PESummary: the code agnostic Parameter Estimation Summary page builder

Charlie Hoy[a,*], Vivien Raymond[a]

[a] *Cardiff University, Cardiff CF24 3AA, UK*

Abstract

PESummary is a Python software package for processing and visualising data from any parameter estimation code. The easy to use Python executable scripts and extensive online documentation has resulted in PESummary becoming a key component in the international gravitational-wave analysis toolkit. PESummary has been developed to be more than just a post-processing tool with all outputs fully self-contained. PESummary has become central to making gravitational-wave inference analysis open and easily reproducible.

Keywords: Parameter Estimation, Python, html, javascript, software

*Corresponding Author
Email address: charlie.hoy@ligo.org (Charlie Hoy)

Required Metadata

C1	Current code version	0.6.0
C2	Permanent link to code/repository used for this code version	https://github.com/pesummary/pesummary/tree/v0.6.0
C3	Legal Code License	MIT
C4	Code versioning system used	git
C5	Software code languages, tools, and services used	Python, Javascript, HTML, CSS, Bootstrap, Unix/MacOS
C6	Compilation requirements, operating environments & dependencies	numpy$\geq$ 1.15.4, h5py, matplotlib, seaborn, statsmodels, corner, tables, deepdish, pandas, pygments, astropy$\geq$ 3.2.3, lalsuite$\geq$ 6.70.0, ligo-gracedb, configparser, gwpy, plotly, tqdm$\geq$ 4.44.0
C7	If available Link to developer documentation/manual	See https://lscsoft.docs.ligo.org/pesummary
C8	Support email for questions	charlie.hoy@ligo.org

Table 1: Code metadata (mandatory)

1. Motivation and significance

Bayesian inference is central for many areas of science [e.g. 1, 2, 3, 4] as it allows for the identification of model parameters which best describes the collected data [e.g. 5, 6], see Section 2.1.1 for more details. Since the first detection of gravitational-waves (GWs) [7], cosmic ripples predicted by Einstein's theory of General Relativity [8, 9], Bayesian inference has been used to infer the astrophysical source properties from GWs measurements [see e.g. 10, 11, 12], understanding the properties of noise within the GW detectors [13] and understanding the astrophysical population of GW sources [14]. For a detailed review of how Bayesian inference is used within the GW community see [15]. In addition, with more areas of science entering the 'open data era', there is a need for a robust, easy to use and code agnostic software to interpret and distribute the output from all Bayesian inference codes.

PESummary, the Parameter Estimation Summary page builder, is a Python package that provides an intuitive, object-oriented user interface for displaying, interpreting, comparing and distributing posterior samples from any parameter estimation code. PESummary's unified input/output (I/O) system enables the comparison of samples from codes that previously stored data in incompatible formats.

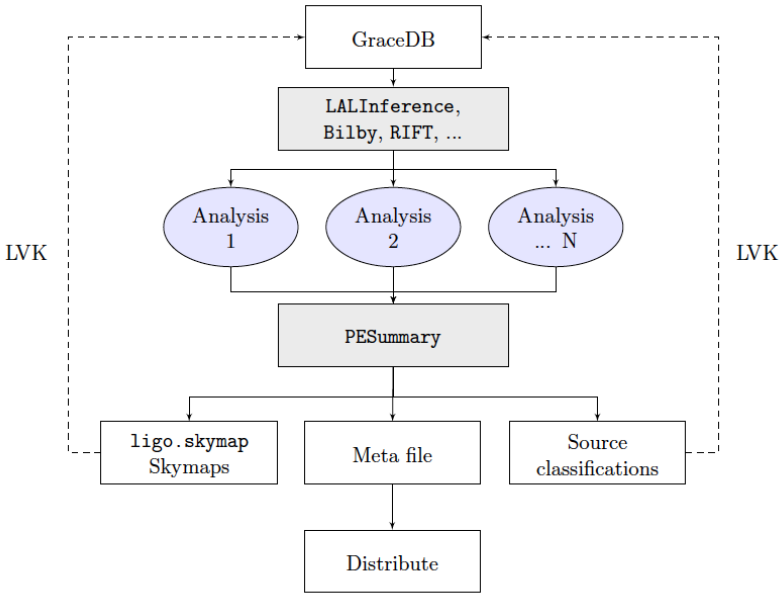

Figure 1: Flow chart showing the role of PESummary within the gravitational wave community. Dashed paths are specific for the LIGO Scientific, Virgo and KAGRA collaborations (LVK). Once a gravitational-wave is observed and uploaded to GraceDB, numerous Bayesian inference analyses are launched to extract the astrophysical source properties from the gravitational-wave measurement. The output data from all analyses is then passed to `PESummary` for post-processing. Alongside webpages for displaying, interpreting and comparing the Bayesian inference data, `PESummary` produces skymaps and source classifications, and a single universal 'metafile' containing information about each Bayesian inference analyses undertaken. This file can then distributed to the wider community. For the LVK workflow, skymaps and source classification probabilities are uploaded to GraceDB for circulation.

Since its first use in 2019, PESummary has grown to be a key component in the workflow of the LIGO Scientific [16], Virgo [17] and KAGRA [18] collaborations (LVK), as shown in Figure 1. It is used to analyse and compare the outputs from the popular GW Bayesian inference software packages: `LALInference` [19], `RIFT` [20, 21, 22], `PyCBC` [23] and `Bilby` [24, 25, 26] (both `PyCBC` and `Bilby` provide an interface for a number of popular sampling packages [e.g. 27, 28]), as well as circulating both skymaps through the GW candidate event database GraceDB [29] and Bayesian inference data through the LIGO Document Control Center. PESummary was critical in the parameter estimation analysis of GW190412 [30], GW190425 [31] and the re-analysis of the first gravitational-wave transient catalog (GWTC-1) [12] using Bilby. The released data were also in the PESummary data format [32, 33].

This article does not present a complete record of all the capabilities of PESummary, for more information, see https://lscsoft.docs.ligo.org/pesummary/stable_docs/index.html.

2. Software description

The main PESummary interface is implemented in pure Python [34] with the core library relying on a number of established scientific programming packages [35, 36, 37, 38, 39, 40, 41, 42, 43] and the GW library requiring custom GW data analysis software: `LALSuite` [44] and `GWpy` [45].

PESummary can be used for more than simply post-processing parameter estimation results. This library includes, but not limited to, new Kernel Density Estimators [46] to estimate PDFs of random variables within specified domains (one and two dimensional), a unified infrastructure for reading data in different formats, a release ready `HDF5` [36] file for storing one or more Bayesian inference analyses, a comprehensive plotting suite and webpage module for analysing and comparing Bayesian inference data, and checkpointing, a requirement for cloud based computing.

PESummary is designed to be as modular and adaptable as possible, ensuring the code will age gracefully with advances in Bayesian inference software. Where possible, the code is kept general meaning additional post-processing techniques can easily be implemented into the PESummary framework.

2.1. Software Architecture

PESummary is structured around a small number of class objects. Each object provides class and instance methods to provide the user with a complete interface for all operations.

2.1.1. Tabular Data

Bayesian inference returns the best fit model parameters through a *posterior* probability distribution function (PDF) calculated through Bayes' theorem [47],

$$p(\theta|d) = \frac{p(\theta)p(d|\theta)}{p(d)} \tag{1}$$

where $p(\theta|d)$ is the posterior probability of the model having parameters θ given the data d, $p(\theta)$ is the prior probability of the model having parameters θ, $p(d|\theta)$ is the likelihood of observing the data given a model with parameters θ and $p(d)$ is the evidence of the data, a model independent quantity. Often the likelihood and the posterior probability distribution function are

unknown analytically. Consequently, most Bayesian inference packages are designed to draw samples from the unknown posterior PDF [48, 49]. Samples approximating a posterior PDF for a single parameter, otherwise known as a marginalized posterior PDF, are then identified by integrating the posterior PDF over all other parameters. Often, these posterior samples are output in the form of a table saved in a 'result file'.

PESummary's stores posterior samples in a custom high level table structure, `pesummary.utils.samples_dict.SamplesDict`, a subclass of the Python builtin `dict` object. This class offers numerous methods for analysing the stored data, including the `.plot` method. Each marginalized posterior distribution is stored as a `pesummary.utils.samples_dict.Array` object, an inherited class of `numpy.ndarray` [50]. This structure provides direct access to the optimised array functions from NumPy [35] and the usability and familiarity of dictionaries.

A popular algorithm for sampling from the posterior distribution is Markov-Chain Monte-Carlo [48]. For this case, multiple Markov chains are often run in parallel to test convergence through the Gelman-Rubin statistic [51]. PESummary stores multiple Markov chains in the `pesummary.utils.samples_dict.MCMCSamplesDict`) class, a dictionary where each chains posterior samples are represented by a `pesummary.utils.samples_dict.SamplesDict` object. Algorithms for removing burnin are available via the `.burnin()` method. Convergence between chains can be measured via the `.gelman_rubin()` method.

Often multiple Bayesian inference analyses are performed to identify how the PDFs vary for different settings. Consequently, PESummary provides the `pesummary.utils.samples_dict.MultiAnalysisSamplesDict` class, inherited from the the Python builtin `dict` object, for storing multiple `pesummary.utils.samples_dict.SamplesDict` tables, each with an assigned label.

2.2. Packages

PESummary has followed the same methodology as Bilby by separating the top level code into 2 packages: `core` and `gw`. The `core` package provides all of the necessary code for analysing, displaying and comparing data files from general inference problems. The `gw` specific package contains GW functionality, including converting posterior distributions, deriving event classifications and GW specific plots, see Sec.2.2.2.

2.2.1. The `core` package

The `pesummary.core` package provides all of the code required for post-processing general inference result files. It is designed to be generic

and therefore work with the output from any Bayesian inference software. It provides a unified interface for producing plots, calculating useful statistics and generating webpages.

Plots are generated via the `pesummary.core.plots.plot` module. One dimensional marginalized posterior distributions are visualised as either a histogram, a kernel density estimate (`scipy.stats.gaussian_kde` or `pesummary.core.plots.plots.bounded_1d_kde.Bounded_1d_kde`) or cumulative distributions. Diagnostic plots displaying the marginalized trace or autocorrelation are also available. Comparison plots between multiple result files can be achieved by producing comparison one dimensional marginalized posterior distributions, cumulative distributions, box plots, violin plots, two dimensional contour and scatter plots or Jenson–Shannon [52] and Kolmogorov–Smirnov [53, 54] statistics.

The `pesummary.core.webpage.webpage` module is a Python wrapper for writing HTML. The `pesummary.core.webpage.webpage.page` class provides functionality for generating multi-level navigation bars, tables of images and/or data, modal carousels and more. The design and functionality of the webpages are controlled through the `pesummary.core.css` and `pesummary.core.js` modules; each containing custom style sheets (CSS) or JavaScript files (JS) respectively.

The `combine_corner.js` script contains functionality to generate custom corner plots by manipulating a pre-made figure [made with e.g. 38]. By providing an interface for the user to specify parameters they wish to compare, the PESummary webpages allow for the PDF, and its correlations, to be interactively explored.

2.2.2. The gw package

The `gw` package provides the functionality for parameter estimation specific to GW astronomy. Building on the `core` package, the `gw` module provides additional methods and classes for handling GW specific data files. Although the `gw` package is tailored for compact binary coalescence data files, we provide GW specific methods which can be applied to the Bayesian inference data from any transient GW source.

The `gw` package provides a comprehensive conversion module (`pesummary.gw.file.conversions`) for deriving alternative posterior distributions from the input, e.g. the primary mass and secondary mass from chirp mass and mass ratio. Assuming a binary black hole merger, the conversion class provides multiple methods for estimating the properties of the final black hole: final mass (or equivalently the energy radiated in gravitational-waves), final spin, peak luminosity and final kick velocity [55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67]. All fits are calibrated

to numerical relativity and are interchangeable via keyword arguments provided to the conversion class.

On the 17th August 2017, alerts [68] were sent out to more than 60 international groups of astronomers notifying them that a GW had just been detected: GW170817 [69]. Each group began observing the night sky to independently observe the source of the GW. GW170817 opened the window of a long-awaited multi-messenger astronomy [70]. Through interfacing with the `ligo.skymap` [71] package, PESummary provides an intuitive method for generating skymaps: data files showing the most likely source location of the GWs. These skymaps are distributed to astronomers through GraceDB, see Figure 1. In the last GW observing period, 26 alerts were sent out regarding possible GW candidates, each containing skymaps [72]. PESummary also interfaces with the `PEPredicates` [73] and `P-Astro` [74] packages to provide source classification probabilities for the type of compact binary merger observed; such as the probability that the system is a binary black hole or a binary neutron star. Both the skymap and source classification probabilities are vital for electromagnetic follow-up campaigns.

PESummary takes advantage of the `plotly` [43] interactive plotting package to produce interactive corner plots for extrinsic and intrinsic parameters. The `pesummary.gw.plots.publication` module also allows for 'publication' quality plots to be produced, for instance those in [12].

2.3. Software Functionalities

2.3.1. Unified input/output

Bayesian inference packages output their posterior samples in varying formats. In just the GW community alone, `LALInference`, `Bilby` and `RIFT` output their data in 3 different formats: `HDF5` [36], `JSON` [75] and `dat`. Although `Bilby` is able to output it's data in `HDF5`, it's posterior samples are stored differently to `LALInference`: `LALInference` stores posterior samples as a `numpy.recarray` while `Bilby` writes each marginalized posterior distribution as a separate dataset, and an alternative software package for reading `HDF5` files is required. Each data file also stores different metadata in different locations. This incompatibility makes it difficult to compare the contents from different Bayesian inference samplers both in and out of the GW community.

PESummary provides a unified infrastructure for reading data in different formats via the `pesummary.io` module. The universal `pesummary.io.read()` function reads data stored in all common file formats via the `pesummary.core.file.formats` module. The GW specific package extends the allowed file formats via the `pesummary.gw.file.formats` module. See Table 2 for a reduced list. Once read, the posterior samples

Package	Format	Class	Description
core	.dat	.default.Default	'.dat' format with parameter names as header
core	.txt	.default.Default	'.txt' format with parameter names as header
core	.h5/.hdf5	.default.Default	'.hdf5' format with data stored in a group called 'posterior' or 'posterior_samples'
core	.json	.default.Default	'.json' format with posterior samples stored in a group called 'posterior' or 'posterior_samples'
core	BILBY	.bilby.Bilby	Result file produced by BILBY
gw	BILBY	.bilby.Bilby	Result file produced by BILBY
gw	LALINFERENCE	.lalinference.LALInference	Result file produced by LALINFERENCE

Table 2: A selection of formats that can be read in with PESummary and accessible through the unified I/O interface for the listed class object(s). The core classes are all prepended by pesummary.core.file.formats, the gw classes are all prepended by pesummary.gw.file.formats.

from different data files can be compared through the common .samples_dict property. All read result files can be written to a specified file format via the common .write method. This method calls the universal pesummary.io.write() function.

PESummary offers the pesummary.core.file.meta_file module for producing a universal file containing posterior samples and metadata for one or more analyses. This single 'metafile' aims to store all information regarding the Baysian inference analysis. This includes prior samples, configuration files, injection information, environment information for reproducibility etc. For GW analyses, the pesummary.gw.file.meta_file module allows for the PSD, skymap data and calibration information to also be saved.

Executable	Description
`summaryclassification`	Generate GW event classification probabilities
`summaryclean`	Clean and convert an input result file
`summarycombine`	Combine multiple result files into a single format
`summarycompare`	Compare the contents of multiple files
`summarydetchar`	Generate GW Detector characterisation plots
`summarygracedb`	Retrieve data from the online GW Candidate Event Database [29]
`summarymodify`	Modify a PESummary meta file
`summarypages`	Generate webpages, plots and a meta file for N result files
`summarypageslw`	Generate webpages, plots and meta files for a select number of parameters
`summaryplots`	Generate specific plots for a given result file
`summarypipe`	Generate a `summarypages` executable given a GW run directory
`summarypublication`	Generate publication plots for N result files
`summaryrecreate`	Launch the GW analysis that was used to generate the PESummary meta file
`summaryversion`	Version of PESummary installed

Table 3: A selection of executable python scripts provided by PESummary

2.3.2. Executables

PESummary provides multiple executable Python scripts to act as an intermediary between the command line and the core functionality. See Table 3. `summarypages` is the main executable and combines `summaryclassification`, `summaryclean`, `summarycombine`, `summarydetchar`, `summaryplots` and `summarypublication`. `summarypages` is the most general executable provided by PESummary. Its GW workflow is described in Fig. 2. The core workflow is similar, except GW specific plots and webpages are not generated and the core classes are used.

The `summarypages` executable takes one or more result files as input via the `--samples` command line argument. A plethora of optional command line arguments are also available to allow for complete customisation[1]. All options are then passed to the `pesummary.gw.inputs.GWInput` class (in-

[1]For details run `summarypages --help` or vist the online documentation available https://lscsoft.docs.ligo.org/pesummary/stable_docs/core/cli/summarypages.html.

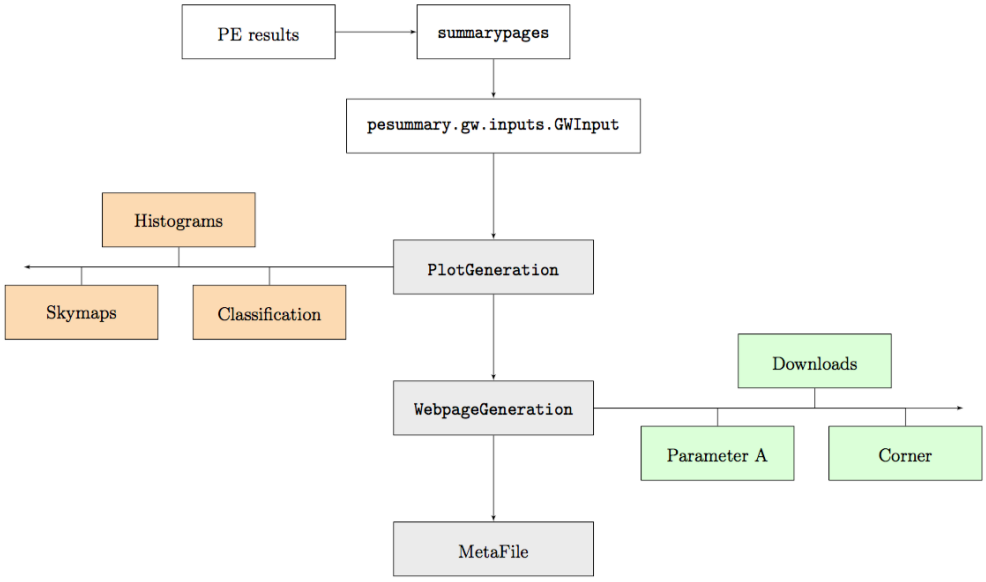

Figure 2: Workflow for the most general executable provided by PESummary: `summarypages`. Here we show the workflow when the `gw` package is used.

herited from the `pesummary.core.inputs.Input` class) which checks the inputs. Amongst these checks, the samples are extracted from all result files (see Section 2.3.1), converted to generate additional PDFs, and stored in a `pesummary.utils.samples_dict.MultiAnalysisSamplesDict` object with an assigned label.

All properties of the `GWInput` class are then used to initiate the `pesummary.cli.summaryplots.PlotGeneration` class. Here, all plots are generated on multiple CPUs, if specified, with the `.generate_plots` method. This includes source classification plots, marginalized posteriors, interactive plots and more. Initial skymaps are generated based on a two-dimensional histogram of the right ascension and declination samples. A subprocess is then launched to generate the more accurate `ligo.skymap` skymap. This implementation allows for the rest of the pipeline to continue without having to wait for the often time-consuming `ligo.skymap` skymap to be produced. Once complete, the initial skymap is overwritten.

All webpages are then produced by running the `.generate_webpages` method once the `pesummary.cli.summarypages.WebpageGeneration` class is initialised with all `GWInput` properties. This includes single webpages for each parameter in each result file, interactive corner pages, comparison pages for all common parameters, a version page providing version and environment information and a downloads page. Finally a single metafile

containing all information from the run is produced. This includes environment information, posterior samples, command line arguments and more. This file is available for download via the downloads page.

The output from `summarypages` is completely self-contained; allowing for it to distributed if required. An example of the `summarypages` output can be seen here: https://pesummary.github.io/GW190412/home.html. This output was produced from Listing 4.

Amidst the collection of optional command line arguments, the `pesummary.gw` module also provides a dynamic argument parser. This allows for dependent arguments to be passed from the command line (see section 3.5).

3. Illustrative Examples

Below we provide a limited set of examples to demonstrate some of the features of PESummary. All data and scripts that are used as part of this section can be downloaded from https://github.com/pesummary/pesummary-paper-data. More examples can be found in the PESummary repository: https://git.ligo.org/lscsoft/pesummary/-/tree/master/examples. The following examples assume that you have cloned the pesummary-paper-data repository and you are in the base directory.

3.1. Example 1: Running with `emcee`

This example shows the flexibility of the PESummary post-processing package. We run the *Fitting a model to data* tutorial provided as part of the `emcee`[76] sampling package. We save the posterior samples to a '.dat' file and run with 8 chains. All post-processing is then handled with PESummary, see Figure 3 for example output. The `emcee` code snippet, as well as the posterior samples can be downloaded from the pesummary-paper-data repository.

Listing 1: Running PESummary on the `emcee` output.

```
python emcee_tutorial.py
chains=($(ls emcee_output/chain_*.dat))
summarypages --webdir ./webpage --samples ${chains[@]}
             --labels tutorial --mcmc_samples --inj_file
                emcee_output/injected.txt
```

3.2. Example 2: Reading a result file

This example demonstrates how to read in a PESummary 'metafile' using the `pesummary.io` module. We use the `posterior_samples.h5` file produced as part of Listing 1 (running Listing 1 is not necessary for this

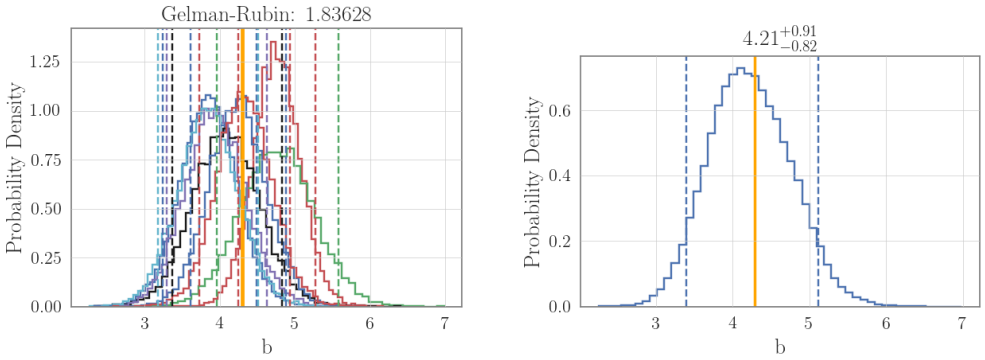

Figure 3: The output marginalized posterior for each chain (Left) and the posterior from combining all chains (Right) resulting from Listing 1. The injected value is shown in orange.

example, the metafile is available in the pesummary-paper-data repository) and specify that we would like to use the `core` package meaning only the core file formats are allowed. We show how to extract the posterior samples for a specified analysis, how to access the marginalized posterior samples, and how to extract the prior samples and any stored configuration files. Some of the attributes of each object are also shown. For full documentation, run the inbuilt Python help function `help` [77].

Listing 2: Example code to read in a PESummary result file with PESummary.

```
from pesummary.io import read

f = read("posterior_samples.h5", package="core")
multi_analysis_samples = f.samples_dict
print(type(multi_analysis_samples))
analysis = multi_analysis_samples.labels[0]
posterior_samples = multi_analysis_samples[analysis]
print(type(posterior_samples))
print(posterior_samples.keys())
marginalized = posterior_samples["m"]
print(type(marginalized))
print(marginalized.average(type="median"))
priors = f.priors["samples"]
config = f.config
```

3.3. (GW) Example 3: analysing public LIGO and Virgo posterior samples

This example demonstrates how to extract and analyse public LIGO and Virgo posterior samples. It includes demonstrations of how to produce 'standard' plots for the 'combined' analysis stored in the PESummary 'metafile' through the `.plot` method, see Figure 4. We use the publicly available

https://dcc.ligo.org/DocDB/0163/P190412/009/posterior_samples.h5 file, which has been copied to the pesummary-paper-data repository. This specific file was chosen since it includes additional information such as the PSD, calibration envelopes, priors etc. Owing to GitHub's strict 100MB file limit, the file located in the pesummary-paper-data repository is stored under `git-lfs`[78]. For instructions on how to successfully download this file, see the pesummary-paper-data README.

Listing 3: Extracting data from public LIGO and Virgo posterior samples

```
from pesummary.io import read

f = read("GW190412_posterior_samples.h5", package="gw")
analysis = "combined"
posterior_samples = f.samples_dict[analysis]
psds = f.psd[analysis]
calibration_envelope = f.priors["calibration"][analysis]
prior_samples = f.priors["samples"][analysis]

hist = posterior_samples.plot("mass_1", type="hist")
hist.show()
skymap = posterior_samples.plot(type="skymap")
skymap.show()
plot = psds.plot(fmin=20)
plot.show()
```

3.4. (GW) Example 4: Producing a summary page for public LIGO and Virgo posterior samples

This example demonstrates how a summary page can be generated from publicly available LIGO and Virgo data files. We use the same publicly available data file as Listing 3. This 'metafile' contains a total of 10 Bayesian inference analyses, each using different standard models used by the LVK. In this example, we choose to compare only a select subset: an analysis conducted with the IMRPhenomPv3HM model [79] and the SEOBNRv4PHM model [80, 81, 55] (see [82] for details). We specify that we would like to use the `gw` package (`--gw`), run on 15 CPUs (`--multi_process 15`), and generate 'publication' quality plots (`--publication`). The output page can be found here: https://pesummary.github.io/GW190412/home.html.

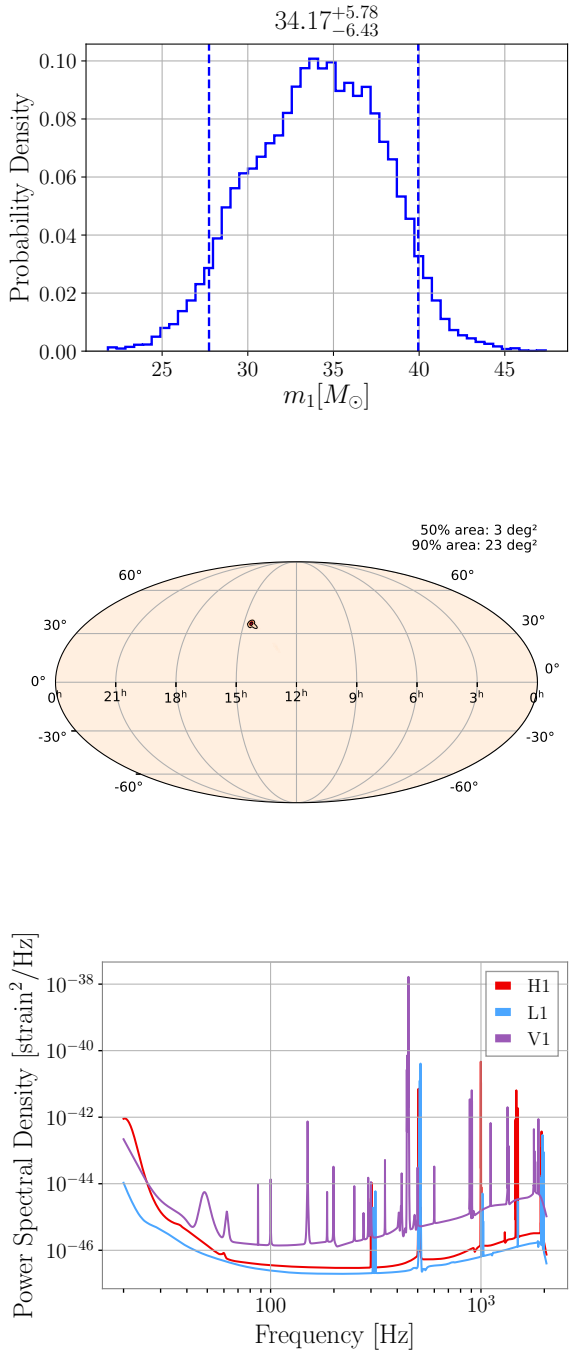

Figure 4: The output marginalized posterior (top), skymap (middle) and PSD plot (bottom) from Listing 3.

Listing 4: Generating a html page to compare and analyse the public LIGO and Virgo posterior samples. This may take some time.

```
summarypages --webdir ./GW190412 \
             --samples GW190412_posterior_samples.h5 \
             --gw --compare_results IMRPhenomPv3HM
                SEOBNRv4PHM \
             --publication \
             --multi_process 15
```

3.5. (GW) Example 5: PESummary's dynamic argument parser

`argparse`, the Python module for handling command-line arguments (CLA) [83], requires a known list of arguments. This means that they cannot depend on another variable. Through the `pesummary.gw.command_line` module, PESummary allows the user to specify CLAs which change depending on the provided label. Below we show how PSDs can be provided to `summarypages` dynamically through the `--{}_psd` CLA. Result files and psd data was created using the 'make_data_for_listing5.py' script made available in the pesummary-paper-data repository.

Listing 5: Example usage of PESummary's dynamic argument parser

```
summarypages --webdir ./webpage \
             --samples test.hdf5 test.json \
             --labels hdf5_example json_example \
             --hdf5_example_psd H1:psd_H1.dat \
             --json_example_psd V1:psd_V1.dat L1:psd_L1.
                dat \
             --gw
```

3.6. (GW) Example 6: Reproducing LIGO and Virgo plots

This example demonstrates how to reproduce a subset of plots in the first GW transient catalog (GWTC-1) [Figures 4 and 5 in Ref. 12]. We use publicly available posterior samples released as part of GWTC-1 (ignoring the GW170809 [12] prior choices file): https://dcc.ligo.org/public/0157/P1800370/005/GWTC-1_sample_release.tar.gz, which have been copied to the pesummary-paper-data repository. Figure 5 shows an example of the output. When running this Listing, PESummary will print multiple warnings to `stdout`. These warnings are expected and shows that PESummary is robust to potential failures and will continue to produce an output while still warning the reader appropriately. One such example is a message warning the user that the './GWTC-1_sample_release/GW170817_GWTC-1.hdf5' cannot be read in and data will not be added to the plot as this file contains 'multiple

posterior sample tables: IMRPhenomPv2NRT_highSpin_posterior, IMRPhenomPv2NRT_lowSpin_posterior' and we have not specified which we wish to load from the command line.

Listing 6: Example code to generate GW plots with PESummary. This may take some time.

```
FILES=($(ls ./GWTC-1_sample_release/*_GWTC-1.hdf5))

LABELS=()
for i in ${FILES[@]}; do
    label=`python -c "print('${i}'.split('_GWTC-1.hdf5')
       [0])"`
    LABELS+=($(python -c "print('${label}'.split('./GWTC-1
       _sample_release/')[1])"))
done

COLORS='#00a7f0' '#9da500' '#c59700' '#55b300' '#f77b00'
   '#ea65ff' '#00b1a4' '#00b47d' '#00aec2' '#9f8eff')
LINESTYLES=(solid dashed solid solid dashed solid solid
   dashed dashed dashed solid)

summarypublication --plot 2d_contour \
                   --webdir ./GWTC-1_sample_release \
                   --samples ${FILES[@]} \
                   --labels ${LABELS[@]} \
                   --parameters mass_1_source
                      mass_2_source \
                   --colors ${COLORS[@]} \
                   --linestyles ${LINESTYLES[@]} \
                   --publication_kwargs xlow:0 xhigh:80
                      ylow:0 yhigh:50

summarypublication --plot violin \
                   --webdir ./GWTC-1_sample_release \
                   --samples ${FILES[@]} \
                   --labels ${LABELS[@]} \
                   --parameters mass_ratio
```

4. Impact

PESummary is now a widely used library in the LVK. In just over one year, PESummary has become the post-processing software for the main LVK GW parameter estimation codes [24, 25, 19, 84, 85, 86, 87] and is relied upon for the open data release of the LVK parameter estimation data products [88]. Rather than releasing multiple data products in different formats spread across numerous URLs, PESummary has greatly simplified this to simply releasing a single file [32, 33]. This can propel future research in the field, as

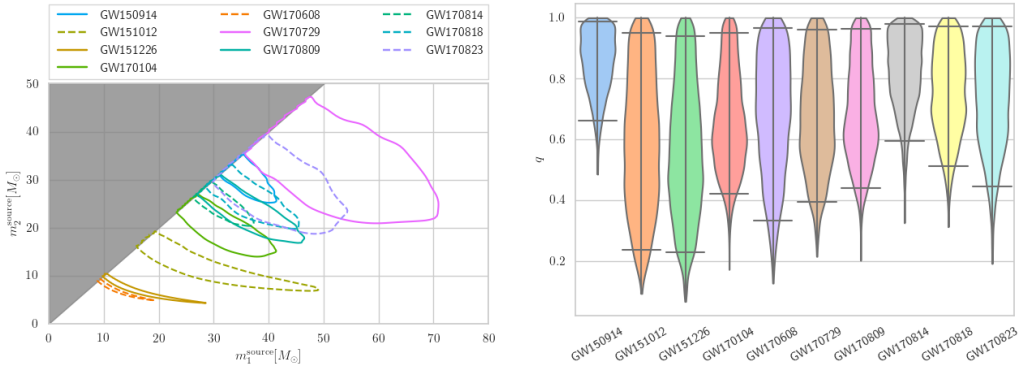

Figure 5: Two plots output from Listing 6. Left: Recreation of Figure 4 in Ref. [12], Right: Recreation of Figure 5 in Ref. [12].

researches no longer have to create custom scripts for combining, retrieving, and recreating the initial analysis.

The flexibility and intuitive Python executables provided by PESummary, has significantly improved the efficiency of researchers (often early-career graduates or graduate scientists), by reducing the need for repetitive tasks. In particular, PESummary's interactive corner plots has led to the increased knowledge of degeneracies between parameters for researchers within the LVK.

PESummary is also fully incorporated into the automated low-latency alert workflow [89]. Posterior based classifications are therefore automatically produced and posted to the online GW Candidate Event Database [29] from the automatic parameter estimation follow-up. This information will greatly aid electromagnetic followup campaigns.

5. Conclusions

In this paper we have described PESummary, the Parameter Estimation Summary page builder. This Python package provides a modern interface for displaying interpreting, comparing and distributing Bayesian inference data from any parameter estimation code. PESummary has been used extensively by the international GW community, and has been crucial for Advanced LIGO's [90] and Advanced Virgo's [91] third gravitational wave observing run.

Although PESummary is primarily used for post-processing GW Bayesian inference data from compact binary coalescenses, looking forward, we plan to incorporate other GW fields, e.g. Tests of General Relativity [see e.g. 92]. This will enable PESummary to be the driving force behind the post-

processing and distribution of all Bayesian inference data output from the LVK. We will also continue to expand our already comprehensive plotting suite to improve its versatility for any Bayesian inference analysis.

6. Conflict of Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Acknowledgements

The authors wish to thank the reviewers for comments and suggestions concerning this manuscript. The authors would also like to thank the LIGO Scientific, Virgo and KAGRA collaboration for all constructive feedback, bug reports and feature requests. A special thanks to Virginia D'Emilio, David Keitel, Nicola De Lillo, Nathan Johnson-McDaniel, and Philip Relton for reviewing PESummary. We wish to thank Gregory Ashton, Christopher Berry, Sylvia Biscoveanu, Gregorio Carullo, Stephen Fairhurst, Edward Fauchon-Jones, Rhys Green, Carl-Johan Haster, Duncan Macleod, Sergeui Ossokine, Isobel Romero-Shaw, Colm Talbot and John Veitch for useful discussions. We thank all past and current contributors; for a full overview, see the list of contributors on Gitlab. The authors are grateful for computational resources provided by the Leonard E Parker Center for Gravitation, Cosmology and Astrophysics at the University of Wisconsin-Milwaukee and Cardiff University, supported by National Science Foundation Grants PHY-1626190 and PHY-1700765 and STFC grant ST/I006285/1.

References

[1] S. Jackman, Bayesian analysis for the social sciences, Vol. 846, John Wiley & Sons, 2009.

[2] A. M. Ellison, Bayesian inference in ecology, Ecology letters 7 (6) (2004) 509–520.

[3] M. A. Beaumont, B. Rannala, The bayesian revolution in genetics, Nature Reviews Genetics 5 (4) (2004) 251–261.

[4] R. Trotta, Bayes in the sky: Bayesian inference and model selection in cosmology, Contemporary Physics 49 (2) (2008) 71–104.

[5] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, D. B. Rubin, Bayesian data analysis, CRC press, 2013.

[6] H. L. Harney, Bayesian Inference, Springer, 2016.

[7] B. P. Abbott, et al., Observation of Gravitational Waves from a Binary Black Hole Merger, Phys. Rev. Lett. 116 (6) (2016) 061102. arXiv:1602.03837, doi:10.1103/PhysRevLett.116.061102.

[8] A. Einstein, Approximative Integration of the Field Equations of Gravitation, Sitzungsber. Preuss. Akad. Wiss. Berlin (Math. Phys.) 1916 (1916) 688–696.

[9] A. Einstein, Über Gravitationswellen, Sitzungsber. Preuss. Akad. Wiss. Berlin (Math. Phys.) 1918 (1918) 154–167.

[10] B. P. Abbott, et al., Properties of the Binary Black Hole Merger GW150914, Phys. Rev. Lett. 116 (24) (2016) 241102. arXiv:1602.03840, doi:10.1103/PhysRevLett.116.241102.

[11] B. P. Abbott, et al., Properties of the binary neutron star merger GW170817, Phys. Rev. X 9 (1) (2019) 011001. arXiv:1805.11579, doi:10.1103/PhysRevX.9.011001.

[12] B. Abbott, R. Abbott, T. Abbott, S. Abraham, F. Acernese, K. Ackley, C. Adams, R. Adhikari, V. Adya, C. Affeldt, et al., Gwtc-1: A gravitational-wave transient catalog of compact binary mergers observed by ligo and virgo during the first and second observing runs, Physical Review X 9 (3) (2019) 031040.

[13] N. J. Cornish, T. B. Littenberg, Bayeswave: Bayesian inference for gravitational wave bursts and instrument glitches, Classical and Quantum Gravity 32 (13) (2015) 135012.

[14] B. P. Abbott, R. Abbott, T. Abbott, M. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. Adhikari, et al., Binary black hole mergers in the first advanced ligo observing run, Physical Review X 6 (4) (2016) 041015.

[15] E. Thrane, C. Talbot, An introduction to Bayesian inference in gravitational-wave astronomy: parameter estimation, model selection, and hierarchical models, Publ. Astron. Soc. Austral. 36 (2019) e010, [Erratum: Publ.Astron.Soc.Austral. 37, e036 (2020)]. arXiv:1809.02293, doi:10.1017/pasa.2019.2.

[16] LIGO Scientific Collaboration, https://www.ligo.org (2021).

[17] Virgo Collaboration, https://www.virgo-gw.eu (2021).

[18] KAGRA Collaboration, https://gwcenter.icrr.u-tokyo.ac.jp (2021).

[19] J. Veitch, V. Raymond, B. Farr, W. Farr, P. Graff, S. Vitale, B. Aylott, K. Blackburn, N. Christensen, M. Coughlin, et al., Parameter estimation for compact binaries with ground-based gravitational-wave observations using the lalinference software library, Physical Review D 91 (4) (2015) 042003.

[20] C. Pankow, P. Brady, E. Ochsner, R. O'Shaughnessy, Novel scheme for rapid parallel parameter estimation of gravitational waves from compact binary coalescences, Phys. Rev. D 92 (2) (2015) 023002. arXiv:1502.04370, doi:10.1103/PhysRevD.92.023002.

[21] J. Lange, et al., Parameter estimation method that directly compares gravitational wave observations to numerical relativity, Phys. Rev. D 96 (10) (2017) 104041. arXiv:1705.09833, doi:10.1103/PhysRevD.96.104041.

[22] D. Wysocki, R. O'Shaughnessy, J. Lange, Y.-L. L. Fang, Accelerating parameter inference with graphics processing units, Phys. Rev. D 99 (8) (2019) 084026. arXiv:1902.04934, doi:10.1103/PhysRevD.99.084026.

[23] C. M. Biwer, C. D. Capano, S. De, M. Cabero, D. A. Brown, A. H. Nitz, V. Raymond, PyCBC Inference: A Python-based parameter estimation toolkit for compact binary coalescence signals, Publ. Astron. Soc. Pac. 131 (996) (2019) 024503. arXiv:1807.10312, doi:10.1088/1538-3873/aaef0b.

[24] G. Ashton, M. Hübner, P. D. Lasky, C. Talbot, K. Ackley, S. Biscoveanu, Q. Chu, A. Divakarla, P. J. Easter, B. Goncharov, et al., Bilby: A user-friendly bayesian inference library for gravitational-wave astronomy, The Astrophysical Journal Supplement Series 241 (2) (2019) 27.

[25] I. Romero-Shaw, C. Talbot, S. Biscoveanu, V. D'Emilio, G. Ashton, C. Berry, S. Coughlin, S. Galaudage, C. Hoy, M. Huebner, et al., Bayesian inference for compact binary coalescences with bilby: Validation and application to the first ligo–virgo gravitational-wave transient catalogue, arXiv preprint arXiv:2006.00714.

[26] R. J. E. Smith, G. Ashton, A. Vajpeyi, C. Talbot, Massively parallel Bayesian inference for transient gravitational-wave astronomy, Mon. Not. Roy. Astron. Soc. 498 (3) (2020) 4492–4502. arXiv:1909.11873, doi:10.1093/mnras/staa2483.

[27] D. Foreman-Mackey, D. W. Hogg, D. Lang, J. Goodman, emcee: The MCMC Hammer, Publ. Astron. Soc. Pac. 125 (2013) 306–312. arXiv:1202.3665, doi:10.1086/670067.

[28] J. S. Speagle, dynesty: a dynamic nested sampling package for estimating bayesian posteriors and evidences, Monthly Notices of the Royal Astronomical Society 493 (3) (2020) 3132–3158.

[29] T. Prestegard, A. Pace, B. Moe, B. Stephens, P. Brady, https://gracedb.ligo.org (2019).

[30] L. S. Collaboration, V. Collaboration, et al., Gw190412: Observation of a binary-black-hole coalescence with asymmetric masses, arXiv preprint arXiv:2004.08342.

[31] B. Abbott, et al., GW190425: Observation of a Compact Binary Coalescence with Total Mass $\sim 3.4 M_{\odot}$, Astrophys. J. Lett. 892 (2020) L3. arXiv:2001.01761, doi:10.3847/2041-8213/ab75f5.

[32] B. Abbott, et al., https://dcc.ligo.org/public/0163/P190412/008/posterior_samples.h5 (2019).

[33] B. Abbott, et al., https://dcc.ligo.org/public/0165/P2000026/001/GW190425_posterior_samples.h5 (2019).

[34] P. L. R. The Python Software Foundation, http://www.python.org.

[35] O. Travis E, a guide to numpy.

[36] A. Collette, Python and HDF5, O'Reilly, 2013.

[37] J. D. Hunter, Matplotlib: A 2d graphics environment, Computing in Science & Engineering 9 (3) (2007) 90–95. doi:10.1109/MCSE.2007.55.

[38] D. Foreman-Mackey, corner.py: Scatterplot matrices in python, The Journal of Open Source Software 1 (2) (2016) 24. doi:10.21105/joss.00024.
URL https://doi.org/10.21105/joss.00024

[39] W. McKinney, Data structures for statistical computing in python, in: S. van der Walt, J. Millman (Eds.), Proceedings of the 9th Python in Science Conference, 2010, pp. 51 – 56.

[40] Astropy Collaboration, et al., Astropy: A community Python package for astronomy, Astronomy and Astrophysic 558 (2013) A33. arXiv:1307.6212, doi:10.1051/0004-6361/201322068.

[41] A. M. Price-Whelan, et al., The Astropy Project: Building an Open-science Project and Status of the v2.0 Core Package, Astronomical Journal 156 (2018) 123. doi:10.3847/1538-3881/aabc4f.

[42] C. da Costa-Luis, tqdm: A fast, extensible progress meter for python and cli, Journal of Open Source Software 4 (37) (2019) 1277.

[43] P. T. Inc., Collaborative data science (2015).
URL https://plot.ly

[44] LIGO Scientific Collaboration, LIGO Algorithm Library - LALSuite, free software (GPL) (2018). doi:10.7935/GT1W-FZ16.

[45] D. M. Macleod, et al., GWpy: a python package for gravitational-wave astrophysics, in preparation (2020).

[46] B. W. Silverman, Density estimation for statistics and data analysis, Vol. 26, CRC press, 1986.

[47] J. Joyce, Bayes' theorem.

[48] N. Metropolis, S. Ulam, The monte carlo method, Journal of the American statistical association 44 (247) (1949) 335–341.

[49] J. Skilling, et al., Nested sampling for general bayesian computation, Bayesian analysis 1 (4) (2006) 833–859.

[50] S. v. d. Walt, S. C. Colbert, G. Varoquaux, The numpy array: a structure for efficient numerical computation, Computing in Science & Engineering 13 (2) (2011) 22–30.

[51] S. P. Brooks, A. Gelman, General methods for monitoring convergence of iterative simulations, Journal of computational and graphical statistics 7 (4) (1998) 434–455.

[52] J. Lin, Divergence measures based on the shannon entropy, IEEE Transactions on Information Theory 37 (1) (1991) 145–151. doi:10.1109/18.61115.

[53] A. N. Kolmogorov, Sulla determinazione empirica di una legge di distribuzione, G. Ist. Ital. Attuari. 4 (1933) 83–91.

[54] N. Smirnov, Table for estimating the goodness of fit of empirical distributions, Ann. Math. Statist. 19 (2) (1948) 279–281. doi:10.1214/aoms/1177730256.
URL https://doi.org/10.1214/aoms/1177730256

[55] S. Babak, A. Taracchini, A. Buonanno, Validating the effective-one-body model of spinning, precessing binary black holes against numerical relativity, PhRvD 95 (2) (2017) 024010. arXiv:1607.05661, doi:10.1103/PhysRevD.95.024010.

[56] S. Ossokine, S. Marsat, R. Cotesta, S. Babak, A. Buonanno, I. Hinder, R. Haas, H. Pfeiffer, M. Pürrer, Effective-one-body multipolar waveforms for spinning, precessing binary black holes, In preparationIn preparation.

[57] S. Husa, et al., Frequency-domain gravitational waves from nonprecessing black-hole binaries. I. New numerical waveforms and anatomy of the signal, Physical Review D 93 (2016) 044006. arXiv:1508.07250, doi:10.1103/PhysRevD.93.044006.

[58] S. Khan, K. Chatziioannou, M. Hannam, F. Ohme, Phenomenological model for the gravitational-wave signal from precessing binary black holes with two-spin effects, PhRvD 100 (2) (2019) 024059. arXiv:1809.10113, doi:10.1103/PhysRevD.100.024059.

[59] S. Khan, F. Ohme, K. Chatziioannou, M. Hannam, Including higher order multipoles in gravitational-wave models for precessing binary black holes, "PhRvD" 101 (2) (2020) 024056. arXiv:1911.06050, doi:10.1103/PhysRevD.101.024056.

[60] V. Varma, S. E. Field, M. A. Scheel, J. Blackman, D. Gerosa, L. C. Stein, L. E. Kidder, H. P. Pfeiffer, Surrogate models for precessing binary black hole simulations with unequal masses, Physical Review Research 1 (3) (2019) 033015.

[61] V. Varma, S. E. Field, M. A. Scheel, J. Blackman, L. E. Kidder, H. P. Pfeiffer, Surrogate model of hybridized numerical relativity binary black hole waveforms, Physical Review D 99 (6) (2019) 064045.

[62] F. Hofmann, E. Barausse, L. Rezzolla, The final spin from binary black holes in quasi-circular orbits, ApJL 825 (2) (2016) L19. arXiv:1605.01938, doi:10.3847/2041-8205/825/2/L19.

[63] N. K. Johnson-McDaniel, et al., Determining the final spin of a binary black hole system including in-plane spins: Method and checks of accuracy, Tech. Rep. LIGO-T1600168, LIGO Project, https://dcc.ligo.org/LIGO-T1600168/public/main (2016).
URL https://dcc.ligo.org/T1600168/public

[64] J. Healy, C. O. Lousto, Remnant of binary black-hole mergers: New simulations and peak luminosity studies, PhRvD 95 (2) (2017) 024037. arXiv:1610.09713, doi:10.1103/PhysRevD.95.024037.

[65] X. Jiménez-Forteza, D. Keitel, S. Husa, M. Hannam, S. Khan, M. Pürrer, Hierarchical data-driven approach to fitting numerical relativity data for nonprecessing binary black holes with an application to final spin and radiated energy, PhRvD 95 (2017) 064024. doi:10.1103/PhysRevD.95.064024.
URL https://link.aps.org/doi/10.1103/PhysRevD.95.064024

[66] D. Keitel, X. J. Forteza, S. Husa, L. London, S. Bernuzzi, E. Harms, A. Nagar, M. Hannam, S. Khan, M. Pürrer, G. Pratten, V. Chaurasia, The most powerful astrophysical events: Gravitational-wave peak luminosity of binary black holes as predicted by numerical relativity, Physical Review D 96 (2017) 024006. doi:10.1103/PhysRevD.96.024006.
URL https://link.aps.org/doi/10.1103/PhysRevD.96.024006

[67] V. Varma, M. Isi, S. Biscoveanu, Extracting the Gravitational Recoil from Black Hole Merger Signals, Physical Review Letters 124 (10) (2020) 101104. arXiv:2002.00296, doi:10.1103/PhysRevLett.124.101104.

[68] LIGO Scientific Collaboration and Virgo Collaboration, GCN 21505, 21509, 21513 (2017).

[69] B. P. Abbott, et al., GW170817: Observation of Gravitational Waves from a Binary Neutron Star Inspiral, Phys. Rev. Lett. 119 (16) (2017) 161101. arXiv:1710.05832, doi:10.1103/PhysRevLett.119.161101.

[70] B. P. Abbott, et al., Multi-messenger Observations of a Binary Neutron Star Merger, Astrophys. J. Lett. 848 (2) (2017) L12. arXiv:1710.05833, doi:10.3847/2041-8213/aa91c9.

[71] L. Singer, ligo.skymap https://lscsoft.docs.ligo.org/ligo.skymap/ (2019).

[72] LIGO Scientific Collaboration and Virgo Collaboration, GCN 25871, 25829, 25753, 25503, 25497, 25324, 25187, 25164, 25115, 25012, 24998, 24950, 24922, 24717, 24632, 24621, 24598, 24570, 24522, 24503, 24377, 24237, 24168, 24141, 24098, 24069 https://gcn.gsfc.nasa.gov/lvc.html (2019).

[73] W. Farr, et al., https://pypi.org/project/pepredicates/ (2019).

[74] D. Chatterjee, et al., https://lscsoft.docs.ligo.org/p-astro/ (2020).

[75] B. Ippolito, et al., https://github.com/python/cpython/blob/3.8/Lib/json/__init__.py (2019).

[76] D. Foreman-Mackey, D. W. Hogg, D. Lang, J. Goodman, emcee: The MCMC Hammer, Publications of the Astronomical Society of the Pacific 125 (925) (2013) 306. arXiv:1202.3665, doi:10.1086/670067.

[77] Python Software Foundation, Built-in functions https://docs.python.org/3/library/functions.html (2021).

[78] B. M. Carlson, C. Darroch, L. Schneider, et al., https://github.com/git-lfs/git-lfs.

[79] S. Khan, F. Ohme, K. Chatziioannou, M. Hannam, Including higher order multipoles in gravitational-wave models for precessing binary black holes, Phys. Rev. D 101 (2) (2020) 024056. arXiv:1911.06050, doi:10.1103/PhysRevD.101.024056.

[80] S. Ossokine, et al., Multipolar Effective-One-Body Waveforms for Precessing Binary Black Holes: Construction and Validation, Phys. Rev. D 102 (4) (2020) 044055. arXiv:2004.09442, doi:10.1103/PhysRevD.102.044055.

[81] Y. Pan, A. Buonanno, A. Taracchini, L. E. Kidder, A. H. Mroué, H. P. Pfeiffer, M. A. Scheel, B. Szilágyi, Inspiral-merger-ringdown waveforms

of spinning, precessing black-hole binaries in the effective-one-body formalism, Phys. Rev. D 89 (8) (2014) 084006. arXiv:1307.6232, doi:10.1103/PhysRevD.89.084006.

[82] GW190412: Observation of a Binary-Black-Hole Coalescence with Asymmetric MassesarXiv:2004.08342.

[83] The Python Software Foundation, argparse https://docs.python.org/3/library/argparse.html (2021).

[84] R. Smith, G. Ashton, Expediting astrophysical discovery with gravitational-wave transients through massively parallel nested sampling, arXiv preprint arXiv:1909.11873.

[85] G. Carullo, W. Del Pozzo, J. Veitch, Observational Black Hole Spectroscopy: A time-domain multimode analysis of GW150914, Phys. Rev. D 99 (12) (2019) 123029, [Erratum: Phys.Rev.D 100, 089903 (2019)]. arXiv:1902.07527, doi:10.1103/PhysRevD.99.123029.

[86] M. Isi, M. Giesler, W. M. Farr, M. A. Scheel, S. A. Teukolsky, Testing the no-hair theorem with GW150914, Phys. Rev. Lett. 123 (11) (2019) 111102. arXiv:1905.00869, doi:10.1103/PhysRevLett.123.111102.

[87] J. Lange, R. O'Shaughnessy, M. Rizzo, Rapid and accurate parameter inference for coalescing, precessing compact binaries, arXiv:1805.10457.

[88] LIGO Scientific Collaboration, https://dcc.ligo.org/LIGO-M1000066/public (2017).

[89] L. P. Singer, et al., GWCelery, https://git.ligo.org/emfollow/gwcelery (2020).

[90] J. Aasi, et al., Advanced LIGO, Class. Quant. Grav. 32 (2015) 074001. arXiv:1411.4547, doi:10.1088/0264-9381/32/7/074001.

[91] F. Acernese, M. Agathos, K. Agatsuma, D. Aisa, N. Allemandou, A. Allocca, J. Amarni, P. Astone, G. Balestri, G. Ballardin, et al., Advanced virgo: a second-generation interferometric gravitational wave detector, Classical and Quantum Gravity 32 (2) (2014) 024001.

[92] B. Abbott, R. Abbott, T. Abbott, S. Abraham, F. Acernese, K. Ackley, C. Adams, R. X. Adhikari, V. Adya, C. Affeldt, et al., Tests of general relativity with the binary black hole signals from the ligo-virgo catalog gwtc-1, Physical Review D 100 (10) (2019) 104036.