

# HasKAL Reference Manual

Edition 0.1 alpha

March 8, 2016

## Contents

<b>1</b>	<b>Monitor Tools</b>	<b>2</b>
1.1	RayleighMonitor . . . . .	2
1.1.1	<b>Introduction</b> . . . . .	2
1.1.2	<b>Function:</b> rayleighMonWaveData . . . . .	2
1.1.3	<b>Example:</b> rayleighMon . . . . .	2
1.2	StudentRayleighMonitor . . . . .	4
1.2.1	<b>Introduction</b> . . . . .	4
1.2.2	<b>Function:</b> studentRayleighMonWaveData . . . . .	4
1.2.3	<b>Example:</b> studentRayleighMon . . . . .	5
1.3	RMSMonitor . . . . .	6
1.3.1	<b>Introduction</b> . . . . .	6
1.3.2	<b>Function:</b> rmsMon . . . . .	6
1.3.3	<b>Function:</b> rmsMonWaveData . . . . .	7
1.3.4	<b>Example:</b> rmsMon . . . . .	7

# 1 Monitor Tools

## 1.1 RayleighMonitor

### 1.1.1 Introduction

RayleighMonitor is a tool for calculated a quantile value of normalized spectrum of  $x(t)$ . The deviation of the calculated quantiles from the expected one in Gaussian noise case shows deviation of the detector noise from Gaussian distribution.

Normalized spectrogram,  $w(t, f)$ , of input signal,  $x(t)$ , is calculated

$$w(t_i, f_j) = \frac{| \text{STFT}[x(t)] |}{S_0(f)},$$

where  $1 \leq i \leq N$ ,  $1 \leq j \leq M$  and  $S_0(f)$  is a normalization factor. Normalization factor can be estimated

$$S_0(f) = | \text{FFT}[x(t)] |.$$

P-quantile value of input signal is calculated from normalized spectrogram as the function of time and frequency,  $Q(P; f_l)$  where  $1 \leq l \leq M/m$ ,  $m(l-1)-1 \leq j \leq ml$  and  $m = df/df_{\text{fft}} = df \, dt_{\text{fft}}$

### 1.1.2 Function: rayleighMonWaveData

```
rayleighMonWaveData p secfft df x0 xt
```

This function compute p-quantile value,  $Q(p; f)$ , of the input signal,  $x(t)$ , as the function of frequency,  $f$ . The arguments are:

- **p**: Input. The list of dimensionless p-values ( $0 \leq p \leq 1$ ).
- **secfft**: Input. The data length for short time Fourier transform in seconds.
- **df**: Input. The frequency resolution of  $Q(p; f)$  in Hertz
- **x0**: Input. The time series signal for estimating averaged spectrum
- **xt**: Input. The time series for calculating quantile value  $Q(p; f)$
- **q**: Output. The quantile value of input signal  $Q(p; f)$ .

### 1.1.3 Example: rayleighMon

This program calculates the  $Q(p; f)$  of the input signal.

**Typical usage:** rayleighMon param.conf file.lst

```
import Data.Maybe (catMaybes)
import System.Environment (getArgs)

import HaskAL.DetectorUtils.Detector (Detector(..))
import HaskAL.FrameUtils.Function (readFrameWaveData')
import HaskAL.Misc.ConfFile (readFileList, readConfFile)
import HaskAL.MonitorUtils.RayleighMon.RayleighMon (rayleighMonWaveData)
import HaskAL.PlotUtils.HR00T.PlotGraph
import HaskAL.WaveUtils.Data (WaveData(..))
import HaskAL.WaveUtils.Function (catWaveData)

main = do
  {-- arg check --}
  args <- getArgs
  (conf, lst) <- case length args of
    2 -> return (args!!0, args!!1)
    _ -> error "Usage: rayleighMon conffile filelist"
```

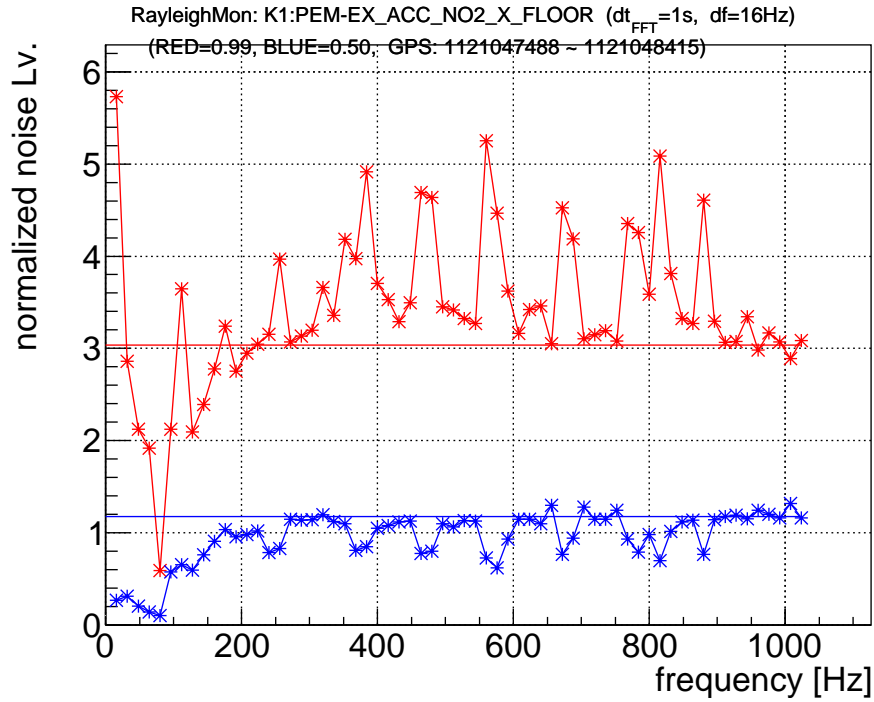


Figure 1: sample plot of rayleighMonitor

```
{-- read param --}
filelist <- readFileList lst
([ch, dtfft, df], [qs]) <- readConfFile conf ["channel", "dtfft", "df"] ["quantile"]

{-- read data --}
mbWd <- mapM (readFrameWaveData' KAGRA ch) filelist
let wd = case catMaybes mbWd of
    [] -> error "Can't find file."
    xs -> catWaveData xs

{-- main --}
let result = rayleighMonWaveData (map read qs) (read dtfft) (read df) wd wd
lineType = concat $ replicate (length qs) [LinePoint, Line]
colors = concatMap (replicate 2) [RED, BLUE, PINK, GREEN, CYAN, YELLOW, BLACK]
title = ch ++ ": " ++ (show . fst . startGPSTime $ wd) ++ " ~ " ++ (show . fst . stopGPSTime $ wd)
oPlotV Linear lineType 1 colors ("frequency [Hz]", "normalized noise Lv.") 0.05
title "X11" ((0,0),(0,0)) $ concatMap \(x,y) -> [x,y] result
```

#### Param file format: param.conf

```
channel: X1:H0GE-XX # channel name
quantile: 0.5 0.95 # list of dimensionless p-value
dtfft: 1 # data length for STFT in seconds
df: 16 # frequency resolution of Q(p;~f) in Hertz
```

#### List file format: file.lst

```
/path/to/framefile/a.gwf
/path/to/framefile/b.gwf
```

contact person: Takahiro Yamamoto (yamamoto@yukimura.hep.osaka-cu.ac.jp)

## 1.2 StudentRayleighMonitor

### 1.2.1 Introduction

StudentRayleighMonitor is a tool for investigating stationarity non-Gaussianity of input signal  $x(t)$  by assuming detector noise distributed the Student-t distribution. In this assumption, non-Gaussianity is represented by only one parameter,  $\nu$ , which shows weight of tail of the distribution. Non-Gaussianity,  $\nu$ , is computed as the function of time,  $t$ , and frequency,  $f$ , from normalized spectrum of  $x(t)$ .

Normalized spectrogram,  $w(t, f)$ , of input signal,  $x(t)$ , is calculated

$$w(t_i, f_j) = \frac{| \text{STFT}[x(t)] |}{S_0(f)},$$

where  $1 \leq i \leq N$ ,  $1 \leq j \leq M$  and  $S_0(f)$  is a normalization factor. Normalization factor can be estimated

$$S_0(f) = | \text{FFT}[x(t)] |.$$

P-quantile value of input signal is calculated from normalized spectrogram as the function of time and frequency,  $Q_P(t_k, f_l)$  where  $1 \leq k \leq N/n$ ,  $1 \leq l \leq M/m$ ,  $n(k-1) + 1 \leq i \leq nk$ ,  $m(l-1) + 1 \leq j \leq ml$ ,  $n = dt/dt_{\text{fft}}$  and  $m = df/df_{\text{fft}} = df/dt_{\text{fft}}$

On the other hand, theoretical quantile value in the Student-t noise case can be described

$$Q_{\text{sr}}(\sigma, \nu; P) = \sigma \sqrt{\frac{\nu(1 - (1 - P)^{2/\nu})}{(1 - P)^{2/\nu}}}$$

Degree of non-Gaussianity  $\nu$  is calculated from P-quantile value of data and theoretical quantile value.

$$\nu(t_k, f_l) = \arg \min_{\nu} |Q_{P=P_0}(t_k, f_l) - Q_{\text{sr}}(\sigma, \nu; P = P_0)|$$

### 1.2.2 Function: studentRayleighMonWaveData

`studentRayleighMonWaveData p secfft chunk dt df x0 xt`

This function compute the non-Gaussianity,  $\nu$ , of the input signal,  $x(t)$ , as the function of time,  $t$ , and frequency,  $f$ . The arguments are:

- **p**: Input. The dimensionless p-value ( $0 \leq p \leq 1$ ).
- **secfft**: Input. The data length for short time Fourier transform in seconds.
- **chunk**: Input. The data length for estimating  $\nu(f)$  in seconds. (**secfft**  $\leq$  **chunk**)
- **dt**: Input. The time resolution of  $\nu(t, f)$  in seconds.
- **df**: Input. The frequency resolution of  $\nu(t, f)$  in Hertz
- **x0**: Input. The time series signal for estimating averaged spectrum
- **xt**: Input. The time series for estimating  $\nu(t, f)$
- **nu**: Output. The dimensionless non-Gaussian parameter  $\nu(t, f)$ .

### 1.2.3 Example: studentRayleighMon

This program calculates the  $\nu(t, f)$  of the input signals.

**Typical usage:** studentRayleighMon param.conf file.lst

```
import Data.Maybe (catMaybes)
import System.Environment (getArgs)

import HaskAL.DetectorUtils.Detector (Detector(..))
import HaskAL.FrameUtils.Function (readFrameWaveData')
import HaskAL.Misc.ConfFile (readFileList, readConfFile)
import HaskAL.MonitorUtils.SRMon.StudentRayleighMon (studentRayleighMonWaveData)
import HaskAL.PlotUtils.HROOT.PlotGraph3D
import HaskAL.WaveUtils.Data (WaveData(..))
import HaskAL.WaveUtils.Function (catWaveData)

main = do
  {-- arg check --}
  args <- getArgs
  (conf, lst) <- case length args of
    2 -> return (args!!0, args!!1)
    _ -> error "Usage: rayleighMon conffile filelist"

  {-- read param --}
  filelist <- readFileList lst
  ([ch, q, dtfft, dt, lap, df], _) <- readConfFile conf ["channel", "quantile", "dtfft"
    , "dt", "overlap", "'df"] []

  {-- read data --}
  mbWd <- mapM (readFrameWaveData' KAGRA ch) filelist
  let wd = case catMaybes mbWd of
    [] -> error "Can't find data"
    xs -> catWaveData xs

  {-- main --}
  let result = studentRayleighMonWaveData (read q) (read dtfft)
    (read dt) (read dt - read lap) (read df) wd wd
    title = ch ++ ": " ++ (show . fst . startGPSTime $ wd)
      ++ " ~ " ++ (show . fst . stopGPSTime $ wd)
  histogram2dM Linear COLZ ("time [s]", "frequency [Hz]", "nu")
    title "X11" ((0,0),(0,0)) $ result
```

**Param file format:** param.conf

```
channel: X1:H0GE-XX # channel name
quantile: 0.95      # dimensionless p-value
dtfft: 1           # data length for STFT in seconds
dt: 128            # time resolution of \nu(t,f) in seconds
overlap: 124       # data overlap in seconds
df: 16             # frequency resolution of \nu(t,f) in Hertz
```

**List file format:** file.lst

```
/path/to/framefile/a.gwf
/path/to/framefile/b.gwf
```

contact person: Takahiro Yamamoto (yamamoto@yukimura.hep.osaka-cu.ac.jp)

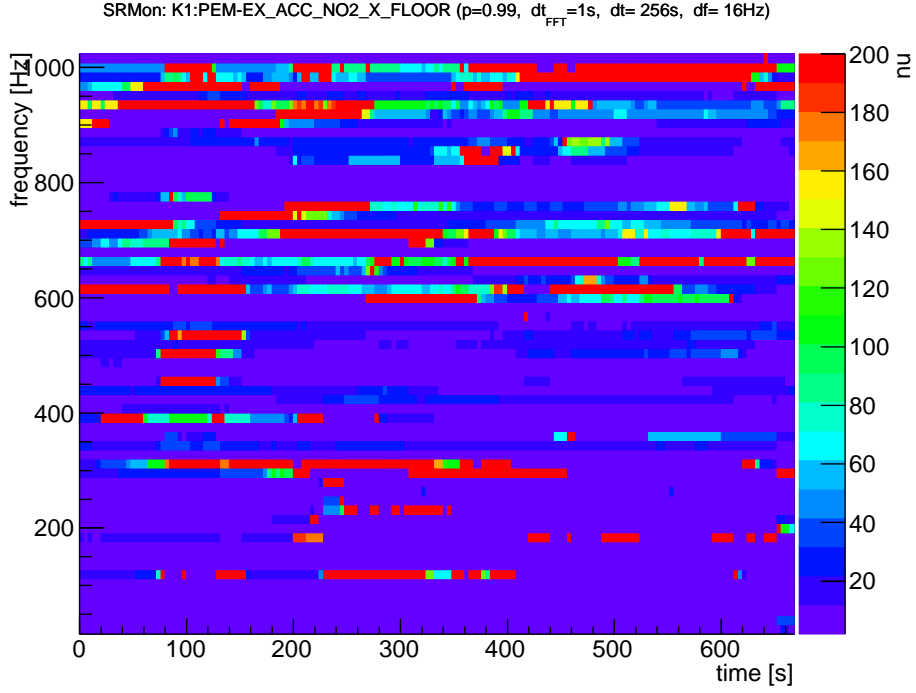


Figure 2: sample plot of studentRayleighMonitor

### 1.3 RMSMonitor

#### 1.3.1 Introduction

The RMSMonitor is a tool to investigate the non-stationary behavior of the input time series  $x(t)$  by calculating the band-limited root-mean square (for short RMS) values.

The RMS values,  $\rho_{RMS}(t)$ , of input time series,  $x(t)$ , are calculated as

$$\rho_{RMS}(t) = \sqrt{\int_{f_1}^{f_2} |\tilde{x}(f)|^2 df} \quad (1)$$

where  $f_1$  and  $f_2$  are the frequency band and  $\tilde{x}(f)$  is the input frequency domain signal calculated by FFT as,

$$\tilde{x}(f) = \text{FFT}[x(t)].$$

#### 1.3.2 Function: rmsMon

`rmsMon nmon fs ys freq`

This function compute the RMS,  $\rho_{RMS}(t)$ , of the input time series  $x(t)$ . The time series  $x(t)$  are divided into small chunks. The RMS value is calculated from each chunk. The number of chunks are calculated by  $N_{ys}/\text{nmon}$ , where  $N_{ys}$  is the number of samples of input time series.

The arguments are:

- **nmon**: [Input] The number of samples in one chunk.
- **fs**: [Input] The sampling frequency of input time series.
- **ys**: [Input] The input time series.
- **freq**: [Input] The frequency bands  $[f_1 : f_2]$  described in Eq. (1).
- **RMS**: [Output] The calculated RMS values.

### 1.3.3 Function: rmsMonWaveData

`rmsMonWaveData nmon freq wd`

This function compute the RMS,  $\rho_{RMS}(t)$ , of the input time series  $x(t)$ . The difference between `rmsMon` and `rmsMonWaveData` is the type of input time series. `rmsMonWaveData` uses `WaveData` type instead of the time series  $x(t)$ . The other arguments are same as `rmsMon`. The arguments are:

- `nmon`: [Input] The number of samples in one chunk.
- `wd`: [Input] The input data (`WaveData` type).
- `freq`: [Input] The frequency bands  $[f_1 : f_2]$  described in Eq. (1).
- `RMS`: [Output] The calculated RMS values.

### 1.3.4 Example: rmsMon

This program calculates the  $\rho_{RMS}$  of the input time series. Examples of `RMSMon` are described in Fig. 3 and 4.

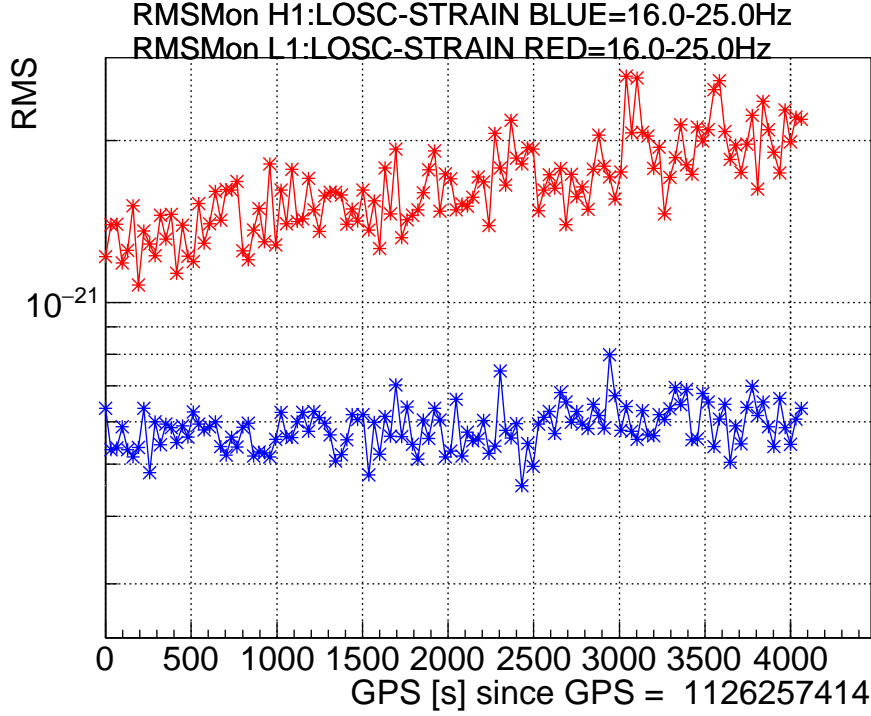


Figure 3: Sample plot of `RMSMon`. The duration of one chunk is 32s. The frequency bands is  $[16:25]\text{Hz}$

contact person: Hirotaka Yuzurihara ([yuzurihara@yukimura.hep.osaka-cu.ac.jp](mailto:yuzurihara@yukimura.hep.osaka-cu.ac.jp))

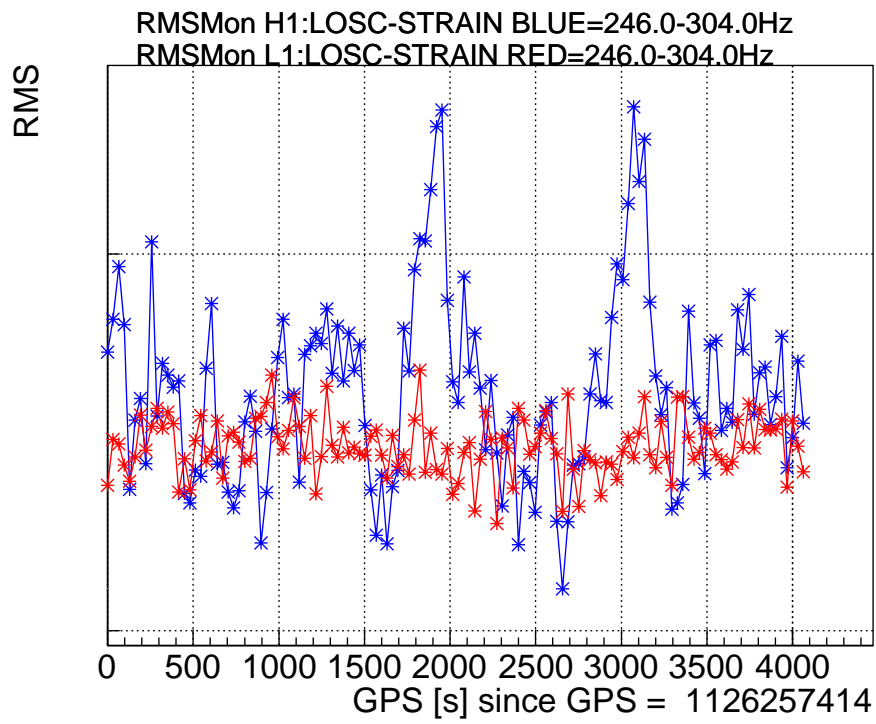


Figure 4: Sample plot of RMSMon. The duration of one chunk is 32s. The frequency bands is [246:304]Hz