RIFT Tutorial

05: Generating Runs on Synthetic Data

Katelyn J. Wagner

Injection and recovery testing is a critical tool for validating and stress-testing our pipeline. We can assess how accurately and precisely the pipeline recovers source parameters under controlled conditions by injecting a known GW signal into simulated or low-noise background data and attempting to recover it. This process allows us to test the pipeline's robustness across a wide range of source properties (e.g., mass, spin, eccentricity), sampling configurations, and integration methods. It also helps us quantify things like the impact of waveform systematics (differences between the injected signal model and the model used for recovery) which can bias results if not properly accounted for. Basically, this type of test provides a ground truth framework for identifying biases, understanding uncertainties, and ensuring the pipeline is functioning as expected before applying it to real detector data.

Injection & Recovery

As usual, begin by sourcing your environment and/or setup script. The most common pitfall for new users is improperly set environment variables and paths. There are ways to automate sourcing these (i.e. modifying your `.bashrc`) but proceed carefully.

We will now create fake gravitational wave data and perform a test RIFT run. This is the procedure we use for systematics studies. The first step is to create the synthetic data. To do this, we will combine a

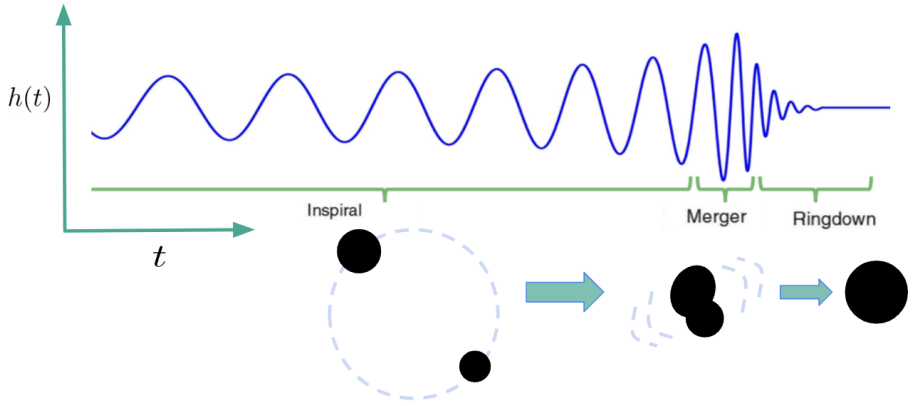

Figure 1: A sample gravitational wave signal showing the merger stages.

fake gravitational wave signal (created using a specified waveform model) with noise frames based on LIGO detector design. As expected, our data = signal + noise. The infrastructure to perform this already exists within the RIFT repository. It's best to checkout a new copy of the repo to perform these tests in. Note

that this **will not be** the source where your conda environment access scripts. Be careful! From your home directory:

```
git clone https://github.com/oshaughn/research-projects-RIT.git
```

Navigate to the `pp` directory:

```
cd research-projects-RIT/MonteCarloMarginalizeCode/Code/test/pp
```

Add this directory to your path so that the scripts can find other supporting scripts within this directory.

```
export PATH=${PATH}:`pwd`
```

The relevant scripts are `pp_RIFT` and `pp_RIFT_with_ini`. New users should use `pp_RIFT_with_ini` unless they have a reason not to, as it directly uses the setup procedures we also use for production level runs on real data. You will modify `sample_pp_w_ini.ini` (the copy that comes in `rift_O4c` should be fine), but you will need to copy PSDs from the author's home directory for now:

```
cp /home/katelyn.wagner/*psd.xml.gz .
```

Open the ini file and change the `username` to match yours in the `[make_workflow]` section of the ini file. All other settings are fine for a test run, but note that these are the values a user modifies to create custom systematics runs. You are ready to build the workflow. From this directory, run:

```
pp_RIFT_with_ini --use-ini sample_pp_w_ini.ini
```

This creates a new directory (named by "working_directory" as specified in the ini file) which contains

- An injection file containing the source parameters: `mdc.xml.gz`
- A directory containing the signal frames: `signal_frames`
- A directory containing signal+noise frames: `combined_frames`
- The event directory: `analysis_event_0/`
- A dag that contains info for condor to submit the job: `master.dag`

Navigate into the event directory and test a single ILE instance to make sure the setup worked:

```
cd test_pp/analysis_event_0/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

```
 --------> 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)}
```

the job to the cluster using HTCondor and watch the progress:

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

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

```
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 just created synthetic signals, joined them with noise to create synthetic LIGO data, setup a RIFT run directory, and submitted your run to cluster computing resources.

For more extensive testing, users typically run this type of analysis on 100 events, and change the waveforms used in the ini file. Some common pitfalls:

- `pp_RIFT_with_ini` only checks if the `signal_frames` and `combined_frames` directories *exist*, but not whether they were properly generated. Many users try to submit runs without these directories properly built. This can happen for a number of reasons, including but not limited to:
  - SSH connection timing out (e.g., getting logged out of the cluster during a build). Try using `screen`.
  - Failing to run `export PATH=${PATH}:pwd` each time you log in to build a rundir, which can cause `add_frames.py` to be unavailable to `pp_RIFT_with_ini`.
- Forgetting to source `setup.sh`, so things like the singularity image and the accounting group/user are not properly set in the sub files.

If you made a rundir but realized your settings were not correct, you can keep the `mdc.xml.gz` file, the `signal_frames` directory, and `combined_frames` directory. Delete the `analysis_event_XX` directories, the `pseduo_ini.ini`, and the `master.dag`. Then, rerun `pp_RIFT_with_ini` using the same ini file. It will detect the existence of the frames and just rebuild the analysis directories. If you'd like to keep the original analysis directories but perform a rerun, simply `cp -r` the `mdc.xml.gz` file, the `signal_frames` directory, and `combined_frames` directory to your new location, change the dir name in your ini file, and rerun `pp_RIFT_with_ini` as usual.

- If you're using the OSG, when you run `pp_RIFT_with_ini`, add the `--use-osg` flag.
- If you're using an SEOB model, when you run `pp_RIFT_with_ini`, add the `--use-gwsignal` flag to access the external waveform interface that is required for this waveform.
- For more settings information, visit the documentation.