RIFT Tutorial

03a: Generating a RIFT directory for a real BBH

Katelyn J. Wagner

Real Data

Setting up a RIFT run for real data is more straightforward once you have your environment and path properly configured. Make sure you have properly sourced your environment and setup your path variables before you begin.

We have an internal script called `util_RIFT_pseudo_pipe.py` which automatically gets the data and settings you need. These default options can be changed by passing an ini file to the script. As an initial test, you can run the command below, and RIFT will collect the relevant data from GraceDB and build the run directory.

First, run the following command:

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

This will provide you a browser link to complete the authentication. Copy and paste the output link into your browser and log in. Return to your terminal to continue. You should see some output like:

```
Storing vault token in /tmp/<rand>
Saving credkey to /home/albert.einstein/.config/htgettoken/credkey-igwn-default
Saving refresh token ... done
```

There is now an authentication token associated with your albert.einstein account and the following commands will work. Next, attempt the build process for GW190521 (190521g):

```
util_RIFT_pseudo_pipe.py --gracedb-id G333631 --approx IMRPhenomD --calibration C01
↪  --add-extrinsic --choose-data-LI-seglen
```

This uses basic default settings in `util_RIFT_pseudo_pipe.py`. However, the **generally preferred** method is to run with an ini file. To do this, you must get the GraceDB `coinc.xml` and you can find the production ini file in PE/O3 repository under the appropriate event ID directory. The structure of the ini file you will need is slightly different than what appears in the production ini file. To set up a run manually like this, it is best to use an ini file from another run and modify it to include the appropriate information for your event. This process is now automated with `asimov`, but it's educational for new users to see the whole process.

The ini files contain the settings used for production PE, and the main information you need from them is the `types` under `[datafind]` and the `channels` under `[data]`. This tells the pipeline where to find the

data for your real event according to information in the `coinc.xml`. You should also pay attention to `flow`, `fhigh`, `chirpmass-min`, and `chirpmass-max`. You can find the `event-time` in the same PE/O3 directory in a text file called something like `gpstime.txt`. You can copy the `example.ini` file from this directory if you do not have your own.

Make sure to modify the items mentioned above, as well as any other RIFT specific settings, such as waveform approximation, spin settings, etc.

There is also another way to do this if you already have the `coinc.xml` which contains important information about event time and central parameters for the particular event. Check GraceDB for the GID of your event. The superevent ID (SID) in GraceDB corresponds to the date of the event, and you can get the GID from the SID page (as long as you are logged in with your LIGO credentials). Then, on the command line, you can download the `coinc.xml`.

```
gracedb get file G333631 coinc.xml coinc.xml
ligolw_no_ilwdchar coinc.xml
```

Then, with the appropriate environments etc sourced, you can run `util_RIFT_pseudo_pipe.py` on the `coinc.xml` with the settings from the ini file:

```
util_RIFT_pseudo_pipe.py --use-ini `pwd`/my_ini.ini --use-coinc `pwd`/coinc.xml
↪  --use-rundir `pwd`/rundir
```

You'll also need to copy the PSDs into your run directory. For simplicity, just take them from the rundir you created above.

```
cd rundir
cp ../G333631*/*psd.xml.gz .
```

Once the run directory is created, check a `command-single.sh` and then submit the job to the cluster, just as for an injection test.

```
cd rundir/
./command_single.sh
```

There will be a bunch of output, but once you see output like the following, you know the job will run on the cluster: There might be some errors about `lalym`, but you can ignore them. Finally, you can submit the

```
 --------> Arguments  ('right_ascension', 'declination', 'phi_orb', 'inclination', 'psi', 'distance')
 .... mcsampler : providing verbose output .....
iteration Neff  sqrt(2*lnLmax) sqrt(2*lnLmarg) ln(Z/Lmax) int_var
 : 10000 12.307066439716408 6.824081525864119 5.944754763710367 -5.613989735491433 0.18414994040088967
  Worthwhile modes :  {(2, -2), (2, 2)}
```

job to the cluster using HTCondor and watch the progress:

```
condor_submit_dag -import_env master_clean.dag
watch condor_q
```

```
OWNER          BATCH_NAME                   SUBMITTED   DONE   RUN    IDLE  TOTAL JOB_IDS
katelyn.wagner master_clean.dag+54824584   3/25 20:31  24580     41    795  43111 54825855.0 ... 54876867.0
```

You always want jobs in the RUN and IDLE columns, otherwise something is wrong:

This is the setup for a BBH event. If you are interested in doing runs for a BNS, see example 03b. In general, we have migrated to use `asimov` to automate PE using RIFT for real events. See the next tutorial folder (04_asimov) to learn how to do this.