# Hveto Documentation

***Release 2.1.3***

**Josh Smith, Duncan Macleod, Alex Urban**

**Jun 16, 2023**

# CONTENTS

Hveto is a package designed for gravitational-wave detector characterisation and data quality. The algorithm identifies statistically significant correlations between transient noise events (triggers) in instrumental or environmental data channels, and those in the calibrated gravitational-wave strain output. Hveto is a key tool in reducing the noise background in searches for transient gravitational-wave signals, and was used, for example, during the detection of the binary black hole signal GW150914.

For full details about the algorithm, please refer to Smith et al. 2011 (Classical and Quantum Gravity).

To get started, simply import the core module:

```
import hveto
```

CHAPTER

ONE

# INSTALLATION

Hveto is best installed with conda:

```
conda install -c conda-forge hveto
```

but can also be installed with pip:

```
python -m pip install hveto
```

Note, users with `LIGO.ORG` credentials have access to a software container with a regularly-updated build of Hveto. For more information please refer to the LSCSoft Conda documentation.

CHAPTER

TWO

# CONTRIBUTING

All code should follow the Python Style Guide outlined in PEP 0008; users can use the flake8 package to check their code for style issues before submitting.

See the contributions guide for the recommended procedure for proposing additions/changes.

The Hveto project is hosted on GitHub:

- Issue tickets: https://github.com/gwdetchar/hveto/issues
- Source code: https://github.com/gwdetchar/hveto

## 2.1 License

Hveto is distributed under the GNU General Public License.

### 2.1.1 On the command-line

The main interface to the Hveto algorithm is the command-line executable *hveto*.

#### Basic instructions

For a full explanation of the available command-line arguments and options, you can run

```
$ hveto --help
usage: hveto [-h] [-V] -i IFO [-j NPROC] [-f CONFIG_FILE] [-p PRIMARY_CACHE]
             [-a AUXILIARY_CACHE] [-S ANALYSIS_SEGMENTS] [-w NSCAN]
             [-o OUTPUT_DIRECTORY] [-e EXTRA_TIMES]
             gpsstart gpsend

Run the HierarchichalVeto (hveto) algorithm over some data

positional arguments:
  gpsstart              GPS start time or datetime of analysis
  gpsend                GPS end time or datetime of analysis

optional arguments:
  -h, --help            show this help message and exit
  -V, --version         show program's version number and exit
  -i IFO, --ifo IFO     IFO prefix for this analysis, default: None
```

(continues on next page)

(continued from previous page)

```
  -j NPROC, --nproc NPROC
                        the number of processes to use when reading data,
                        default: 1
  -f CONFIG_FILE, --config-file CONFIG_FILE
                        path to hveto configuration file, can be given
                        multiple times (files read in order)
  -p PRIMARY_CACHE, --primary-cache PRIMARY_CACHE
                        path for cache containing primary channel files
  -a AUXILIARY_CACHE, --auxiliary-cache AUXILIARY_CACHE
                        path for cache containing auxiliary channel files,
                        files contained must be T050017-compliant with the
                        channel name as the leading name parts, e.g.
                        'L1-GDS_CALIB_STRAIN_<tag>-<start>-<duration>.<ext>'
                        for L1:GDS-CALIB_STRAIN triggers
  -S ANALYSIS_SEGMENTS, --analysis-segments ANALYSIS_SEGMENTS
                        path to file containing segments for the analysis flag
                        (name in data file must match analysis-flag in config
                        file)
  -w NSCAN, --omega-scans NSCAN
                        generate a workflow of omega scans for each round,
                        this will launch automatically to condor, requires the
                        gwdetchar package
  -e EXTRA_TIMES, --extra-times EXTRA_TIMES
                        extra times to add to the list of primary triggers to
                        include specific times that may not meet the
                        thresholds used to find triggers

Output options:
  -o OUTPUT_DIRECTORY, --output-directory OUTPUT_DIRECTORY
                        path of output directory, default: .
```

### Detailed instructions

A few of the command-line options require special formatting, which is detailed in this section.

#### `gpsstart` and `gpsend`

Each of the required GPS start and stop times can be given as GPS integers, or as date strings, e.g.

```
hveto 1135641617 1135728017 ...
```

will produce the same analysis as

```
hveto "Jan 1 2016" "Jan 2 2016" ...
```

**Note:** It is important to use quote marks when passing date strings on the command-line. This will ensure that the shell passes these to `python` as individual arguments.

#### -j/--nproc

Wall-clock processing time for *hveto* is dominated by two things:

- Reading event files for the auxiliary channels
- Determining the most-significant channel for a given round

Each of these can be sped up by using multiple CPUs on the host machine by supplying `--nproc X`, where `X` is any integer.

> **Warning:** Beware that using too many CPUs could overload the machine at runtime, or otherwise cause problems for yourself and other users.
>
> You can check how many processors are available with the following unix command:
>
> ```
> cat /proc/cpuinfo | grep processor | wc -l
> ```

#### -p/--primary-cache

This flag should receive a LAL-format cache file (see `glue.lal` for details). If needed, you can use `hveto-cache-events` for this purpose (see below).

#### -a/--auxiliary-cache

This flag requires a LAL-format cache file (similar to `--primary-cache`), but with a specific format:

- Each trigger file should contain events for only a single channel
- The cached trigger files should map to their channel as follows: if a channel is named `X1:SYS-SUBSYS_NAME`, each filename should start `X1-SYS_SUBSYS_NAME` according to the T050017 convention. Additional underscore-delimited components can be given, but will not be used to map to the channel.

If needed, `hveto-cache-events` can be used to create this cache (see below).

---

**Note:** If `channels` is not given in the `[auxiliary]` section of the configuration file, it will be populated from the unique list of channels parsed from the `--auxiliary-cache`.

---

### Caching trigger events

The `hveto-cache-events` utility can be used to save both gravitational-wave and auxiliary channel triggers:

```
$ hveto-cache-events --help
usage: hveto-cache-events [-h] [-V] -i IFO [-j NPROC] [-f CONFIG_FILE]
                          [-p PRIMARY_CACHE] [-a AUXILIARY_CACHE]
                          [-S ANALYSIS_SEGMENTS] [--append]
                          [-o OUTPUT_DIRECTORY]
                          gpsstart gpsend

Create a (temporary) cache of event triggers for analysis by `hveto`
```

(continues on next page)

(continued from previous page)

```
This method will apply the minimal SNR thresholds and any frequency cuts
as given in the configuration files

positional arguments:
  gpsstart              GPS start time or datetime of analysis
  gpsend                GPS end time or datetime of analysis

optional arguments:
  -h, --help            show this help message and exit
  -V, --version         show program's version number and exit
  -i IFO, --ifo IFO     IFO prefix for this analysis, default: None
  -j NPROC, --nproc NPROC
                        the number of processes to use when reading data,
                        default: 1
  -f CONFIG_FILE, --config-file CONFIG_FILE
                        path to hveto configuration file, can be given
                        multiple times (files read in order)
  -p PRIMARY_CACHE, --primary-cache PRIMARY_CACHE
                        path for cache containing primary channel files
  -a AUXILIARY_CACHE, --auxiliary-cache AUXILIARY_CACHE
                        path for cache containing auxiliary channel files,
                        files contained must be T050017-compliant with the
                        channel name as the leading name parts, e.g.
                        'L1-GDS_CALIB_STRAIN_<tag>-<start>-<duration>.<ext>'
                        for L1:GDS-CALIB_STRAIN triggers
  -S ANALYSIS_SEGMENTS, --analysis-segments ANALYSIS_SEGMENTS
                        path to LIGO_LW XML file containing segments for the
                        analysis flag (name in segment_definer table must
                        match analysis-flag in config file)
  --append              append to existing cached event files, otherwise,
                        start from scratch (default)

Output options:
  -o OUTPUT_DIRECTORY, --output-directory OUTPUT_DIRECTORY
                        path of output directory, default: .
```

### Tracing event triggers

The `hveto-trace` utility can be used to determine whether event triggers are vetoed by a given *hveto* run:

```
$ hveto-trace --help
usage: hveto-trace [-h] [-V] -t TRIGGER_TIME -d DIRECTORY [-v]

Check whether a specified trigger time was vetoed by an hveto analysis

optional arguments:
  -h, --help            show this help message and exit
  -V, --version         show program's version number and exit
  -t TRIGGER_TIME, --trigger-time TRIGGER_TIME
                        GPS time of the trigger
  -d DIRECTORY, --directory DIRECTORY
```

(continues on next page)

(continued from previous page)

```
                      path to the hveto output directory containing a
                      summary-stats.json file
  -v, --verbose       Log verbose output
```

### 2.1.2 hveto.config module

#### Configuration files for Hveto

#### How to write a configuration file

The *hveto* command-line executable can run without a configuration file, and will auto-select the primary channel, and attempt to auto-detect which auxiliary channels are available. It will also inset a minimal list of 'unsafe' channels so that the analysis isn't totally meaningless.

However, the way that things are auto-detected means that the resulting output isn't exactly reproducable after-the-fact in general. If someone adds or removes data for an auxiliary channel, a rerun will operate over a different channel lists, and confusion will reign.

So, it is highly recommended that you use a custom configuration file for your analyses. It could be that the configuration file never changes, but the very fact that it exists should mean that you are in much greater control over what the output will look like.

Each of the below sections lists the valid options and a description of what the value should be, and then an example of what that section might look like in the INI format:

#### [hveto]

| | |
|---|---|
| snr-thresholds | A comma-separated list of signal-to-noise ratio thresholds over which to test each auxiliary channel |
| time-windows | A comma-separated list of time windows (in seconds) over which to test each auxiliary channel |
| minimum-significance | The significance value below which to stop the analysis |

```
[hveto]
; comma-separated list of SNR thresholds (pass >=)
snr-thresholds = 7.75, 8, 8.5, 9, 10, 11, 12, 15, 20, 40, 100, 300
; comma-separated list of time windows
time-windows = .1, .2, .4, .8, 1
minimum-significance = 5
```

### [primary]

| | |
|---|---|
| `channel` | The name of the primary channel |
| `trigger-generator` | The name of the primary trigger generator |
| `snr-threshold` | The minimum threshold on signal-to-noise ratio for primary channel events to be included in the analysis |
| `frequency-range` | The `(low, high)` frequency range of interest for this analysis. Note that for CBC trigger generators, the `'template_duration'` column is used here. |
| `read-format` | The `format` name to use when reading files for this trigger generator |
| `read-columns` | The list of three column names equivalent to `time-like,frequency-like,snr-like` for this trigger generator |

```
[primary]
; use Omicron calibrated strain as primary
channel = %(IFO)s:GDS-CALIB_STRAIN
trigger-generator = omicron
; SNR threshold (pass >=)
snr-threshold = 6
; flow, fhigh
frequency-range = 0, 2048.
; format
read-format = ligolw
read-tablename = sngl_burst
; read-columns to read
read-columns = peak, peak_frequency, snr
```

**Note:** In this section the `%(IFO)s` interpolation variable has been used. This can be used throughout the configuration so that you only have to write one for all interferometers, with the correct two-character prefix being inserted automatically at run time.

**Note:** As well as the above, users can specify arguments to be used when automatically discovering triggers. Any option that begins `trigfind-` will be passed to the `trigfind.find_trigger_urls` function as a keyword argument.

Specifically, to specify the specific run associated with the `daily-cbc` event generator, you can give

```
[primary]
trigger-generator = daily-cbc
trigfind-run = bbh_gds
trigfind-filetag = 16SEC_CLUSTERED
read-format = ligolw
read-tablename = sngl_inspiral
read-columns = end,template_duration,snr
```

### [auxiliary]

| | |
|---|---|
| trigger-generator | The name of the auxiliary trigger generator |
| frequency-range | The (low, high) frequency range of interest |
| read-format | The format name to use when reading files for this trigger generator |
| read-columns | The list of three column names equivalent to time-like,frequency-like,snr-like for this trigger generator |
| channels | a tab-indented, line-delimited list of auxiliary channel names |

```
[auxiliary]
trigger-generator = omicron
; flow, fhigh
frequency-range = 0, 2048
; file format
read-format = ligolw
read-tablename = sngl_burst
; read-columns to read
read-columns = peak, peak_frequency, snr
; give tab-indented, line-separated list of channels
channels =
    %(IFO)s:ASC-AS_B_RF45_I_PIT_OUT_DQ
    %(IFO)s:ASC-AS_A_RF45_Q_PIT_OUT_DQ
    %(IFO)s:ASC-X_TR_B_NSUM_OUT_DQ
    %(IFO)s:ISI-ITMX_ST2_BLND_X_GS13_CUR_IN1_DQ
    %(IFO)s:ASC-REFL_A_RF9_I_YAW_OUT_DQ
    %(IFO)s:ASC-AS_A_RF45_Q_YAW_OUT_DQ
```

**Note:** As with the [primary] section, users can specify trigfind- arguments to customise trigger-file discovery.

### [segments]

| | |
|---|---|
| url | The URL of the segment database |
| analysis-flag | The name of the data-quality flag indicating analysable times |
| padding | The (pre, post) padding to apply to the analysis segments [note both pre and post operate forward in time, so to pad out at the start of a segment (or in at the end), use a negative number] |
| veto-definer-file | The path of a veto-definer file to apply before analysis (can be a remote URL) |
| veto-definer-categories | Comma-separated list of category integers to apply, defaults to all flags in veto-definer file |

```
[segments]
url = https://segments.ligo.org
; require full observation mode
analysis-flag = %(IFO)s:DMT-ANALYSIS_READY:1
; no padding by default
padding = 0, 0
```

### [safety]

| | |
|---|---|
| unsafe-channels | The list of unsafe channels (tab-indented, line-delimited) |

```
[safety]
unsafe-channels =
    %(IFO)s:GDS-CALIB_STRAIN
    %(IFO)s:CAL-DELTAL_EXTERNAL_DQ
    %(IFO)s:OAF-CAL_DARM_DQ
    %(IFO)s:OMC-DCPD_SUM_OUT_DQ
```

### Module API

**class** hveto.config.**HvetoConfigParser**(*ifo=None*, *defaults={}*, ***kwargs*)

Bases: ConfigParser

**Attributes**

**converters**

### Methods

| | |
|---|---|
| add_section(section) | Create a new section in the configuration. |
| clear() | |
| get(section, option, *[, raw, vars, fallback]) | Get an option value for a given section. |
| getint(section, option, *[, raw, vars, fallback]) | |
| has_option(section, option) | Check for the existence of a given option in a given section. If the specified `section' is None or an empty string, DEFAULT is assumed. If the specified `section' does not exist, returns False. |
| has_section(section) | Indicate whether the named section is present in the configuration. |
| items([section, raw, vars]) | Return a list of (name, value) tuples for each option in a section. |
| keys() | |
| options(section) | Return a list of option names for the given section name. |
| pop(k[,d]) | If key is not found, d is returned if given, otherwise KeyError is raised. |
| popitem() | Remove a section from the parser and return it as a (section_name, section_proxy) tuple. |
| *read*(filenames) | Read and parse a filename or an iterable of filenames. |
| read_dict(dictionary[, source]) | Read configuration from a dictionary. |
| read_file(f[, source]) | Like read() but the argument must be a file-like object. |
| read_string(string[, source]) | Read configuration from a given string. |
| readfp(fp[, filename]) | Deprecated, use read_file instead. |
| remove_option(section, option) | Remove an option. |
| remove_section(section) | Remove a file section. |
| sections() | Return a list of section names, excluding [DEFAULT] |
| set(section, option[, value]) | Set an option. |
| setdefault(k[,d]) | |
| update([E, ]**F) | If E present and has a .keys() method, does: for k in E: D[k] = E[k] If E present and lacks .keys() method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k, v in F.items(): D[k] = v |
| values() | |
| write(fp[, space_around_delimiters]) | Write an .ini-format representation of the configuration state. |

| | |
|---|---|
| **defaults** | |
| **getboolean** | |
| **getfloat** | |
| **getfloats** | |
| **getparams** | |
| **optionxform** | |
| **set_hveto_defaults** | |

```
HVETO_DEFAULTS = {'auxiliary': {'frequency-range': (30, 2048),
'trigger-generator': 'Omicron'}, 'hveto': {'minimum-significance': 5,
'snr-thresholds': (8, 10, 12, 15, 20, 50, 100, 300), 'time-windows': (0.1, 0.2,
0.5, 1, 2, 5)}, 'primary': {'channel': '%(IFO)s:GDS-CALIB_STRAIN',
'frequency-range': (30, 2048), 'snr-threshold': 8, 'trigger-generator':
'Omicron'}, 'safety': {'unsafe-channels': ['%(IFO)s:GDS-CALIB_STRAIN',
'%(IFO)s:LDAS-STRAIN', '%(IFO)s:CAL-DELTAL_EXTERNAL_DQ', '%(IFO)s:DER_DATA_H',
'%(IFO)s:h_4096Hz', '%(IFO)s:h_16384Hz']}, 'segments': {'analysis-flag':
'%(IFO)s:DMT-ANALYSIS_READY:1', 'padding': (0, 0), 'url':
'https://segments.ligo.org'}}
```

**getfloats**(*section*, *option*)

**getparams**(*section*, *prefix*)

**read**(*filenames*)

Read and parse a filename or an iterable of filenames.

Files that cannot be opened are silently ignored; this is designed so that you can specify an iterable of potential configuration file locations (e.g. current directory, user's home directory, systemwide directory), and all existing configuration files in the iterable will be read. A single filename may also be given.

Return list of successfully read files.

**set_hveto_defaults**()

hveto.config.**comma_separated_floats**(*string*)

### 2.1.3 API

#### hveto package

#### Subpackages

#### hveto.cli package

#### Subpackages

#### hveto.cli.tests package

#### Submodules

### hveto.cli.tests.test_trace module

Tests for *hveto.cli.trace*

hveto.cli.tests.test_trace.**test_main**(*caplog*, *tmpdir*)

hveto.cli.tests.test_trace.**test_main_empty_input**(*caplog*, *tmpdir*)

hveto.cli.tests.test_trace.**test_main_no_input**(*caplog*, *tmpdir*)

### Module contents

Unit tests for hveto.cli

### Submodules

### hveto.cli.cache_events module

Create a (temporary) cache of event triggers for analysis by *hveto*

This method will apply the minimal SNR thresholds and any frequency cuts as given in the configuration files

hveto.cli.cache_events.**create_parser**()

Create a command-line parser for this entry point

hveto.cli.cache_events.**main**(*args=None*)

Run the cache_events tool

### hveto.cli.trace module

Check whether a specified trigger time was vetoed by an hveto analysis

hveto.cli.trace.**create_parser**()

Create a command-line parser for this entry point

hveto.cli.trace.**main**(*args=None*)

Run the hveto-trace command-line tool

### Module contents

Command-line interfaces for hveto

The modules contained within this subpackage are all rendered as console_script entry points.

### hveto.tests package

#### Submodules

#### hveto.tests.test_core module

Tests for *hveto.core*

hveto.tests.test_core.**test_create_round**()
Test creation of a *hveto.core.HvetoRound* object

hveto.tests.test_core.**test_significance**(*n*, *mu*, *sig*)
Test *hveto.core.significance()*

#### hveto.tests.test_html module

Tests for *hveto.html*

hveto.tests.test_html.**test_banner**()

hveto.tests.test_html.**test_bold_param**(*args*, *kwargs*, *result*)

hveto.tests.test_html.**test_navbar**()

hveto.tests.test_html.**test_write_hveto_page**(*tmpdir*)

hveto.tests.test_html.**test_write_null_page**(*tmpdir*)

#### hveto.tests.test_plot module

Tests for *hveto.plot*

hveto.tests.test_plot.**test_drop_plot**(*num*, *tmpdir*)

#### hveto.tests.test_segments module

Tests for *hveto.segments*

hveto.tests.test_segments.**test_query**(*dqflag*)

hveto.tests.test_segments.**test_read_veto_definer_file**(*dqflag*, *tmpdir*)

hveto.tests.test_segments.**test_write_segments_ascii**(*ncol*, *tmpdir*)

hveto.tests.test_segments.**test_write_segments_ascii_failure**()

### hveto.tests.test_triggers module

Tests for *hveto.triggers*

hveto.tests.test_triggers.**test_aux_channels_from_cache**()

hveto.tests.test_triggers.**test_get_triggers**()

### hveto.tests.test_utils module

Tests for *hveto.utils*

hveto.tests.test_utils.**test_channel_groups**(*n*, *out*)

hveto.tests.test_utils.**test_primary_vetoed**(*mock_table*)

hveto.tests.test_utils.**test_primary_vetoed_type**()

hveto.tests.test_utils.**test_write_lal_cache**(*tmpdir*)

### Module contents

Test for Heto

### Submodules

### hveto.const module

Constants for *hveto.utils*

hveto.const.**get_hvetopath**(*gpstime*)

Returns the path of hveto output files

Given a gpstime, the path of the folder containing hveto trigger files is returned

**Parameters**

**gpstime**
[str or float] start time of the day for this analysis

**Returns**

_____

**path**
[str] path to the hveto output file on the local filesystem

**Example**

_____

**>>> from hveto.const import get_hvetopath**
**>>> get_hvetopath(1257811218)**
**'/home/detchar/public_html/hveto/day/20191115/latest'**

### hveto.core module

Core of the HierarchichalVeto algorithm

class hveto.core.**HvetoRound**(*round*, *primary*, *segments=None*, *vetoes=None*, *plots=[]*, *files={}*, *rank=None*)

Bases: object

**Attributes**

**cum_deadtime**
**cum_efficiency**
**deadtime**
**efficiency**
**files**
**livetime**
**n**
**plots**
**primary**
**rank**
**scans**
**segments**
**use_percentage**
**vetoes**
**winner**

cum_deadtime

cum_efficiency

property deadtime

efficiency

files

property livetime

n

plots

primary

rank

scans

segments

use_percentage

vetoes

winner

class hveto.core.**HvetoWinner**(*name=None*, *significance=None*, *snr=None*, *window=None*, *segments=None*, *events=None*, *ncoinc=0*, *mu=None*)

Bases: object

**Attributes**

**events**
**mu**
**name**
**ncoinc**
**segments**
**significance**
**snr**
**window**

### Methods

| get_segments | |
|---|---|

**events**

**get_segments**(*times*)

**mu**

**name**

**ncoinc**

**segments**

**significance**

**snr**

**window**

hveto.core.**coinc_significance**(*a*, *b*, *dt*, *livetime*)

Calculate the significance of coincidences between two time arrays

**Parameters**

**a**
[numpy.ndarray] first array

**b**
[numpy.ndarray] second array

**dt**
[float] coincidence window

**livetime**
[float] the livetime of the analysis

**Returns**

**coincs**
[numpy.ndarray] the indices of array a that were coincident with an entry in b

**significance**
[float] the Poisson significance of the number of coincidences found as compared to the number expected by random chance

hveto.core.**find_all_coincidences**(*triggers*, *channel*, *snrs*, *windows*)

Find the number of coincs between each auxiliary channel and the primary

**Parameters**

**primary**
[numpy.ndarray] an array of times for the primary channel

**auxiliary**
[numpy.recarray] an array of triggers for a set of auxiliary channels

**snrs**
[list of float] the SNR thresholds to use

**window**
[list of float] the time windows to use

hveto.core.**find_coincidences**(*a*, *b*, *dt=1*)

Find the coincidences between values in two numpy arrays

**Parameters**

**a**
[numpy.ndarray] first array

**b**
[numpy.ndarray] second array

**dt**
[float, optional] coincidence window

**Returns**

**coinc**
[numpy.ndarray] the indices of all items in a within [-dt/2., +dt/2.) of an item in b

hveto.core.**find_max_significance**(*primary*, *auxiliary*, *channel*, *snrs*, *windows*, *livetime*)

Find the maximum Hveto significance for this primary-auxiliary pair

**Parameters**

**primary**
[numpy.recarray] record array of data from the primary channel

**auxiliary**
[numpy.recarray] record array from the auxiliary channel

**snrs**
[list of float] the SNR thresholds to use

**window**
[list of float] the time windows to use

**Returns**

**winner**
[*HvetoWinner*] the parameters and segments generated by the (snr, dt) with the highest significance

hveto.core.**significance**(*n*, *mu*)

Calculate the significance of n coincidences, when mu were expected

**Parameters**

**n**
[int] the number of coincidences found

**mu**
[float] the number of coincidences expected from a Poisson process

hveto.core.**veto**(*table*, *segmentlist*)

Remove events from a table based on a segmentlist

A time `t` will be vetoed if `start <= t <= end` for any veto segment in the list.

**Parameters**

**table**
[numpy.recarray] the table of event triggers to veto

**segmentlist**
[segmentlist] the list of veto segments to use

**Returns**

**keep**
[numpy.recarray] the reduced table of events that were not coincident with any segments

hveto.core.**veto_all**(*auxiliary*, *segmentlist*)

Remove events from all auxiliary channel tables based on a segmentlist

**Parameters**

**auxiliary**
[dict of numpy.recarray] a dict of event arrays to veto

**segmentlist**
[segmentlist] the list of veto segments to use

**Returns**

**survivors**
[dict of numpy.recarray] a dict of the reduced arrays of events for each input channel

**See also:**

**core.veto**
for details on the veto algorithm itself

### hveto.html module

HTML utilities for hveto

hveto.html.**banner**(*ifo*, *start*, *end*)

Initialise a new markup banner

**Parameters**

**ifo**
[str] the interferometer prefix

**start**
[int] the GPS start time of the analysis

**end**
[int] the GPS end time of the analysis

**Returns**

**page**
[markup.page] the structured markup to open an HTML document

hveto.html.**bold_param**(*key*, *value*, ***attrs*)

Write a (key, value) pair in HTML as key: value

**Parameters**

**key**
[str] the element to be rendered in bold

**value**
[str] the explanation/description of the key

****attrs**
HTML attributes for the containing  tag

**Returns**

**html**
[str]

hveto.html.**navbar**(*ifo*, *gpstime*, *winners=[]*)

Initialise a new markup.page

**Parameters**

**ifo**
[str] the interferometer prefix

**gpstime**
[float] the central GPS time of the analysis

**winners**
[list] list of round winners for navbar table of contents

**Returns**

**page**
[markup.page] the structured markup to open an HTML document

hveto.html.**wrap_html**(*func*)

Decorator to wrap a function with init_page and close_page calls

This allows inner HTML methods to be written with minimal arguments and content, hopefully making things simpler

hveto.html.**write_about_page**(*configfile*, *prog=None*)

Write a page explaining how an hveto analysis was completed

**Parameters**

**ifo**
[str] the prefix of the interferometer used in this analysis

**start**
[int] the GPS start time of the analysis

**end**
[int] the GPS end time of the analysis

**configfile**
[str] the path of the configuration file to embed

**prog**
[str, optional] name of the program which produced this page, defaults to the script run on the command-line

**outdir**
[str, optional] the output directory for the HTML

**Returns**

**index**
[str] the path of the HTML written for this analysis

hveto.html.**write_hveto_page**(*rounds*, *plots*, *context='default'*)

Write the Hveto results to HTML

**Parameters**

**ifo**
[str] the prefix of the interferometer used in this analysis

**start**
[int] the GPS start time of the analysis

**end**
[int] the GPS end time of the analysis

**rounds**
[list of HvetoRound] the rounds produced by this analysis

**plots**
[list of str] the list of summary plots

**outdir**
[str, optional] the output directory for the HTML

**winners**
[list of str, optional] list of channels that won each round

**context**
[str, optional] the bootstrap context class for this result, see the bootstrap docs for more details

**Returns**

**index**
[str] the path of the HTML written for this analysis

hveto.html.**write_null_page**(*reason*, *context='info'*)

Write the Hveto results to HTML

**Parameters**

**ifo**
[str] the prefix of the interferometer used in this analysis

**start**
[int] the GPS start time of the analysis

**end**
[int] the GPS end time of the analysis

**reason**
[str] the explanation for this null result

**context**
[str, optional] the bootstrap context class for this result, see the bootstrap docs for more details

**outdir**
[str, optional] the output directory for the HTML

**Returns**

**index**
[str] the path of the HTML written for this analysis

hveto.html.**write_round**(*round_*, *context*)

Write the HTML summary for a specific round

**Parameters**

**round_**
[HvetoRound] the analysis round object

**context**
[str] context for bootstrap objects, default: info

**Returns**

**page**
[page] the formatted HTML for this round

hveto.html.**write_summary**(*rounds*, *plots=[]*, *header='Summary'*, *plotsperrow=4*, *tableclass='table table-sm table-hover'*)

Write the Hveto analysis summary HTML

**Parameters**

**rounds**
[list of HvetoRound] the list of round objects produced by this analysis

**plots**
[list of str, optional] the list of summary plots to display underneath the summary table

**header**
[str, optional] the text for the section header (<h2>)

**plotsperrow**
[int, optional] the number of plots to display in each row

**tableclass**
[str, optional] the class for the summary <table>

**Returns**

**page**
[page] the formatted markup object containing the analysis summary table, and images

### hveto.plot module

Plotting routines for hveto

hveto.plot.**before_after_histogram**(*outfile*, *x*, *y*, *label1='Before'*, *label2='After'*, *bins=100*, *histtype='stepfilled'*, *range=None*, *figsize=[9, 6]*, ***kwargs*)

Plot a histogram of SNR for two event distributions

hveto.plot.**get_column_label**(*column*)

hveto.plot.**hveto_roc**(*outfile*, *rounds*, *figsize=[9, 6]*, *constants=[1, 5, 10, 20]*, ***kwargs*)

hveto.plot.**significance_drop**(*outfile*, *old*, *new*, *show_channel_names=None*, ***kwargs*)

Plot the signifiance drop for each channel

hveto.plot.**veto_scatter**(*outfile*, *a*, *b*, *label1='All'*, *label2='Vetoed'*, *x='time'*, *y='snr'*, *color=None*, *clim=None*, *clabel=None*, *cmap=None*, *clog=True*, *figsize=[9, 6]*, ***kwargs*)

Plot an x-y scatter of all/vetoed events

### hveto.segments module

Segment utilities for hveto

hveto.segments.**integer_segments**(*f*)

hveto.segments.**query**(*flag*, *start*, *end*, *url='https://segments.ligo.org'*)

Query a segment database for active segments associated with a flag

hveto.segments.**read_veto_definer_file**(*vetofile*, *start=None*, *end=None*, *ifo=None*)

Read a veto definer file, downloading it if necessary

hveto.segments.**write_ascii**(*outfile*, *segmentlist*, *ncol=4*)

### hveto.triggers module

Trigger I/O utilities for hveto

hveto.triggers.**find_auxiliary_channels**(*etg*, *gps='*'*, *ifo='*'*, *cache=None*)

Find all auxiliary channels processed by a given ETG

If `cache=None` is given (default), the channels are parsed from the ETG archive under `/home/detchar/triggers`. Otherwise, the channel names are parsed from the files in the `cache`, assuming they follow the T050017 file-naming convention.

**Parameters**

**etg**
[`str`] name of the trigger generator

**gps**
[`int`, `tuple`, optional] GPS reference time, or pair of times indicating [start, end), at which to find channels

**ifo**
[`str`, optional] interferometer prefix for which to find channels

**cache**
[list of str, optional] list of files from which to parse channels

**Returns**

**channels**
[list of str] the names of all available auxiliary channels

hveto.triggers.**find_trigger_files**(*channel*, *etg*, *segments*, ***kwargs*)

Find trigger files for a given channel and ETG

**Parameters**

**channel**
[str] name of channel to find

**etg**
[str] name of event trigger generator to find

**segments**
[segmentlist] list of segments to find

****kwargs**
all other keyword arguments are passed to gwtrigfind.find_trigger_urls

**Returns**

**cache**
[list of str] cache of trigger file paths

**See also:**

**gwtrigfind.find_trigger_urls**
for details on file discovery

hveto.triggers.**get_triggers**(*channel*, *etg*, *segments*, *cache=None*, *snr=None*, *frange=None*, *raw=False*, *extra_times=None*, *trigfind_kwargs={}*, ***read_kwargs*)

Get triggers for the given channel

### hveto.utils module

General utilities for hveto

hveto.utils.**channel_groups**(*channellist*, *ngroups*)

Separate a list of channels into a number of equally-sized groups

**Parameters**

**channellist**
[list] large list to separate into chunks

**ngroups**
[int] number of output groups

**Returns**

**iterator**
[iterator list of list] a generator sequence yielding a sub-list on each iteration

hveto.utils.**primary_vetoed**(*starttime=None*, *hveto_path=None*, *snr=6.0*, *significance=5.0*)

Catalogue all vetoed primary triggers from a given analysis

This utility queries the output of an hveto analysis for the triggers vetoed from its primary channel over all rounds (up to thresholds on signal-to-noise ratio and round significance).

**Parameters**

**starttime**
[str or float] start GPS time for this analysis

**hveto_path**
['str'] path of the hveto files directory, not required if `starttime` is given

**snr**
[float, optional] signal-to-noise ratio threshold on triggers, default: 6.0

**significance**
[float, optional] hveto significance threshold on auxiliary channels, default: 5.0

**Returns**

**catalogue**
[EventTable] a tabular catalogue of primary triggers vetoed in the hveto run

hveto.utils.**write_lal_cache**(*target*, *paths*)

### Module contents

The HierarchichalVeto algorithm

### hveto

# PYTHON MODULE INDEX

## h

# INDEX

## U



## V



## W