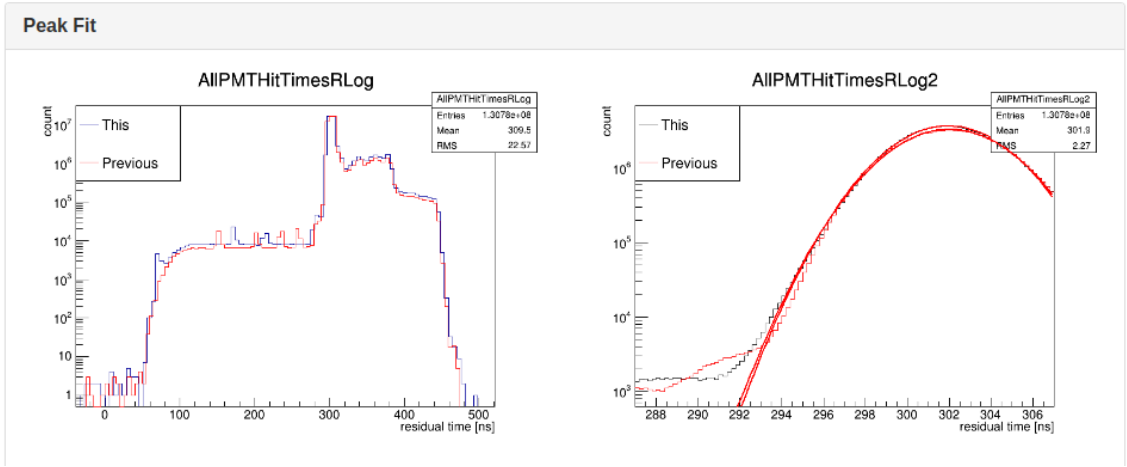
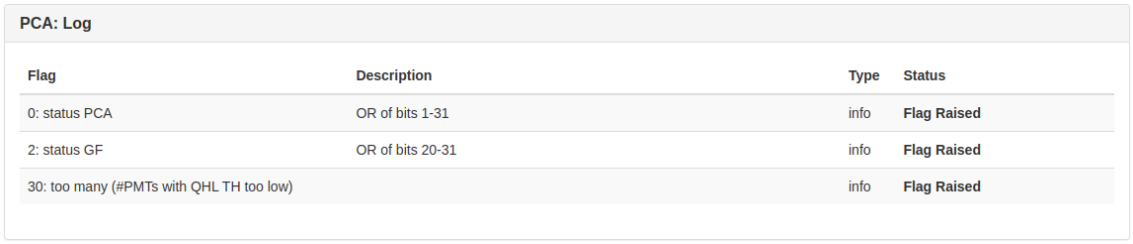


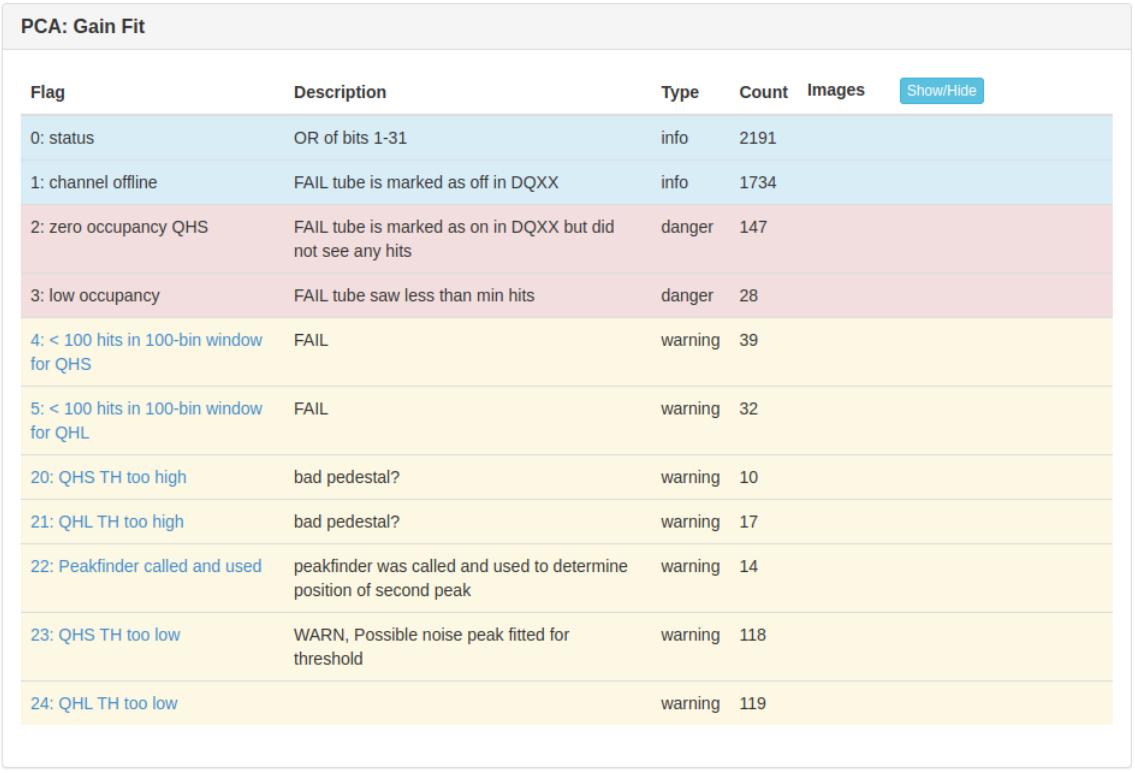
Figure 50: `Benchmarking`: Peak fit plots.
These plots shown the residual hit times from a test laserball run, with the newly obtained, and old (previous set) cable delays applied to the PMT timings, respectively. Due to the difference of the actual number of PMTs that were calibrated, there will be difference. What is important is to look at the peak of this distribution (right plot).



PCA: Log

Flag	Description	Type	Status
0: status PCA	OR of bits 1-31	info	**Flag Raised**
2: status GF	OR of bits 20-31	info	**Flag Raised**
30: too many (#PMTs with QHL TH too low)		info	**Flag Raised**

Figure 51: `PCA logs`: Status.
This is data parsed from the PCA processor main log file. It lists bits that failed.



PCA: Gain Fit

Flag	Description	Type	Count	Images	Show/Hide
0: status	OR of bits 1-31	info	2191		
1: channel offline	FAIL tube is marked as off in DQXX	info	1734		
2: zero occupancy QHS	FAIL tube is marked as on in DQXX but did not see any hits	danger	147		
3: low occupancy	FAIL tube saw less than min hits	danger	28		
4: < 100 hits in 100-bin window for QHS	FAIL	warning	39		
5: < 100 hits in 100-bin window for QHL	FAIL	warning	32		
20: QHS TH too high	bad pedestal?	warning	10		
21: QHL TH too high	bad pedestal?	warning	17		
22: Peakfinder called and used	peakfinder was called and used to determine position of second peak	warning	14		
23: QHS TH too low	WARN, Possible noise peak fitted for threshold	warning	118		
24: QHL TH too low		warning	119		

Figure 52: `PCA logs`: Gain fit.
This is data parsed from the PCA processor GF log file. It lists PMTs that failed particular checks, by group. The groups can be expanded, showing crate maps highlighting the PMTs in question.

PCA: Time Walk

Flag	Description	Type	Count	Images	Show/Hide
0: status	OR of bits 1-31	info	2084		
1: channel offline	same as GF	info	1734		
2: zero occupancy	no hits on this PMT, same as GF	danger	147		
3: low occupancy	Less than min hits on this PMT, could be different than GF	danger	28		
8: RMS is too high		warning	9		
9: high Q tail was not fitted		warning	69		
10: Tstep warning	Possible bad ADC charge conversion. This will affect a complete card.	warning	38		
11: Flate warning	The fraction of late light is too large	warning	6		
12: Fout warning	The fraction of hits <QHSmin and < Tpeak-10 is too large	warning	27		
13: Gradient warning	The gradient of the fit to high charge tail is positive	warning	2		

Figure 53: `PCA logs`: Time-walk fit.
This is data parsed from the PCA processor TW log file. It lists PMTs that failed particular checks, by group. The groups can be expanded, showing crate maps highlighting the PMTs in question.

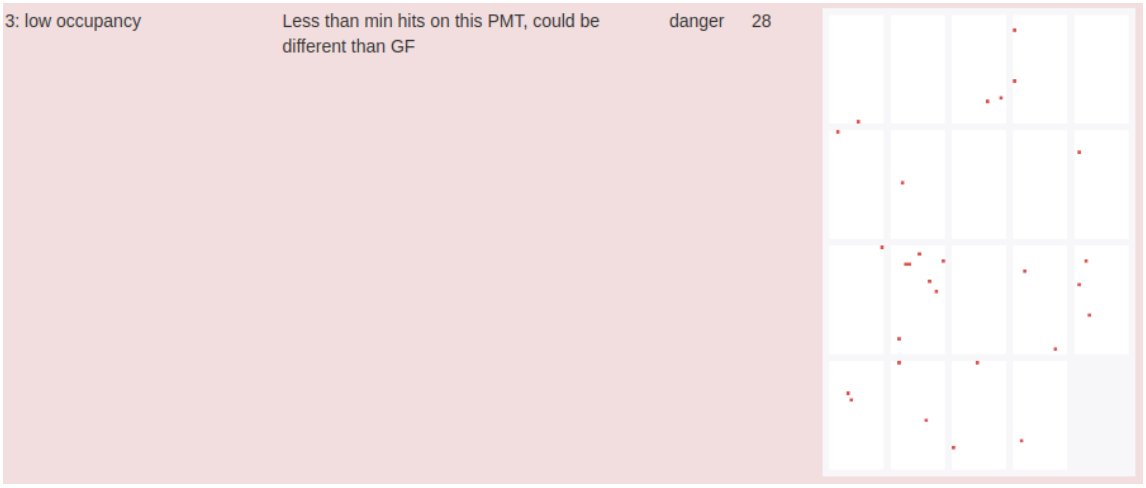


Figure 54: `PCA logs`: Example crate map.
This is an example crate map highlighting PMTs that failed the low occupancy check. Same map is available for each check where PMTs that failed exist. The maps are shown by clicking the 'Show/Hide' button.

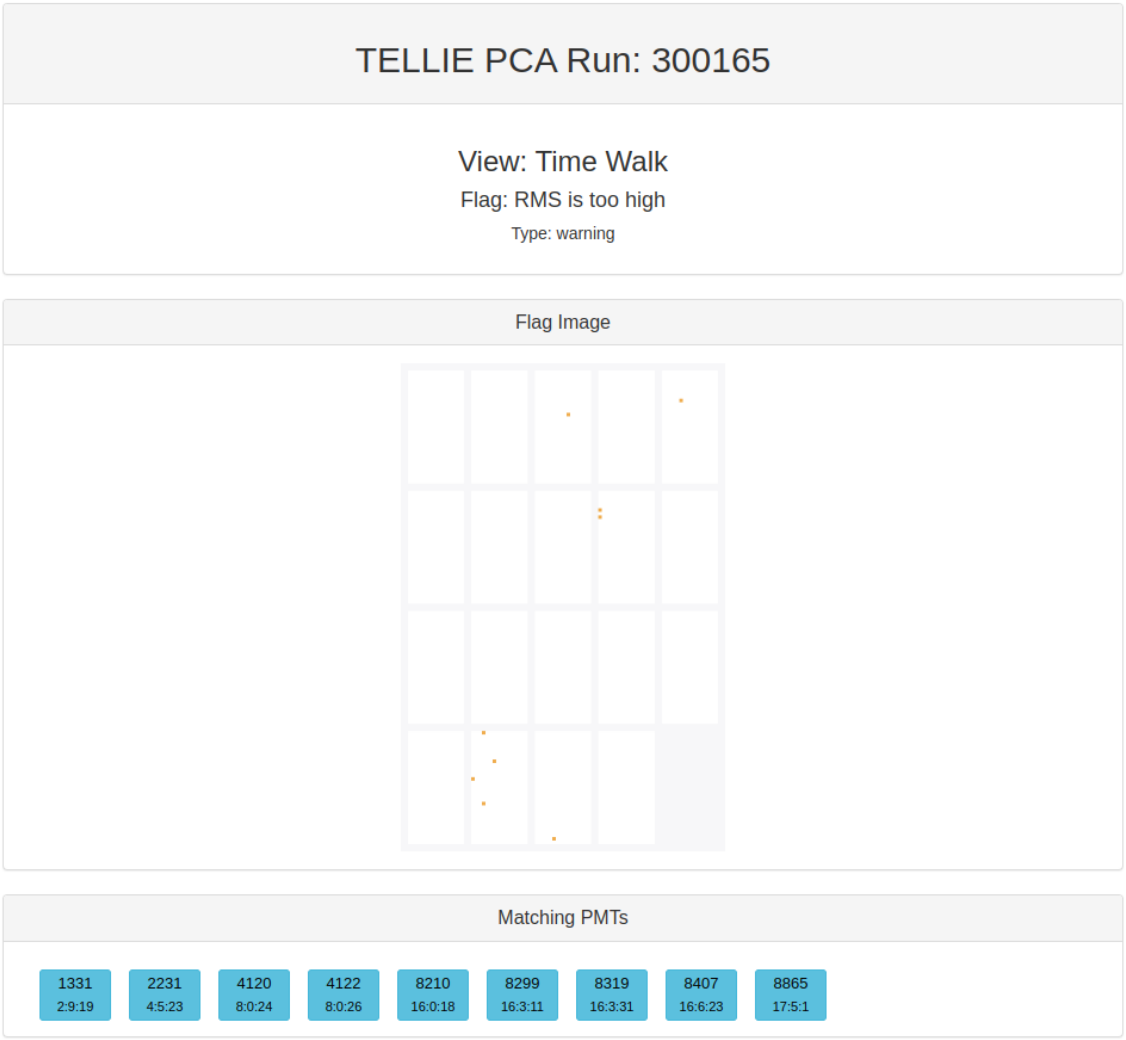


Figure 55: `PCA logs`: Example check detailed page.
After clicking any of the checks on the log pages, a detailed page for the check is shown. This shows the crate map and lists the offending PMTs. Each PMT can be clicked again for more detailed page for that PMT.

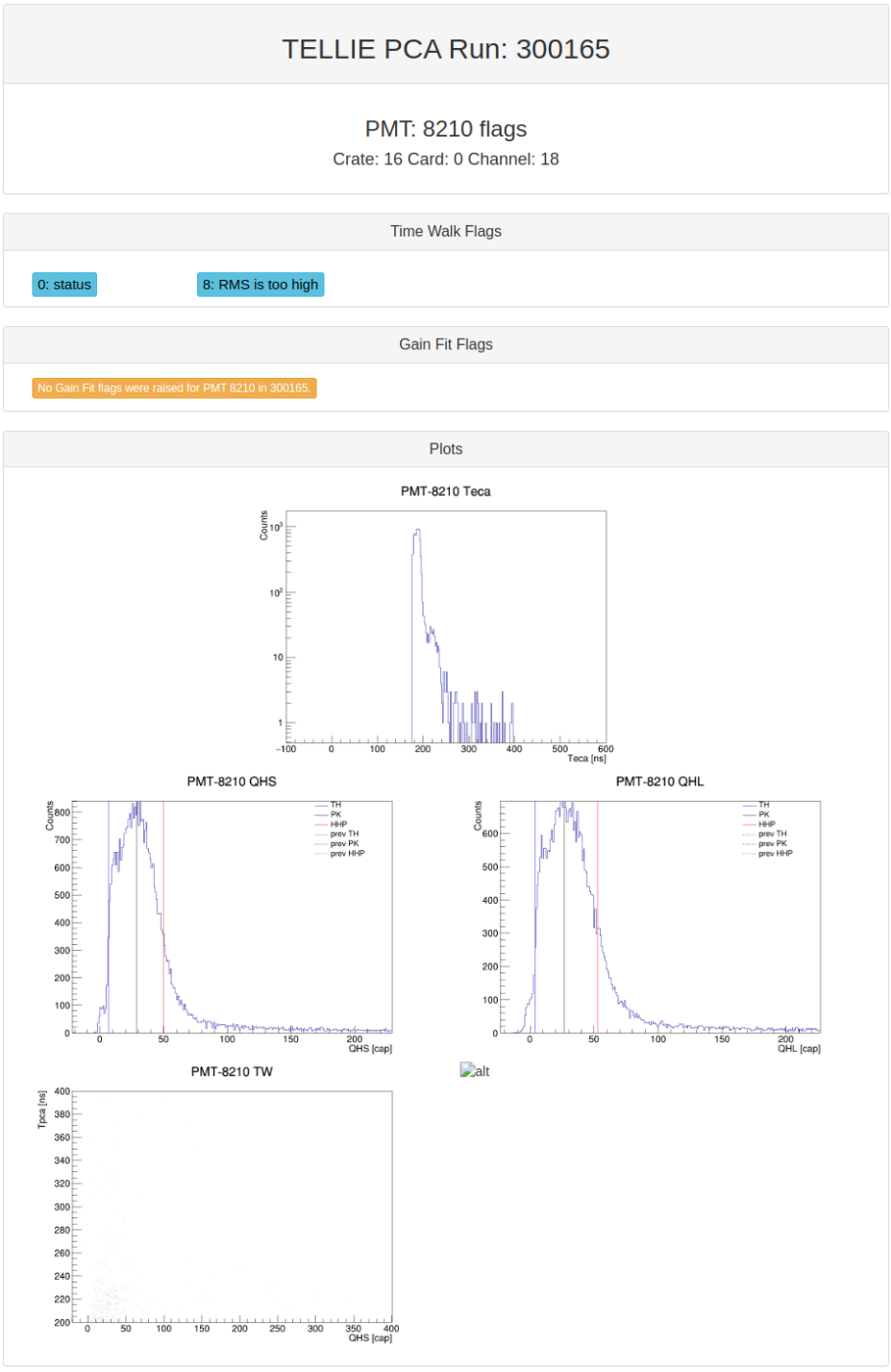


Figure 56: `PCA logs`: PMT page.
After a PMT is selected on one of the check pages, its status for that particular calibration is shown. Plots for time and charge fits are displayed alongside the status words for this PMT.

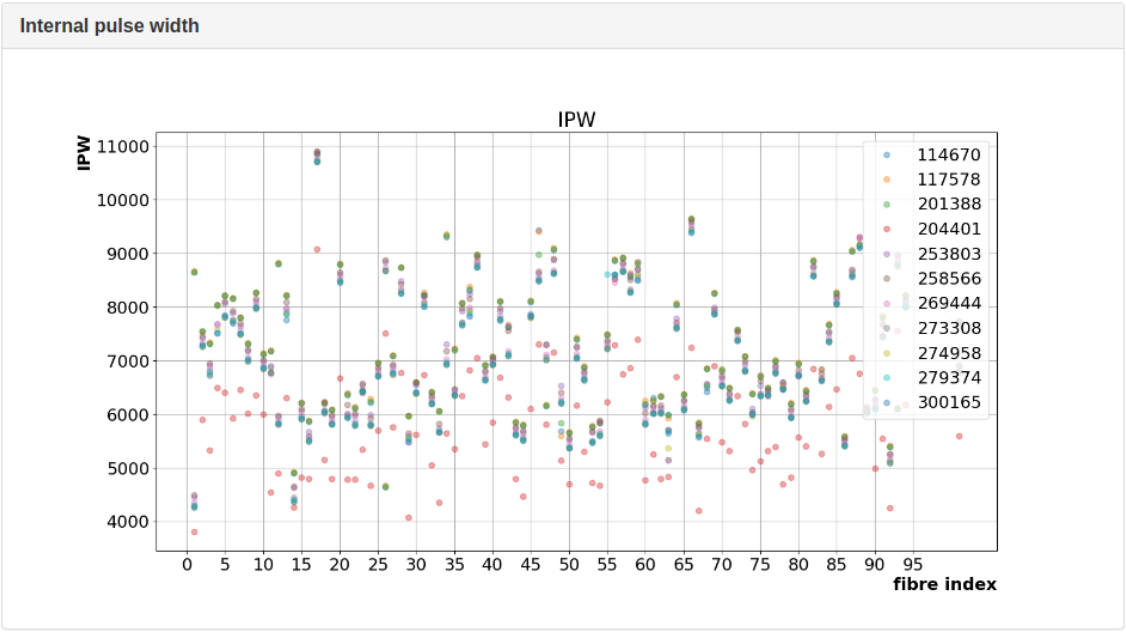


Figure 57: `PCA tables`: Comparing the PCA tables across all available datasets.
IPW (internal pulse width) parameter shown.

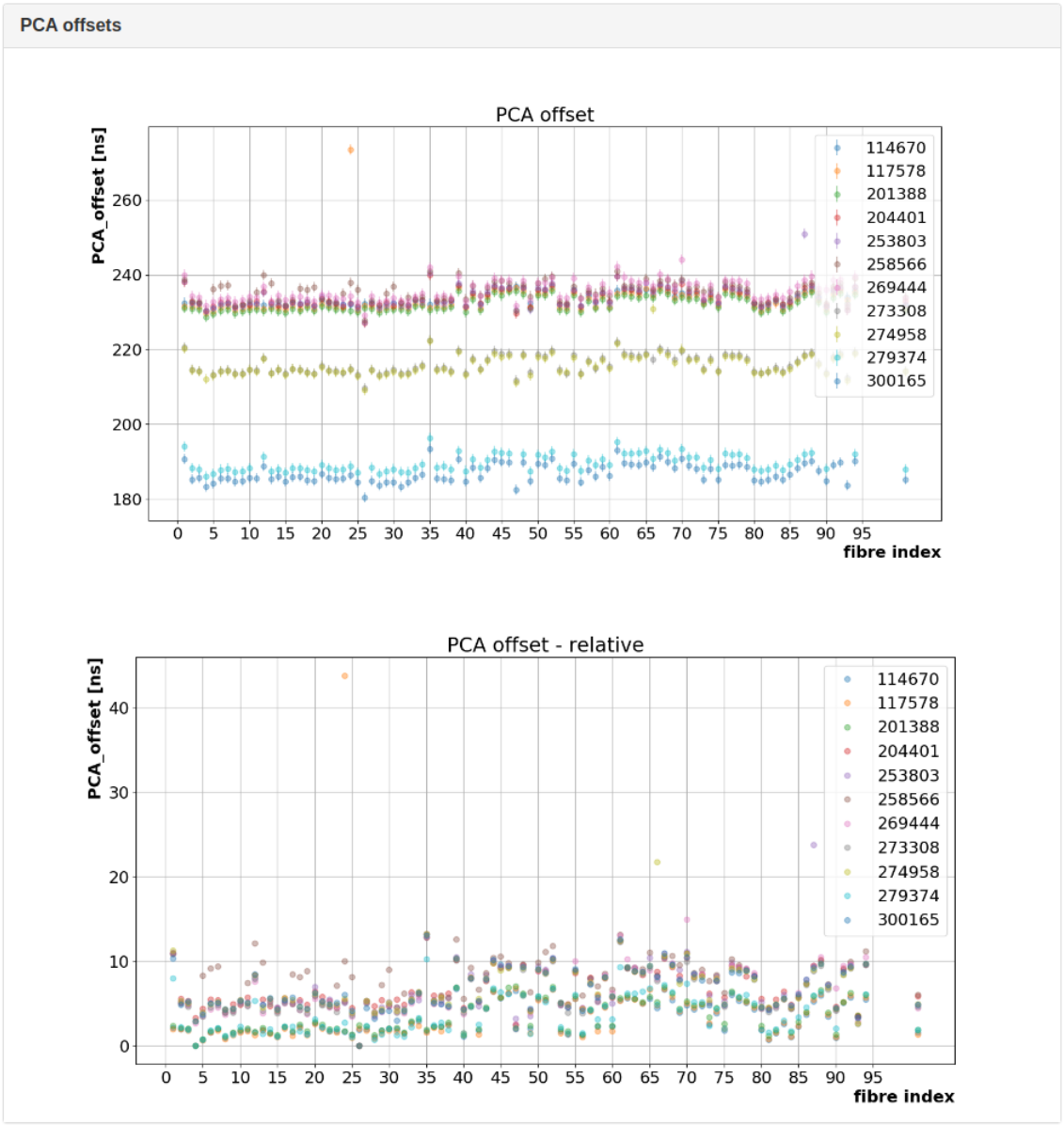


Figure 58: `PCA tables`: Comparing the PCA tables across all available datasets.
PCA offset (injection time) shown. Relative version on the bottom.

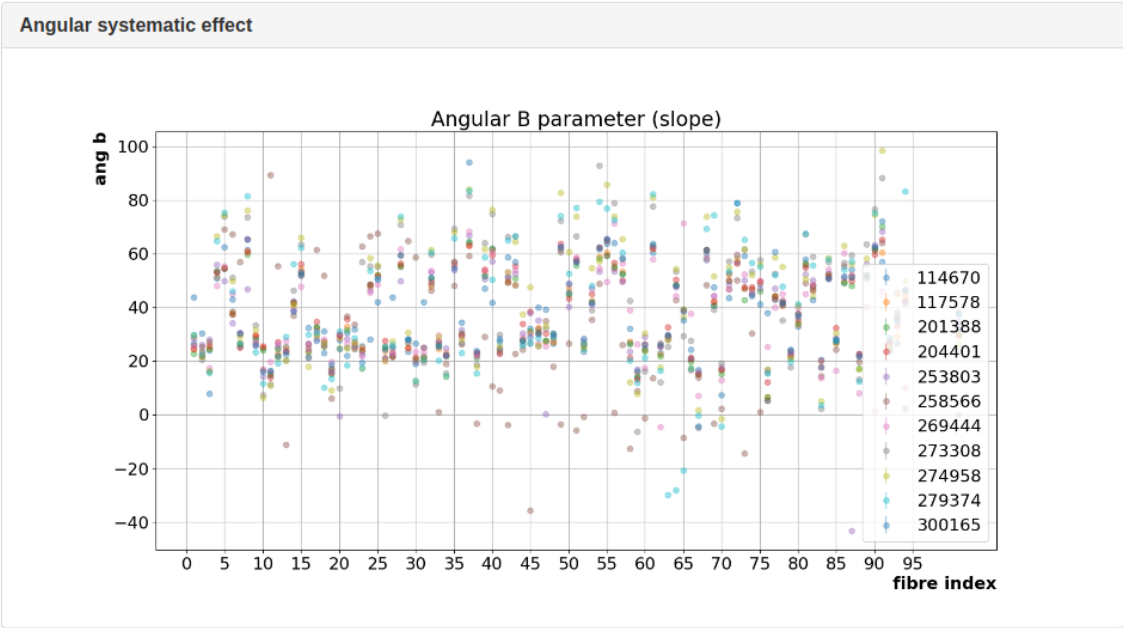


Figure 59: `PCA tables`: Comparing the PCA tables across all available datasets.
The angular b parameter from angular systematic study.

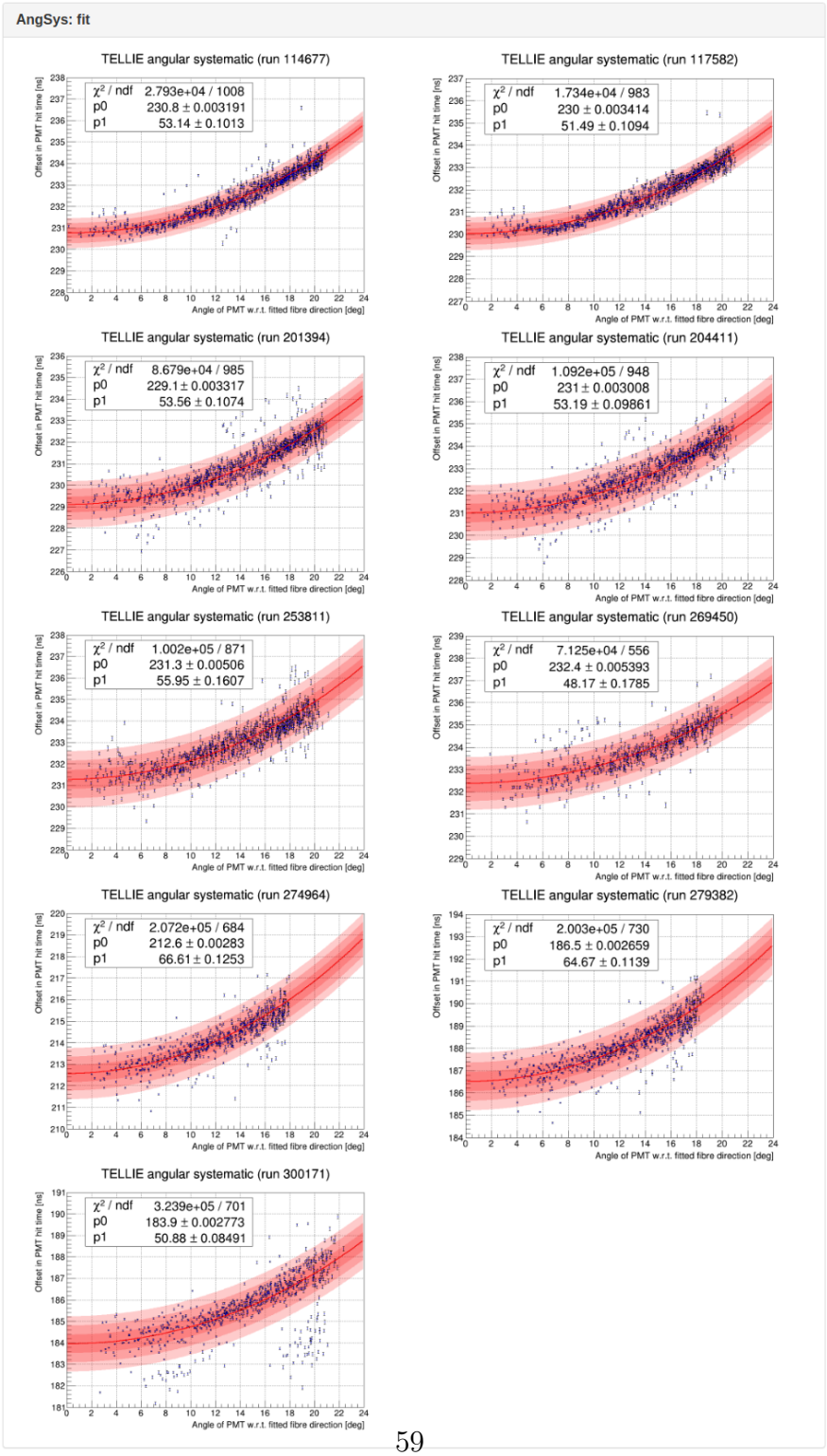


Figure 60: `Fibre data across datasets`: Angular systematic fit. Clicking a fibre name in the table for a dataset brings a page displaying data for this fibre across all available datasets. Example for ang b parameter shown.

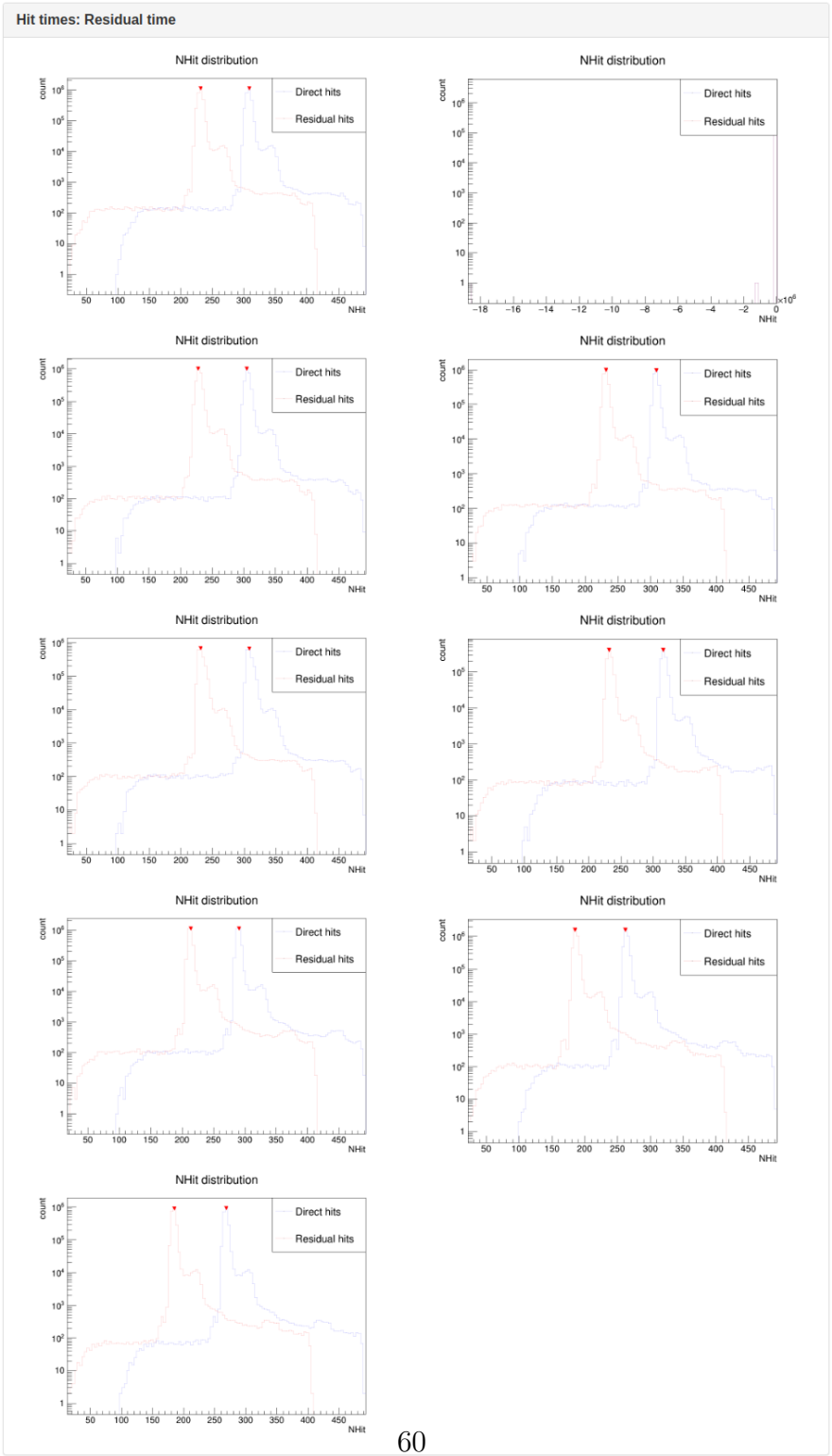


Figure 61: `Fibre data across datasets`: Residual hit times. licking a fibre name in the table for a dataset brings a page displaying data for this fibre across all available datasets. Example for residual hit times shown.

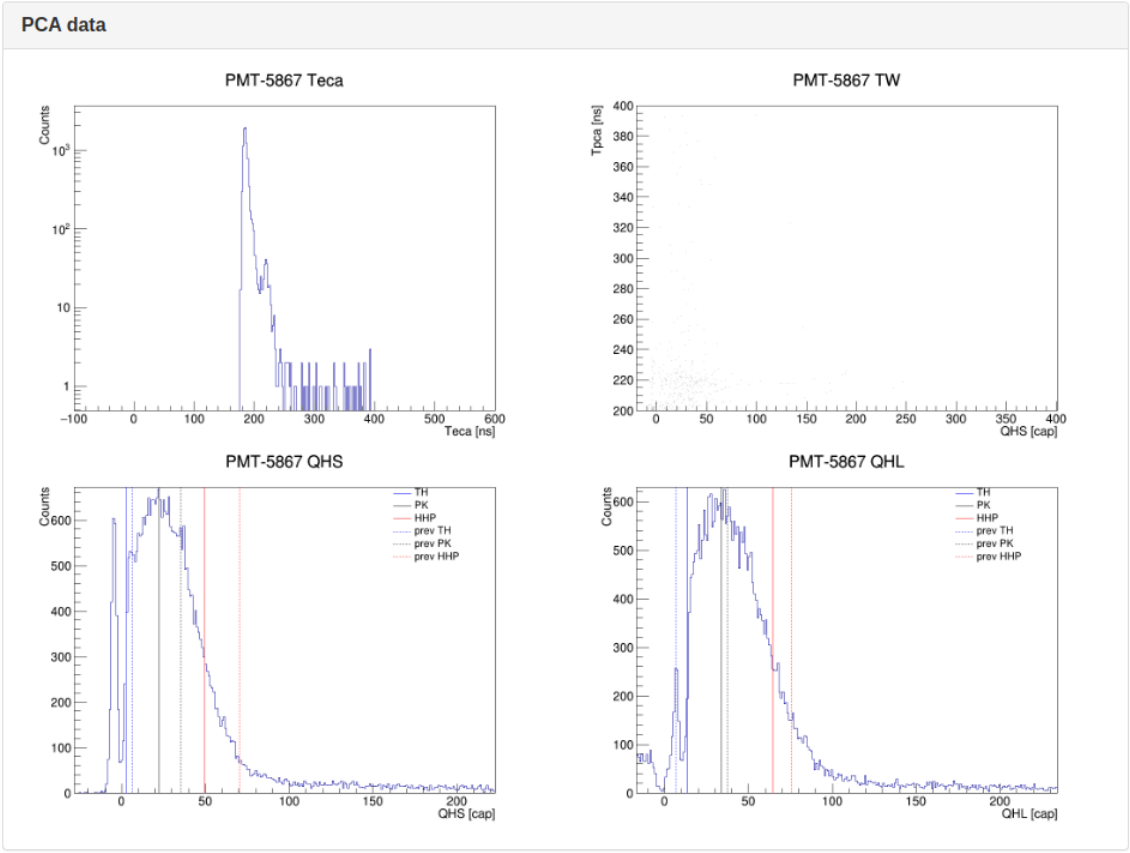


Figure 62: `PMT plots`: teca, TW, charges.
Selecting a PMT from the PCA log section or benchmarking will show its current calibration data compared to previous dataset. This includes Teca, TW, QHS, and QHL.

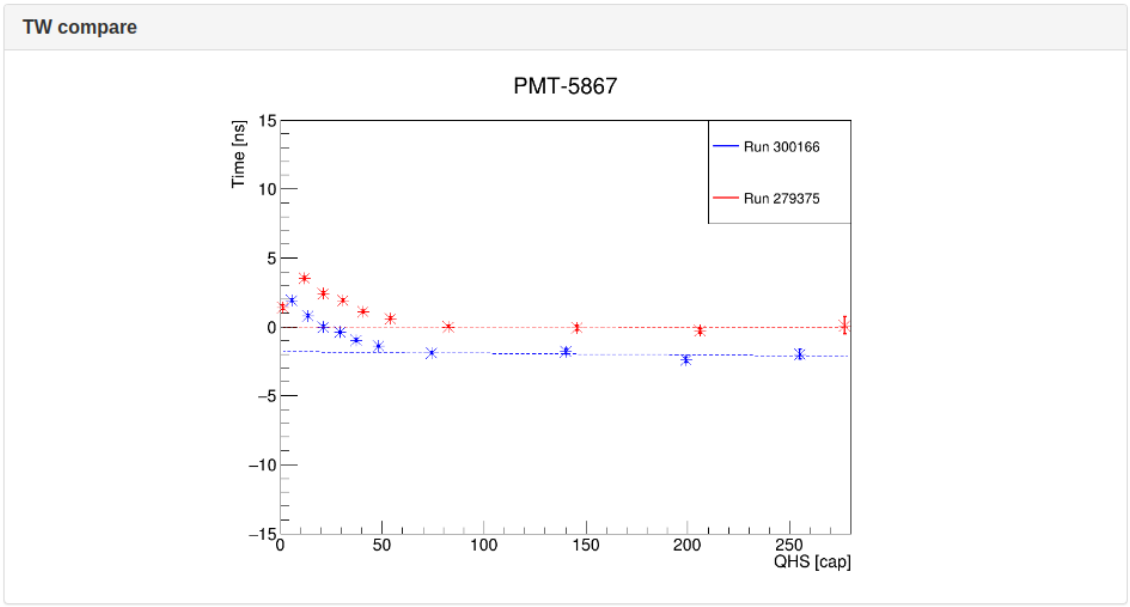


Figure 63: `PMT plots`: cable delay.
electing a PMT from the PCA log section or benchmarking will show its current calibration data compared to previous dataset. This includes cable delays.

5 System

This section will describe some of the features of the overall processing.

5.1 Simple

The system is based on being very simple and with minimum human input. As it stands, it only needs to be started, providing a single text file with runs corresponding to a dataset.
Example runlist text file:

```
300165
300167
300372
300171
300173
...
```

5.2 Modular

The whole system is a network of multiple individual steps, managed by one master script. The master script spawns subprocessed. Individual steps can be (re)run. Modules can be easily modified, separately.

5.3 Submission platform

To process the whole dataset, many individual jobs need to be run. There are 6 steps for each run: `Validation #1`, `position fit`, `angular systematic fit`, `injection fit`; and addition steps to be run per dataset: `PCA Processor`. There are several other global jobs: creating PCA table, comparing PCA tables, checking PCA output, comparing time-walk data, benchmarking (multiple steps), final compare scripts (multiple). For a single set of 95 runs (one per fibre), this ends up totalling close to 600 (95*6 + global + other) individual processes.

To deal with this, the master scripts has inbuilt submission platform. It spaws child processes (up to a configurable limit) and monitors their status. There is a queue that holds jobs to be submitted. The whole system is run in steps, as usually consecutive steps require the output of previous steps.

If processes fail, they are retried up to a limit. After that, the failed processes need to be investiaged and rerun manually (if needed).

5.4 Customizable

The checks in validation steps are often based on threshold values. There are loaded from environment and should be tuned before running. There are comments explaining what the threshold values are, please see Figure 64.

5.5 Linked

The processing system is linked to multiple databases. The CouchDB holds data on: belly plates, benchmarking, environment constants, list of fibres, runlists, and document for each run/fit combination. An overview of views in the CouchDB is shown in Figure 65, while an example environment CouchDB document is shown in Figure 66.

Additionally, the system requires access to ratDB. Finally, there is a plethora of plots created. These need to be linked to minard, usually via nfs mount over network.

5.6 Regulation

The system was developped to have unified cuts, event selection, checks, and ranges.

5.7 Evaluative

There are bitwords used in `Validation #1` and `Validation #2`. There are a proxy for the quality of the data. If tuned correctly, these could be used to ignore specific runs, however, this is not currently implemented.

```
### LIMITS FOR DIRECT AND REFLECTED LIGHT (DEG)
DIR_LIGHT_ANG=48
REF_LIGHT_ANG=20
LOCALITY=10
ANG_SYS_ANG=24
MIN_DIST=900
### VALIDATION 1
TOT_EVS=200000          ### predicted number of events
TOT_EVS_DEV=1           ### deviation for above [%]
TOT_HITS=20             ### threshold = percentage of passed hits
TH_EXTA=10              ### threshold = percentage of NON-EXTA events
TH_BADCHAN=1.75         ### threshold = percentage of bad channel hits
TH_BADECA=10            ### threshold = percentage of bad ECA hits
TH_BADPCA=5             ### threshold = percentage of bad PCA hits
TH_XTALK=0.1            ### threshold = percentage of cross-talk hits
TH_NOTEN=0.1            ### threshold = percentage of not enabled channel hits
TH_OFFPMT=0.1           ### threshold = percentage of offline PMT hits
TH_OFFCHAN=0.1          ### threshold = percentage of offline channel hits
TH_DAQEN=0.1            ### threshold = percentage of not DAQ enabled hits
TH_NNORM=0.1            ### threshold = percentage of NOT normal PMT hits
TH_BADPOS=0.01          ### threshold = percentage of bad position PMT hits
TH_LPCTIR=8             ### threshold = percentage of LPC-total internal reflection hits
TH_LPCIP=0.1            ### threshold = percentage of LPC-invalid path hits
TH_LPCLOC=0.01          ### threshold = percentage of LPC-locality hits
TH_LPCPATH=0.052        ### threshold = percentage of LPC-weird path (not throught AV) hits
TH_ANG=55               ### threshold = percentage of hits cut due to angular cut (0 - 12 deg)
TH_NEARREF=20           ### threshold = percentage of near relfection hits
NHIT_DIV=0.2            ### threshold = nhit deviation tolerance (fibre stability)
NHIT_MEAN=42            ### mean of cleaned NHit dist
NHIT_MEAN_DEV=12        ### allowed deviation for the mean NHit
NHIT_RMS=10             ### allowed RMS for the cleaned NHit dist
SUB_NHIT=10             ### mean of cleaned NHit dist, for a subrun
SUB_DEV=1               ### allowed deviation for the mean NHit, for a subrun
HIT_PEAK=302            ### the assumed mean of the direct hit time distribution
HIT_PEAK_DEV=10         ### allowed deviation for above [value]
TIME_DIV=0.75           ### threshold = time deviation tolerance (fibre stability)
PMTS_BS=200             ### # PMTs in beamspot
PMTS_BSP=2              ### # PMTs in beamspot, percentage
PMTS_BS_GO=150          ### # PMTs in beamspot, with good occupancy
PMTS_BS_GOP=1.5         ### # PMTs in beamspot, with good occupancy, percentage
PMTS_BS_RAT=70          ### ratio of PMTs in beamspot with good occupancy, to all PMTs in beamspot
RUNTIME=2500            ### allowed length of run in seconds
FREQ=1000               ### aimed frequency
FREQ_DEV=7.5            ### tolerance for real frequency (%)
EV_SUB=5000             ### expected evs in subrun
EV_SUB_DEV=0.5          ### tolerance for above (%)
N_SUBS=40               ### number of allowed subruns
INTEG_ALL=30            ### integral of PMTs with good occupancy (all PMTs)
INTEG_BS=70             ### integral of PMTs with good occupancy (beamspot PMTs)
BS_PMTS=150             ### number of PMTs with good occupancy in the beamspot
PIN_RMS=3               ### RMS of the pin distribution
COV=0.0                 ### minimal covariance
CORF=0.6                ### minimal correlation factor
TOF_MEAN=82             ### expected mean for time of flight [ns]
TOF_RMS=0.75            ### maximal allowed RMS for time of flight distribution
BUCK_MEAN=0.47          ### expected mean for bucket time distribution [ns]
BUCK_RMS=0.005          ### maximal allowed RMS for above
AS_MEAN_MIN=0           ### minimal allowed mean for ang sys distribution [ns]
AS_MEAN_MAX=2           ### maximal allowed mean for ang sys distribution [ns]
AS_RMS=0.4              ### maximal allowed RMS for above
COR_DEV=3               ### allowed deviation for the means (tof, buc, angsys) [%]
TOF_MIN=78              ### min allowed for tof
TOF_MAX=85              ### max allowed for tof
BUC_MIN=0.45            ### min allowed for bucket
BUC_MAX=0.48            ### max allowed for bucket
AS_MIN=0.0              ### min allowed for ang sys
AS_MAX=1.5              ### max allowed for ang sys
RESID_PEAK=220          ### the assumed mean of the ersidual hit time distribution
RESID_PEAK_DEV=12       ### allowed deviation for above [value]
DIR_RESID_0=-78         ### fit (pol1) direct-residual time relation, parameter 0: y-intercept
DIR_RESID_1=1.05        ### fit (pol1) direct-residual time relation, parameter 1: slope
FIT_SLOPE=0             ### fit (pol1) both direct and resid vs angle, parameter 1: slope
FIT_SLOPE_DEV=0.1       ### max allowed deviation for above
```


Figure 64: The threshold parameters, used for validation. These can be tuned as required.

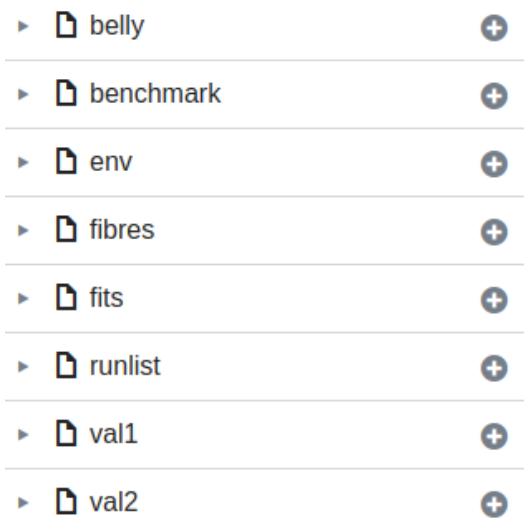


Figure 65: `CouchDB`: An overview of views in CouchDB database.

```
{
  "_id": "1a8b42cf505ca5e765500c80071406a3",
  "_rev": "3-332d0357729c8e27825a0e2d20ba9d22",
  "REF_LIGHT_ANG": "20",
  "COV": "0.0",
  "NHIT_MEAN": "42",
  "TH_LPCLOC": "0.01",
  "TOT_HITS": "20",
  "BS_PMTS": "150",
  "FIT_SLOPE_DEV": "0.1",
  "TH_LPCIP": "0.1",
  "LOCALITY": "10",
  "TH_BADPOS": "0.01",
  "TH_BADPCA": "5",
  "TH_BADCHAN": "1.75",
  "AS_MAX": "1.5",
  "TOF_MIN": "70",
  "TH_XTALK": "0.1",
  "PMTS_BSP": "2",
  "COR_DEV": "3",
  "type": "env",
  "NHIT_DIV": "0.2",
  "TOF_MAX": "85",
  "TH_DAQEN": "0.1",
  "HIT_PEAK_DEV": "10",
  "TOF_RMS": "0.75",
  "timestamp": "20-Oct-2022 00:09:58.222304",
  "NHIT_RMS": "10",
  "PMTS_BS_GOP": "1.5",
  "AS_MIN": "0.0",
  "BUC_MAX": "0.48",
  "FREQ": "1000",
  "FIT_SLOPE": "0",
  "TH_NNORM": "0.1",
  "N_SUBS": "40",
  "MIN_DIST": "900",
  "DIR_LIGHT_ANG": "48",
  "PMTS_BS_GO": "150",
  "EV_SUB_DEV": "0.5",
  "TH_NEARREF": "20",
  "BUC_MIN": "0.45",
  "FREQ_DEV": "7.5",
  "ANG_SYS_ANG": "24",
  "TH_OFFPMT": "0.1",
  "NHIT_MEAN_DEV": "12",
  "PMTS_BS_RAT": "70",
  "TH_NOTEN": "0.1",
  "TOF_MEAN": "82",
  "TH_ANG": "55",
  "CORF": "0.6",
  "RESID_PEAK_DEV": "12",
  "TOT_EVS_DEV": "1",
  "TH_LPCPATH": "0.052",
  "AS_RMS": "0.4",
  "INTEG_ALL": "30",
  "TH_LPCTIR": "8",
  "INTEG_BS": "70",
  "BUCK_RMS": "0.005",
  "TIME_DIV": "0.75",
  "TH_OFFCHAN": "0.1",
  "HIT_PEAK": "302",
  "EV_SUB": "5000",
  "RESID_PEAK": "220",
  "SUB_DEV": "1",
  "DIR_RESID_1": "1.05",
  "DIR_RESID_0": "-78",
  "AS_MEAN_MIN": "0",
  "RUNTIME": "2500",
  "TH_BADECA": "10",
  "BUCK_MEAN": "0.47",
  "AS_MEAN_MAX": "2",
  "TH_EXTA": "10",
  "TOT_EVS": "2000",
  "PMTS_BS": "200",
  "SUB_NHIT": "10",
  "PIN_RMS": "3"
}
```


Figure 66: `CouchDB`: An example CouchDB document (in this case holding the thresholds for checks.

6 Code

This section will provide some comments on the code (TELLIE Automation repository).

6.1 Structure

- **Data-taking**: This folder holds documents, notes, and early attempts on the data-taking automation (no actual repository for automated TELLIE PCA data-taking!).
- **Doc**: This folder holds documentation (including this doc), reports, and notes on the automated TELLIE PCA processing. Please have a look at this.
- **Processing**: This contains the actual code performing the automated TELLIE PCA processing.
  - *angular_systematic*: Processor for the angular systematic fit, see Subsection 4.1.3.
  - **bench_root*: Holds the output root files from benchmarking for each processed set, see Subsection 4.5.
  - *benchmark*: Scripts to extract comparison plots after benchmarking, see Subsection 4.5.
  - *checkPCA*: Scripts to extract plots from the output of PCA processor, see Subsection 4.4.
  - *fits*: This holds text files with the results of the fit processors: direction, ang sys, and pca offset. See Section 4.1.
  - *logs**: The log files holding results of fits, these are uploaded to CouchDB.
  - *minard*: Plots for minard monitoring are stored here. See Subsection 4.6.
  - *pca_constants*: Holds output ratDB and log files from the PCA processor, Subsection 4.4.
  - *pca_offset*: Processor for the pca offset (injection time) fit, see Subsection 4.1.4.

- *pca_tables*: Scripts to compare PCA tables. Also holds copies of PCA tables. See Subsection 4.2.
- *postition_fit*: Processor for the position (beamspot) and direction fit, see Subsection 4.1.1 and Subsection 4.1.2.
- *runtime*: A folder to hold temporary files while processing a dataset.
- *scripts*: Other useful scripts, usually called by the master script.
  - create_bench_apply_mac.py: Creates RAT macro to run benchmarking - loads the new constants and applies on laserball run.
  - create_pca_proc_mac.py: Creates RAT macro to run PCA processor for given dataset.
  - delete_old_couchdb.py: Helper script to delete CouchDB documents by type.
  - get_belly_fibres.py: Helper script to create ratDB file containing the list of fibres affected by belly plates (loaded from CouchDB document).
  - load_db.py: Helper script to load default fibre directions from ratDB.
  - make_ratdb_table.py: Creates ratDB table from PCA table. This is loaded by the PCA processor to apply corrections to hit times.
  - master.py: The brain of operations. See Subsection 6.2.
  - upload_bench.py: Parses data from benchmark comparison scripts and uploads a CouchDB document.
  - upload_env.py: Parses the environment variables and uploads a CouchDB document.
  - upload_fits.py: Parses the data from fits and uploads a CouchDB document.
  - upload_radtb.py: Uploads ratDB table to postgres.
  - upload_val1.py: Parses the data from `Validation #1` step and uploads a CouchDB document.
  - upload_val2.py: Parses the data from `Validation #2` step and uploads a CouchDB document.

– *validate*: Processor to perform the `Validation #1` step of the processing suite.

– *validate2*: Processor to perform the `Validation #2` step of the processing suite.

[1]:

[1]*: these are not part of the github repository (since they are just holding data), and have to be added manually.

6.2 Master script

This is the main script of the whole suite. This is the one called by the user to start the processing. It handles almost every portion of the suite.

The master script requires the run list, in the form of a text file, holding the run numbers for the TELLIE runs forming the dataset. It has inbuilt submission platform, and it will spawn processes to automatically run the TELLIE processing. This is done in specific order, as often consecutive steps require the output of previous step(s). The platform also monitors the output of the processes, and retries them (up to a limit).

Here are the steps, in order:

- *loads environment*: The environment holds script names and locations.
- *parse arguments*: Parses the supplied command line arguments (see Section 8).
- *upload environment (opt)*: Uploads the environment variables to CouchDB.
- *parse runlist*: Parses the provided runlist (from text file).
- *create dataset*: Uploads a document holding dataset information.
- *check data exists*: Checks that physical files exist for the provided runlist.
- *set job limit*: Sets a limit on how many processes can be run at each time.
- *sort runs*: Order runlist.
- *call Validation #1*: Runs the `Validation #1` step over all runs, using the submission platform.
- *upload Validation #1*: Uploads the results of `Validation #1` to CouchDB.
- *call position fit*: Runs the position and direction fit step over all runs, using the submission platform.

- *call ang sys fit*: Runs the angular systematic fit step over all runs, using the submission platform.
- *call pca offset fit*: Runs the pca offset fit step over all runs, using the submission platform.
- *upload fits*: Uploads the results of fit steps (together) to CouchDB.
- *call Validation #2*: Runs the `Validation #2` step over all runs, using the submission platform.
- *upload Validation #2*: Uploads the results of `Validation #2` to CouchDB.
- *log cleanup*: Removes (now obsolete) logs.
- *create run folders*: Create folders for each run in the designated minard location.
- *move plots*: Move plots (created by previous stages) to the minard location.
- *make pca table*: Create PCA table from the fit logs.
- *move fits*: Move the fit (text) files to their location.
- *compare tables two*: Call script to compare new PCA table to previous one.
- *compare tables all*: Call script to recreate plots comparing all PCA tables.
- *upload ratdb table*: Call script to upload ratdb table to postgres.
- *set new names*: Pre-set names for files (to expect) from PCA processor.
- *create pca proc mac*: Call script to create PCA processor RAT macro.
- *call pca proc*: Spawn RAT process to run the PCA processor.
- *get global offset*: Retrieve the global offset from the PCA log file.
- *move pca const*: Move the output files from the PCA processor.

- *call checkPCA*: Call scripts to compare the PCA results to previous set.
- *call compareTW*: Call scripts to compare TW results to previous set.
- *move pca plots*: Move plots created by scripts above to minard location.
- *create bench apply mac*: Call script to create benchmarking RAT macro.
- *setup tw tables*: This function prepares the TW and GF ratdb tables. These need to be temporarily modified to overwrite the defauls from the ratDB.
- *call bench apply*: Spawn RAT process to run the benchmarking.
- *call cd compare*: Call script to compare cable delays from benchmarking.
- *call tw compare*: Call script to compare time-walk data from benchmarking.
- *call peak compare*: Call script to compare residual times peaks from benchmarking.
- *upload bench*: Uploads the results of benchmarking steps (together) to CouchDB.
- *move bench plots*: Move plots created by scripts above to minard location.
- *move bench root*: Move the benchmarking root files to their defined location.
- *move pca to minard*: Move PCA ratDBs to minard location (these are needed for online monitoring).
- *cleanup*: Move logs (from fits, PCA processor, and benchmarking), and remove (now obsolete) mac, and ratdb files.
- *log cleanup*: Remove unwanted logs (from RAT).
- *get final job count*: Print out the final job statistics (running, succesful, failed).

There are several other helper functions that are not called directly, but rather called from within other functions. This includes the submission platform and monitorin. For these, please look at the master script.
The script is here:

```
Automation/Processing/scripts/master.py
```

```
// Trigger type
trig = ev.GetTrigType();
if (!(trig & 0x8000)) continue;    // EXT trigger only
```

Figure 67: `Checks`: trigger type.
The trigger type check to only use events with EXT bit.

```
const RAT::DS::CalPMTs& pmts = ev.GetCalPMTs();   // calib PMTs
for(int iPMT=0; iPMT<pmts.GetNormalCount(); iPMT++) {
  RAT::DS::PMTCal pmt = pmts.GetNormalPMT(iPMT);
  int pmtID = pmt.GetID();

  const RAT::DU::PMTCalStatus& pmtStatus = RAT::DU::Utility::Get()->GetPMTCalStatus();
  const RAT::DU::ChanHWStatus& chs = RAT::DU::Utility::Get()->GetChanHWStatus();
  const RAT::DU::PMTInfo& pmtinfo_loop = RAT::DU::Utility::Get()->GetPMTInfo();
  unsigned int status = pmtStatus.GetHitStatus(pmt);
  if(status & (1<<pmtStatus.kCHSBit)) continue;
  if(status & (1<<pmtStatus.kECABit)) continue;
  if(status & (1<<pmtStatus.kPCABit)) continue;
  if(status & (1<<pmtStatus.kXTalkBit)) continue;
  if ( !chs.IsTubeOnline(pmtID) ){ continue; }
  if ( !chs.IsEnabled() ){ continue; }
  if ( !chs.IsChannelOnline(pmtID) ){ continue; }
  if ( !chs.IsDAQEnabled(pmtID) ){ continue; }
  if ( pmtinfo_loop.GetType(pmtID) != 1 ){ continue; }

  // Get info for this PMT
  const RAT::DU::PMTInfo& pmtinfo = RAT::DU::Utility::Get()->GetPMTInfo();
  TVector3 pmtPos = pmtinfo.GetPosition(pmtID);       // position [mm]
  TVector3 pmtDir = pmtinfo.GetDirection(pmtID);      // direction
  if (pmtPos.Mag()==0) { continue; }
```

Figure 68: `Checks`: PMT checks.
This highlights the checks on the PMT status, type, PCA & ECA bitwords, X-talk, PMT position, and more.

6.3 Cuts and checks

This subsection lists some of the more important cuts and checks that are applied throught the many steps of the processing automation. Please see Figures 67, 68, 69, and 67.

```
// LPC checks
if (lpc.GetTIR() == 1) { continue; }            // total internal reflection
if (lpc.GetPathValid() == 0) { continue; }      // check whether path parameters are valid
if (lpc.GetResvHit() == 1) { continue; }        // whether end point was within locality
```

Figure 69: `Checks`: LPC features.
Checks on LPC related features such as the total internal reflection, validity of the path, and the locality.

```
if (lpc.GetTotalDist() <= 12000){nearL++;} else {farL++;}
if (lpc.GetTotalDist() <= 12000){continue;}         // this rejects near reflections
if (lpc.GetDistInInnerAV() <= 7000){continue;}      // this rejects other weird paths
```

Figure 70: `Checks`: LPC values.
Additional checks on the actual LPC values to ensure only direct, unreflected, physical paths are used.

7 Deploying

This section summarizes some notes from the deployment.

7.1 Requirements

There are some requisites for the software to operate successfully.
First, because the base of the software is RAT processors, a functioning RAT is required.
Second, the plotting, creating macros and tables, and interfacing with couchDB is done via python scripts. Most of the modules are (usually) present with python by default, such as: os, sys, re, argparse, time, shlex, subprocess, math, json, signal, datetime. Some may need to be installed if not present, such as: couchdb, numpy, and matplotlib.

7.2 External

The data from the suite is stored in CouchDB. The CouchDB database is `tellie_auto`. There are several views available:

- *belly*: A single document holding list of fibres affected by belly plates.
- *benchmark*: View holding documents from the benchmarking step.
- *env*: A single document holding the environment variables (tune thresholds).
- *fibres*: A single document holding list of active fibres.
- *fits*: View holding documents from the fits steps.
- *runlist*: View holding documents defining the runlists (runs and fibres).
- *val1*: View holding documents from the validation 1 step.
- *val2*: View holding documents from the validation 2 step.

These views and single documents (belly, env, fibres) need to be created when deploying. Additionally, the CouchDB users need to be given access to this database. Finally, the address, and log in credentials for CouchDB need to be set up in the environment file.

Additionally, some run data is loaded from RatDB. Therefore, the credentials need to stored in the usual .pgpass file.

7.3 Setting up

The repository is located at: https://github.com/snoplus/pcatellie.This is a fork from the original: https://github.com/MRiganSUSX/Automation. After cloning, one needs to create some additional directories, which hold data that is not, by default, part of the repository, such as: mianrd, pca_constants, runtime, logs. These are mentioned in Subsection 6.1. After that, the environment file (inside Processing) needs to be set up. This holds: credentials, database details, paths, scripts locations, path to RAT, and threshold values for checks.

After that, one needs to build the processors and comparison scripts. Run 'scons' in angular_systematic, benchmark, pca_offset, position_fit, validate, and validate2 folders. Also run 'make' in benchmark/cd_compare, benchmark/peak_compare, benchmark/tw_compare, and checkPCA. Rerun these commands when a particular processor or plotting script is changed.

Additionally, one should make sure the template RAT macros in the processor directories (angular_systematic, pca_offset, position_fit, validate, validate2) and the scripts making RAT macros (create_bench_apply_mac.py and create_pca_proc_mac.py) are correct for the given geometry.

Because all datasets were already processed, the data was also copied over, including minard plots (in /minard), the benchmarking root files (in bench_root), fit files (in logs), pca constants (in pca_constants), pca tables (in pca_tables/tables), and laserball root files (in pca_constants/LB).

It is also important to NFS mount the minard location - where all the plots are (this is in Processing/minard) to the minard server. The server hosting the TELLIE PCA processing automation is:

```
autopca.sp.snolab.ca
(192.168.80.56)
```

The minard server is:

```
minard.sp.snolab.ca
(192.168.80.128)
```

This is how the minard location is mounted:

```
/home/snotdaq/Automation/Processing/minard @ autopca
->
/mnt/pcatellie @ minard
```

and also:

```
/home/snotdaq/Automation/Processing/pca_constants @ autopca
->
/mnt/pca_constants @ minard
```

Finally, symlinks are made @ minard as:

```
/mnt/pca_constants -> /mnt/pcatellie/pca_constants
/mnt/pcatellie -> /home/tlatorre/minard/minard/static
```

[2]

The user at 'autopca' is 'snotdaq' with the usual password. The location of the repository is:

```
/home/snotdaq/Automation (@ autopca)
```

Othe locations (@ autopca):

```
/home/snotdaq/data : TELLIE PCA data to be processed
/home/snotdaq/env.sh : RAT env is loaded here
(Called by the Automation/Processing/env.sh)
/home/snotdaq/rat : RAT install
```

8 Running

In general, the running is supposed to be autonomous - after being started. However, I suggest monitoring the application thoroughly the first few times it is run (as the development was done with different setup). It should also be monitored occasionally when tuned, as there will be indications if things go wrong - such as high number of jobs failing.

[2]The firewall needed to be modified @ autopca to allow connections to NFS ports.

For actual start-up, I recommend running this in screen. Firstly, the environment needs to be sourced. The `env.sh` file contains: CouchDB address and credentials, threshold parameters for checks, names and locations of script files, and ratDB credentials. This file is not included with the repository, but is placed in the deployed location manually.

Several folders and default files need to be in place for the application to run succesfully. These are mentioned in Section 7.

The only input required is the text file holding the run numbers for TELLIE PCA runs (one for each fibre). An example of this is shown in Subsection 5.1.

Finally, one just calls the master script. There are few command line arguments that can be provided, as shown in Figure 71. The only required argument is '-f', providing the path to the runlist tex file. Other options of interest are '-c', selecting the number of cores: this corresponds to the number of simultaneous jobs that can be spawned; and '-e', which is an option to reupload the environment. All other arguments allow to select which steps to run from the full suite. In normal conditions - running processing on full TELLIE PCA dataset - it is always recommended to run all steps.

An example run procedure:

```
screen -S tellie_pca
cd <Automation repository>
source env.sh
cd runtime
python ../scripts/master.py -f <path/to/runlist.txt> -c <cores>
```

Then one can detach from the screen, and attach occassionaly to monitor.

```
parser = argparse.ArgumentParser(description='The master script for TELLIE automation - processing.')
parser.add_argument("-f", dest="run_list_file", help="File containing the run list (full path)", type=str, required=True)
parser.add_argument("-c", dest="cores", help="How many cores to use (ie consecutive jobs) ?", default=4, type=int)
parser.add_argument("-e", dest="upload_env", help="Should reupload env ?", default=0, type=int)
parser.add_argument("-val1", dest="val1", help="Should run validate (1) ?", default=1, type=int)
parser.add_argument("-val2", dest="val2", help="Should run validate (2) ?", default=1, type=int)
parser.add_argument("-fit1", dest="fit1", help="Should run position fit ?", default=1, type=int)
parser.add_argument("-fit2", dest="fit2", help="Should run ang sys fit ?", default=1, type=int)
parser.add_argument("-fit3", dest="fit3", help="Should run pca offset fit ?", default=1, type=int)
parser.add_argument("-pca_tab", dest="pca_tab", help="Should create pca table ?", default=1, type=int)
parser.add_argument("-pca_tab_comp", dest="pca_tab_comp", help="Should compare pca table ?", default=1, type=int)
parser.add_argument("-pca_tab_upl", dest="pca_tab_upl", help="Should upload pca table ?", default=0, type=int)
args = parser.parse_args()
return args
```

Figure 71: `Master script`: The available command line arguments.

9 ToDos

This section briefly lists some open questions:

- *TELLIE PCA data*: TELLIE PCA data needs to be placed in '/home-/snotdaq/data' @ autopca. This can be either done manually, via script, or one can perhaps NFS mount the raid folder with live data.
- *RAT integration*: the code currently consist of custom user processors. These could all be made official part of RAT once finalised. Same goes for the PCA processor, which is currenly a heavily modified version.
- *Data-taking modes*: #7612
- *Tagging events outside Orca*: if one decides for continuous calibration, TELLIE is run 'alongside' ORCA. EXTA events need to be tagged externally, so that they can be extracted from data offline.
- *Ensure recent ECA*: recent ECA is essential for PCA calibration. This should be enforced just before running PCA.
- *When (from) to apply new PCA constants*: Depending on the data-taking mode, it can take days to weeks to collect enough data for PCA calibration. PCA constants can only be extracted from a full dataset. Once this is done one needs to decide whether to apply the constants from (i) the start of the set, (ii) the end of the set or, (iii) some other time (like middle of the set).
- *RatDB tables cache*: because we process every run multiple times (over multiple processors), the ratDB tables are loaded multiple times. In some cases (like ECA tables), this is significant size, and we also have to account for the number of consecutive ratDB connections, which sometimes causes issues (time out). One improvement would be to pre-load the tables and keep them locally while the runs are being processed.

10 Other documentation

Other useful TELLIE / PCA documentation:

- PCA calibration with SNO+ RAT
- TELLIE hardware manual
- TELLIE user manual
- TELLIE angular systematic study
- TELLIE debugging
- TELLIE fibre validation
- Michal's thesis
- Also look at the documentation folder in the repository:

```
Automation/Doc
```