Boot camp for RESCEU KAGRA project

Seonjun Kwon

RESCEU, Department of Physics, The University of Tokyo

Contents

1 Basic Settings

We assume that you use LINUX OS.

There are many packages for Python language which CANNOT be used in Windows environment.

In this section, you can learn how to install packages, revise programs, etc..

1.1 LINUX

1.1.1 Start LINUX

Open terminal. You can see that for example for me,

```
seonjunkwon@resceuX13:~$
```

Of course your name and your machine's name are different from mine; The important thing is that you can enter command after the symbol $.

The window of this terminal you are seeing now is your home directory.

1.1.2 Directory

Doing all thing in home directory is NO-efficient. So, we need to move to directory we use.

Open terminal, enter command below.

```
$ cd Documents/
```

Tip: Enter Docu and push Tab-key, you can write 'Documents' more easily.

You moved to Documents directory.

Next, check the recent directory.

After # is annotation.

```
$ ls # Check what is in this directory
$ ls / # See root directory
$ ls .. # See the directory above a step
$ ls ~ # See home directory
$ ls -a # See all files including hidden files
etc.
```

For more information and more options, please enter

```
$ ls -help
```

We can make/remove the directory(file).

```
$ mkdir NewDirectory # Make an empty directory
$ rmdir NewDirectory # Remove an EMPTY directory
$ rm -rf NewDirectory # Remove a directory and all the files in which
```

With rmdir, we can remove empty directory only. If you want to remove a directory in which files exist, you can use 'rm -rf'.

We can make/remove a test file.

```
$ vi test.txt # Make an empty text file
```

You will enter to VI. Type ':wq' to exit(make an empty file).

```
$ mkdir TestDirectory # Make a directory
$ mv test.txt TestDirectory # Move test file to TestDirectory
$ vi test2.txt
 # Make a new empty text file: In VI, please type ':wq' and enter(return).
$ mv test2.txt renamed.txt # Rename test2.text to renamed.txt
$ mv TestDirectory/test.txt ./
 # Move test.txt in TestDirectory to current directory
$ cp test.txt TestDirectory/ # Copy text.txt to TestDirectory
```

You can make, move and copy a file.

1.2 VI

1.2.1 Start VI

VI has command mode, visual mode and insert mode.

Turn on a terminal and make a file.

```
$ vi text.txt
```

(If there exists text.txt, we can revise that file.)

The terminal will be changed to VI editor, especially command mode.

In command mode, you can enter to visual mode by entering below.

```
v : Enter to visual mode (Select per letters)
V : Enter to visual line mode (Select per sentences)
```

You can go back to command mode by enteing 'esc' key.

In command mode, you can enter to insert mode by entering (1 of) those commands.

```
a : Enter from the next letter of cursor
A : Enter from the end of this column
i : Enter from the previous letter of cursor
I : Enter from the start of this column
o : Enter with empty column under this column
O : Enter with empty column above this column
s : Enter deleting a letter on cursor
```

In insert mode, you can go back to command mode by entering 'esc' key.

You can save the file and exit VI command mode, and go back to terminal by entering below.

```
:wq
```

1.2.2 Command Mode

For more information, enter below IN TERMINAL.

```
$ vi -h
```

In command mode in VI, we can use move keys, especially

```
h : left key
```

```
j : down key
k : up key
l : right key
w : Move to the next word
b : Move to the previous word
gg : Move to the first column
G : Move to the last column
( : Move to the previous sentence
) : Move to the next sentence
{ : Move to the pervious paragraph
} : Move to the next paragraph
etc.
```

You can delete column(s) by

```
dd : Delete this column
d3 : Delete 3+1=4 columns
etc.
```

You can copy and paste column(s) by

```
yy : Copy this column
y5 : Copy 5+1=6 columns
p : Paste next to cursor
P : Paste in front of cursor
etc.
```

And you can do below.

```
/[word] : Find [word]
n : Next find result
u : Undo
[ctrl] r : Redo
etc.
```

You can check and use another convenient methods to revise files by

```
$ vi -h
```

1.2.3 Visual Mode

You can enter visual mode from command mode, by entering

```
v
```

You can select from the cursor position before entering visual mode.

```
[Arrow keys] : Select an area
~ : Convert the letter type and go back to command mode;
big to small or small to big
d : Delete the area and go back to command mode
c : Delete the area and ENTER TO INSERT MODE
y : Copy the area and go back to command mode
[esc] : Do nothing and go back to command mode
```

1.2.4 Insert Mode

You can revise the file in insert mode, and this is almost same as that of Windows memo or wordpad.

1.3 SSH

1.3.1 Create an Account

Using SSH, you can enter to the supercomputer, RESCEUBBC or IGWN computer, etc. for example.

At first, talk to your supervisor and create your account. If you received email about SSH, please follow that guideline.

If you want to have your own SSH key(public and private), open the terminal and enter below.

```
$ ssh-keygen
```

SSH keys are stored in the '∼/.ssh' folder.

You will have 'id_rsa' file or 'id_ed25519' file and that's '.pub' file. The file '.pub' is a public file, and the file without an extension is a private file. You should not disclose your private file to anyone other than yourself. The file to be uploaded to the authentication website is also a public file, so you do not need to upload a private file. The private file on your computer is used to log in using SSH, with the public file uploaded to the authentication website.

1.3.2 Login

In terminal, enter below.

```
$ ssh [account name]@[address]
$ ssh seonjun.kwon@resceubbc.resceu.s.u-tokyo.ac.jp # for example
```

You can login with your own password, or public-private keys.

For various options, you can enter below, but I think it is necessary without any options.

```
ssh
scp help
```

1.3.3 Upload and Download

You can upload and download the file or directory with 'scp'.

In local terminal,

```
$ scp [file] [account name]@[address]:[target directory] # Upload a file
$ scp -r [directory] [account name]@[address]:[target directory]
 # Upload a directory
$ scp [account name]@[address]:[directory]/[file] [Downloading directory]
 # Download a file
```

1.4 GIT

You can upload your files to GIT in order for your colleagues to see and check the file, and also you can download the file from your GIT. It is very useful for your remote works.

1.4.1 Initial Settings

Talk to your supervisor, and make your git address for your thesis. It should have the 'readme.md' file.

After that, you should do a little work in local.

In your local computer, you should make a local directory to share your file.

```
$ mkdir gitshare
```

You can set the directory name for your own, not only 'gitshare'.

Make an initial environment.

```
$ cd gitshare
~gitshare/$ git init
~gitshare/$ git remote add origin [Your git address]
~gitshare/$ git branch -M main
~gitshare/$ git push -u origin main
```

Once you added the remote address, you don't have to add the remote git address.

You have to check your git's branch,

```
~gitshare/$ git branch
```

You can see your all branch, and the branch with '*' is the branch selected.

You can make a branch and change the selected branch.

```
~gitshare/$ git branch [branch name] # Make a branch
~gitshare/$ git checkout -b [branch name]
 # Checkout from the branch and make a new branch
~gitshare/$ git branch -d [branch name] # Delete a branch
```

For more information, you can see Git book.

1.4.2 Upload

First, you should 'add' your file and second 'commit', and push(means upload) the committed file.

```
~gitshare/$ git add [file name] # Upload a file
~gitshare/$ git add .
 # Upload all files, CAUTION FOR THIS COMMAND!!
   IF THERE DOESN'T EXISTS THE FILE IN LOCAL, THAT EXISTS IN GIT,
   THE FILE WILL BE DELETED.
~gitshare/$ git commit -m [message] # Commit
~gitshare/$ git push # Push
~gitshare/$ git push -f # Push forcely
```

For more information, you can see Git book.

1.4.3 Download

You can download(merge) files from git to local.

```
~gitshare/$ git pull # Pull(Download), it can occur an error.
~gitshare/$ git pull -f
 # Pull forcely,
   IF YOU ARE CONFIDENT THAT
   YOU CAN CHANGE ALL YOUR LOCAL FILES TO THAT IN GIT.
```

You can undo your previous commitments, add tag, fetch forcely, etc.. For those information, you can see Git book.

1.4.4 Set the Access Token

You can set the access token, and SHOULD set it if you are using Github.

LIGO git: Go to the "User Settings - Access tokens". And then click "Add new token" and set the token's name, expire date, and the scopes(permissions).

Github: Go to your profile and enter "Settings - Developer settings". Enter "tokens(classic)" and click "Generate new token". Set the token's name, expire date, and the permissions.

You have to save the token in your own storage: this should not be leaked. If not saved, you cannot see the token so you have to regenerate a token.

And then set the remote path.

```
~gitshare/$ git remote hub https://<Username>:<Token>@github.com/<Address>.git
```

If you are using LIGO git, it is almost same.

```
~gitshare/$ git remote origin https://<Username>:<Token>@git.ligo.org/<Address>.git
```

Then, you can copy and link your project in LIGO git to github, which does not expire even if you graduate. (We assume that the local path is 'gitshare'.)

```
$ git pull origin main
$ git push hub main
```

If you worked locally,

```
$ git add .
$ git commit -m "<Your commit message>"
$ git push origin main
$ git push hub main
```

Caution: Uploading a file over 100MB in Github is very annoying. Make sure to check if you have files over 100MB in size locally.

1.5 Conda and Python

When you are doing research, you should make a program for your research, using Python3.

1.5.1 Download Conda

First, download installation file.

```
$ wget https://repo.anaconda.com/archive/Anaconda3-2024.06-1-Linux-x86_64.sh
```

I recommend to download the newest version.

Run the installation file.

```
bash Anaconda3-2024.06-1-Linux-x86_64.sh
```

We have to set the path.

```
$ source ~/.bashrc
$ vi ~/.bashrc
```

In VI, you add

```
export PATH="/home/username/anaconda3/bin:$PATH"
```

where 'username' is your user name.

1.5.2 Use Python 3.10 with Conda

I recommend the Python version 3.10, and you can also use another version of Python.

Let's create a virtual environment.

```
$ conda create -n python3.10 python=3.10
```

We can see that d

```
Proceed ([y]/n)?
```

Type y.

You can see the environments using below.

```
$ conda info -e
```

To run python file, you have to activate the python environment.

```
$ conda activate python3.10
```

1.5.3 Set Path

You have to set path to run your package. It can be set automatically, but if not, you cannot run the package you installed.

PLEASE DO THIS SECTION ONLY IF AN ERROR OCCURED IMPORTING DOWNLOADED PACKAGE. If it works without any path settings, you can move to next section.

PLEASE ASK YOUR SUPERVISOR OR LINUX EXPERT ABOUT THAT PROBLEM IF YOU DON'T HAVE A CONFIDENCE.

If you use local, you have to check where is your Python. And edit '.bashrc' file.

```
$ which python3
$ vi .bashrc
```

In VI, you can add that phrase in the bottom of that file,

```
export PATH=[Your Python path]:$PATH
```

If you made a mistake, a temporarily solution is to enter this phrase in the bottom of that file.

```
export PATH=$(getconf PATH)
```

AND PLEASE ASK YOUR SUPERVISOR OR LINUX EXPERT.

1.5.4 Install Packages

We have to install the necessary packages.

```
$ conda activate python3.10
$ pip install numpy matplotlib bilby gwpy pycbc
```

And the 'argparse' package will help you to do testing and running file, or multi-processing.

```
$ pip install argparse
```

If you have to install more packages, please install WITH CONDA, not pip.

```
$ conda install -n python3.10 [package-name]
```

If this doesn't work,

```
$ conda install -n python3.10 -c conda-forge [package-name]
```

When we have to install a specific version of the package, we can use version number. for example,

```
$ pip install numpy==1.14.5
$ conda install tensorflow=2.5
```

You can see we have 2 =s in pip, and 1 = in conda.

For more information about pip and conda,

```
$ pip help
$ conda
```

1.5.5 Python Tutorial

You have to study Python before make your own program.

We consider you to have a basic knowledge of the Python language. If not, please see and study Python Tutorial.

1.5.6 Run .py File

You can run your own Python file you made.

```
$ python3 test.py
```

If you set your Python as 'python3.10' not 'python3'(especially for example Conda environment), you can run with

```
$ python3.10 test.py
```

1.5.7 Make a Program

You can run a file, which isn't '.py' file. For example, take that the file name is 'test', not 'test.py'.

In the first of file, you have to add

```
#!/usr/bin/env python3
```

And run the file

```
$ ./test
```

The running language will be Python 3.

1.5.8 Argparse Packge

You can run a file with various options using argparse package.

Make a file(with Python language). For convenient, the file name will be 'test'.

python3

```python
#!usr/bin/env python3

# Set initial environment
import argparse
parser = argparse.ArgumentParser(
    prog = 'ProgramName',
    description = 'What the program does',
    epilog = 'Text at the bottom of help')
parser.add_argument('-c',
                    '--count',
                    type=int,
                    default=1,
                    help='The number of simulation times')
args = parser.parse_args()

# Define a function
def run(n):
    print(n)
for n in range(args.count):
    run(n)
```

And print from 1 to 10.

```
$ ./test -c 10
```

Of course, you can run with entering this command,

```
$ python3 test -c 10
```

You can use this package for other purposes, and for more information, you can see Argparse Tutorial.

1.5.9 Nohup and Tail

Using 'nohup', you can activate a program in background, when you have to logout from ssh server.

```
$ nohup python3.10 test.py &
```

If you didn't enter '&', the program is activated in foreground, so if you have to logout during the program running, you have to type '&' at the last of the command.

If the program is running in foreground, you cannot interact with the shell(terminal) until the end of the running.

'tail' prints the last 10 lines of each file to standard output.

To see the last 10 lines of the nohup file, you can enter below.

```
$ tail nohup.out
```

If you want to see the status in real time, you can enter the following.

```
$ tail -f nohup.out
```

Of course, this state can escape by entering Ctrl-c.

If you did multi-processing so nohup says nothing, you can use 'nohup' and '-u'. For example,

```
$ nohup python3.10 -u test.py &
```

For more information about 'tail', you can type that

```
$ tail --help
```

1.6 Data Access

1.6.1 GWOSC

1.6.2 O4 data

2 Mathematics, Physics, and Statistical Methods

2.1 Mathematics

2.1.1 Gaussian Distribution

The probability density function of Gaussian distribution is as follows, where μ is the mean and σ is the standard deviation.

$$\mathrm{N}(\mu, \sigma^2): \quad \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{1}$$

So, we can see that the log of that is a quadratic function.

$$\ln P(x) = -\frac{(x-\mu)^2}{2\sigma^2} - \ln\sigma - \frac{1}{2}\ln 2\pi \tag{2}$$

For more information, please move into Wikipedia: Normal Distribution.

2.1.2 Chi Distribution

The chi distribution is a probability distribution with positive real numbers as its domain. We have to distinguish between this and the chi-square distribution.

When k variables x_k have a standard normal distribution, we define that the following value y has a k-degree chi distribution.

$$y = \sqrt{\sum_{i=1}^{k} {x_i}^2} \tag{3}$$

Two famous examples are the Rayleigh distribution as a 2-degree chi distribution and the Maxwell-Boltzmann distribution as a 3-degree chi distribution.

The probability density distribution is as follows.

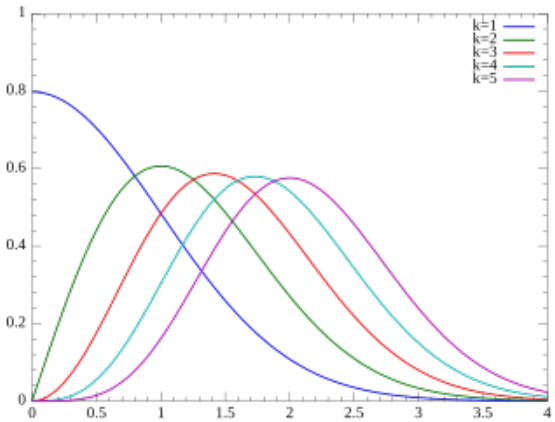

Figure 1: The probability density function of chi-distribution.

$$f(x;k) = \frac{x^{k-1}e^{-x^2/2}}{2^{(k/2)-1}\Gamma(k/2)} \quad (x > 0) \tag{4}$$

For more information such as cumulative distribution function, visit Wikipedia: Chi Distribution.

2.1.3 Spherical Coordinate System

To exhibit an isotropic distribution, an understanding of the spherical coordinate system is required.

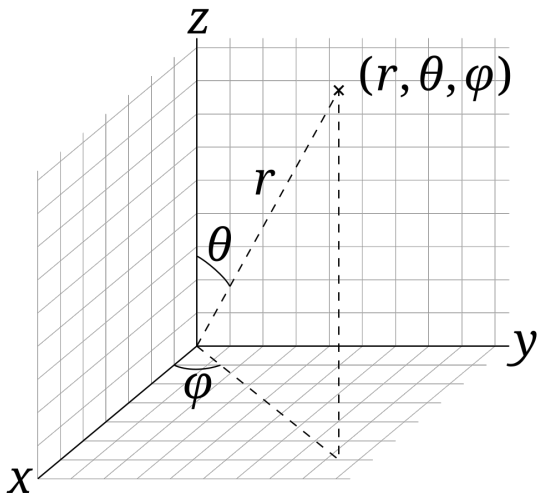

Figure 2: Spherical Coordinate.

We define the origin as O, the point we think as P, and the projection of P to xy-plane as P'.

The variables are as follows.

r: Radius, Distance from coordinate origin ($r \geq 0$)

θ: Inclination, Angle from the direction $+z$ to the line OP ($0 \leq \theta \leq \pi$)

φ: Azimuth, Angle from the direction $+x$ to the line OP' ($0 \leq \theta < 2\pi$)

In the integration of orthogonal and spherical coordinates, the relationship follows.

$$\int dV = \int d^3\mathbf{x} = \int_0^\infty dr \int_0^\pi r d\theta \int_0^{2\pi} r \sin\theta d\varphi \tag{5}$$

Let us assume that events occur isotropically and the inclination angle is $\theta \sim \theta + d\theta$. Of course, the radius and azimuth angle have any possible value.

Given θ, the number of events is proportional to $\sin\theta$(Sine distribution). It is easy to see by referring to equation (5).

For more information such as coordinate conversion, differentiation or integration, visit Wikipedia: Spherical Coordinate System or read *Arfken and Weber, 7th edition.*

2.1.4 Fourier Transformation

Fourier transform is a transformation that decomposes a function over time or space into frequency components. The definition is as follows.

$$\hat{h}(f) = \int_{-\infty}^{\infty} dt\ h(t) e^{-2\pi i f t} \tag{6}$$

Note that the output of the Fourier transform is a complex domain of frequency, not a real number domain.

The Fourier inverse transformation is defined as follows.

$$h(t) = \int_{-\infty}^{\infty} dt\ \hat{h}(f) e^{2\pi i f t} \tag{7}$$

The Fourier transform is represented by the letter $\mathcal{F}$.

Depending on the literature or Python packages, the proportional constant may be multiplied during Fourier transform/inverse transform, so check it carefully.

The following basic properties are established in Fourier transform.

$$ah(x) + bg(x) \overset{\mathcal{F}}{\Longleftrightarrow} a\hat{h}(f) + b\hat{g}(f) \qquad \text{linearity} \tag{8}$$

$$h(t - t_0) \overset{\mathcal{F}}{\Longleftrightarrow} e^{-2\pi i f t}\hat{h}(f) \qquad \text{time-shift} \tag{9}$$

$$e^{2\pi i f_0 t}h(t) \overset{\mathcal{F}}{\Longleftrightarrow} \hat{h}(f - f_0) \qquad \text{frequency-shift} \tag{10}$$

$$h(at) \overset{\mathcal{F}}{\Longleftrightarrow} \frac{1}{|a|}\hat{h}\left(\frac{f}{a}\right) \qquad \text{time-scaling} \tag{11}$$

The conversion theorem is as follows.

$$(g * h)(t) \equiv \int_{-\infty}^{\infty} g(\tau)h(t - \tau)\ d\tau \overset{\mathcal{F}}{\Longleftrightarrow} \hat{g}(f)\hat{h}(f) \tag{12}$$

Some important Fourier transformations are as follows.

$$1 \overset{\mathcal{F}}{\Longleftrightarrow} \delta(f) \tag{13}$$

$$\delta(t) \overset{\mathcal{F}}{\Longleftrightarrow} 1 \tag{14}$$

$$e^{iax} \overset{\mathcal{F}}{\Longleftrightarrow} \delta\left(f - \frac{a}{2\pi}\right) \tag{15}$$

Fourier transform is used in various fields such as solving linear differential equations, quantum mechanics, signal processing, and analysis.

For more information such as the applications, Fourier transform tables, etc., visit Wikipedia: Fourier Transformation.

2.2 Physics

2.2.1 Chirp Mass

The chirp mass(In the 2-body system) determines the leading-order orbital evolution of the system as a result of energy loss from emitting gravitational waves.(Wikipedia: Chirp Mass)

We can define the chirp mass as

$$\mathcal{M} = \frac{(m_1 m_2)^{3/5}}{(m_1 + m_2)^{1/5}} = \mu^{3/5} M^{2/5} \tag{16}$$

Where

$$M = m_1 + m_2, \quad \mu = \frac{m_1 m_2}{m_1 + m_2} \tag{17}$$

The frequency of the gravitational wave (generated by the binary orbit) evolves using the chirp mass,

$$\dot{f} = \frac{96}{5}\pi^{\frac{8}{3}}\left(\frac{G\mathcal{M}}{c^3}\right)^{\frac{5}{3}} f^{\frac{11}{3}} \tag{18}$$

So we can know the chirp mass as

$$\mathcal{M} = \frac{c^3}{G}\left(\frac{5}{96}\pi^{-\frac{8}{3}} f^{-\frac{11}{3}} \dot{f}\right)^{\frac{3}{5}} \tag{19}$$

For more information, visit Wikipedia: Chirp Mass.

2.2.2 Polarizations of Gravitational Waves

There are two polarizations (plus, cross) in gravitational waves.

Example 2.1. Let's consider the following metric perturbations.

$$g_{\mu\nu} = \eta_{\mu\nu} + h_{\mu\nu}, \quad h \equiv \eta^{\mu\nu} h_{\mu\nu} \tag{20}$$

The plane wave solutions of gravitational waves (in the general relativity) traveling in the positive z-direction satisfy the following.

$$\bar{h}_{\mu\nu} \equiv h_{\mu\nu} - \frac{1}{2}h, \quad \Box \bar{h}_{\mu\nu} = 0 \tag{21}$$

If you think about it in the same way you learn from electro-magnetics, these positive z-direction plane waves have two polarizations. This solution defines the following TT gauge(Transverse-traceless gauge).

$$h^{\mathrm{TT}}_{ab}(t,z) = \begin{pmatrix} h_+ & h_\times \\ h_\times & -h_+ \end{pmatrix} \cos\left[\omega(t - z/c)\right] \tag{22}$$

This gravitational wave acts on the interval ds^2 as follows.

$$\begin{aligned} ds^2 = &- c^2 dt^2 + dz^2 + \{1 + h_+ \cos\left[\omega(t - z/c)\right]\} dx^2 \\ &+ \{1 - h_+ \cos\left[\omega(t - z/c)\right]\} dy^2 + 2h_\times \cos\left[\omega(t - z/c)\right] dx dy \end{aligned} \tag{23}$$

Here I only listed the results, and see pages 3 to 9 of *Maggiore* (2008) for a detailed explanation.

The gravitational wave can have more polarizations (if we don't assume that the general relativity is correct), but this is not covered here.

Figure 3: The plus(left) and cross(right)-polarizations of the gravitational waves.

2.2.3 Waveform of Gravitational Waves

Before two large objects (such as black holes) collide with each other, they have time to orbit. At this time, energy is released in the form of gravitational waves. Initially, low frequencies and low amplitude gravitational waves occur. And as the two objects continue to revolve and get closer to each other, their frequencies and amplitudes increase. Eventually, when the two objects approach a certain level or higher, the frequencies and amplitude increase dramatically.

Here is the waveform of the GW150914, which is the first detected gravitational-wave signal.

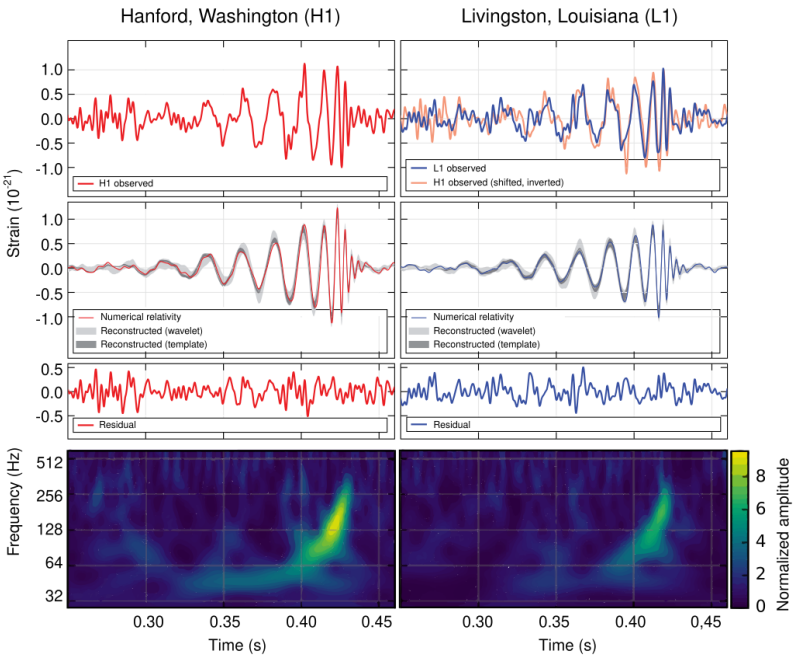

Figure 4: The waveform of GW150914. Figure from Wikipedia: First Observation of Gravitational Waves

Let's assume that two massive objects are orbiting each other in a circular orbit.

We define as $t_{\text{ret}} = t - r/c$ (when the gravitational wave waveform we observe occurs) and f as the frequency of the gravitational wave. θ is the angle between the observer and the straight line connecting the gravitational source and the axis of rotation of the gravitational source celestial body (the co-latitude) and ϕ is the phase.

And then we can estimate the waveform of the gravitational-wave.

$$h_+(t) = \frac{4}{r}\left(\frac{G\mathcal{M}}{c^2}\right)^{\frac{5}{3}}\left(\frac{\pi f}{c}\right)^{\frac{2}{3}}\frac{1+\cos^2\theta}{2}\cos\left(2\pi f t_{\text{ret}} + 2\phi\right) \tag{24}$$

$$h_\times(t) = \frac{4}{r}\left(\frac{G\mathcal{M}}{c^2}\right)^{\frac{5}{3}}\left(\frac{\pi f}{c}\right)^{\frac{2}{3}}\cos\theta\sin\left(2\pi f t_{\text{ret}} + 2\phi\right) \tag{25}$$

For more information such as the derivation of the formula, see *Maggiore*(2008) Sec. 3 and 4.

2.2.4 The Time to Coalescence

About elliptical orbit, let's think that we have the samimajor axis a_0, the eccentricity e_0 and the orbital period T_0. The subscript 0 means that they are initial value. And $M = m_1 + m_2$ is the total mass, μ is the reduced mass.

We can see the orbital period from the fact that the gravitational wave frequency f is twice the orbital frequency.

We have the coalescence time τ_0 as

$$\tau_0(a_0, e_0) \simeq 9.829\text{Myr}\left(\frac{T_0}{1\text{hr}}\right)^{\frac{8}{3}}\left(\frac{M_\odot}{M}\right)^{\frac{2}{3}}\left(\frac{M_\odot}{\mu}\right)f(e_0) \tag{26}$$

$$F(e_0) = \frac{48}{19}\frac{1}{[g(e_0)]^4}\int_0^{e_0} de\ \frac{[g(e)]^4(1-e^2)^{5/2}}{e(1+\frac{121}{304}e^2)} \simeq \frac{768}{429}(1-{e_0}^2)^{\frac{7}{2}} \tag{27}$$

$$g(e) = \frac{e^{12/19}}{1-e^2}\left(1+\frac{121}{304}e^2\right)^{\frac{870}{2299}} \tag{28}$$

For circular orbit case, because eccentricity e_0 is 0, so the coalescence time τ_0 is

$$\tau_0 \simeq 9.829\text{Myr}\left(\frac{T_0}{1\text{hr}}\right)^{\frac{8}{3}}\left(\frac{M_\odot}{M}\right)^{\frac{2}{3}}\left(\frac{M_\odot}{\mu}\right) \tag{29}$$

For more information such as the derivation of the formula, see *Maggiore*(2008) from Sec. 4.1.1 to 4.1.3.

2.2.5 Gravitational Lens

2.2.6 Signal

The time-domain signal-strain-data we can see from the detector is expressed as a linear combination of h_+ and $h_\times$.

$$h(t) = F_+(\hat{\mathbf{n}})h_+(t) + F_\times(\hat{\mathbf{n}})h_\times(t) \tag{30}$$

Where $\hat{\mathbf{n}}$ is the direction vector. This is determined from polarization and azimuth. For more information, see *Maggiore*(2008) Sec. 7.2.

The Python package Pycbc has a function of entering hplus, hcross, polarization, and azimuth to project it. For more information about this, see Pycbc Waveform Documents.

2.2.7 Signal and Noise

When detecting gravitational waves, the amplitude of noise is greater than that of ordinary signals. Therefore, it is necessary to identify the frequency characteristics of the signal in noise by Fourier transforming them.

[Physical part: formula]

2.2.8 Detection Analysis

2.2.9 Interferometers

2.3 Statistical Methods

2.3.1 Bayes Theorem

Given two probability distributions and conditional probabilities, the probability of two occurring simultaneously is as follows.

$$P(A \cup B) = P(A|B)P(B) = P(B|A)P(A) \tag{31}$$

Bayes' theorem is a way to find a conditional probability $P(B|A)$ when you know a conditional probability $P(A|B)$. The formula is as follows.

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \tag{32}$$

The general formula is as follows.

$$P(x_0|A) = \frac{P(x_0)P(A|x_0)}{P(A)} = \frac{P(x_0)P(A|x_0)}{\int dx\ P(x)P(A|x)} \tag{33}$$

For more information such as application example, visit Wikipedia: Bayes Theorem.

2.3.2 Maximum Likelihood Method

The likelihood is a probability known empirically beforehand. By formula(33), we define the likelihood $\mathcal{L}$ as follows.

$$\mathcal{L}(A|x) = P(x|A) \tag{34}$$

Here, A is the surrounding environment (parameters, such as the mass of the black holes, the distance from observer, etc.) that produces the result value, and x is the result value. We know the result(x) from observations, and we need to find the surrounding environment(A) that produces the result.

Let the parameters such that the probability distribution $P(x|A)$ is maximized be called the estimated value A_0. In this case, we can estimate that the observation value x is more likely to be generated by parameter A_0 compared to other parameters.

Therefore, when x is fixed, we need to find A so that the likelihood has an extreme value.

$$A_0 = \arg\max_A \mathcal{L}(A|x) \tag{35}$$

We can obtain parameters that derive the maximum likelihood in the preferred method of the two. (the latter is usually preferred here)

$$0 = \frac{\partial \mathcal{L}(A|x)}{\partial A} \tag{36a}$$
$$0 = \frac{\partial \ln \mathcal{L}(A|x)}{\partial A} \tag{36b}$$

For a thorough mathematical description or an explanation of an algorithm, etc., visit Wikipedia: Maximum Likelihood Estimation.

2.3.3 Metropolis-Hastings Method

Markov Chain Monte Carlo(MCMC) method is an example of rejection method. In this method, we need a probability distribution(it is OK if not-renormalized).

Metropolis-Hastings method is one of the examples of MCMC. In this method, we need a probability distribution and a proposal distribution. Here I explain the algorithm only, not proof.[1][2]

The set of all states is $\{\Gamma\}$. We have a probability distribution $P(\Gamma)$ and a proposal distribution $q(\Gamma|\Gamma')$,

Step 0: Prepare an initial state $\Gamma_0 \in \{\Gamma\}$

loop t

1. Make next candidate state Γ' randomly from the proposal distribution $q(\Gamma'|\Gamma_t)$
2. Make a uniform random number $r \in [0, 1]$
3. Select the next state Γ_{t+1} based on r as

$$\Gamma_{t+1} = \begin{cases} \Gamma' & (r \leq a(\Gamma_t \to \Gamma') \\ \Gamma_t & (\text{else}) \end{cases} \tag{37}$$

Where the acceptance probability is as follows.

$$a(\Gamma_t \to \Gamma') = \min\left(1,\ \frac{P(\Gamma')q(\Gamma_t|\Gamma')}{P(\Gamma_t)q(\Gamma'|\Gamma_t)}\right) \tag{38}$$

The example problem is as follows.

Example 2.2. The probability density function is as follows.

$$p(x) = 0.3\frac{1}{2\sqrt{2\pi}}\exp\left(-\frac{(x+5)^2}{8}\right) + 0.7\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(x-3)^2}{2}\right) \tag{39}$$

The proposal density funcion is a normal distribution function, as follows.

[1] The notation and explanation are based on Okubo(2022), https://github.com/compsci-alliance/many-body-problems

[2] The proof is in Chib & Greenberg(1995).

$$q(x^*|x) = \frac{1}{10\sqrt{2\pi}} \exp\left(-\frac{(x^* - x)^2}{200}\right) \tag{40}$$

Set initial state as random normal distribution with mean 0 and standard deviation 10. The probability density is as follows.

$$p_0(x_0) = \frac{1}{10\sqrt{2\pi}} \exp\left(-\frac{{x_0}^2}{200}\right) \tag{41}$$

Calculate the logarithm of the acceptance rate.

$$\ln a = \ln p(x^*) - \ln p(x_{t-1}) + \ln q(x_{t-1}|x^*) - \ln q(x^*|x_{t-1}) \tag{42}$$

Make a random r from $(0, 1]$ and compare r and the acceptance rate.

$$\ln \alpha = \min(0, \ln a) \tag{43}$$

$$\ln r \leq \ln a \rightarrow x_t = x^* \tag{44}$$

$$\ln r > \ln a \rightarrow x_t = x_{t-1} \tag{45}$$

Repeat this 5000 times($t = 5000$).

The code is as follows.

Python3

```
import numpy as np
import math
import scipy.stats as stats
import matplotlib.pyplot as plt
def p(x):
    return 0.3 * (2 * math.sqrt(2 * math.pi)) ** (-1) * np.exp(-(x + 5) ** 2 /
    ↪  8) + 0.7 * (math.sqrt(2 * math.pi)) ** (-1) * np.exp(-(x - 3) ** 2 / 2)
def logp(x):
    return np.log(p(x))
T = 5000
sigma = 10
for i in range(1, T+1):
    candidate = np.random.normal(size = 1, loc=learn[i-1], scale=sigma)
    loga = logp(candidate) - logp(learn[i-1]) + stats.norm(candidate,
    ↪  sigma).logpdf(learn[i-1]) - stats.norm(learn[i-1],
    ↪  sigma).logpdf(candidate)
    logalpha = min(0, loga)
    r = np.random.uniform(size=1)
    logr = math.log(r)
    if logr <= logalpha:
        learn = np.append(learn, candidate)
    else:
```

```
        learn = np.append(learn, learn[i-1])
grid = np.arange(-10, 10, 0.01)
plt.plot(grid, p(grid))
plt.hist(learn, bins = 40, density = True)
plt.show()
```

The result is as follows.

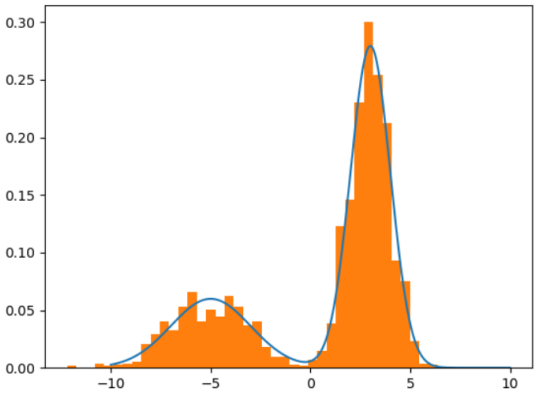

Figure 5: Metropolis-Hastings example code result

This part deals with similar content as Wikipedia: Metropolis-Hastings Algorithm, so you can read and compare them.

2.3.4 Inverse Transform Method

When we have a probability distribution $p(x)$, we can make a simulation program to generate that distribution.

The cumulative probability distribution $P(x)$ is as follows.

$$P(x) = \int_{-\infty}^{x} p(x)\ dx \tag{46}$$

Let's think about the slope of $P(x)$. This is a probability distribution $p(x)$.

$$y \equiv P(x) \tag{47}$$

$$dy = p(x)dx \tag{48}$$

Therefore, we can know that dy is proportional to the probability distribution. What that does mean?

Let us generate random numbers y that have uniform distribution in the rages of $[0, 1]$. If we choose a specific $y_0 \sim y_0 + dy$, we can get the number of y such that $y_0 \leq y \leq y_0 + dy$. The number is proportional to $p(x)$.

$$dy = p(x)dx \propto p(P^{-1}(y)) \tag{49}$$

Therefore, we can correspond y to x, as follows.

$$x = P^{-1}(y) \tag{50}$$

So the x have the probability distribution $p(x)$.

I think the explanation might be difficult to understand, so let's see an example.

Example 2.3. The range of inclination angle in GWpy package is $-\pi/2 \leq \theta \leq \pi/2$, so if we want the isotropic spherical distribution, we should use cosine distribution rather than sine distribution. See equation (5).

The distribution is as follows.

$$p(x) = \begin{cases} \frac{1}{2}\cos x & (-\pi/2 \leq x \leq \pi/2) \\ 0 & (\text{else}) \end{cases} \tag{51}$$

$$P(x) = \begin{cases} 0 & (x \leq -\pi/2) \\ \frac{1}{2}\sin x + \frac{1}{2} & (-\pi/2 \leq x \leq \pi/2) \\ 1 & (x \geq \pi/2) \end{cases} \tag{52}$$

$$x = P^{-1}(y) = \arcsin(2y - 1) \quad (0 \leq y \leq 1) \tag{53}$$

We will generate 10,000 number of random uniform numbers in $[0, 1]$ and make cosine distribution. And for comparison, we will draw an exact cosine distribution function. The code is as follows.

Python3

```
import numpy as np
import matplotlib.pyplot as plt
a = []
for _ in range(10000):
    x = np.random.uniform(0,1)
    y = np.arcsin(2 * x - 1)
    a.append(y)
plt.hist(a, bins=100, density=True)
plt.plot(np.linspace(-np.pi/2, np.pi/2,100), 0.5*np.cos(np.linspace(-np.pi/2,
↪  np.pi/2,100)))
plt.show()
```

The result is as follows.

When making a prior set using bilby, we can use a cosine distribution provided in the package.

Python3

```
import numpy as np
import matplotlib.pyplot as plt
import bilby
priors = bilby.core.prior.PriorDict()
```

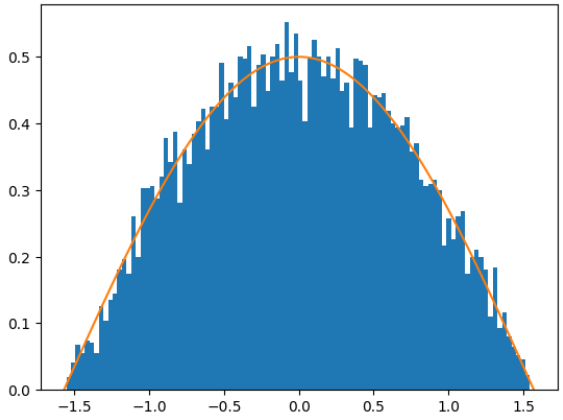

Figure 6: Code result and cosine distribution

```
priors['dec'] = bilby.core.prior.Cosine(name = 'dec')
samples = priors.sample(10000)
plt.hist(samples['dec'], bins=100, density=True)
plt.plot(np.linspace(-np.pi/2, np.pi/2,100),
         0.5*np.cos(np.linspace(-np.pi/2, np.pi/2,100)))
plt.show()
```

The result is almost same as Figure 6.

For more information, visit Wikipedia: Inverse Transform Sampling.

2.3.5 Confusion Matrix and ROC Curve

The confusion matrix is used to evaluate the performance of the algorithm. It can be expressed as follows.

	Prediction: Positive	Prediction: Negative
Actual result: Positive	True Positive (TP)	False Negative (FN)
Actual result: Negative	False Positive (FP)	True Negative (TN)

Table 1: Confusion Matrix

False positive rate(False alarm rate) is the proportion of data predicted as positive among data that are actually negative.

$$\text{False positive rate} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{54}$$

Sensitivity(True positive rate) is the proportion of data that is actually positive, and the prediction result is also positive.

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{55}$$

We can get the likelihood, accuracy, etc.. For more information about those, visit Wikipedia - Confusion Matrix.

Receive Operating Characteristic(ROC) curve is a plot that illustrates the performance of a binary classifier model at varying threshold values. The horizontal axis of the ROC curve is false positive rate, and the vertical axis is sensitivity.

The comparison curve is the $y = x$ straight line, which means that if the ROC curve is above $y = x$, the judgment was more correctly made than complete random classification, and if it is below, the classification was worse than complete random classification. For more information about those, visit Wikipedia - Receive Operating Characteristic.

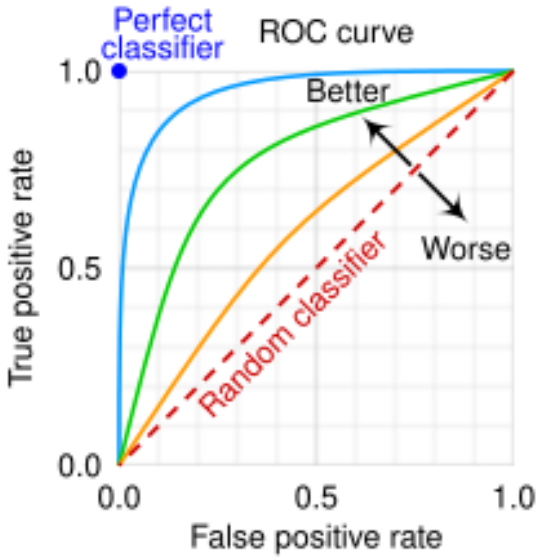

Figure 7: ROC curve - by Wikipedia

3 Data Analysis Tutorial

GWOSC Tutorial: Optional.[3]

3.1 Tutorial 1

3.1.1 Tutorial 1.1 - View Data with GWOSC Package

With 'gwosc' pacakge, you can

1. Know what event are in a (particular) catalog: for example, you can know GW150914, GW151012,... are in GWTC-1(confident) event catalog.

2. Find gravitational-wave datasets for a specific time

3. Know the event time(GPS) of the gravitational-wave detection

4. Get the URLs of a specific gravitational-wave event data

5. Filter data satisfying certain parameters

etc..

For more understanding, you can visit GWOSC 0.7.1 documentation.

3.1.2 Tutorial 1.2 - Fetch Timeseries

'gwpy', 'pycbc', 'bilby' are Python packages which is used to analyse gravitational-wave data.

Depending on the analyzing method, there are cases in which data formatted by a specific package should be used. For a few sections from here you learn:

How to open and plot the data,

How to convert the timeseries-data into a frequencyseries-data,

How to generate and inject signal data,

How to change the format so that the data to be compatible with each other,

etc..

In this section, you fetch open-data(timeseries data) with 'gwpy', 'pycbc' and 'bilby' package.

1. GWpy[4]

Prepare the environment.[5]

Python3

```
import warnings
warnings.filterwarnings("ignore", "Wswiglal-redir-stdio")
import gwpy
from gwpy.timeseries import TimeSeries
```

[3]You can do GWOSC TUTORIAL 2024: tutorial 1.1-3.3 first. But this tutorial referred to GWOSC tutorial much, so I recommend doing one and reading another as a dictionary.

[4]https://gwpy.github.io/docs/stable/overview/

[5]It will be explained next time, please distinguish it from PyCBC TimeSeries.

Fetching open-data, we need to

1. Know the event time(GPS).

2. Choose the interferometer.

At first, we use GWOSC package to know the GPS time. In this step, we fetch GPS time and set the segment time(for this example, GPS segment to 10 seconds around GPS time).

You can print gps and segment for check also.

python3

```
import gwosc
from gwosc.datasets import event_gps

gps = event_gps('GW190412')
segment = (int(gps)-5, int(gps)+5)
```

And then, we need to download open data from interferometer data.

The interferometers we can use in GWOSC package are as follows.

'L1': LIGO-Livingston

'H1': LIGO-Handford

'G1': GEO-600

'V1': (Advanced) Virgo

'K1': KAGRA

Using TimeSeries (from gwpy.timeseries), we can fetch the data.

python3

```
FetchedData = TimeSeries.fetch_open_data('L1', *segment, verbose = True)
```

You can plot easily using GWOSC TimeSeries.

python3

```
plot = FetchedData.plot()
plot.show() # If you can't see the plot by previous line, run this line also.
```

2. PyCBC

In PyCBC, we don't need to know GPS time exactly.(So we don't need GWOSC.)

We can use interferometers in PyCBC almost same as in GWpy.

python3

```
import pycbc
from pycbc.catalog import Merger
import matplotlib.pyplot as plt
```

```
merger = Merger("GW150914")
strain = merger.strain('H1')
plt.plot(strain.sample_times, strain)
plt.show()
```

The default sampling time is 1/4096 second, and we will learn downsampling and removing low-frequence in the next section.

3. bilby

In bilby, we need to fetch data from GWpy.

python3

```
import bilby
import gwosc
from gwosc.datasets import event_gps
import gwpy
from gwpy.timeseries import TimeSeries

#Using GWpy, download the gravitional-wave data.
gps = event_gps('GW150914')
Data = TimeSeries.fetch_open_data('L1', gps - 2, gps + 2, sample_rate = 4096,
↪  cache = True)

#Set interferometer and strain data using bilby
L1 = bilby.gw.detector.get_empty_interferometer('L1')
L1.set_strain_data_from_gwpy_timeseries(Data)
```

3.1.3 Tutorial 1.3 - Generate a Signal Timeseries

With PyCBC, we can generate a signal. In this example, using the approximant 'IMRPhenomD', we generate a binary-black-holes signal.

python3

```
from pycbc.waveform import get_td_waveform
sampling_rate = 4096
hp, hc = get_td_waveform(approximant = 'IMRPhenomD',
                         mass1 = 36,
                         mass2 = 29,
                         delta_t = 1 / sampling_rate,
                         f_lower = 20,
                         distance = 410)
```

You can use other approximants (for example TaylorT4 for binary-neutron-stars) or add other parameters (for example spins). See here.

3.1.4 Tutorial 1.4 - Resample Timeseries Data

1. GWpy

From tutorial 1.2, we have fetched data named FetchedData.

First, let's check sampling rate and delta t(the inverse of the sampling rate).

python3

```
print(FetchedData.sample_rate)
print(FetchedData.dt)
```

We can resample(downsample) the data easily.

python3

```
ResampledData = FetchedData.resample(1024)
print(ResampledData.sample_rate)
print(ResampledData.dt)
```

2. PyCBC

From tutorial 1.2, we have fetched data named strain.

First, let's check sampling rate and delta t.

python3

```
print(strain.sample_rate)
print(strain.delta_t) #not dt
```

We can resample(downsample) the data with 'resample_to_delta_t'.

python3

```
from pycbc.filter import resample_to_delta_t

resampled = resample_to_delta_t(strain, 1/1024)
print(resampled.sample_rate)
print(resampled.delta_t)
```

3.1.5 Tutorial 1.5 - Resize a TimeSeries

With PyCBC, we can resize the PyCBC timeseries.

1. Time Slice

(0,4) means that we only have time from 0 to 4 seconds left in these timeseries.

python3

```
import pycbc
from pycbc.waveform import get_td_waveform
hp, hc = get_td_waveform(approximant = "IMRPhenomD",
```

```
                          mass1 = 36,
                          mass2 = 29,
                          delta_t = 1 / sampling_rate,
                          f_lower = 20,
                          distance = 410)
hp.start_time = hp.start_time + 2
hc.start_time = hc.start_time + 2
hpslice = hp.time_slice(0,4)
hcslice = hc.time_slice(0,4)
```

2. Crop

(0.5,0.5) means that we throw away the first 0.5 seconds and the last 0.5 seconds in these timeseries.

python3

```
import pycbc
from pycbc.waveform import get_td_waveform
hp, hc = get_td_waveform(approximant = "IMRPhenomD",
                          mass1 = 36,
                          mass2 = 29,
                          delta_t = 1 / sampling_rate,
                          f_lower = 20,
                          distance = 410)
hp.start_time = hp.start_time + 2
hc.start_time = hc.start_time + 2
hpcrop = hp.crop(0.5,0.5)
hccrop = hc.crop(0.5,0.5)
```

3. Resize

(65536) means that we resize the data, length to 65536. The length is (duration time)×(time interval). Here, 'resize' adjusts the length by adding a sample before or after the data if necessary.

python3

```
import pycbc
from pycbc.waveform import get_td_waveform
hp, hc = get_td_waveform(approximant = "IMRPhenomD",
                          mass1 = 36,
                          mass2 = 29,
                          delta_t = 1 / sampling_rate,
                          f_lower = 20,
                          distance = 410)
hp.start_time = hp.start_time + 2
hc.start_time = hc.start_time + 2
hpcrop = hp.resize(65536)
hccrop = hc.resize(65536)
```

3.1.6 Tutorial 1.6 - Generate FrequencySeries Data

With PyCBC, we can generate a Frequencyseries data. The method is similar to the Timeseries part. The return is a frequency domain gravitational waveform.

python3

```
from pycbc.waveform import get_fd_waveform
hp, hc = get_td_waveform(approximant = 'IMRPhenomD',
                         mass1 = 36,
                         mass2 = 29,
                         delta_f = 0.25,
                         f_lower = 20,
                         distance = 410)
```

For more information, see PyCBC Document: Waveform

3.1.7 Tutorial 1.7 - Change Format

We can change data format, for example, PyCBC to GWpy, etc..

We will change a PyCBC timeseries to GWpy timeseries. The basic method is to copy and paste the numbers of time data and the corresponding time series data, so it is easy to convert to the format not introduced here. Try it yourself.

Example material: PyCBC timeseries(named 'sample', time interval: 1/4096 s, start time: 0 s)

python3

```
import pycbc
from pycbc.types import TimeSeries
import gwpy
from gwpy.timeseries import TimeSeries as GWTimeSeries
GWsample = GWTimeSeries(
sample,
dt = 1/4096,
t0 = 0)
```

3.2 Tutorial 2

3.2.1 Tutorial 2.1 - Generate a White Gaussian Noise

When generating a white gaussian noise, the PSD must be flat.

You can generate the PSD with bilby and generate a white gaussian noise. In this example, we make a flat PSD whose frequency range is from 0 to 2048Hz and frequency interval is 0.25Hz

python3

```
import numpy as np
import bilby
freq = np.linspace(0,2048,8193)
psd = 8193 * [1. * 1e-40]
```

```
Ifo = bilby.gw.detector.get_empty_interferometer('K1')
Ifo.power_spectral_density = bilby.gw.detector.PowerSpectralDensity(
    frequency_array = freq,
    psd_array = psd)
```

Let's make a white gaussian noise from the flat PSD we made.

python3

```
Ifo.set_strain_data_from_power_spectral_density(
    sampling_frequency=4096,
    duration=4096,
    start_time=0)
```

You can check if this noise is gaussian.

python3

```
plt.hist(Ifo.strain_data.time_domain_strain, bins = 100)
plt.show()
```

You can check if this noise is white.

python3

```
plt.loglog(Ifo.strain_data.frequency_array,
↪  abs(Ifo.strain_data.frequency_domain_strain))
plt.show()
```

3.2.2 Tutorial 2.2 - Fetch PSD and Generate Coloured Noise

With bilby package, you can generate a coloured PSD for simulation. Each of the interferometers in the bilby package has its own PSD by default.

python3

```
import bilby
Ifo = bilby.gw.detector.get_empty_interferometer('K1')
Ifo.set_strain_data_from_power_spectral_density(
    sampling_frequency=4096,
    duration=4096,
    start_time=0)
```

We fetched KAGRA simulation PSD and set strain data from the PSD. You can choose 'L1', 'H1' or 'V1', etc for other interferometers.

You can see the PSD.

python3

```
import matplotlib.plot as plt
```

```
plt.loglog(Ifo.power_spectral_density.frequency_array,
↪  Ifo.power_spectral_density.psd_array)
plt.show()
```

You can check if this noise is gaussian.

python3

```
plt.hist(Ifo.strain_data.time_domain_strain, bins = 100)
plt.show()
```

You can check if this noise is coloured.

python3

```
plt.loglog(Ifo.strain_data.frequency_array,
↪  abs(Ifo.strain_data.frequency_domain_strain))
plt.show()
```

You can plot the noise timeseries.

python3

```
plt.plot(Ifo.strain_data.time_array, Ifo.strain_data.time_domain_strain)
plt.show()
```

3.2.3 Tutorial 2.3 - Inject a Signal to the Noise

We generated a noise and signal. To inject, the two timeseries must be compatible (have the same format), and the time interval and duration of the two must be the same.

If the signal's format is GWPy(I recommend), then convert the bilby noise to GWPy timeseries and inject. PyCBC is also recommended, and the method is almost same. See PyCBC Document and GWPy Document.

1. GWPy

python3

```
from gwpy.timeseries import TimeSeries
Ifo.set_strain_data_from_power_spectral_density(
    sampling_frequency=4096,
    duration=4,
    start_time=0)
Noise = TimeSeries(
    Ifo.strain_data.time_domain_strain,
    dt = 1/4096,
    t0 = 0)
```

2. bilby

We can inject a signal to the noise, using bilby. I only put it here for the reference that there is an example of this. See bilby Document for more information.

3.3 Tutorial 3

3.3.1 Tutorial 3.1 - Q Transforms

We can do Q-transform with GWpy. For more informations such as basic knowledge, visit this article.

Material: GWpy timeseries (named 'data')

python3

```
import warnings
warnings.filterwarnings("ignore", "Wswiglal-redir-stdio")
import gwpy
import matplotlib.pyplot as plt

qt = data.q_transform(frange=(30, 500))
plot = qt.plot()
plot.colorbar(label="Normalised energy")
```

We can set colorbar range also. This helps to see signals hidden in loud noise (e.g., shot noise, etc.).

```
plot.colorbars[0].mappable.set_clim(0,20)
plot.refresh()
plot
```

3.3.2 Tutorial 3.2 - Matched Filtering Test

We use PyCBC.

Material. The formats should be PyCBC timeseries.

1. PSD

2. Strain data (ex. noise of the detector, fetched strain data, or signal injected to the noise, etc)

If you generate noise strain data, please use the PSD data. If you fetch strain data, please make(estimate) the PSD from the strain data fetched. The sampling frequency interval has to be same between PSD and the strain data.

3. Sample waveform data (hplus timeseries)

The sampling time interval of the sample has to be same as that of the strain data.

Let's assume that we have the strain data(named 'strain'), PSD data(named 'psd'), and the sample waveform data(named 'hp').

First, match the length of hp and sample.

python3

```
import numpy as np
import pycbc
from pycbc.waveform import get_td_waveform
```

```
from pycbc.filter import matched_filter
hp.resize(len(sample))
```

The time of the strain data has to be different from that of the sample. If you generated the noise data, please do this work. (If you fetched strain data, this work is not necessary.)

python3

```
hp.start_time = hp.start_time - (len(sample) * hp.delta_t + 1)
```

Do matched filtering. We should remove time corrupted by the template filter and the psd filter. We assume that the time-length of the sample is long enough, so 2e set the cropping time to the first 4 seconds and the last 4 seconds. (This time can change fluidly of course.)

```
snr = matched_filter(hp,
                     sample,
                     psd=psd,
                     low_frequency_cutoff=20)
snr = snr.crop(4, 4)
peak = abs(snr).numpy().argmax()
snrp = noisesnr[peak]
snrp = abs(snrp)
time = snr.sample_times[peak]
print(time)
print(snrp)
```

3.3.3 Tutorial 3.3 - Parameter Estimating

Using bilby, we can estimate the parameter(The source of the gravitational-wave).

Material

1. Strain data

You can fetch the strain data, or use generated noise data or signal injected to noise.

2. Waveform Arguments (bilby)

3. Prior (bilby)

First, set prior. Bilby runs Monte Carlo process or Dynesty process to estimate the parameters. In this example, we will estimate the GW150914 parameters.

python3

```
import bilby
from bilby.core.prior import Uniform, PowerLaw
prior = bilby.core.prior.PriorDict()
prior['chirp_mass'] = Uniform(name = 'chirp_mass',
                              minimum = 26.0,
                              maximum = 30.0)
prior['mass_ratio'] = Uniform(name = 'mass_ratio',
```

```
                                minimum = 0.5,
                                maximum = 1)
prior['phase'] = Uniform(name = 'phase',
                         minimum = 0,
                         maximum = 2 * np.pi)
prior['geocent_time'] = Uniform(name = 'geocent_time',
                                minimum = 2 - 0.1,
                                maximum = 2 + 0.1)
prior['a_1'] = 0.0
prior['a_2'] = 0.0
prior['luminosity_distance'] = PowerLaw(alpha = 2,
                                        name = 'luminosity_distance',
                                        minimum = 50,
                                        maximum = 2000,
                                        unit = 'Mpc',
                                        latex_label = '$d_L$')
```

3.3.4 Tutorial 3.4 - Visualing the Results

3.4 Tutorial 4

3.4.1 Turotial 4.1 - Loop, Parallel Processing

Using 'argparse' package, we can do parallel processing.

We set variables and make a program that prints the number.

python3

```
import argparse
import multiprocessing
from multiprocessing import Pool
parser = argparse.ArgumentParser(
    prog = 'ProgramName',
    description = 'What the program does',
    epilog = 'Text at the bottom of help')
parser.add_argument('-c',
                    '--count',
                    type=int,
                    default=1,
                    help='The number of simulation times')
parser.add_argument('-p',
                '--process',
                type=int,
                default=1,
                help='The number of processes for multi-processing')
args = parser.parse_args()
def run(n):
```

```
    print(n+1)
pool = Pool(args.process)
inputs = range(args.count)
pool.map(run, inputs)
```

In bash, we will run this program running 20 steps with 4 processes. (The file name is test.py for example.)

```
$ python3.10 test.py -c 10 -p 4
```

The result will be mixed number, not the order of 1,2,3,... .

This is because 20 tasks are divided and executed by 4 processes. If we set '-p 1', the result will be the order of 1,2,3,... .

We cannot do bilby multi-processing by this method. Instead, multiprocessing is done inside the bilby package. In tutorial 3.3(parameter estimation), we set run_sampler as follows.

python3

```
result_short = bilby.run_sampler(
        likelihood,
        prior,
        sampler = 'dynesty',
        outdir = 'short',
        label = 'GW150914',
        conversion_function = bilby.gw.conversion.generate_all_bbh_parameters,
        nlive = 250,
        dlogz = 1.,
        clean = True,
        npool = args.process)
```

Then bilby will send message that we will multitask with (the set number of) processes.

3.4.2 Tutorial 4.2 - Saving and Reading CSV Files

After the iterative work, we can store the collected data. In this part, we will save the data as csv format.

python3

```
import csv
result = []
def run(n):
    result.append(n)
for n in range(30):
    run(n)
with open('result.csv',
      'w',
      newline = '') as saveresult:
csv.writer(saveresult).writerow(result)
```

Also, we can read a csv file.

python3

```
data = 'result.csv'
output = []
with open(data,
          'r',
          newline = '') as f:
    reader = csv.reader(f)
    for row in reader:
        output.append(row)
print(output)
```

3.4.3 Tutorial 4.3 - Drawing ROC Curve

The ROC curve is the plot of the true positive rate (TPR) against the false positive rate (FPR) at each threshold setting. For more information, visit Wikipedia page.

The higher the lower area of the curve, the better this data is classified.

Let's say we have positive and negative data separately(Named PositiveData and NegativeData as python lists). The data value at this time should be a real number.

python3

```
import numpy as np
import sklearn
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import train_test_split
print(len(PositiveData))
print(len(NegativeData))
```

Let's give positive data '1' and negative data '0', and do an analysis.

python3

```
DataTrue = [0] * len(NegativeData) + [1] * len(PositiveData)
Scores = NegativeData + PositiveData
fpr, tpr, thresholds = roc_curve(DataTrue, Scores)
```

We can plot the ROC curve.

python3

```
import matplotlib.pyplot as plt
plt.plot(fpr, tpr)
plt.legend(loc = 2)
plt.xlabel('Flase Alarm Probability')
plt.ylabel('Detection Probability')
plt.title('ROC Curve')
```

```
plt.show()
```

For more information, visit Scikit-learn homepage.

4 GstLAL

4.1 Connecting to a LIGO Data Grid

4.1.1 How to Add a SSH Key

When you are using GstLAL, it's very convenient to connect to LIGO Data Grid.

This guide follows the LIGO Computing Guide.

Because we are KAGRA member, please enter GW-Astronomy Registry.[6]

Click 'Services' on the left and click 'Click to Request' of 'LDG Account Request' among the main items.

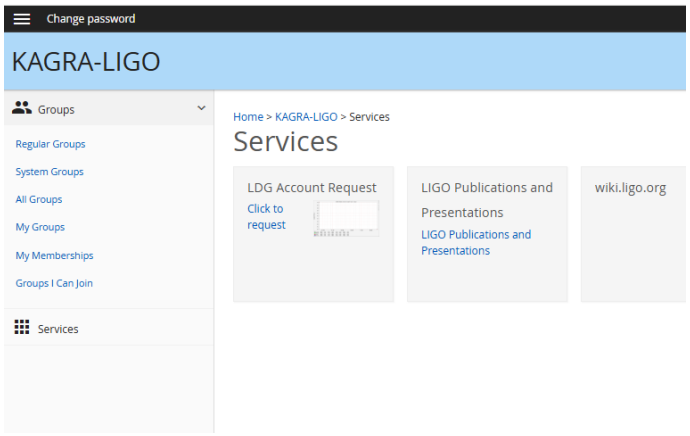

Figure 8: Add a SSH key

Affiliation has to be selected as 'Student', and the purpose of the current study has to be roughly written in the Justification. And click 'SUBMIT'.

Read the authorize form and aggree, and go to the next step.

And then click 'Authenticators'. Then you can see 'Add LDG SSH key' page, and in that page, you can add your SSH PUBLIC key. If you don't have an SSH key, see section 1.3.1.

4.1.2 Login to the LIGO Data Grid

If you uploaded the SSH key, you can enter to the LIGO Data Grid(the remote computer). It may take some time for the SSH key to be uploaded and logged in.

You can see the login hosts of Hawk, Caltech, IUCAA, LIGO-Hanford, LIGO-Livingston, Nemo(UWM) or Gwave(PSU). I usually login to Caltech host.

```
$ ssh seonjun.kwon@ldas-pcdev2.ligo.caltech.edu
```

4.2 Install the GstLAL

You can use GstLAL in LIGO Grid, or another supercomputer(for example, RESCEUBBC).

This guide was written with the help of Dr. Alvin Li.

[6]The link for offline(paper-book) users: https://registry.gw-astronomy.org/

4.2.1 Use GstLAL in LIGO Grid

Login to LIGO Grid and make a directory.

```
$ mkdir Lensing
$ cd Lensing
```

There is a package in Dr. Alvin Li's directory that he has organized for download in batches. Let's create a Conda virtual environment by copying it.(Enter without a new line)

```
$ conda create --name testlensing --clone
    /home/alvin.li/.conda/envs/tesla_o4a_analysis_240331
```

This is the end of the installation.

The name of the package should not overlap. For example, if you set it to igwn-py39, you get an error that the name overlaps.

4.2.2 Use GstLAL in Another Supercomputer

Basically, you can follow Dr. Alvin Li's TESLA guide in its entirety. It has been approved that it is okay to copy-paste all of this guide, and parts that are not in the guide will also be described here.

SSH Login

You need permission to connect to LIGO GIT on that supercomputer. Let's create an SSH key in the Ed25519 format from the home directory.

```
$ ssh-keygen -t ed25519
$ cd .ssh
$ vi id_ed25519.pub
```

Go to the SSH key page of the User Settings page of the LIGO GIT.(You need to login to the LIGO GIT.) And click 'Add new key' button.

Copy and paste the entire contents of your 'ed_25519.pub' file into the 'key' column. And click 'Add key'. Then you will get permission to clone the package files.

Conda Environment

You will need to first create a working python3 conda environment. We have a suggested conda YAML file included in this git repo tesla_conda.yaml. Download the conda yml file via

```
$ mkdir lensingenv
$ cd lensingenv
$ git archive --remote=git@git.ligo.org:alvin.li/tesla.git
    o4a_version_1.0 tesla_conda.yaml | tar -x
```

Vim into tesla_conda.yaml. In the last line, modify name: igwn-py39 to the name of the conda environment you want, for example, tesla_o4a_py39_231228.

Of course, you don't need to vim that file and change the environment name, if you don't have 'igwn-py39' already.

Enter below.

```
$ conda env create --file tesla_conda.yaml
```

Activate the environment.

```
$ conda activate igwn-py39
```

python-ligo-lw

From now on, proceed with the igwn-py39 virtual environment in operation. (If not activate, enter below.)

```
$ conda activate igwn-py39
```

Because the cluster's ligolw is not up to date (especially for ligolw_sqlite), we will need to install our own version of ligolw. To do so, first go to the conda environment directory that you have built.

For example, you can check the directory with

```
$ vi ~/.conda/environments.txt
```

And go into the directory that says igwn-py39 is in this text file. For example, (The directory may vary depending on the person or environment.)

```
$ cd ~/anaconda3/envs/igwn-py39
```

Make a new directory via mkdir src and change directory into it via cd src. Then clone the python-ligo-lw git repository. (Note that -b 1.8.x is essential.)

```
$ mkdir src
$ cd src
$ git clone git@git.ligo.org:kipp.cannon/python-ligo-lw.git -b 1.8.x
```

And then change directory, and install the python package.

```
$ cd python-ligo-lw
$ pip3 install . --prefix ~/anaconda3/envs/igwn-py39/
```

doxygen

To install doxygen, simply run

```
$ conda install -c conda-forge doxygen
```

Boost Library

For some reason, the Boost library on the clusters are deprecated. You will have to update it manually by running

```
$ conda install boost boost-cpp libboost libboost-devel libboost-headers
    libboost-python libboost-python-devel
```

GstLAL

We will be using the O3 version of GstLAL. First, go to the src directory in your conda environment.

```
$ cd ~/anaconda3/envs/igwn-py39/src
```

Clone the gstlal git repository. (Note that the -b mcvt_plus_fastpath_updated is important.)

```
$ git clone git@git.ligo.org:lscsoft/gstlal.git -b mcvt_plus_fastpath_updated
```

Change directory into gstlal.

```
$ cd gstlal
```

Then run the following command to install gstlal.(./00init.sh to -j 8; is the same line.) If you are confused, copy the contents of the guide and modify only the 'path to conda environment' part to your conda environment location, and paste it.

```
$ folders=(gstlal gstlal-ugly gstlal-calibration gstlal-burst gstlal-inspiral)
  path_to_conda_environment=~/anaconda3/envs/igwn-py39/
  for folder in "${folders[@]}" ; do \
  echo ${folder} ; \
  cd ${folder} ; \
  ./00init.sh ; ./00init.sh && ./configure --prefix=${path_to_conda_environment}
          --with-zlib=$CONDA_PREFIX && make -j 8 && make install -j 8 ; \
  cd ../ ; \
  done
```

gwpopprior

Gwpopprior is used to create a targeted population model for the targeted search.

Go to the src directory in your conda environment.

```
$ cd ~/anaconda3/envs/igwn-py39/src
```

Clone the numerical-mass-model git repository. (Note that the -b o4a_tesla_version_1.0 is important.)

```
$ git clone git@git.ligo.org:heather-fong/numerical-mass-model.git -b
    o4a_tesla_version_1.0
```

Change directory into numerical-mass-model.

```
$ cd numerical-mass-model
```

Then install gwpopprior via

```
pip3 install . --prefix ~/anaconda3/envs/igwn-py39/
```

configobj

You will also need to install configobj. To do so, run the following command. (This can be done in any directory.)

```
$ pip3 install configobj --prefix ~/anaconda3/envs/igwn-py39/
```

numexpr

If you use the igwn-py39.yml file to set up the conda environment, most likely you will also need to upgrade numexpr. To do so, run the following command. (This can be done in any directory.)

```
$ pip3 install numexpr -U --prefix ~/anaconda3/envs/igwn-py39/
```

skymap-overlap

Skymap-overlap is used to compute the overlap between two given skymaps.

Go to the src directory in your conda environment.

```
$ cd ~/anaconda3/envs/igwn-py39/src
```

Clone the skymap-overlap git repository via

```
$ git clone https://github.com/ricokaloklo/skymap-overlap.git
```

Change directory into skymap-overlap.

```
$ cd skymap-overlap
```

Then install skymap-overlap.

```
$ pip3 install . --prefix ~/anaconda3/envs/igwn-py39/
```

TESLA

Go to the src directory in your conda environment.

```
$ cd ~/anaconda3/envs/igwn-py39/src
```

Clone the tesla git repository. (Note that the -b o4a_version_1.0 is important.)

```
$ git clone git@git.ligo.org:alvin.li/tesla.git -b o4a_version_1.0
```

Change directory into tesla.

```
$ cd tesla
```

Then install tesla via

```
$ pip3 install . --prefix ~/anaconda3/envs/igwn-py39/
```

Go to your conda environment directory. Run tesla_download_gstlal_resources. This will download all the bank files, reference psds and mass model files used in GstLAL from all the previous observing runs, and will take a while (around 15 minutes).

```
$ cd ~/anaconda3/envs/igwn-py39
$ tesla_download_gstlal_resources
```

If you want to install lalsuite, enter below.

```
$ pip3 install swig --prefix ~/anaconda3/envs/igwn-py39/
```

4.3 Initialize TESLA

Make sure you have activate the conda environment.

```
$ conda activate igwn-py39
```

Set up a main project directory and go there. For example,

```
$ cd ~
$ mkdir gstlalrun
$ cd gstlalrun
$ mkdir tutorial
$ cd tutorial
```

Run TESLA with the suitable arguments, for example,

```
$ tesla_generate_base_config --group-user seonjun.kwon --verbose
```

Your name is not mine, so enter your name instead of seonjun.kwon. For the explanations for more arguments, see the guide's description.

Before we move on, you should generate a 'SciToken' to connect to gracedb. In this analysis, we cannot use X509 now, so you have to use a SciToken for LIGO and KAGRA.

```
htgettoken -a vault.ligo.org -i igwn
```

You will then be prompted to access a link. You can then copy and paste the link into your Internet browser and log in to the organization's website.

And then you will have a permission to access the GraceDB data.

Vim into the base config file.

```
$ vi base_config.yml
```

You will need to modify the following items:

[basic_information] –> project_dir: The project directory

[basic_information] –> kde_params: Default is mchirp:chieff, you can change it to mass1:mass2 (recommended for fast-TESLA-X).

[basic_information] –> x509_proxy: You don't have to revise that part.

[general_settings] –> webdir (if for some reason you want to specify a particular web directory. Otherwise leave it blank.)

Then, you should choose the event's SID(superevent ID) or GID(event ID). You can search at GraceDB. If you analyze an event of O4, type O4 in the search box. For example, I will analyze for GID G527355 (SID S241124j).

4.4 GstLAL

You are recommend to do this part in the LIGO computing grid.

4.4.1 Search for GW Superevents(Events)

At first, you can enter GraceDB and login to your account. You can enter the query in the search box, for example, interferometer name, GPS time, or ID. You can use the option about whether the **super**event is or isn't a GW event by 'is_gw' option. For help, visit Queries Help Document.

For example, you can search 'G416203' event.

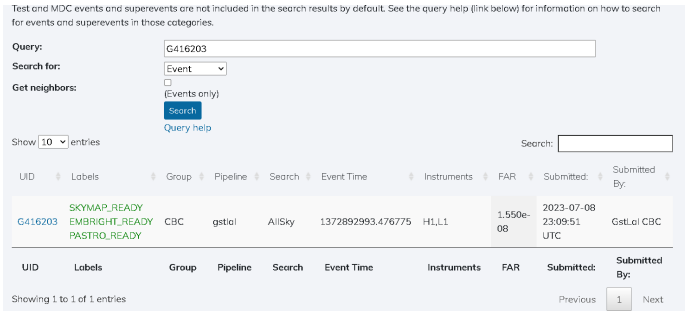

Figure 9: Search G416203 at GraceDB

4.4.2 See an Event

Activate the environment. The environment name is that you have set.

```
$ conda activate testlensing
```

If you know the superevent ID(SID) or the event ID(GID), you can see the property of the (super)event.

```
$ mkdir seeevent
$ cd seeevent
$ gracedb get event G416203
```

4.4.3 Analyze an event

First, initialize the environment and login.

```
$ conda actiavte testlensing
$ htgettoken -a vault.ligo.org -i igwn
```

Go to the home directory and make a directory to analyze.

```
$ cd
$ mkdir Projects
$ cd Projects
$ mkdir tesla_o4_runs
$ cd tesla_o4_runs
$ mkdir production
$ cd production
```

And set the environment.

```
$ cp /home/lensing.o4/Projects/tesla_o4_runs/production/base_config.yml
$ ln -s /home/lensing.o4/Projects/tesla_o4_runs/production/S230518h
$ ln -s /home/lensing.o4/Projects/tesla_o4_runs/production/S230529ay
$ ln -s /home/lensing.o4/Projects/tesla_o4_runs/production/S230601bf
$ ln -s /home/lensing.o4/Projects/tesla_o4_runs/production/S230605o
```

Vim into the base-config file.

```
$ vi base_config.yml
```

And change the part:

project_dir: (The project folder you analyze)

Webdir: /home/(your name)/public_html/Projects/tesla_o4_runs/production/

Please see inside the public_html folder.

Start analysis.

```
$ tesla_initiate_event_analysis --base-config base_config.yml --event G416203
```