Infrasound Data Analysis

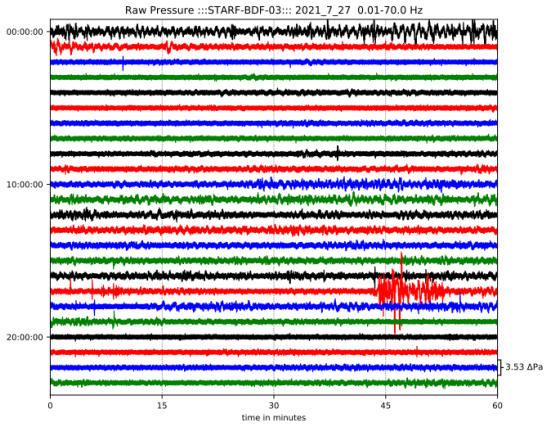
a brief introduction

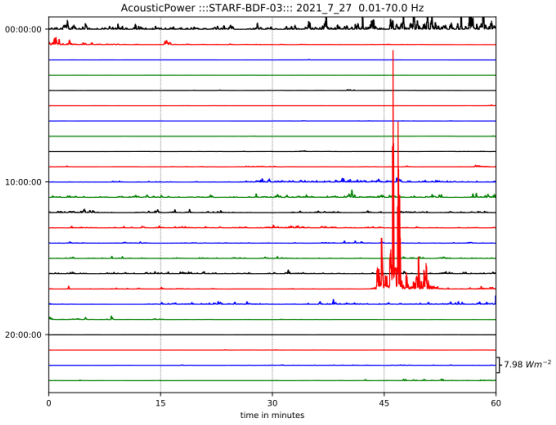
Ian Robinson [*] Nathan Robinson [†] Saul Robinson [‡]

July 20, 2022

revision 2.1

ian@schoolphysicsprojects.org





[*] Association for Science Education
[†] Durham University
[‡] schoolphysicsprojects.org

Contents

1 Introduction

So, we have a shiny, new infrasound monitoring station, such as the one described at schoolphysicsprojects.org. How to view and analyse the data?

The monitoring program running on the PI saves data and data-plots hourly. Data are saved in *.mseed* format with 24 files per day. Upon saving the previous hour's data the program generates and saves two '24 hour' plots, the day's raw pressure readings and the acoustic power. These can be automatically uploaded via ftp to a remote server such as a website[1].

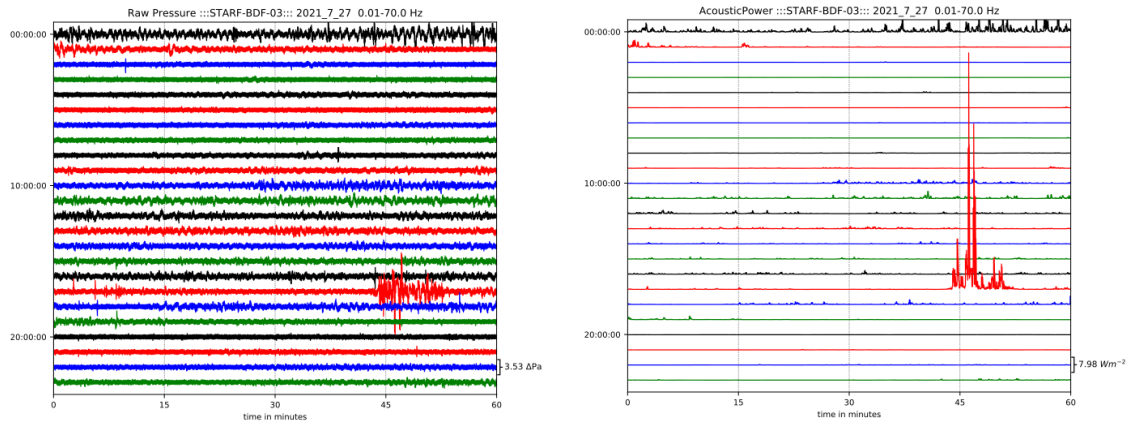


Figure 1: Daily plots generated by R.Pi showing a short, violent thunderstorm at 17:40 UTC

Let's have a go at analysing this further...
I'll assume that having seen the daily plots from above and using an ftp program we have transferred the data off the PI from
/home/pi/InfraSound/Data/2017/7/21 to a folder on our PC. The actual data

Personally I use a folder *Infrasound* on my PC. I simply use ftp to copy over the entirety of the Data and Plots folders from the PI into this. The PI folders can be then safely deleted and will be recreated on the next data save.
The data was gathered on 27/7/21 and is in the associated file *Thunderstorm.zip*.

2 Reading & Plotting 1 Day

Open the InfrasoundAnalyser.py program in your favourite editor.
At about line 50 we have

[1] http://schoolphysicsprojects.org/projects/infrasound/infrasound.html

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
###~~~read in single datafile or an entire folder
## select on one of the two options below

#st1 = opendataFile()  # select a data file to work on

###~~~or an entire folder
st1 = readInFolder(resamplefreq)
```

We'll read in the whole day's data so *#st1 = opendataFile()* is commented out and we are running *st1 = readInFolder(resamplefreq)*.

This generates a file browse box which allowing us to navigate into the folder /2021/7/27/ . Clicking OK causes the 24 hourly files to be read, joined and stored as a Stream datastructure **st1**. As we always work with a copy the data is copied into a trace object **tr1**[2].
At about line 100 we have

```
        simplePlot(tr1)
        plotDayplot(tr1, lowCut, highCut)
```

If we now run the program , navigating to the folder /2021/7/27 we should see in our terminal window,

```
############### Data Read ########################
network: IR
station: STARF
location: 03
channel: BDF
starttime: 2021-07-27T00:00:00.003531Z
endtime: 2021-07-27T23:59:59.991185Z
sampling_rate: 81.0
delta: 0.012345679012345678
npts: 6998400
calib: 1.0
_format: MSEED
mseed: AttribDict({'dataquality': 'D', 'number_of_records': 509,
 'encoding': 'FLOAT32', 'byteorder': '>', 'record_length': 4096,
  'filesize': 2084864})
processing: ["ObsPy 1.2.2: resample(no_filter=True::sampling_rate
=81.0::strict_length=False::window='hanning')"]
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

A plot window should appear.

[2] a Stream object many contain multiple traces e.g. one may have 2 detectors

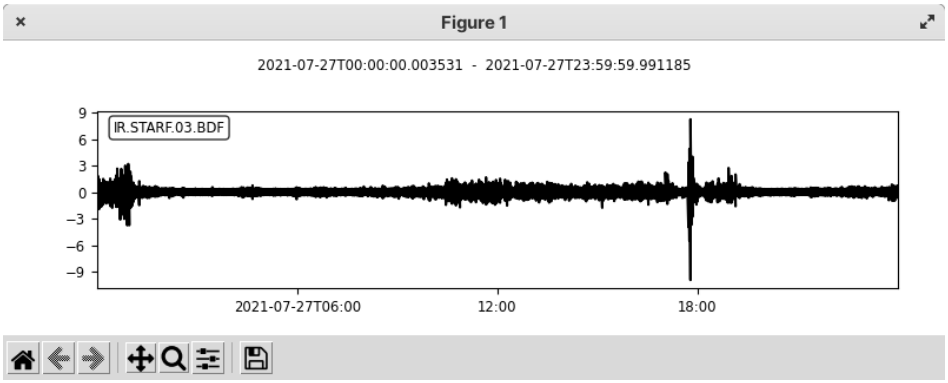


Figure 2: Simple plot

Play around with the zoom (magnifying glass). Note the precise x –time coordinate is displayed in the lower right. The plot can be saved with the save icon. [3]
We do seem to have something interesting at about 18:00, however this all a bit compressed.

On closing the 'simplePlot' window a 'dayplot' appears - [fig 3].

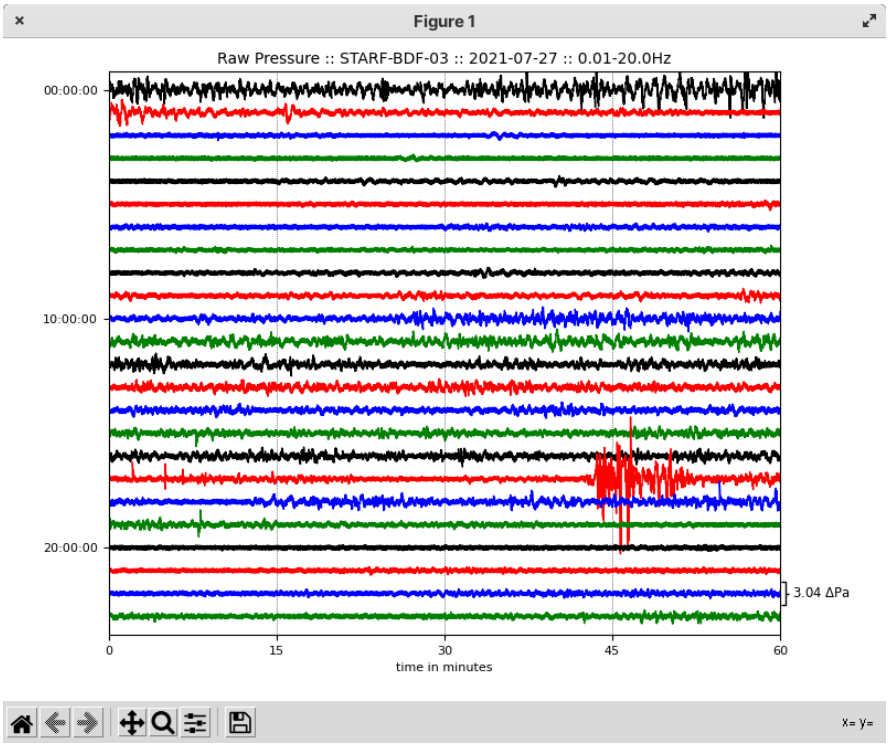


Figure 3: Dayplot

Here we can clearly see a loud event between approx 17:40-17:55. We'll look at this further.

[3] the system automatically saves the plot in the same directory as the Analyser code as simplePlot.png.

3 Reading Single Files

Close down the running program. In the python editor we'll select reading in a single file by commenting out the *ReadInFolder*, uncommenting the *opendataFile*.

```
    st1 = opendataFile()  # select a data file to work#############
    ###~~~or an entire folder
        #st1 = readInFolder(resamplefreq)#
```

also comment out the line

```
        #plotDayplot(tr1, lowCut, highCut)
```

Run the Analyser again. A browse box should appear, navigate to the previous folder and select the file *17.mseed*. This contains the 17:00→18:00 data.

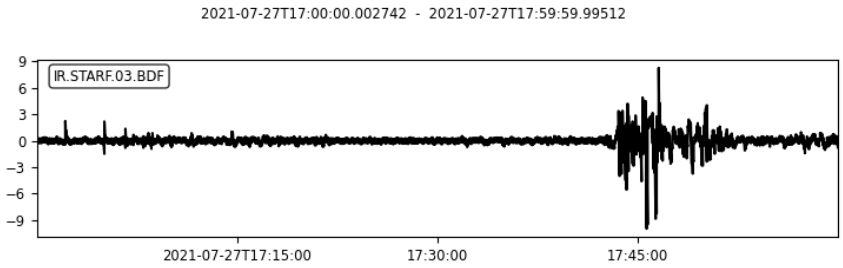


Figure 4: Hour Plot

In the plot - [fig 4] we can clearly see a strong signal 17:41-17:54.
What next?

4 Spectral Analysis Example 1

It is likely that the signal is concentrated in particular frequency ranges. There are 3 main forms of analysis to view this:

1. the *Wavelet Transform*
2. *filtering* then plotting frequency bands e.g. 0.1 - 1.0 Hz
3. the *Power Magnitude Spectrum*.

4.1 Wavelet Transform

I would start with the wavelet transform. Note, this is computational intensive and is impractical for more than a couple of hours of data. I recommend sticking to 1 hour of data.
As before, we'll read in the datafile 17.mseed and uncomment the line

```
plotWaveletTransform(tr1, lowCut, highCut)
```

This will probably take a couple of minutes to process.

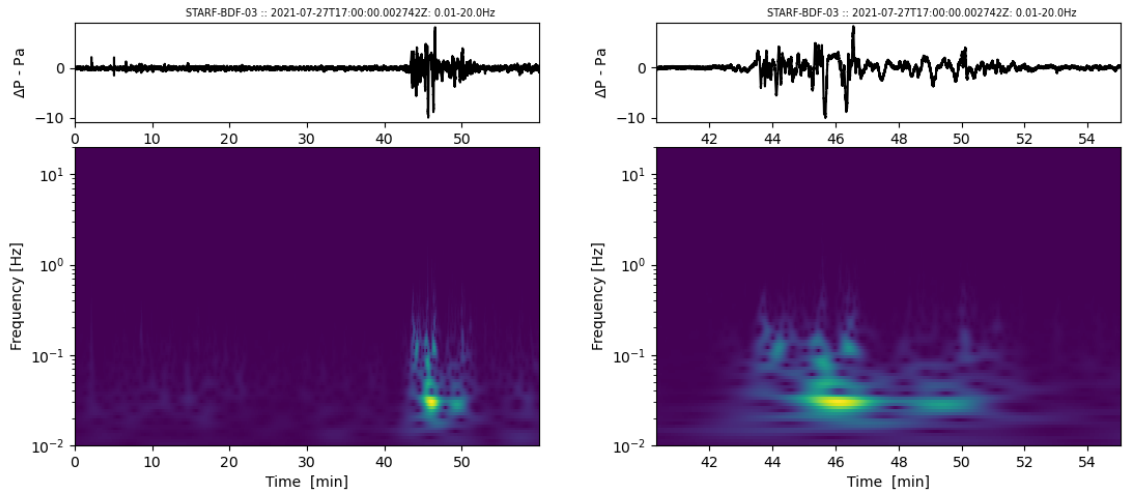


Figure 5: Wavelet transform - right view zoomed

We can see a strong signal, yellow at about 0.04 Hz (note: log scale). I feel that the wavelet is the best general visualiser.

4.2 Power Density Plots

We can also perform a WelchPowerDensity plot or a Power magnitude Plot – two alternate Fourier analysis to the wavelet transform.
We'll comment out the line

```
#plotWaveletTransform(tr1, lowCut, highCut)
```

and uncomment

```
plotWelchPeriodogram(tr1, lowCut, highCut)
plotPwrMagnitudeSpectrumLog(tr1)
```

Do we see anything dramatic - i.e large spikes at well defined frequencies? [fig 6]

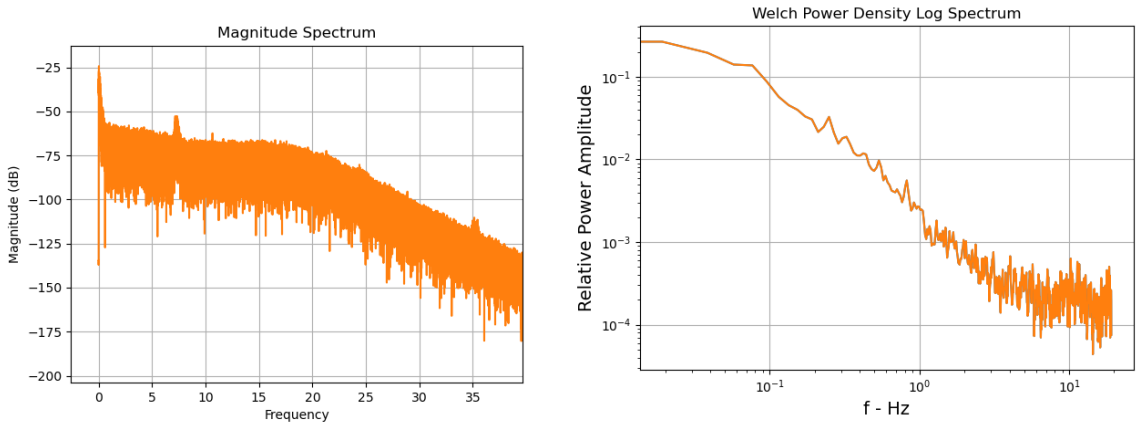


Figure 6: Power Magnitude and Welch Periodogram

Well – somewhat uninspiring. Apart from a peak at 7.5Hz – a fairly broad spectrum. Likely our thunderstorm did not have any particularly strong specific frequencies.
However – the plot above covers the whole hour – why not repeat just for the period 17:43-17:50 as shown in the wavelet transform?

4.3 Selecting a slice of data to work on

At about L83 we have the 'slicing code'

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# #this allows a slice of the trace to be selected
# ### ----- select slice of data to work on
startMinute =  43            #edit this value
endMinute =  50             #edit this value

tracestart = tr1.stats.starttime
startSec = (startMinute * 60)
endSec = (endMinute * 60)
tr1.trim(tracestart + startSec, tracestart + endSec)
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Uncomment the lines as shown, setting startMinute=43 and endMinute=50. Tihs slices the trace down to this period,. We'll run the wavelet transform, Welch and PwrMagnitudeSpectrumLog as before.
This should give us, [figs 7, 8]

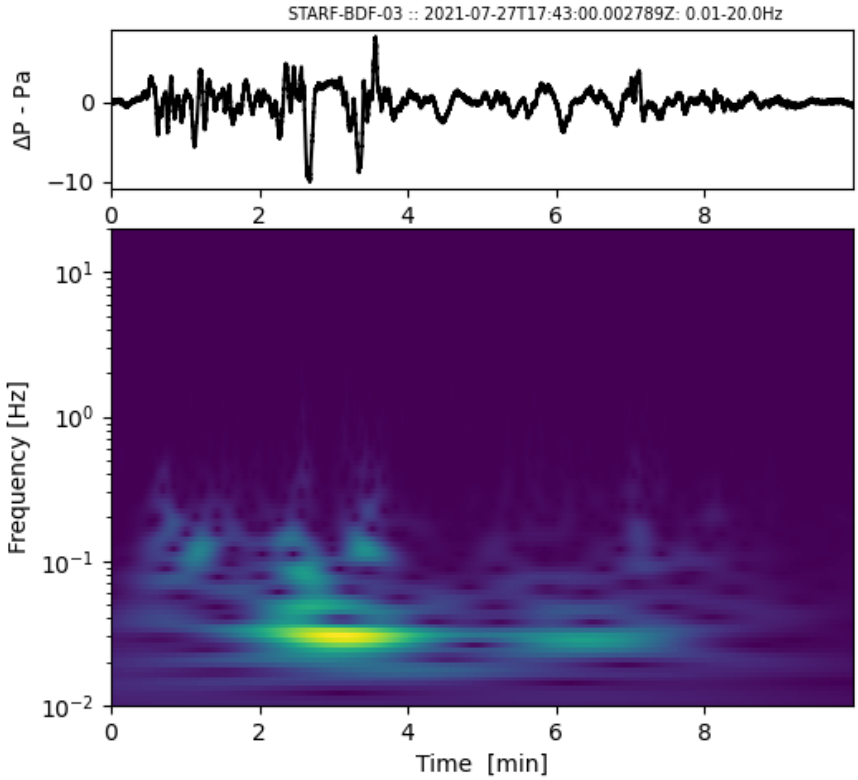


Figure 7

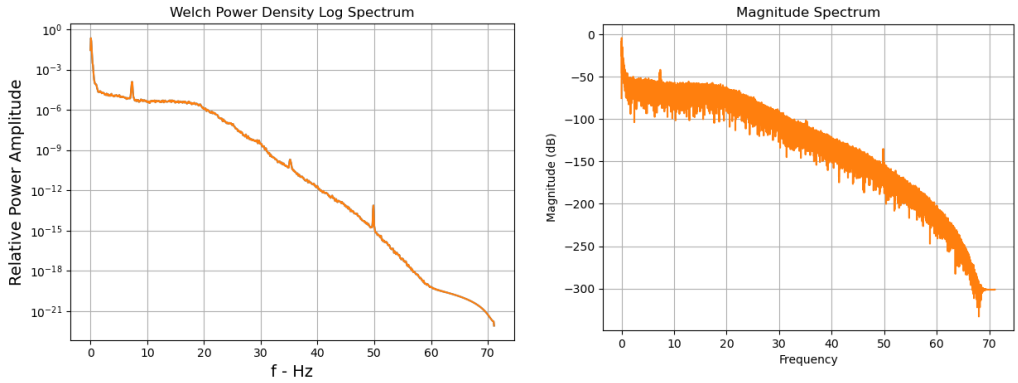


Figure 8

Well, the plots of power bands, wavelet transform, and pwrmagnitude all show that we have infrasound though there are no clear dominant frequencies.

4.4 Plotting Frequency Bands

In the editor uncomment the line

```
plotBands(tr1,deltaT)
```

and rerun as above
After closing the initial plot window we will see something interesting. The original signal has been fiiltered into bands - [fig 9].

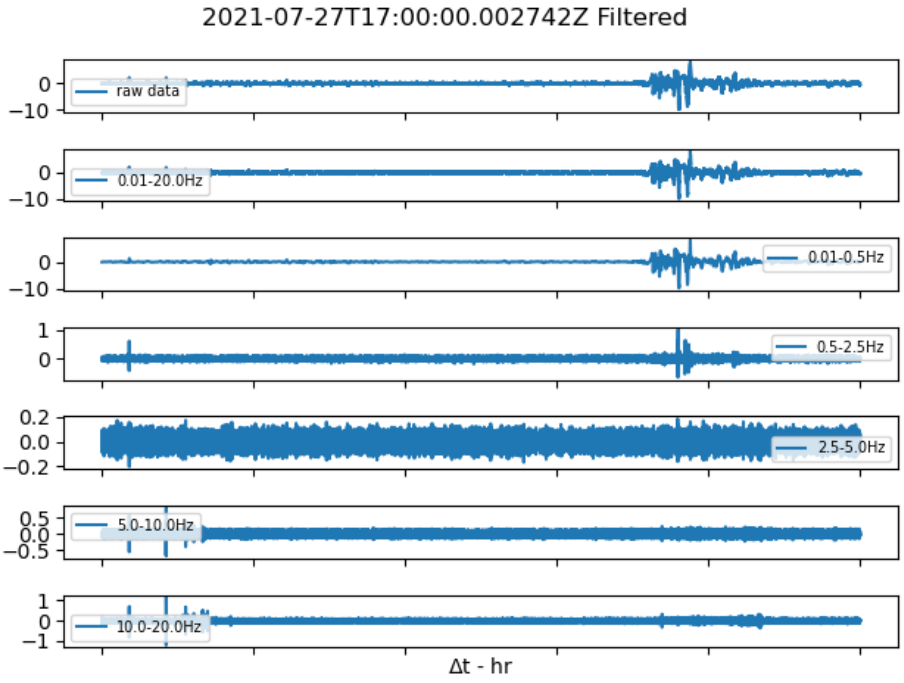


Figure 9: Pressure filtered into frequency bands

Note the vertical scales, in Pascals (atmospheric pressure 101kPa). Most of the signal is in the 0.01-0.5Hz range. Zooming on any one plot zooms all. Closing this window causes a similar plot of acoustic power [4] to appear - [fig 10].

[4] acoustic power is calculated from the square of the raw amplitude and thus emphasises changes

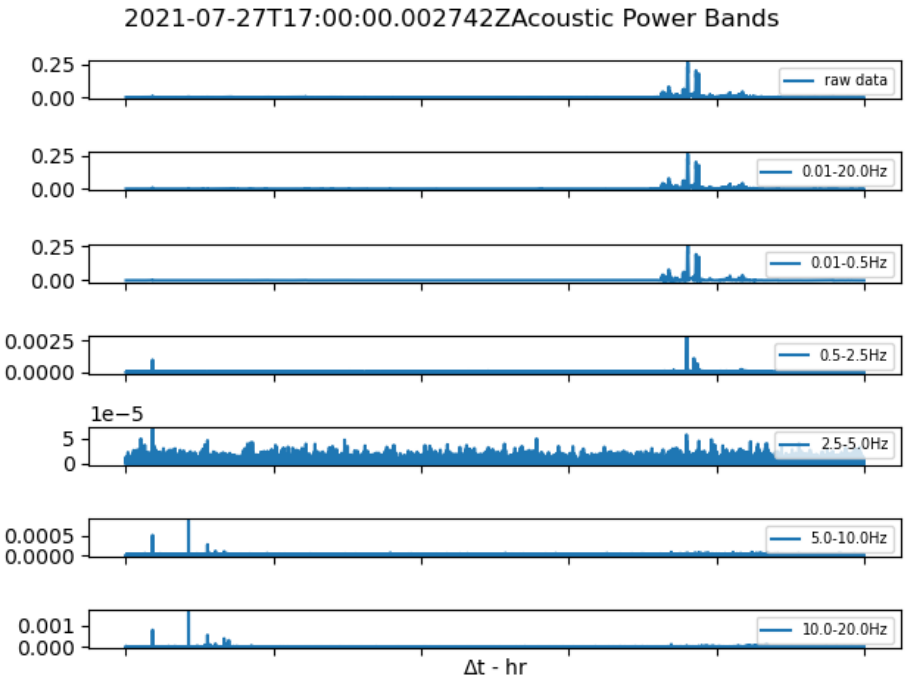


Figure 10: Acoustic power filtered into frequency bands

Again, note the very different vertical scaling.
These filtered plots can give us a good general idea of background infrasound levels and may show up major large signals such as storms, explosions, or industrial noise

5 Spectral Analysis Example 2

Now going to look at an initially uninspiring recording. We took the sensor to a local windfarm an a moderately blustery day. *Windfarm.zip*.

```
############### Data Read ########################
         network: IR
         station: Beta
        location:
         channel: 1
       starttime: 2019-04-14T11:54:39.825550Z
         endtime: 2019-04-14T13:14:40.143274Z
   sampling_rate: 38.281007751937985
           delta: 0.026122614286437505
            npts: 183762
           calib: 1.0
         _format: MSEED
```

Running the Analyser program and loading the file *seamer.mseed* we see that the data runs from 11:54-13:14.

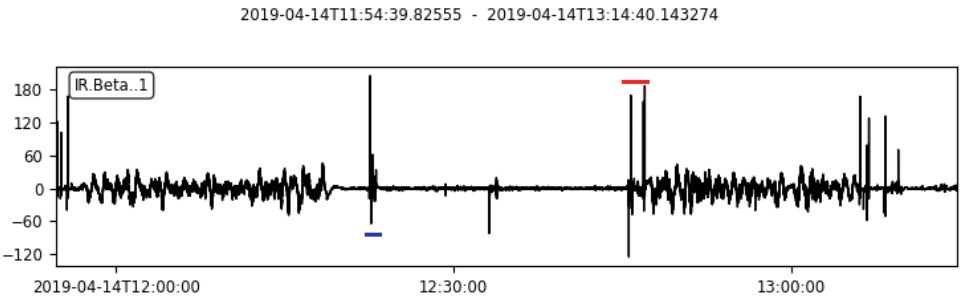


Figure 11

This needs some background

- We drove to the windfarm – lots of infrasound in car.
- On arrival we set up the sensor outside connecting a sensor hose (blue).
- We left it running for some 30 minutes until the red spike where we disconnected and drove home. item The small negative spike in the centre of this region was swapping over the hose to a different filter.

i.e very little signal outside at the windfarm– lots in car journey there and back.

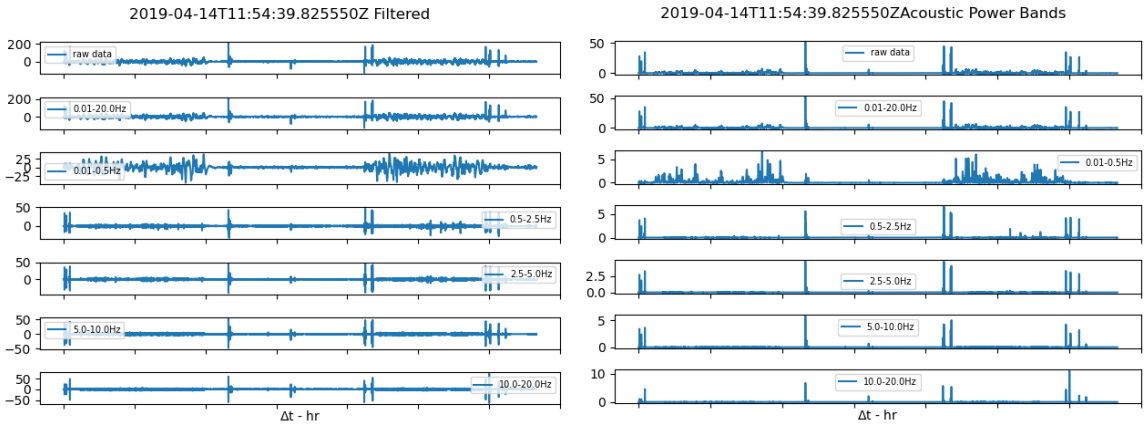


Figure 12

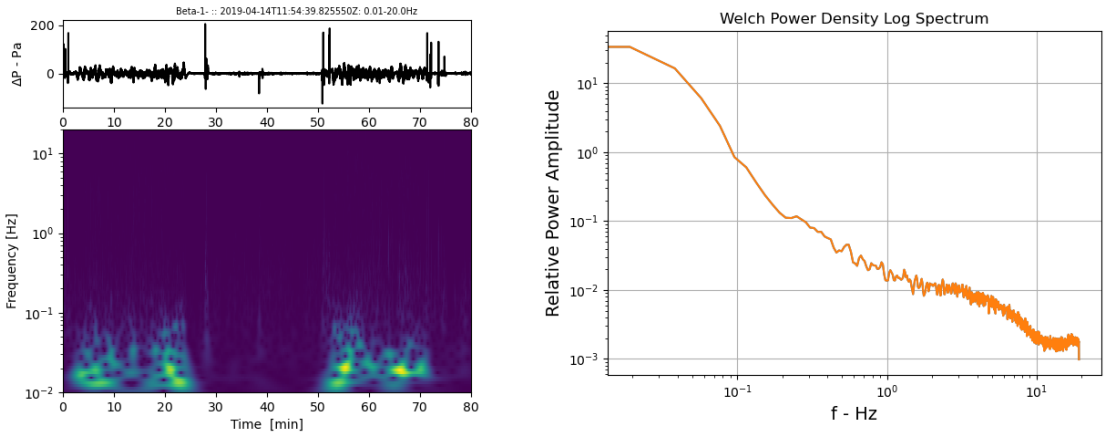


Figure 13

Somewhat disappointing [figs 12, 13].
However Saul extracted the section when we were outside and repeated. *(note, moving the mouse over the first plot window allows very precise measurement of times)* [fig 12].

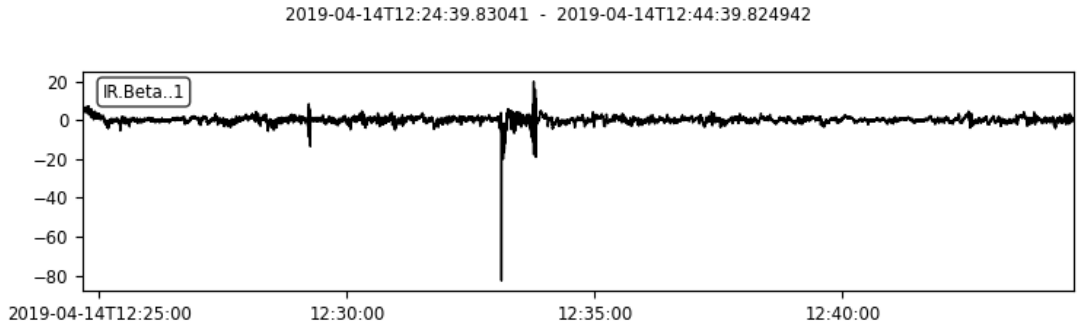


Figure 14

Using the Obspy python routines Saul analysed the signals. Having filmed the turbine he determined a 'blade pass frequency' of 0.75Hz. He could reasonably expect this to be the fundamental frequency f_0 and see signals at this and its harmonics nf_0. After trying various F.F.T. visualisations he did, exactly where predicted except for f_0 which, whilst strong, was masked by noise (note -log scale) Fig 15.

6 Sonification

Our tendancy when visualising data is to .. visualise it - i.e. turn it into an image. After all, much of our processing capability appears dedicated to visual images. This neglects our very capable aural processing system. We have looked at displaying the Infrasound data as audible sound. This requires shifting the frequencies up to well above 20Hz, preferably above 100Hz. It occurs there are 2 ways of doing this..

1. Perform a Fourier Transform to discover the frequency components, multiply them up, then recombine to produce a frequency shifted output signal

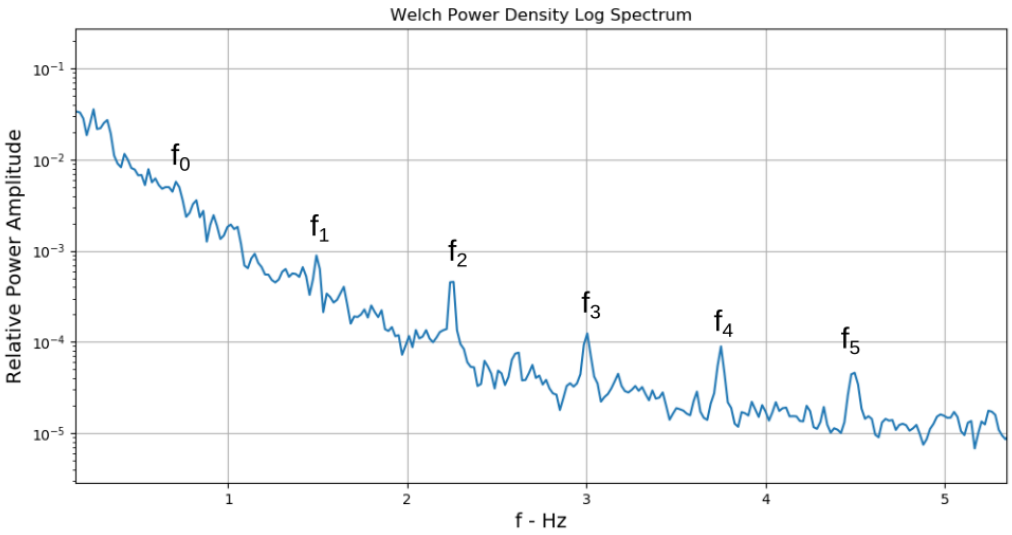


Figure 15: Wind Turbine: f.f.t. showing fundamental and harmonics

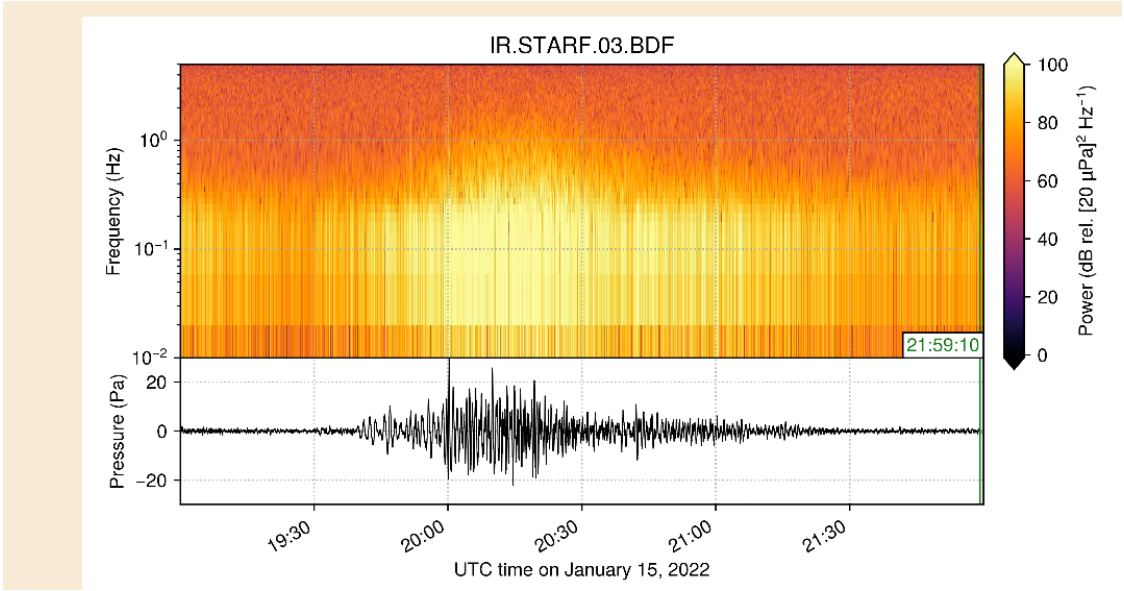


Figure 16: Sonification of Tonga Pulse

2. Speed up the signal. Playing back 500 times faster multiplies the frequences by 500.

We are still working on 1. We found outstanding 'Sonification' code (it won an award![5] at https://sonify.readthedocs.io/en/latest/index.html .
The original read data from an online repository. We made minor changes to read from a single local file.

[5] 2020 John Hunter Excellence in Plotting Contest

6.1 Sonify Installation

sonify is a python script that requires its own anaconda installation environment.

- follow the installation instructions at https://sonify.readthedocs.io/en/latest/index.html.
- replace the file

  `../sonify/sonify/sonify.py`

  with the new version of sonify which should have come with this document.

6.2 Running Sonify

from the command line

```
conda activate sonify
sonify --freqmin 0.01 --freqmax 20 --speed_up_factor 500
--fps 2 --spec_win_dur 8 --db_lim -80 80
```

–fps 1 is fastest, –fps 60 is extremely slow.
This will open a browse box with which you can open a single *.mseed* file. To sonify more than 1 hour then use the analyser program to open a directory, concatenate the files then save them as *.mseed*.

7 General Thoughts

Background infrasound topically has a broad spectrum, i.e. without clearly defined frequencies. In this case plots of acoustic power, wavelet transforms and ‘banded’ plots may give all the salient information contained in the signal.
Rather more rarely specific events can give rise to very clearly defined frequencies e.g. windfarms and helicopters. In these cases it may be worthwhile chasing specific frequencies with a Welch Power Density Spectrum.

A Installing the Analyser Program

assuming that you have a recent anaconda installation. From the command-line.

```
conda create --name obspy python=3.8
conda config --add channels conda-forge
conda activate obspy
conda install obspy
```

B Joining .mseed Files

At about line 50 we have

```
        #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        ###~~~read in single datafile or an entire folder
        ## select on one of the two options below

        #st1 = opendataFile()  # select a data file to work on

    ###~~~or an entire folder
        st1 = readInFolder(resamplefreq)#


        st1 += readInFolder(resamplefreq)#
        st1 += readInFolder(resamplefreq)#
        st1 += readInFolder(resamplefreq)#
        st1.sort(['starttime'])
        st1.merge(method=1, fill_value=0.00)

        #save_as_mseed(st1)
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

We have 2 options here. Either read in a single file or an entire folder. The *readInFolder* will not read recursively. i.e. all the files must be in a single folder, sub-directories will not be read. The function can be called multiple times to read multiple folders.

Note: when reading in multiple files they are *resampled* before concatenation. This because there may be slight differences in the sample rate between files. The *resamplefreq* shoudl be set a few Hz below the lowest sample rate. As a rule-of-thumb is the detector samples at, say 80Hz resampling at 75Hz will probably be fine.

C Saving .mseed Files

The function

```
save_as_mseed(st)
```

will output a stream object as a single *.mseed* file. This may be useful after concatenating a bunch of files.

D Infrasound Monitor - Data Directory Structure

The Infrsound Monitor described at schoolphysicsprojects.org saves the data files in

```
\Data
    --> Year
          --> Month
                 --> Day
                          --> Hour
```

e.g. The data for 10-11 am, 19/07/2022

```
\Data
    --> 2022
          --> 7
                 --> 19
                          --> 10.mseed
```