Apéndice C

Manual códigos unificados: Captura e Inversión Sísmica automática

C1. Introducción

La rama inicial de la detección, asociada a instrumentación, es la construcción y configuración/calibrado de maquinas que monitorean, capturan y guardan las vibraciones de la Tierra en datos computacionales (sismografos). La segunda rama (de la que trata este manual) es el proceso computacional para reunir estos datos de manera consistente para nuestro objetivo, procesarlos sísmicamente, detectar y localizar un evento sísmico de acuerdo a los conocimientos actuales sobre sismicidad. Este documento pretende reescribir distintos códigos previos de la disciplina con las herramientas del lenguaje Python.

Correcciones, consultas, sugerencias y críticas se reciben al correo juviveros@udec.cl

C1.1. Entorno de trabajo

Para aplicar los códigos de este método se recomienda crear un entorno dentro de python que contenga los siguientes paquetes:

- **- ObsPy**
- **- Cartopy**
- **- Subprocess**

- **- Numpy**
- **- Matplotlib**

Es muy útil utilizar el software libre Anaconda[1] para manejar entornos de trabajo fuera de los archivos formales de Python. En caso de no usar Anaconda, un entorno se crea de la siguiente forma:
Instalar los motores de descarga de librerias *pip* y el creador de entornos *virtualenv* :

```
python3 -m pip install -user -upgrade pip
python3 -m pip -version
```

```
python3 -m pip install -user virtualenv
```

Crear el entorno propiamente dicho :

```
python3 -m venv .../NOMBRE_ENTORNO
```

Ingresamos al entorno en nuestra terminal:

```
source activate ../NOMBRE_ENTORNO
```

Finalmente instalamos las librerias deseadas:

```
python3 -m pip install NOMBRE_PAQUETE
```

¡No olvide entrar al entorno antes de lanzar los códigos a continuación!

[1] https://www.anaconda.com

C2. Resumen de códigos para el método

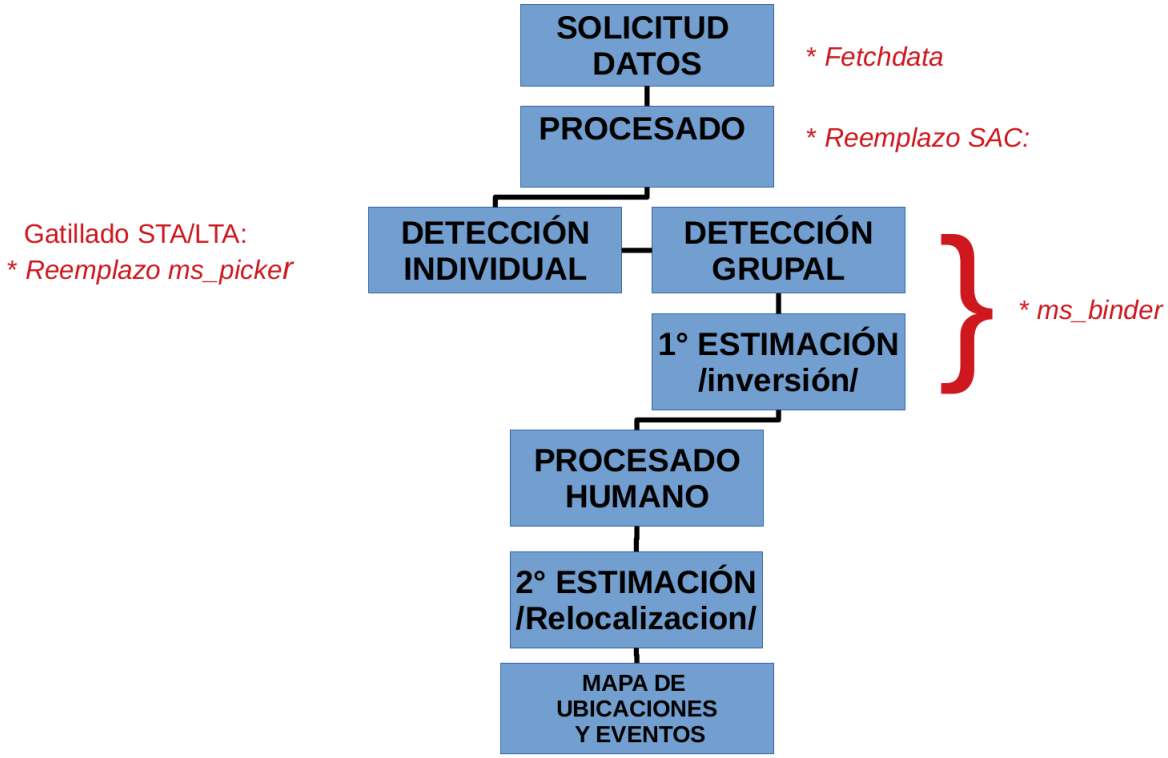


Figura C2.1: Algoritmo de las tareas requeridas para una localización automática exitosa. En rojo algunos códigos previos reemplazados por nuestras propias versiones en Python o acondicionados para su uso en caso contrario (*Binder*).

A continuación detallamos cada etapa y mencionamos los códigos usados para su resolución, recordando que el objetivo de este anexo es compartir cada script del método para su uso libre.

1. **Solicitud de datos/Descarga**: La solicitud de datos de amplitud sísmica a los servidores de la red unificada IRIS. Se requiere manejar el script fetchdata , y definir una serie de parámetros asociados a una red de estaciones sismograficas.
   Códigos y Auxiliares:
   - *FetchData.sh*
   - *fetchdata_descarga.sh*
   - *fetchdata_dayloop.sh*

- *info_estaciones.txt*
- *corte_horario.py* (opcional)

2. **Calibrado inicial**: De acuerdo a la extensión y características de los datos/proyecto/estaciones que estemos trabajando, se hace necesario decidir una serie de parámetros para los siguientes pasos. Primero utilizamos un espectrograma sobre una muestra de series para definir las bandas de frecuencia y otros datos del procesado. Luego utilizamos la función característica de una serie con un evento confirmado de antemano para definir parámetros del gatillado individual sobre algunas muestras al azar.
   Códigos:
   - *Espectrograma.py*
   - *fcaracteristica.py*
3. **Procesado**: Eliminamos fuentes de ruido no asociado a sismicidad, corregimos limites entre trazas, respuesta del instrumento, verificamos estaciones sin data presente, entre otros.
   Código:
   - sismogramas.py
4. **Gatillado individual**: Utilizamos el método STA/LTA, debidamente calibrado en pasos anteriores, para obtener una lista periodica de anomalías detectadas.
   Código:
   - sismogramas.py
5. **Gatillado grupal + Localización inicial**: Utilizamos el código **Binder_nosc** debidamente calibrado para realizar un gatillado grupal, que filtra los resultados de la etapa anterior, y desde ahí resuelve un problema de inversión con el método de búsqueda en grilla. Se obtiene una lista de hipocentros estimados.
   Código:
   - *Binder_nosc_lanzador.py*
   - *info.dat* (parámetros)

- *param.txt*
- *velmod.hdr*

6. **Inspección manual**: El siguiente paso sería obtener una inversión mas precisa de los supuestos hipocentros. Capturar con exactitud las llegadas S se vuelve necesario, pero métodos automáticos como las Redes neuronales aún no se han demostrado suficientemente precisos. Este paso exige inspeccionar manualmente los eventos estimados y eliminar falsos positivo.
   Código:
   - *sismogramas_MINI.py*
   - *filtro_falsospositivos.py*

C3. Solicitud de Datos

Para realizar la adquisición de datos nos valemos de un par de scripts donde se aplica el comando **Fetchdata** para una serie de parámetros que definen la extensión de los datos requeridos. Existen muchas otras formas de conseguir datos [2]

C3.1. ¿Que es fetchdata?

Descargar datos adecuados es la primera tarea a la hora de cualquier estudio en las Ciencias de la Tierra. Es una tarea ardua, tendiendo en cuenta la cantidad de instituciones (que capturan datos), tipos de instrumento, tipos de datos, frecuencias, catálogos, fechas, otros parámetros, etc; elecciones que tienen un impacto en el resultado final.

Una de las instituciones mas importantes en el mundo de la sismología es el **IRIS** (*Incorporated Research Institutions for Seismology*) y su DMC (*Data Management Center*) asociado[3], que recopilan información de muchas redes sísmicas, con multitud de instrumentos, a disposición del publico.

Esta descarga se puede hacer desde la misma pagina institucional, o directamente utilizando el comando de back-end en la terminal desde la que se solicita la información, llamado FetchData. En este manual básicamente aprenderemos a utilizar este comando y automatizarlo en un script para configurar de forma fácil los diversos parámetros con los que IRIS estandariza las descargas.

C3.2. Instalando Fetchdata:

Fetchada a su vez debe ser descargado en su ultima versión desde el repositorio ([4]) y puesto a punto para ser llamado desde la terminal:

[2] Vease el manual de sismología observacional Bormann (2002), Capitulo 8. Enlace de descarga https://moodle2.units.it/pluginfile.php/294221/mod_resource/content/1/manual_seismological_observatory-2002.pdf

[3] Sitio oficial https://ds.iris.edu/ds ; Descarga manual data: https://ds.iris.edu/gmap

[4] https://seiscode.iris.washington.edu/projects/ws-fetch-scripts/files

1. Hecha la descarga, extraemos el archivo en una carpeta de fácil acceso (ej: FetchData). Cambiamos sus permisos.

2. Abrimos desde la terminal el **.bashrc** y añadimos un acceso a la carpeta.
   ```
   > export PATH=$PATH:/home/Descargas/FetchData
   ```

3. Finalmente creamos un acceso directo en la carpeta:
   ```
   > ln -s /home/Descargas/FetchData/FetchData-2018.337
    /home/Descargas/FetchData/fetchdata
   ```

Luego basta con escribir fetchdata con sus parámetros asociados, como explicaremos a continuación y estar conectados a internet, para solicitar data desde el DMC de IRIS.

C3.3. Elección de Canal

La información que nos provee el comando, se configura con diversos parámetros relacionados a las frecuencias capturadas en distintos tipos de instrumentos hallados en cada estación. Con el fin de personalizar la información solicitable por fetchdata se maneja un código de tres letras llamado Canal (Frecuencia de banda-esquina / Tipo de instrumento / Dirección (N-E-Z)).

Band Code

The first letter specifies the general sampling rate and the response band of the instrument. (The "A" code is reserved for administrative functions such as miscellaneous state of health.)

Band code	Band type	Sample rate (Hz)	Corner period (sec)
F	...	≥ 1000 to < 5000	≥ 10 sec
G	...	≥ 1000 to < 5000	< 10 sec
D	...	≥ 250 to < 1000	< 10 sec
C	...	≥ 250 to < 1000	≥ 10 sec
E	Extremely Short Period	≥ 80 to < 250	< 10 sec
S	Short Period	≥ 10 to < 80	< 10 sec
H	High Broad Band	≥ 80 to < 250	≥ 10 sec
B	Broad Band	≥ 10 to < 80	≥ 10 sec
M	Mid Period	> 1 to < 10	
L	Long Period	≈ 1	
V	Very Long Period	≈ 0.1	
U	Ultra Long Period	≈ 0.01	
R	Extremely Long Period	≥ 0.0001 to < 0.001	
P	On the order of 0.1 to 1 day[1]	≥ 0.00001 to < 0.0001	
T	On the order of 1 to 10 days[1]	≥ 0.000001 to < 0.00001	
Q	Greater than 10 days[1]	< 0.000001	
A	Administrative Instrument Channel	variable	NA
O	Opaque Instrument Channel	variable	NA

1. These are approximate values. The sample rate should be used for the correct Band Code.

Instrument Code and Orientation Code

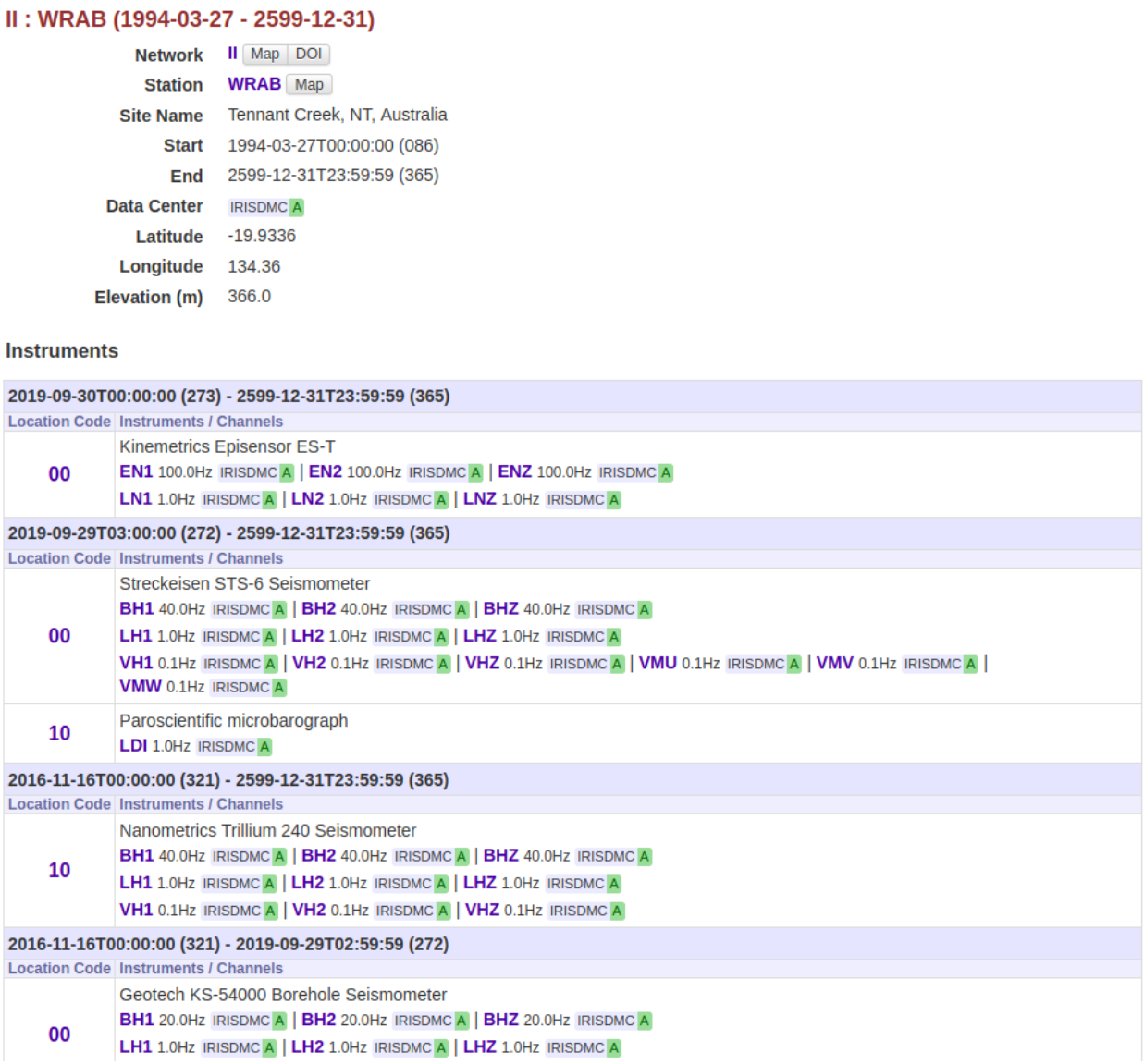
The second letter specifies the family to which the sensor belongs. In essence, this identifies what is being measured. Each of these instrument types are detailed in this section.

The third letter in the channel name is the Orientation Code, which provides a way to indicate the directionality of the sensor measurement. This code is sometimes used for a purpose other than direction, which is instrument-specific. When orthogonal directions are used, there are traditional orientations of North (N), East (E), and Vertical (Z), as well as other orientations that can readily be converted to traditional ones. These options are detailed with each instrument type. Use N or E for the orientation when it is within 5 degrees of north or east. Use 1 or 2 when orientations are more than 5 degrees from north or east. Put the actual orientation of the sensor in the dip and azimuth fields of blockette 52.

Seismometer: Measures displacement/velocity/acceleration along a line defined by the dip and azimuth.

Instrument Code	
H	High Gain Seismometer
L	Low Gain Seismometer
G	Gravimeter
M	Mass Position Seismometer
N*	Accelerometer
	* historically some channels from accelerometers have used instrumentation codes of L and G. The use of N is the FDSN convention as defined in August 2000.
Orientation Code	
Z N E	Traditional (Vertical, North-South, East-West)
A B C	Triaxial (Along the edges of a cube turned up on a corner)
T R	For formed beams (Transverse, Radial)
1 2 3	Orthogonal components but non traditional orientations
U V W	Optional components
Dip/Azimuth:	Ground motion vector (reverse dip/azimuth if signal polarity incorrect)
Signal Units:	M, M/S, M/S**2, (for G & M) M/S**2 (usually)
Channel Flags:	G

Figura C3.1: Detalle del manual oficial de manejo de data MSEED. Documento suministrado por el FDSN (International federation of digital seismograph networks: http://www.fdsn.org/pdf/) donde se explica a fondo el significado de cada letra del canal.



II : WRAB (1994-03-27 - 2599-12-31)

Network	II Map DOI
Station	WRAB Map
Site Name	Tennant Creek, NT, Australia
Start	1994-03-27T00:00:00 (086)
End	2599-12-31T23:59:59 (365)
Data Center	IRISDMC A
Latitude	-19.9336
Longitude	134.36
Elevation (m)	366.0

Instruments

Location Code	Instruments / Channels
2019-09-30T00:00:00 (273) - 2599-12-31T23:59:59 (365)	
00	Kinemetrics Episensor ES-T **EN1** 100.0Hz IRISDMC A | **EN2** 100.0Hz IRISDMC A | **ENZ** 100.0Hz IRISDMC A **LN1** 1.0Hz IRISDMC A | **LN2** 1.0Hz IRISDMC A | **LNZ** 1.0Hz IRISDMC A
2019-09-29T03:00:00 (272) - 2599-12-31T23:59:59 (365)	
00	Streckeisen STS-6 Seismometer **BH1** 40.0Hz IRISDMC A | **BH2** 40.0Hz IRISDMC A | **BHZ** 40.0Hz IRISDMC A **LH1** 1.0Hz IRISDMC A | **LH2** 1.0Hz IRISDMC A | **LHZ** 1.0Hz IRISDMC A **VH1** 0.1Hz IRISDMC A | **VH2** 0.1Hz IRISDMC A | **VHZ** 0.1Hz IRISDMC A | **VMU** 0.1Hz IRISDMC A | **VMV** 0.1Hz IRISDMC A | **VMW** 0.1Hz IRISDMC A
10	Paroscientific microbarograph **LDI** 1.0Hz IRISDMC A
2016-11-16T00:00:00 (321) - 2599-12-31T23:59:59 (365)	
10	Nanometrics Trillium 240 Seismometer **BH1** 40.0Hz IRISDMC A | **BH2** 40.0Hz IRISDMC A | **BHZ** 40.0Hz IRISDMC A **LH1** 1.0Hz IRISDMC A | **LH2** 1.0Hz IRISDMC A | **LHZ** 1.0Hz IRISDMC A **VH1** 0.1Hz IRISDMC A | **VH2** 0.1Hz IRISDMC A | **VHZ** 0.1Hz IRISDMC A
2016-11-16T00:00:00 (321) - 2019-09-29T02:59:59 (272)	
00	Geotech KS-54000 Borehole Seismometer **BH1** 20.0Hz IRISDMC A | **BH2** 20.0Hz IRISDMC A | **BHZ** 20.0Hz IRISDMC A **LH1** 1.0Hz IRISDMC A | **LH2** 1.0Hz IRISDMC A | **LHZ** 1.0Hz IRISDMC A

Figura C3.2: Observemos una estación de la red IRIS. Arriba tenemos información general de la estación, y abajo la cantidad de instrumentos y los canales que cubren, amén del tiempo desde que registra datos.

C3.4. Uso básico fetchdata:

Fetchdata requiere una serie de parámetros con los cuales se solicita la información. Los parámetros mas relevantes están explicados en la documentación IRIS para FetchData ([5]), y hallamos explicado al detalle de sus canales en el apéndice del

[5] https://seiscode.iris.washington.edu/projects/ws-fetch-scripts/wiki/Running_the_scripts_and_examples

manual para data MSEED (C3.1).

Desglosemos su uso con un ejemplo: (Para la red sísmica C de una serie de estaciones con iniciales GO para los canales BHN/BHZ/BHE/HNZ/HNN/HNE entre las fechas indicadas, utilizando el formato .mseed)

```
> fetchdata  -NC -SGO??  -C'BH,HN?'  -s 2012-03-25T22:37:06
-e 2012-03-25T23:07:06 -o evento.mseed -m evento.metadata
-sd /home/Descargas
```

- -**N**: Red de instrumentos.
- -**S**: Estaciones.
- -**C**: Canal/es.
- -**L** : (Location code) Cual instrumento de la estación estamos llamando.
- -**s** : Fecha de inicio de descarga de datos (en formato UTC).
- -**e** : Fecha de termino de descarga de datos.
- -**o** : Dirección/archivo con la serie de datos en formato miniseed.
- -**m** : Dirección/archivo con la metadata.
- -**sd** : Dirección/archivos de los polos y ceros de las estaciones involucradas.

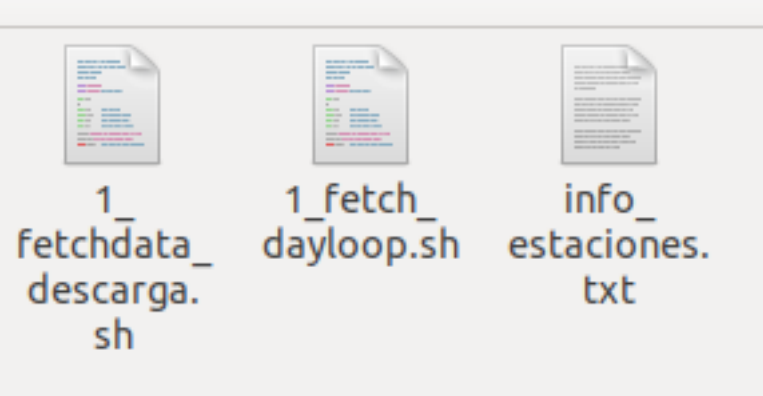
Los espacios rellenados con "?" buscarán todas las coincidencias de las siglas anteriores en la base de datos del DMC de IRIS, (ej: BH? = BHZ,BHN,BHE)

De esta manera podemos solicitar fácilmente datos de instrumentos sísmicos de manera estandarizada. Sin embargo, esto es poco practico a la hora de requerir muchas estaciones donde por lo demás la data está recargada en un solo archivo. Este es el motivo del script que vamos a explicar a continuación.

C3.5. Archivos requeridos:

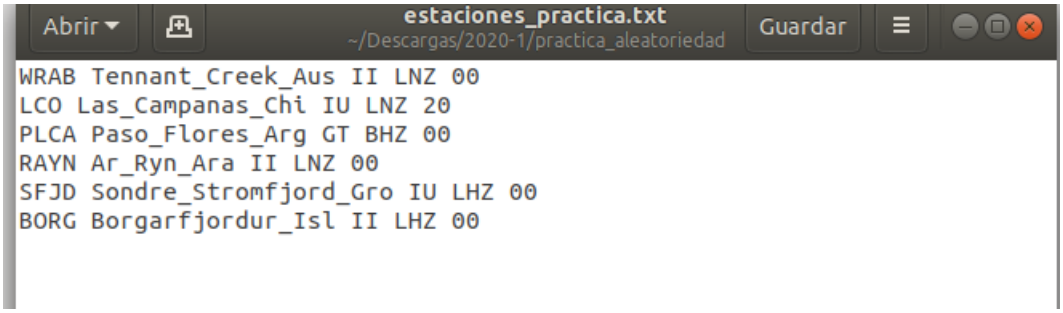
De esta manera construimos 3 archivos para manejar nuestras descargas automáticamente:

1. fetchdata_descarga.sh : El archivo de control donde se lanza el comando fetchdata.
2. fetch_dayloop.sh : Archivo donde se define el periodo de descargas y los subintervalos en segundos que podremos ajustar a conveniencia para crear carpetas/archivos. Además traspasa los tiempos de entrada desde el formato Juliano al Unix requerido por fetchdata.
3. info_estaciones.txt : Documento de texto donde se alojan 5 columnas (Estación,Info, Red,Canal/es, Location code), con tantas filas como estaciones con datos deseemos incluir.



C3.6. Uso del script:

Antes de lanzar nuestro script inspeccionemos el archivo auxiliar(*info_estaciones.txt*) desde el cual almacenamos parámetros de cada estación que serán leidos por fetchdata. Revise la línea que coincide con la estación mostrada en la figura C3.2.



Observamos cada columna:

1. Código estación
2. Descripción (no se usará mas adelante)
3. Nombre Red instrumental
4. Canal
5. Código del instrumento (dentro de la estación)

Esta serie de estaciones y los parámetros que convengan los recopilamos de los mapas del DMC [6] según el objetivo de la investigación como vemos en la imagen siguiente.

[6] https://ds.iris.edu/gmap/

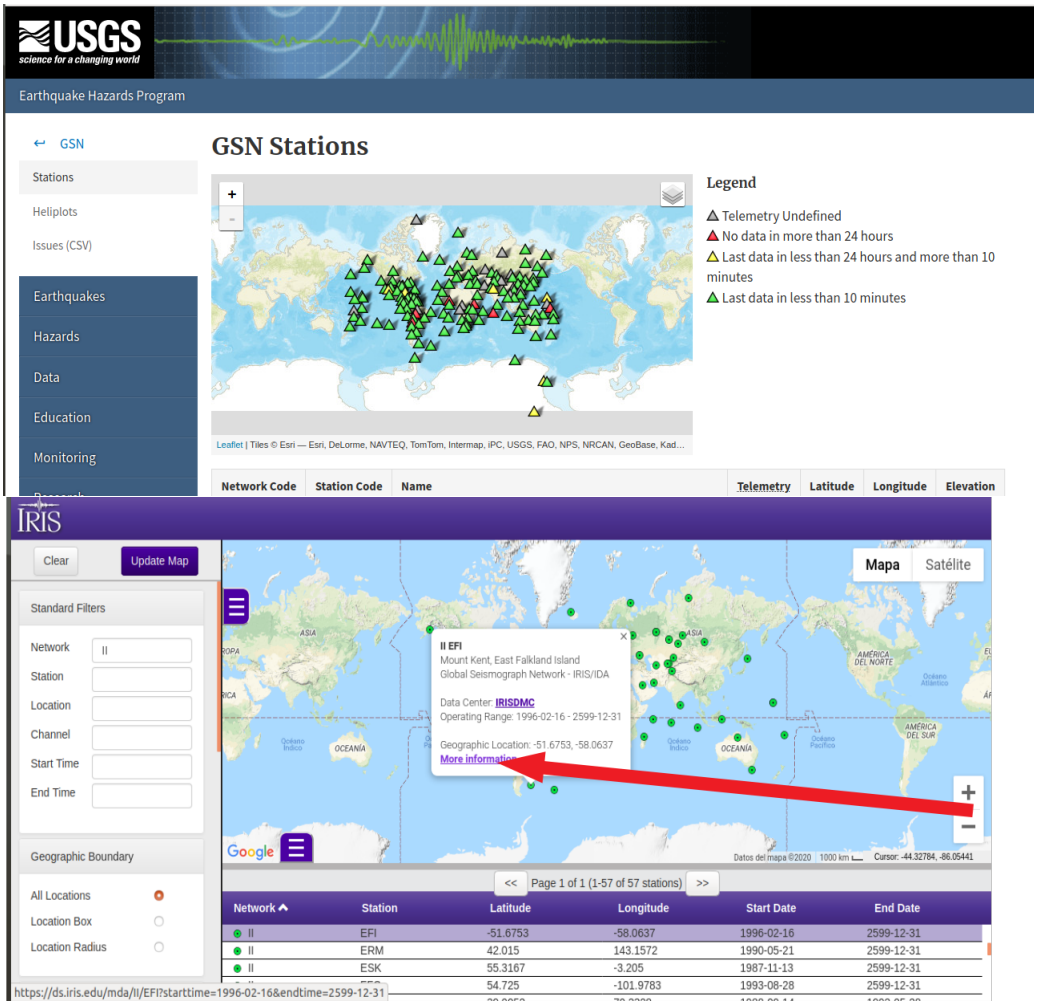


Figura C3.3: Entorno de la pagina interactiva de redes sísmicas de IRIS.En la primera captura, la red unificada de las estaciones mas completas en cuanto a datos de la Global Seismographic Network(GSN). En la figura inferior vemos desplegado el buscador personalizado de estaciones y redes; en la flecha roja la opción para conocer los detalles desplegados en C3.2

Pasemos ahora a explicar otro aspecto relevante del script auxiliar (*fetch_ dayloop.sh*) que determina el periodo de tiempo en el que se extraen datos y se guardan en series distintas, Además pasa nuestros datos de entrada en día juliano (formato sencillo), a formato UNIX que fetchdata lee.

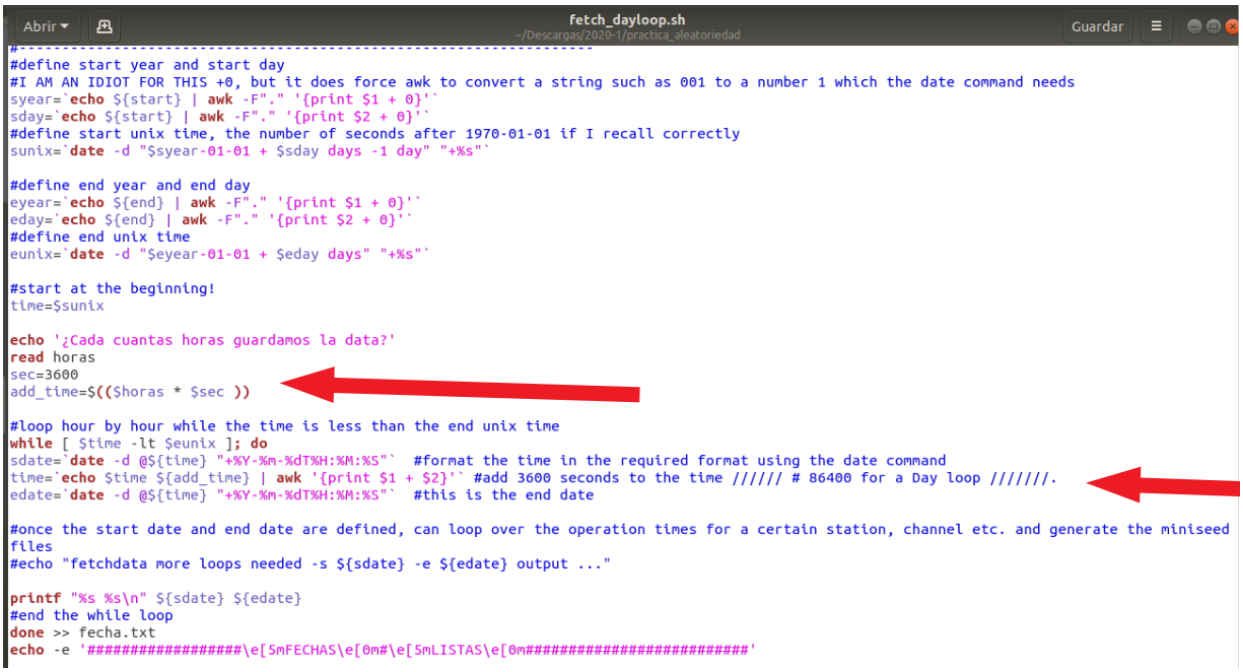


Figura C3.4: Marcamos con flechas las líneas que nos indican cada cuantas horas queremos generar una serie.

Por ejemplo, si mantenemos 1 hora(3600[s]), se generarán 24 carpetas por día, con archivos de N estaciones en el interior de c/u. En cambio si queremos generar una sola serie, calculamos el total de horas de nuestro intervalo de tiempo.
Por ultimo, ubicamos los días julianos (JULDAY DOY[7]) de interés en nuestro estudio. Por ejemplo el 1 y 2 de Marzo del 2020 → 2020.61 y 2020.62 respectivamente.



Figura C3.5: Iniciamos la descarga propiamente dicha, lanzando nuestro script de control.

[7]Enlace de interés: https://www.esrl.noaa.gov/gmd/grad/neubrew/Calendar.jsp

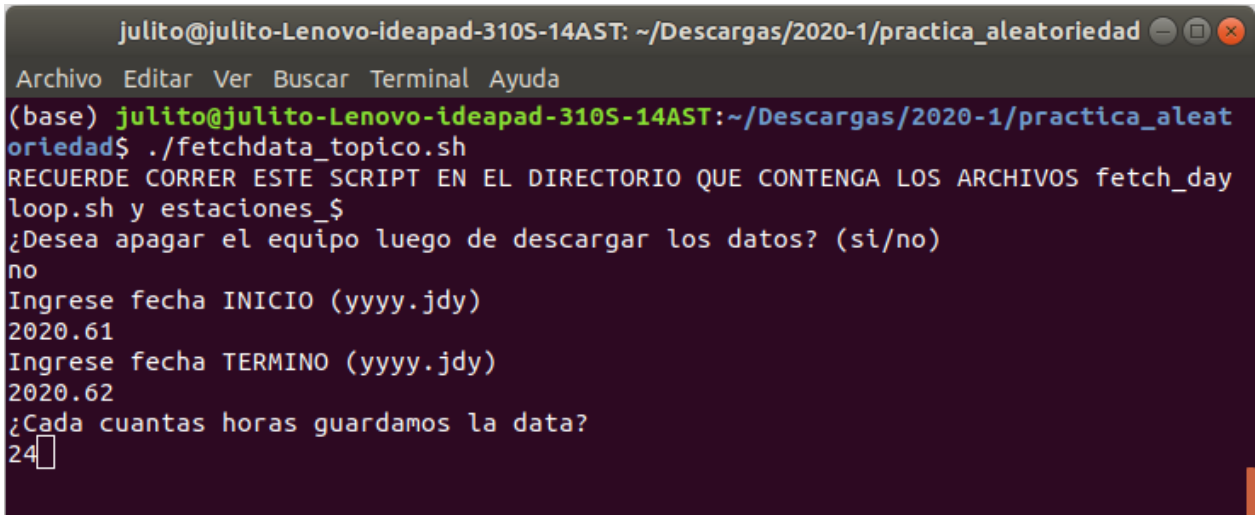


Figura C3.6: Decidimos si apagar o no el equipo al final de la descarga, ingresamos los extremos del total del intervalo en formato juliano indicado e indicamos el numero de horas en las que se extiende cada archivo descargado entre las dos fechas.

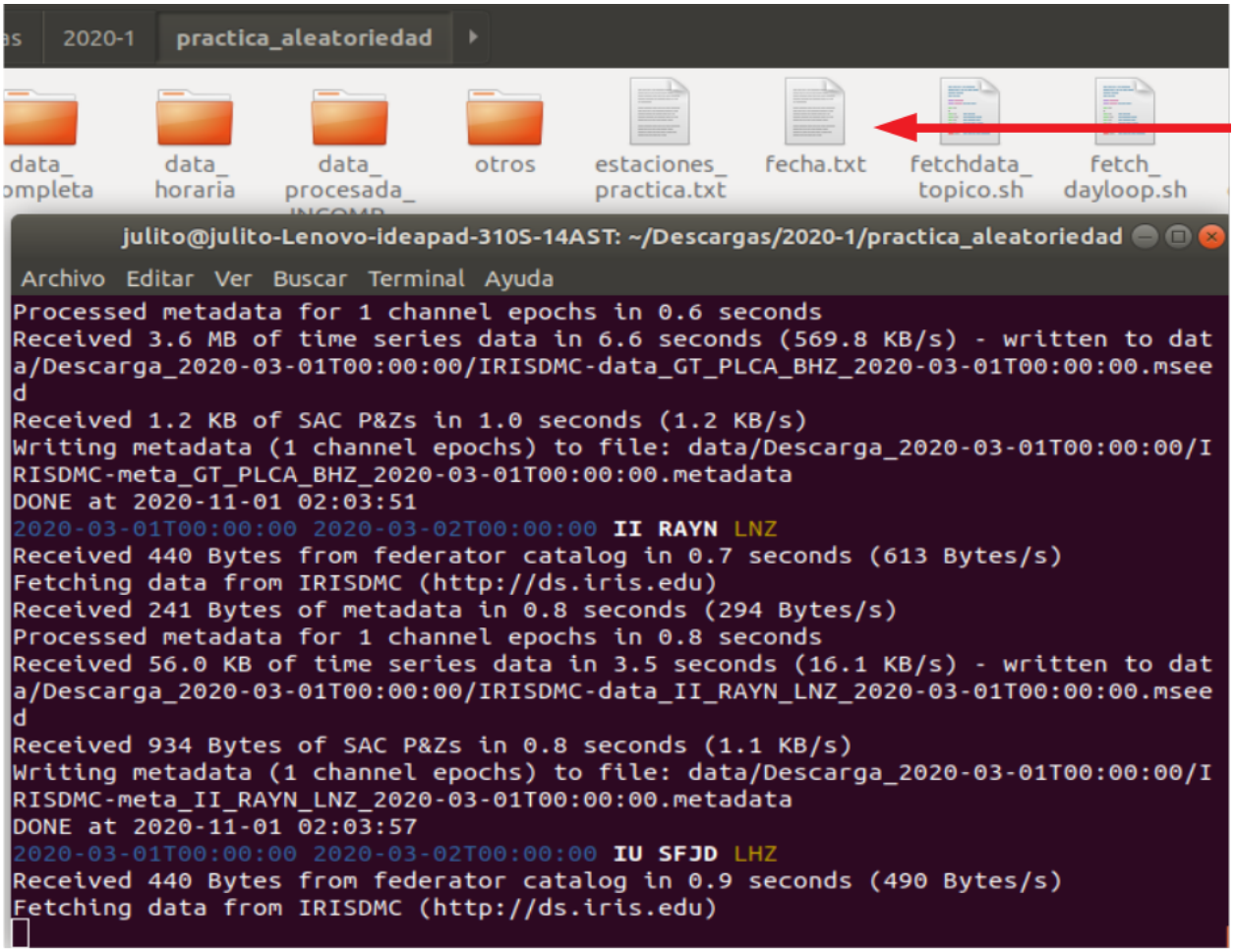


Figura C3.7: Se inicia la descarga; Notamos que se ha generado un archivo temporal llamdo **fecha.txt**, donde están almacenados los intervalos de tiempo en que solicitamos cada dato entre inicio y final.
Si nos fijamos en la descarga propiamente dicha, en azul se muestra el intervalo del que se está bajando información, en blanco la estación/red ídem, y en naranjo el canal. Además se muestra el peso de la data y sus archivos asociados (metadata y Polos & Ceros del instrumento).



Figura C3.8: Una vez completada la descarga comprobamos que se generó una carpeta para cada 24 horas de información solicitada, durante dos días, tal como le indicamos al script.

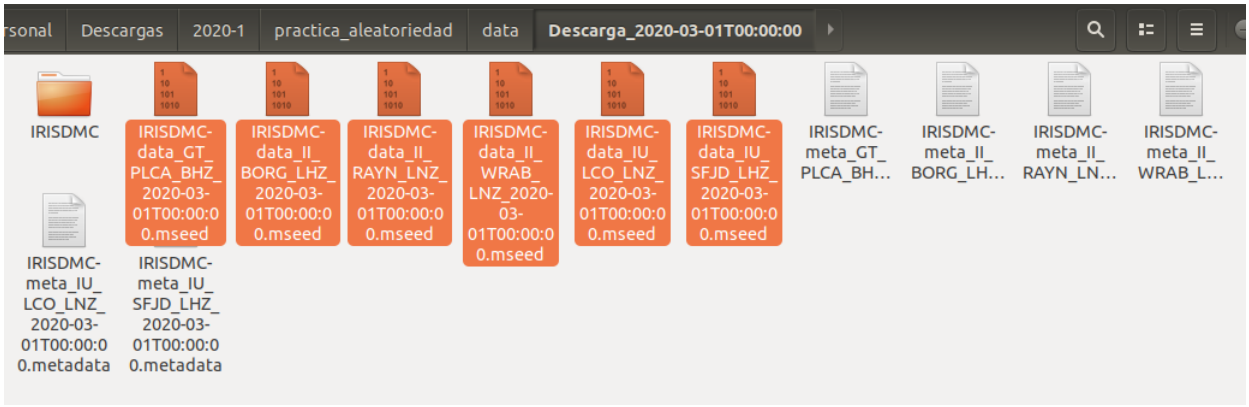


Figura C3.9: Dentro de cada carpeta, se descargó un archivo/serie de amplitud sísmica en formato .mseed para cada estación guardada en el .txt auxiliar.

En caso de realizar una descarga equivocada y requerir un trozado posterior del archivo .mseed, se incluye un script auxiliar llamado *corte_horario.py* que abre cada traza y la guarda en otro archivo.

C4. Calibrado

Antes de hacer el procesado de los datos completos, conviene definir una serie de parámetros del procesado y gatillado individual que son únicos a las estaciones, lugar y datos que estemos analizando. Los parámetros del gatillado grupal y de la inversión serán definidos en su respectiva sección.

C5. Espectrograma:

Queremos conocer las bandas de frecuencia donde se ubica el ruido no sísmico de la mayoría de las estaciones; Programamos en python un espectrograma de los datos para cada hora/estación/componente.
Para calibrar nuestra banda de filtrado, elegimos algunas muestras aleatorias del total de datos donde finalmente determinamos nuestras bandas útiles (entre los 5-19 [Hz] en la imagen).

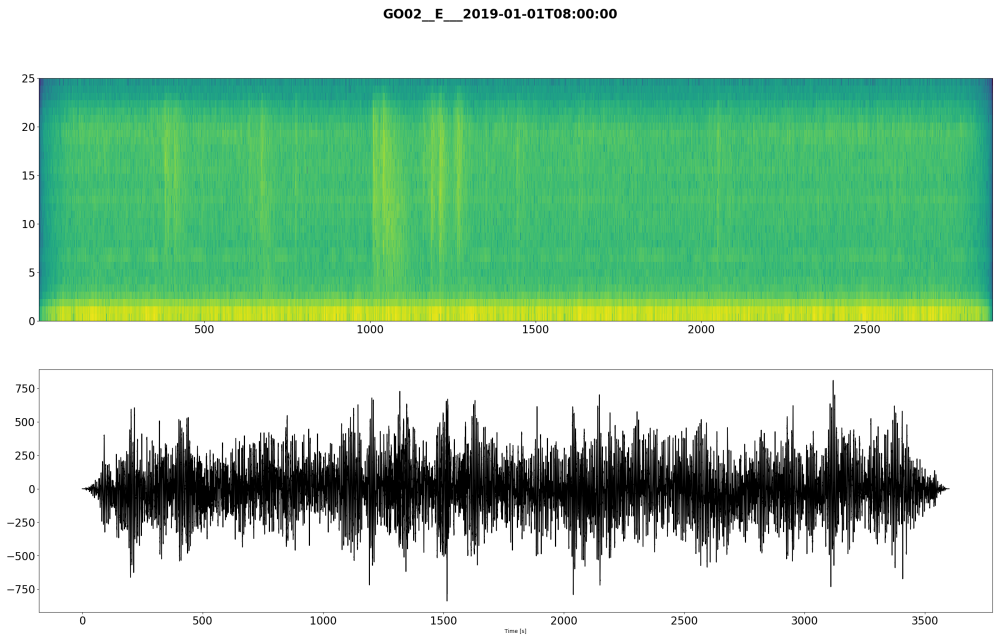


Figura C5.1: Determinamos el ruido ambiental en la franja amarilla continua del espectrograma (∼0 2 [Hz]), En cambio un posible evento sísmico llega en el segundo 1000. Notar como el ruido oculta esta anomalía al visualizar la serie original sin filtrar.

C6. Función Característica

IMPORTANTE: Esta etapa del calibrado exige primero entender que es el gatillado (Triggering) de un posible evento y el método STA/LTA de gatillado. Vea la introducción del siguiente capitulo, sección gatillado individual, antes de continuar.

Para lograr "visualizar", la amplitud umbral donde provocar un gatillado, nos servimos de la gráfica de función característica (*envelope function*) que utiliza curvas tangentes al borde de la serie (ya procesada en este script) para exagerar los saltos abruptos de la traza.

Luego, para calibrar tomamos una serie (filtrada) donde sepamos de antemano que hay un evento sísmico. Entonces determinamos las ventanas de promedio comparativo (STA/LTA) y los umbrales de amplitud de salto del gatillo (threshold) visualizando la función característica, siempre velando que aparezca la menor cantidad de falsos positivos.
Finalmente confirmamos que la detección se ajusta precisamente al evento en la serie original (filtrada) para la mayoría de estaciones.

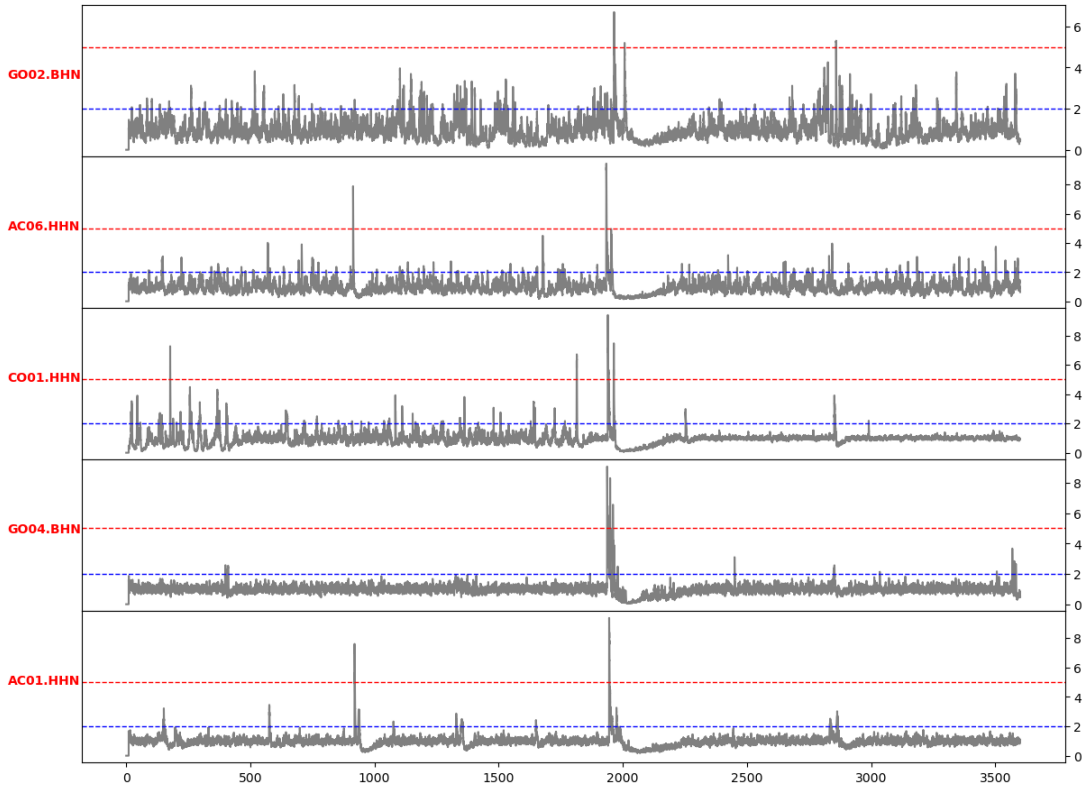


Figura C6.1: En la imagen tenemos un ejemplo de Función Característica, con un evento sísmico a los $\sim 2000[s]$. El eje vertical corresponde a la proporción de incrementos del promedio de una sección movil STA frente a una LTA en el gráfico original. La línea roja (*threshold on*), indica el umbral donde queremos definir un salto en el gatillo (*pick*), la línea azul(*threshold off*) en cambio indica el fin de la detección posterior a un salto.

Por ejemplo, en la figura C6.1 , solo se capturan saltos para promedios cortos que superen las 5 veces el promedio largo y en caso de hallarse uno, se considera que el evento termina cuando el promedio de la ventana corta es 2 veces la ventana larga, entre medio no se capturan nuevos saltos. Finalmente estos parámetros se expresan proporcionalmente en la serie original de tiempo vs amplitud.

C7. Procesado y Gatillado individual (STA/LTA)

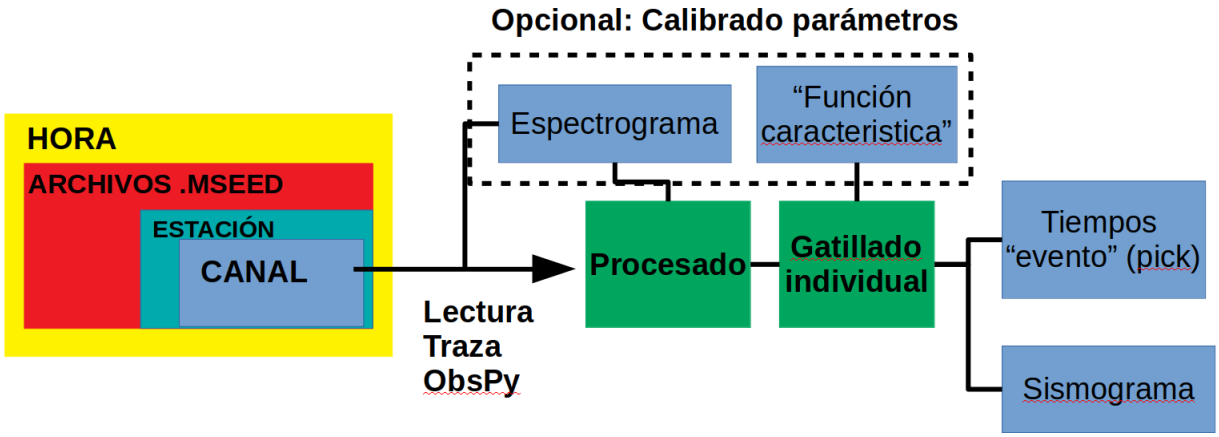


Figura C7.1: Diagrama de flujo de la etapa de procesado de la traza y gatillado individual.

C7.1. Procesado

En Python nos servimos del paquete de herramientas de sismología llamado ObsPy. Este nos provee de fácil acceso a funciones que permiten leer, sincronizar en tiempo, procesar, graficar ,etc.. paquetes de datos sísmicos (trazas) en diversos formatos. Se sugiere crear un ambiente en Python para alojar este paquete.
El procesado de este script consiste en:

- Normalizar y remover la tendencia.
- Unificar trazas (Un paquete de datos puede contener múltiples trazas, sobre todo cuando ocurren eventos sísmicos que interrumpen el registro).
- Remover respuesta del instrumento.
- Filtrar bandas para eliminar el ruido no-sísmico[8].

[8]Se configuran las herramientas mas relevantes para el filtraje: El par de bandas donde dejaremos pasar un ancho de banda, Con la opción *Corners* se decide el Orden/Frecuencia Esquina con la que decidiremos si cortar la banda del resto de el espectro abruptamente (*Corner*= 0) o atenuando las amplitudes para frecuencias fronterizas (1 ≥Corner). Por ultimo, para mayor calidad del filtro este se puede lanzar dos veces, desde el inicio y final de la traza al costo de de sufrir un leve desfase con la opción *Zerophase*. Además es posible aplicar un taper sobre la traza, que le asignará un peso a las frecuencias dentro de la ventana de filtrado respecto al centro de esta, con distintas funciones para elegir (opción *Taper*)

* Por motivos de eficiencia, se combina el Procesado inicial con el Gatillado individual en un solo script(*sismogramas.py*).

C7.2. Gatillado Individual (STA/LTA):

Esta etapa está contenida en el script de procesado: *sismogramas.py*. El script usado en Python reemplaza al script **ms_picker.sh** sin ser exactamente igual.

Básicamente existen 2 mecanismos complementarios entre sí para que el computador afirme que sucedió un evento sísmico. La detección individual y la grupal. A grandes rasgos: La primera busca detectar una anomalía proporcional en un rango de amplitud comparándola a otra ventana. La segunda, busca un cierto numero de anomalías en estaciones diferentes cercanas en un rango de tiempo, para afirmar que está ocurriendo un evento sísmico (Mas adelante veremos que *Binder_nosc* robustece esta ultima a través del calculo iterativo de hipocentro en grilla, con el objetivo de definir cuantos picks efectivamente son un evento y están relacionados entre sí).

Para la detección individual se utilizó el método STA/LTA Recursivo (Short term Average/Long term Average):

Se corren dentro de la traza dos ventanas móviles de tiempo(Short time Average/Long time Average) donde se calculan y comparan sus promedios de amplitud; Superado cierto umbral de tamaño proporcional (*threshold on*) elegido por el usuario, se guarda el tiempo capturado (pick), que corresponde típicamente a la fase P en este caso al ser la primera llegada. Además se fija un segundo umbral de final de la fase (*threshold off*) para no capturar posibles gatillos de la coda . Por ultimo se añade un tiempo muerto antes de permitir al gatillo registrar otra fase. La llegada S por desgracia no es capturable en todo evento con este método al estar en cierta manera escondida en la coda P, la irregularidad de las capturas S obliga mas adelante a replantear el método aplicado al problema de inversión (de inversión simple a iteración con grilla).

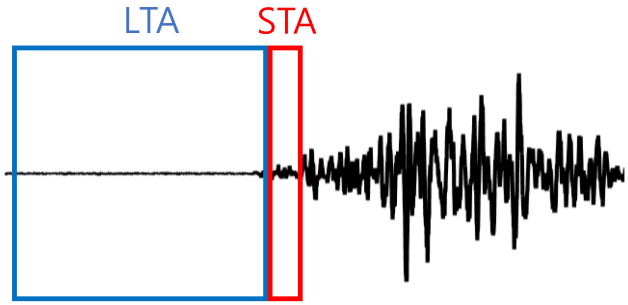


Figura C7.2: Ejemplo gráfico del STA/LTA detectando en una seria una amplitud notablemente superior al promedio hasta el momento.

A continuación, ejemplos de comparativa STA/LTA tanto en la serie original como en la función característica para ilustrar los conceptos de umbrales de inicio y final de la fase. Verifique en que unidad está el eje x de la traza. Generalmente los archivos de datos sísmicos están en Muestras (*Samples*). Puede o no ser útil pasarlo a segundos.

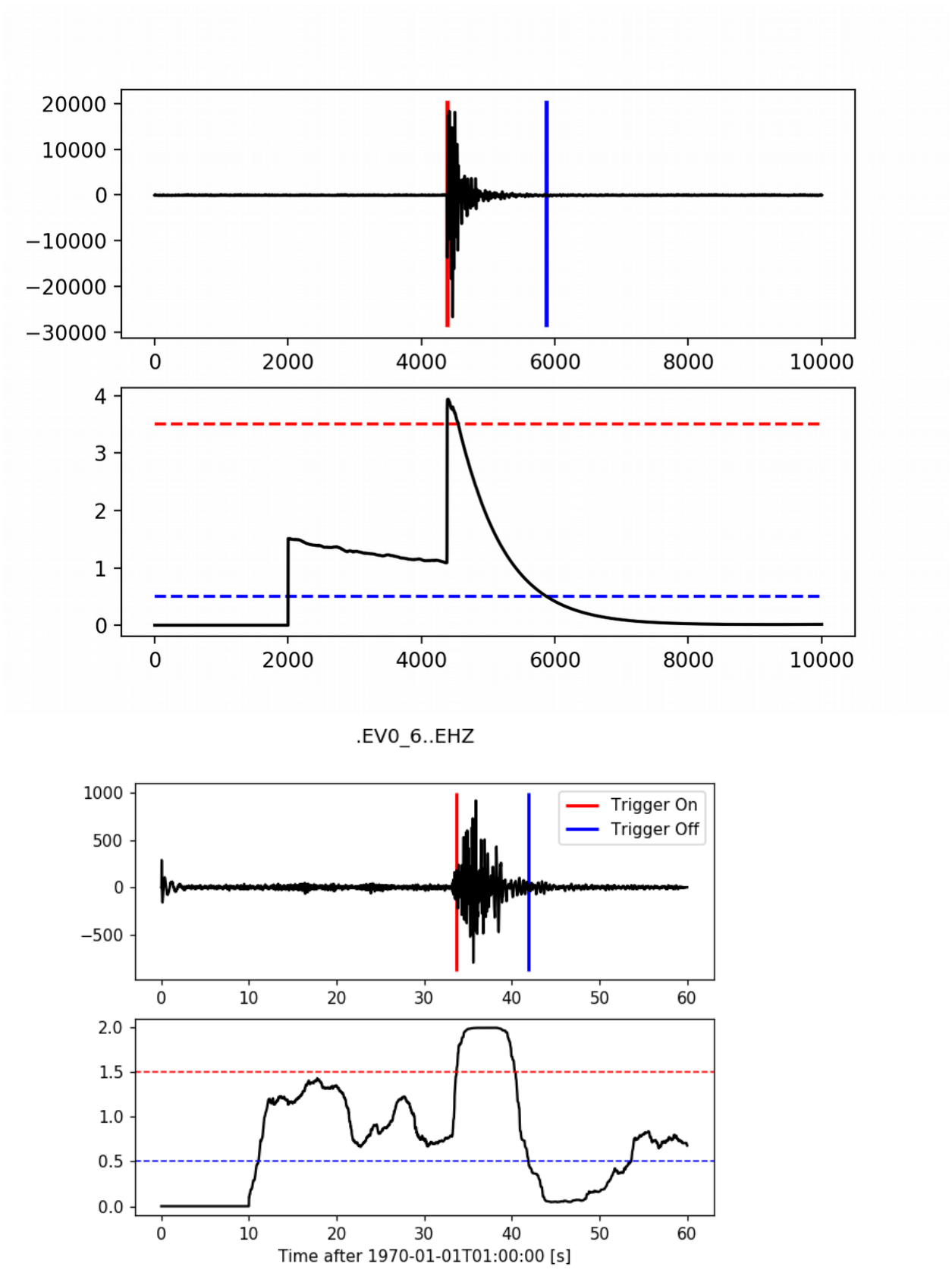


Figura C7.3: Dos ejemplos comparativos entre el salto de los gatillados en la serie original y su respectiva función característica con la que se ajustó el gatillado.

Observe que este método no captura la primera llegada de la onda P, puesto que la detección depende de ventanas, a menor tamaño de las ventanas LTA/STA y mas bajo el umbral, mas precisa la detección, pero también habrán mas falsos positivos.

Así, durante el procesado de la traza, recopilamos un archivo de gatillados en todos los canales anotados en el formato UNIX-TIME[9]. Estas llegadas aún no se las puede declarar como eventos sísmicos.
Importante: El archivo de picks deberá ir ordenado por tiempo(unix) de menos a más, de otra la localización con *Binder_nosc* no funcionará.

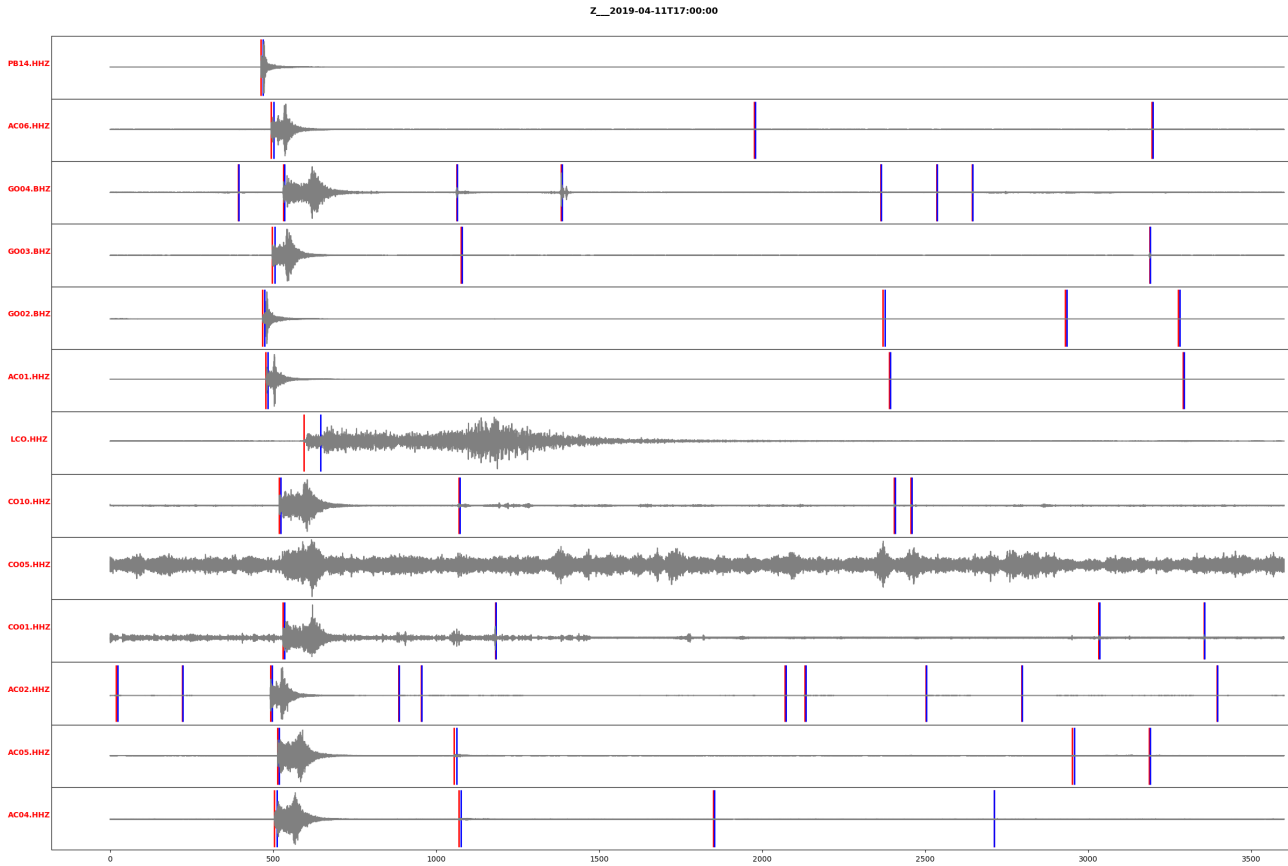


Figura C7.4: Un sismograma ejemplar de procesado y aplicación del gatillado automático individual para múltiples estaciones. Se obtuvieron buenos resultados en esta hora con un rango STA/LTA de 1:10 [s] y Umbrales (thresholds) de 10 y 8 veces sobre el promedio.

[9]Segundos después de 1970-01-01 UTC

C8. Gatillado Grupal e Inversión

La siguiente etapa lógica sería utilizar el archivo de salida de la etapa anterior (picks (P) del procesado y gatillado individual en los canales para la traza de tiempo), para afirmar la existencia de eventos sísmicos y localizarlos.

Entonces tenemos dos tareas separadas: confirmar un evento sísmico y ubicarlo (resolviendo un problema de inversión de datos). Para lograrlo, se utiliza un script auxiliar llamado Binder_nosc_AR, debido a lo complejo del metodo. Convendría trasladarlo definitivamente a Python en el futuro.

La lógica detrás de la localización y la iteración en grilla está explicada detalladamente en el Marco Teórico (3) de la Tesis adjunta. Solo es relevante para este manual mencionar que la falta de llegadas S que permitan manejar un tiempo de viaje, fuerzan a replantear el problema y la incógnita deseada. La búsqueda de un tiempo artificial obliga indirectamente a hallar hipocentros tentativos resolviendo un problema directo en cada trigger aceptable iterando puntos de la grilla hasta dar con el que mejor se acerque a la llegada observada. De ahí que debamos definir tantos parámetros y un modelo de velocidad en los archivos auxiliares. El gatillado grupal se logra en este código al comparar cercanía de tiempo y cercanía espacial de los nodos asociados a cada pick.

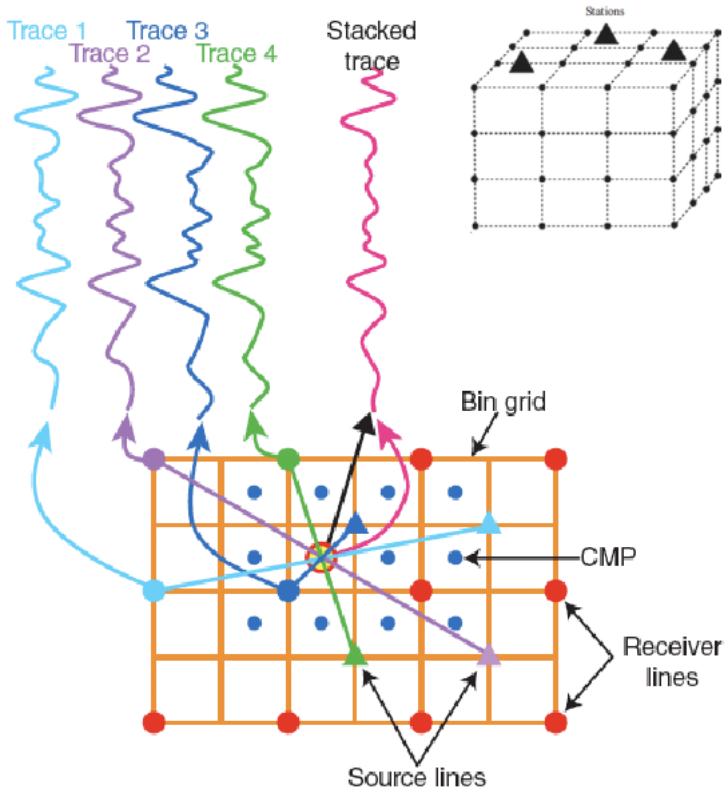


Figura C8.1: A partir de cálculos directos iterativos de distancias en nodo-estación con picks (en colores), se encuentran hipocentros tentativos, los cuales se intersectan resolviendo el sistema de ecuaciones planteado como inversión para dar con la incognita real: Hipocentro artificial, el cual a su vez se puede seguir iterando con la siguiente mejora en la ubicación y disminución del residual con los datos de tiempo llegada observados/extraídos del gatillado.

C8.1. Descarga Binder_nosc_AR + instalación:

Obtenido el script, configuramos su ubicación con un alias en el .bashrc para llamarlo desde la terminal, puede requerir permisos. El script de control en Python que conectará con el original *binder_nosc_AR.sh* en este programa de ubicación sísmica se llama *Binder_nosc_lanzador.py*.

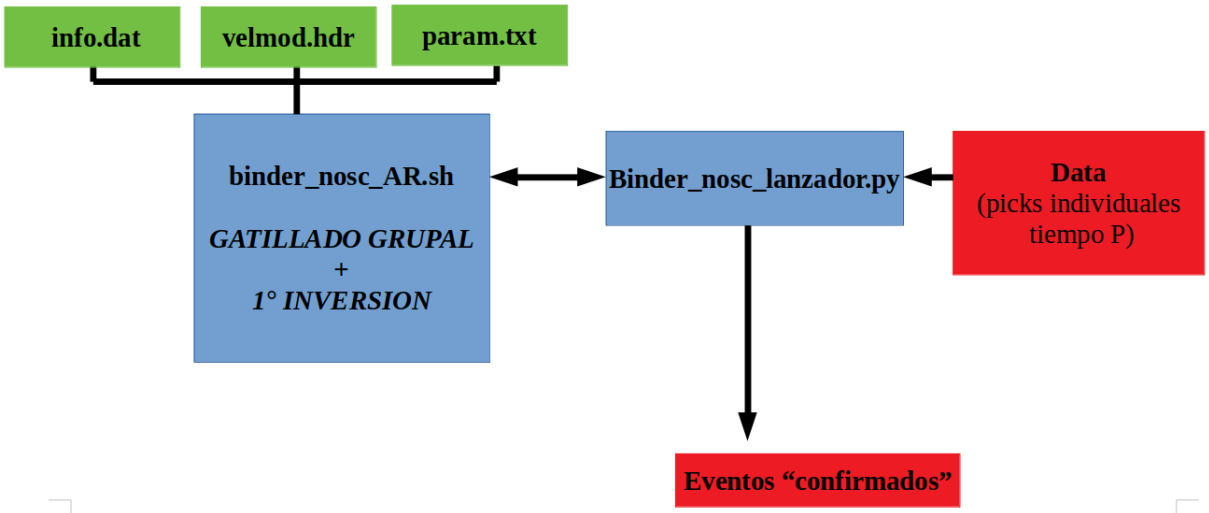


Figura C8.2: Algoritmo de uso de nuestros códigos de localización relacionados a Binder.

C8.2. Uso Binder_nosc_AR:

Procedemos a explicar el código desde su comando en la terminal:

```
/binder_nosc_AR picker_input.txt param.txt
```

Archivos de entrada:

1. **picker_input.txt**: Nombre del archivo de picks para determinado periodo. Este es el output del gatillado individual en el siguiente formato:

```
[ FECHA_UNIX   ESTACIÓN   CANAL   FASE(P ó S)   0   _ ]
```

2. **param.txt:**

   - *eventThreshold*: Cantidad mínima de triggers contiguos en un rango de tiempo para asignar un evento (5 recomendado para este estudio).
   - *AssociationThreshold*: error en $[s]$ que un trigger puede tener, comparado con la llegada teórica de una onda de un potencial sismo

en la zona (Según un modelo de velocidades (1-D en este caso, para P) aplicadas en un enjambre de potenciales sismos debajo de la red, donde el programa calcula el tiempo ideal en que llegarían las fases de las estaciones), para que el trigger esté asignado a ese potencial sismo (2-3[s] recomendado).

- *p/sPickresidualCutoff*: Si la modelacion de un "evento" cumple los requisitos de eventThreshold y asociationThreshold, intenta localizar el evento usando los triggers con un hipocentro estimado dado por el modelo. Así esta línea elimina los picks que lancen residuales mayores al parámetro escrito (recomendado 1.0 – 1.5 [s]).

- *p/sEventWindow:* La cantidad máxima de segundos entre 2 triggers que pueden venir del mismo Evento. Se estima como la cantidad máxima de segundos que una onda p/s demora en cruzar la red/cantidad de estaciones completa Se comparan los triggers que caen dentro de ventanas de este largo. Se recomienda entre 3-5[s].

- *eventCutoff:* 120[s]. debe ser mas grande que p/sEventWindow. Probablemente el tiempo durante el cual se modela el viaje de los rayos del "hipocentro" teórico. (125 % de p/sEventWindow recomendado).

- *mstackWindow* + *stackSpacing* + *stackDepth*: (90.0, 10.0 y 100.0 recomendados) Parámetros asociados a distancias entre nodos de prueba para posibles sismos.

- *depthStart* + *depthStep* + *maxDepthTry*: Parámetros relacionados al esparcimiento de los nodos de manera vertical. Estos mentados nodos son la "red" y sub-cubiculos donde se lanza el modelo bajo la red superficial y se prueban muchos hipocentros teóricos ubicados en cada nodo. La profundidad puede depender de la morfología de la zona sísmica (Recomendados 0.0, 2.0 y 100.0 en este estudio) .

- *eventFilename*: (events.txt→ Archivo de salida: nombre del archivo de salida con las propiedades de los eventos que binder_nosc nos declara como sísmicos).

- *unusedPickFilename*: (unused.txt → Archivo de salida: triggers no usados por que no corresponden a nigun evento según el programa).

- *networkConfigFilename*: (info.dat → Archivo de entrada: Info de las estaciones: lat,lon, elevacion,etc.) y bordes de la ventana sísmica.
- *velmodFilename*: (velmod.hdr→ Archivo de entrada: modelo de velocidad con el cual queremos calcular los tiempos de llegada teoricos y encontrar/localizar eventos).

3. **velmod.txt** : Modelo de velocidad para la ventana donde estamos probando/ubicando eventos sísmicos. Es importante verificar con un para de pruebas que el modelo elegido de resultados lo mas precisos posibles.

```
[ VP [m/s]    PROFUNDIDAD [km] ]
```

4. **info.dat** : Info de las estaciones con las que se modela la primera inversión. Las cuatro ultimas líneas corresponden a las fronteras de grilla donde se modelarán las inversiones sísmica. El modelo no hará cálculos para posibles eventos fuera de esta ventana. A continuación el formato seguido; mantener fijas las constantes numéricas y los valores que dicen IRIS:

```
[ XXX(n°est.) IRIS 1.0 24 3T XXX(num°est.) xx(n°est.) xxxx(nomb.estación) ...
Lat. Lon. Eleva. f.inic(yyyymmdd) horainic f.fin(yyyymmdd) horafin IRIS ]
```

Ej:

```
> 001 IRIS 1.0 24 3T 001 01 AC01 -26.14 -70.60 0346 2019.01.01 00:00 2019.11.31 23:59 IRIS
> 020 GRID 1.0 24 ZZ 020 20 L901 -24.00 -73.00 0000 2019.01.01 00:00 2019.11.31 23:59 Outline
```

Donde se finaliza el texto con cuatro lineas que contengan las esquinas de la grilla (**Outline**)

Archivos de salida:

1. **events.txt**: Registra los picks que gatillaron, según el script, un evento sísmico. Por cada evento detectado y localizado se anota una primera fila de datos del evento seguido de tantas filas como picks asociados a este, repitiendo dentro del archivo tantas veces como eventos se detecten para la

traza:

1)° FILA (evento ubicado)

```
[ unix_time year month day hour min sec lat lon depth ....
 #picks GAP RMS(rootmeansquare) ]
```

2)° FILA (picks gatillantes de un evento según Binder)

```
[ estacion unix_time residuo_inversión ...
; tiempo_absoluto_viaje_onda (?) fase _ ]
```

* A través del script de Python auxiliar, se crea un archivo de salida para cada carpeta de resultados existente

* Los resultados obtenidos también llevan un tiempo muerto para despejar las llegadas S que puedan manchar la primera estimación.

2. **unused.txt** : Picks no usados, que no se pudieron combinar con al menos 4 otros para confirmar un evento según Binder_nosc_AR.

C8.3. Comentario Final:

En vista de los numerosos parámetros, conviene hacer pruebas con un rango menor de tiempo donde sepamos de antemano algunos eventos en la zona para calibrar y mejorar la calidad de la estimación. Conviene ajuste la grilla agrandando un grado(∼111[km]) cada extremo (últimas líneas de *info.hdr*) amén de tener incluidas todas las estaciones presentes en el estudio, con el fin de no quitar eventos cercanos al borde. Verifique que el modelo de velocidad esté bien ajustado buscando literatura para la zona, compare resultados con eventos ubicados por otra institución. Finalmente verifique que el resto de parámetros de la estimación sean los adecuados.

C9. Inspección Manual y Relocalización

Sin embargo, la inversión y primera estimación del apartado anterior tiene una gran limitación: Es bastante imprecisa debido a que se fabrica exclusivamente con datos de onda P, puesto que detectar automáticamente la S en todas y cada una de las detecciones está mas allá de lo que nuestros gatillados ofrecen. Esto provoca errores a la hora de localizar pues el método de iteraciones no siempre logra calcular el hipocentro artificial, debiendo entregar el hipocentro tentativo que está clavado en algún nodo de la grilla. Capturando manualmente llegadas S, o lanzando otro método sobre las horas confirmadas con eventos son las mejores formas hasta ahora de relocalizar exitosamente los sismos detectados.

Por otra parte, Binder erróneamente calcula eventos con llegadas producto de falsos positivos; para eliminarlos primero reagrupamos todos los eventos repartidos en las carpetas-resultado con *buscaeventos.py*, creamos simogramas menores acotados alrededor de 1-2 minutos luego de cada supuesto evento en cuestión (según Binder, utilizando *sismogramas_MINI.py*). Luego aplicamos el script de inspección manual *filtro_falsospositivos.py* para revisar cada bloque de sismogramas e ir descartando los eventos falsos.

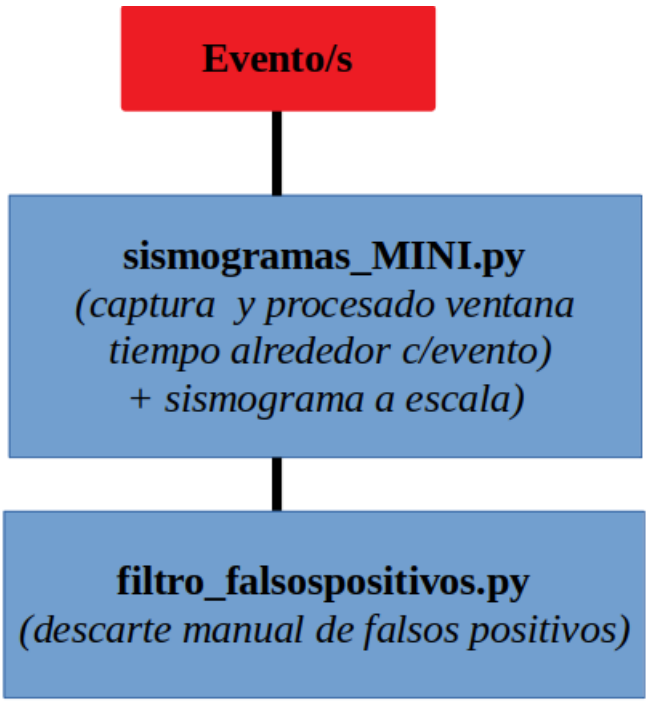


Figura C9.1: Diagrama de flujo para la detección manual de falsos positivos en el catalogo de eventos confirmados por Binder. *sismogramas_MINI.py* es exactamente el mismo script clonado de *sismogramas.py* solo que esta vez se recoge el catalogo emitido por *buscaeventos.py*; A pesar de utilizar hasta el momento solo la imagen resultante del sismograma para la detección manual de filtro_falsospositivos.py, un afinado de los parámetros de gatillado en "MINI" que maximice la cantidad de llegadas detectadas serviría para un posible re-calculo de hipocentros con Binder en una suerte de relocalización.

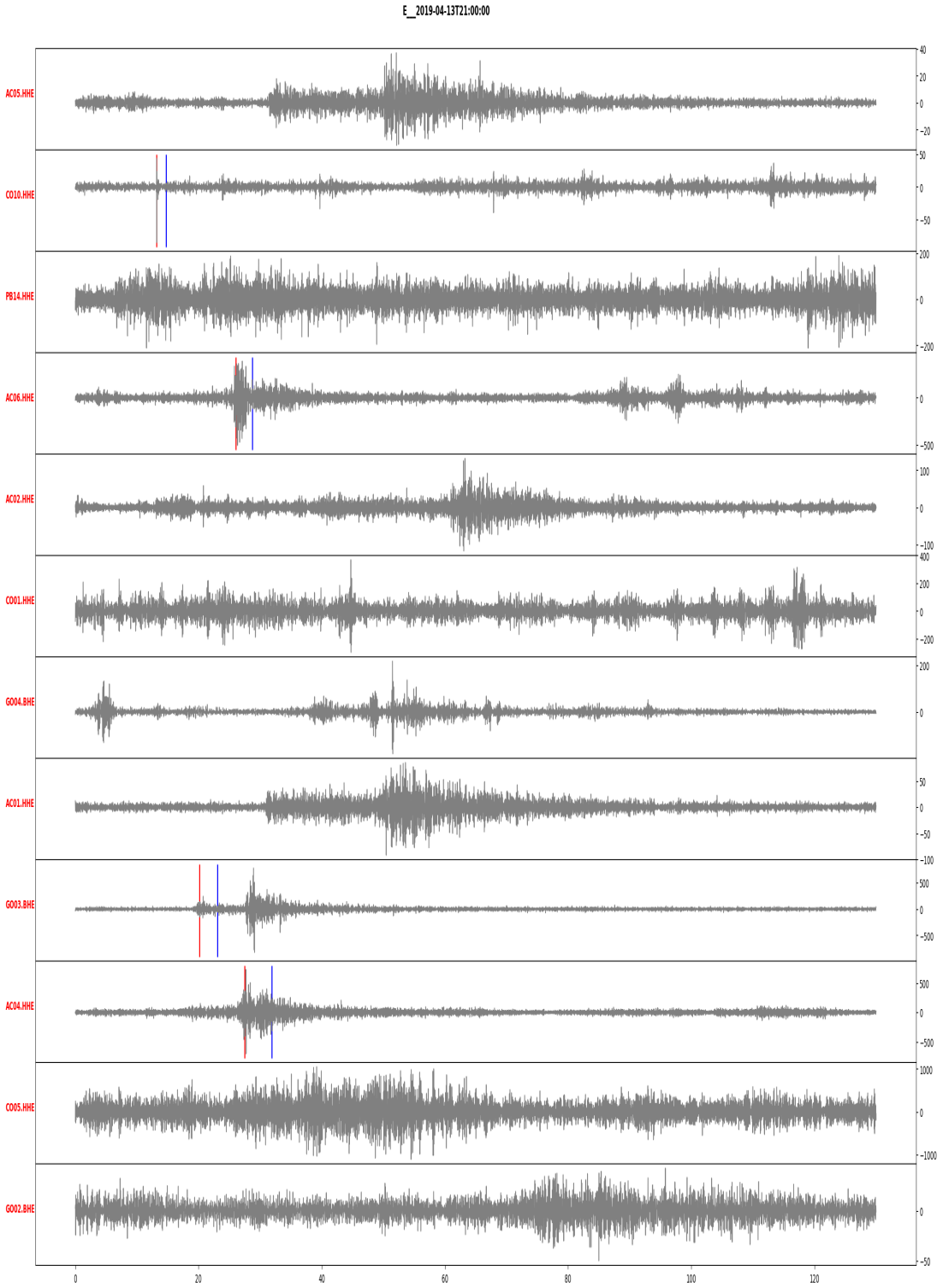


Figura C9.2: Ejemplo de un evento **no-localizable**. A pesar de que se observan un par de estaciones que efectivamente tendrían un origen sísmico(GO03 y AC04), la mayoría de las picks detectadas con las que se ubicó el supuesto evento, no tienen significado sismico o no son relamente observables. Con solo dos picks correctos no se puede hacer correctamente un calculo de 4 incógnita; se sugiere que este evento es de insuficiente magnitud y está demasiado lejos de la red de estaciones para ser localizado.