Wire-Cell Validation "Systems"

Brett Viren
(for W-C team)

Physics Department

BROOKHAVEN
NATIONAL LABORATORY

μB Reco
2017 Dec 12

Outline

Wire-Cell Feature Overview

Toolkit:

- **noise filter** coherent, harmonic, PMT, etc. $\sim \mu$B-specific.
- **signal processing** broad+fine 2D deconvolution and ROI-finder.
- **signal simulation** drift physics, broad+fine 2D field/elec responses, ADC.
- **noise simulation** spectral sampling with MicroBooNE empiric noise model.

Toolkit functionality available via provided `wire-cell` command line program, through user-application or, to some extent, via *art*/LArSoft integration (next slide).

Prototype:

- Incubator for code that eventually goes into the toolkit.
- Now incubating: **3D imaging**, **clustering** and **pattern recognition**.

Prototype functionality available via its integration test programs.

(various software design, development and user technical features omitted here)

Wire-**C**ell / **LAr**Soft Integration

"WC/LS"

- Design overview
  - call WCT code on each event from inside *art* module via a *art* tool.
  - formalize WC $\leftrightarrow$ LS data product conversion
  - job aggregation is fully dynamic (via FHiCL + WC config langs)
- Single `WireCellToolkit_module` and `WCLS_tool` provided.
  - Covers most/all use cases, but tool can be used from user modules.
- All integration code in LArSoft as `larwirecell` package.
  - A `wirecell` UPS product build from WCT releases.
  - All "heavy lifting" code in WCT shared libraries.

Current status and near term:

- $\times$ Obsolete: old style NF-only module only in MCC 8.x branch.
- ✓ Done: Noise Filter + Signal Processing targeting MicroBooNE.
- $\rightarrow$ Next: Simulation, helped immensely by LArG4 reorg effort.

Wire-Cell Prototype Validation

- Initial concepts are developed in the WC **Prototype** code base.
- Conceptual/algorithm validation procedures might be described as intensive, data driven and not reproducible (in a CI sense).
- A number of `main()` programs provide both integration level testing as well as main applications for initial processing and analysis.
  - Test code tends to be somewhat monolithic but with much functionality sequestered to the WCP libraries.

→ We treat results from the prototype as the benchmark for what WCT code must reproduce.

Wire-Cell Toolkit Validation "System"[1]

Design: validation suite runs as a **build system** using WAF.

+ Failures halt the "build" and can not be denied.
+ Test successes need not be rerun after dealing with failures.
+ Runs various tests, some which depend on others.
+ Runs tests in parallel to the extent possible given dependencies.
+ Generates JSON summaries and templated (Jinja2) web pages.
+ Results: logs, plots, **diffs**, job errors.
+ Some hist diffs done automatically, lead to "build" failure.
- Relies on a few GB of MicroBooNE data files.
- Not directly conducive to integrating into "real" CI system.
- Waf's `wscript` is powerful but not for everyone.

[1] Scare quotes because many improvements really should be made.

Do a validation build

```
$ source setup_wcls_environment_is_exercise_for_user
$ ./waf configure  \
      --prefix=$HOME/public_html/wctv \
      --wct-build=/path/to/wct/build \
      --wct-data=/path/to/data/validation
$ ./waf
$ firefox $HOME/public_html/wctv/html
```

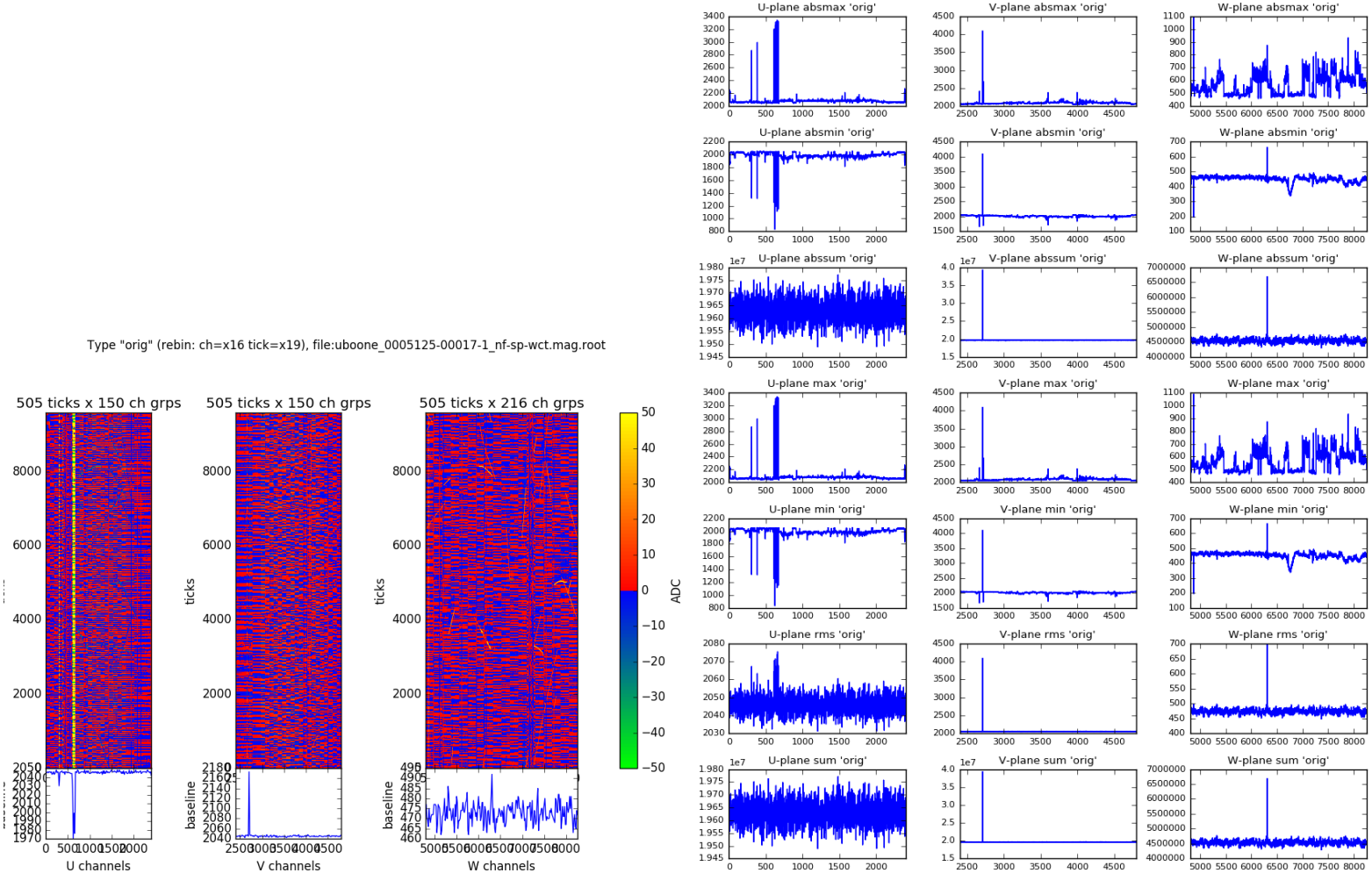
Click to see results in all their stunning lack of CSS glory

and, here is the code repo:
https://github.com/WireCell/wire-cell-validate

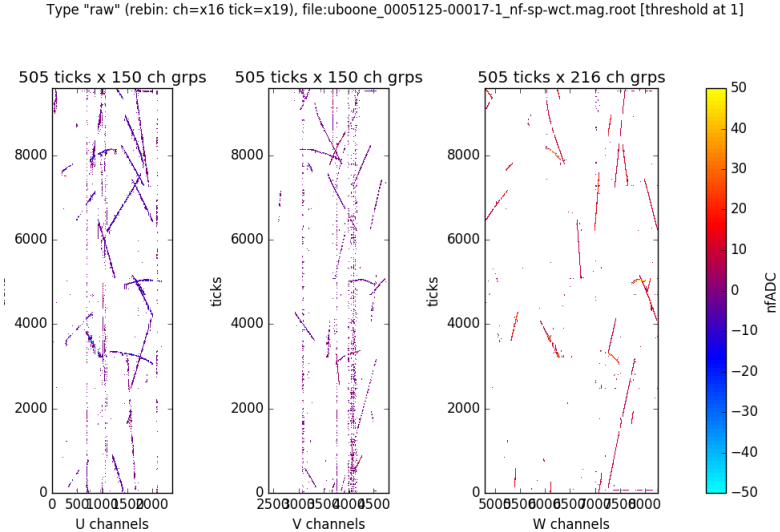
Current validation contents

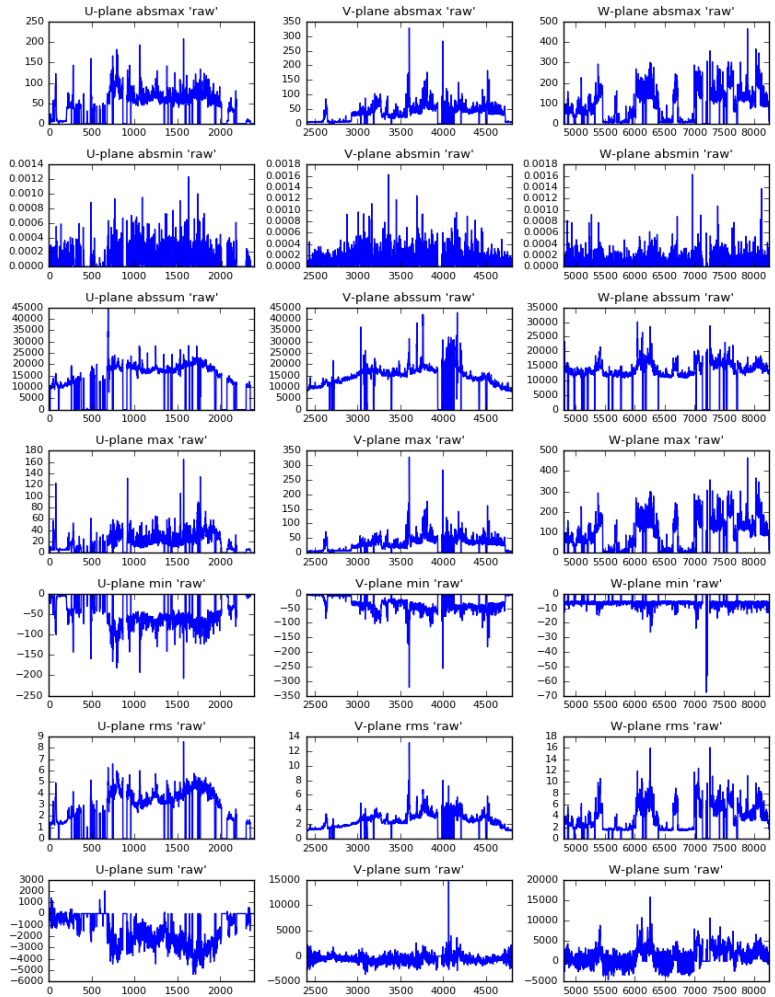
- Idea is to add a new `subdir/wscript` file for each test scope.
- So far only scope is: NF+SP as WCT vs WC/LS
- Run same WCT code as pure `wire-cell` and `art` jobs.
- Form histograms and event displays for each.
  - $\sim$ (uses Numpy/Matplotlib but ROOT could also be used)
- "Diffs" between corresponding WCT and WC/LS results.
- Produce reduced and full resolution event displays.
- Also produce summary/profile hists.
- Look for any non-zero content in subtracted hists/displays.
- `foreach` file, `foreach` event in file, `foreach` tier.
- Processing "tiers":
  - $\rightarrow$ original ADC, NF'ed "raw", "wiener" and "gauss" filtered signals,

Original ADC

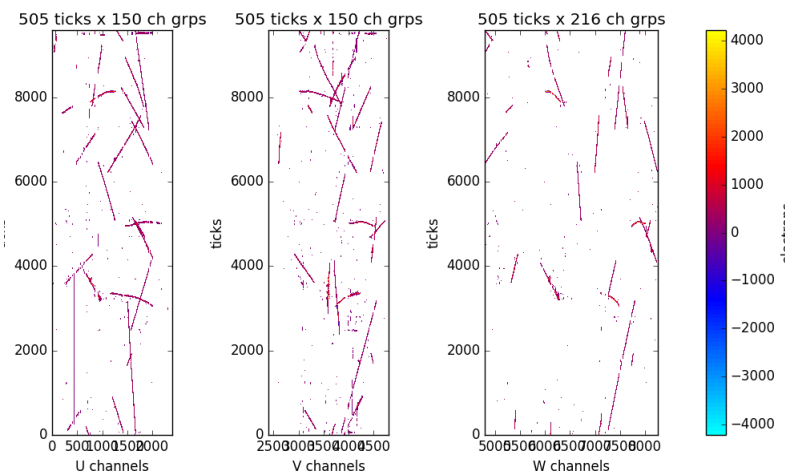


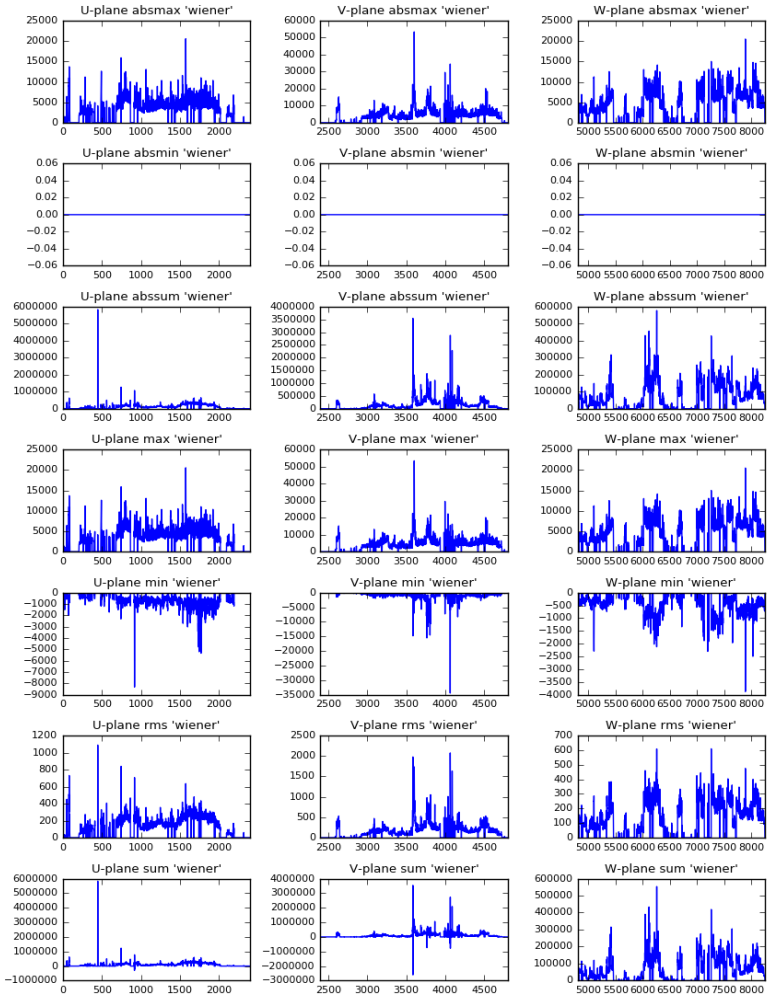
Noise Filtered "raw"



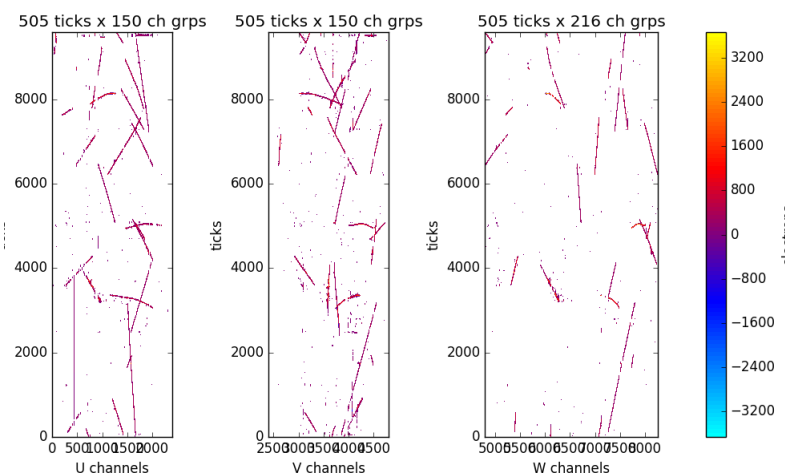


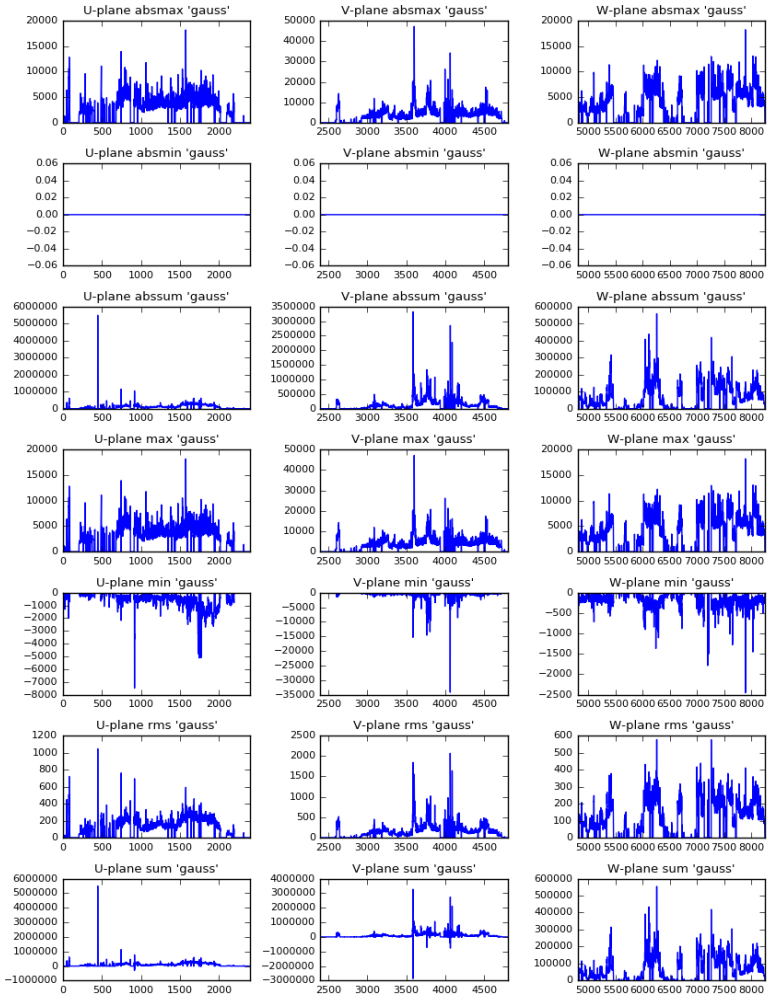
Signal Processed with Wiener Filter



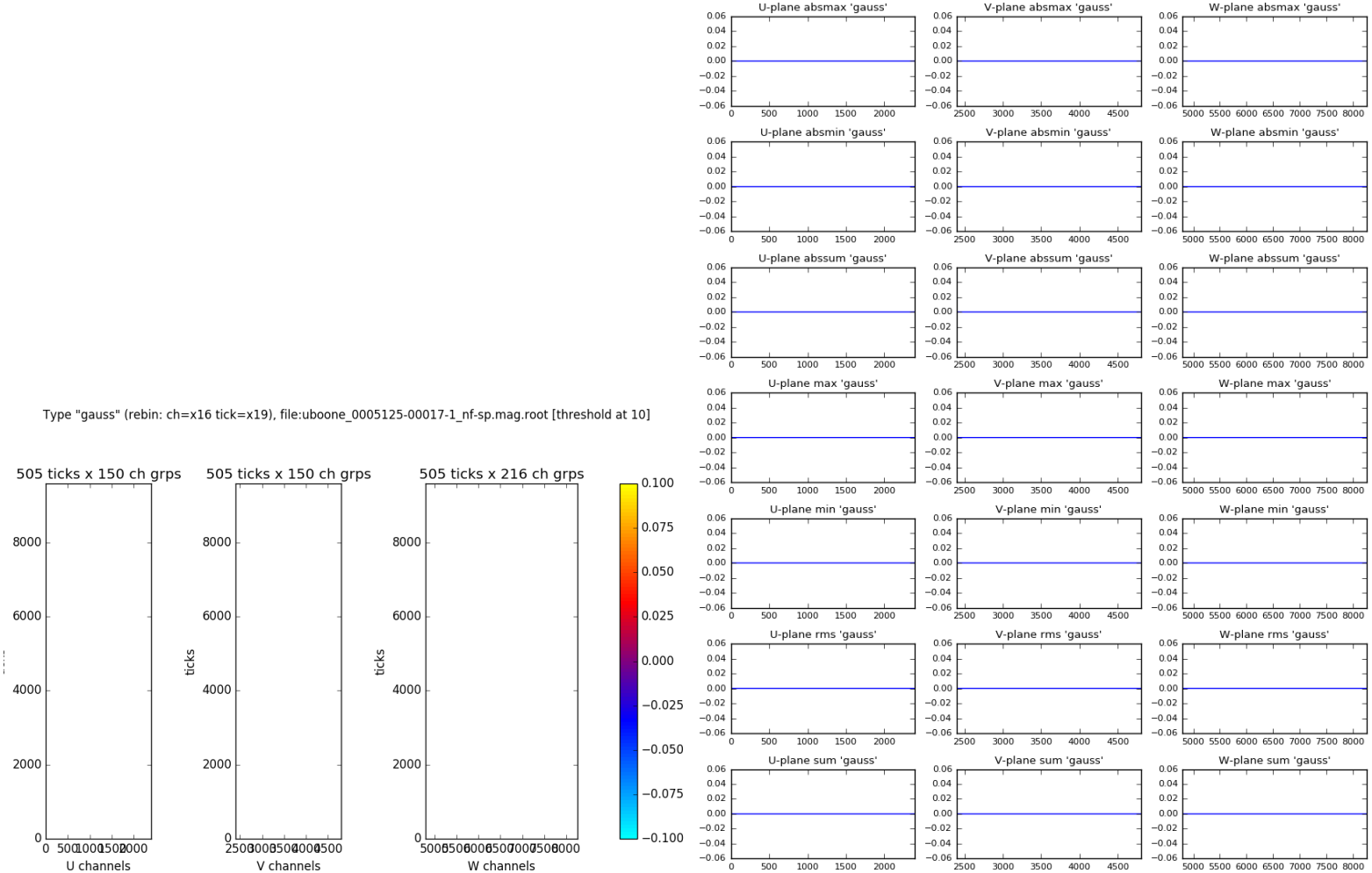


Signal Processed with Gauss Filter





Super Exciting WCT-WC/LS "diffs"



Unsolicited opinions on a uBoone CI

- Provide a limited and **fixed input data** file set spanning all detector running. MC can/should be run as a test itself.
- System must allow for **dependencies** between test jobs.
  - Test validity of intermediate output files.
- Allow tests in various languages/systems (not just art/C++).
- Allow various output files and post process them for validity.
- But, also make standardized JSON summary file for every test.
- **Integration tests likely dominate** over unit tests.
- Crucial to have **blessed output** which is automatically compared to output of each run. $\rightarrow$ Just as crucial to make it **easy to rebless**.
- Must have **fast notification** of failures to **responsible parties**.
- Failures must have **real repercusions** or they won't get addressed.
- Probably obvious: uBoone should try to exploit FNAL Jenkins.
- Likely `wire-cell-validate` will continue in its weird form but can be mined as a source of jobs for a "real" CI.