ECT* workshop,
June 2022

# Joint Bayesian inference of gravitational-wave & NICER data

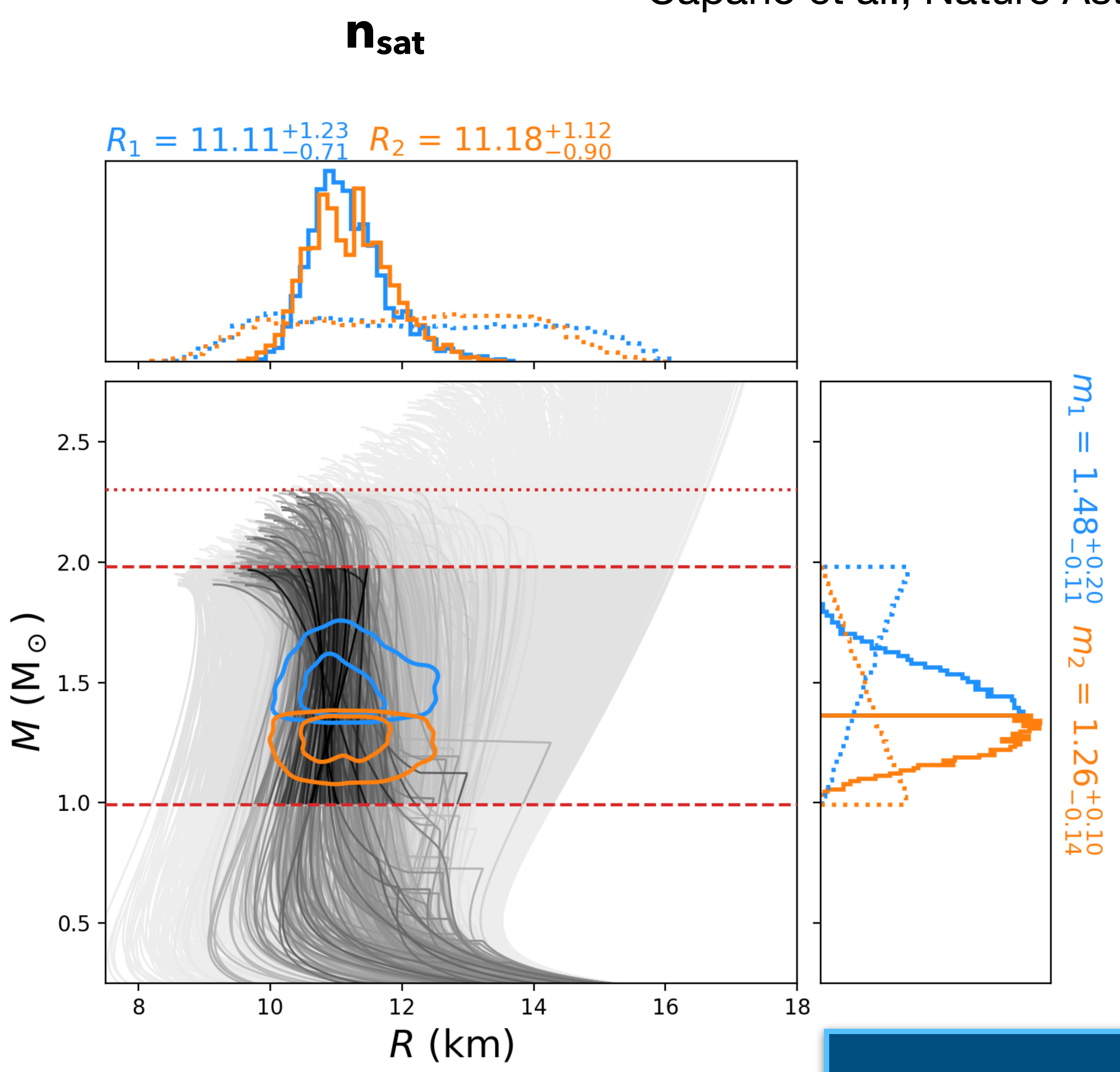

**Collin Capano**
**UMass Dartmouth &**
**Max Planck Institute for Gravitational Physics Hannover**

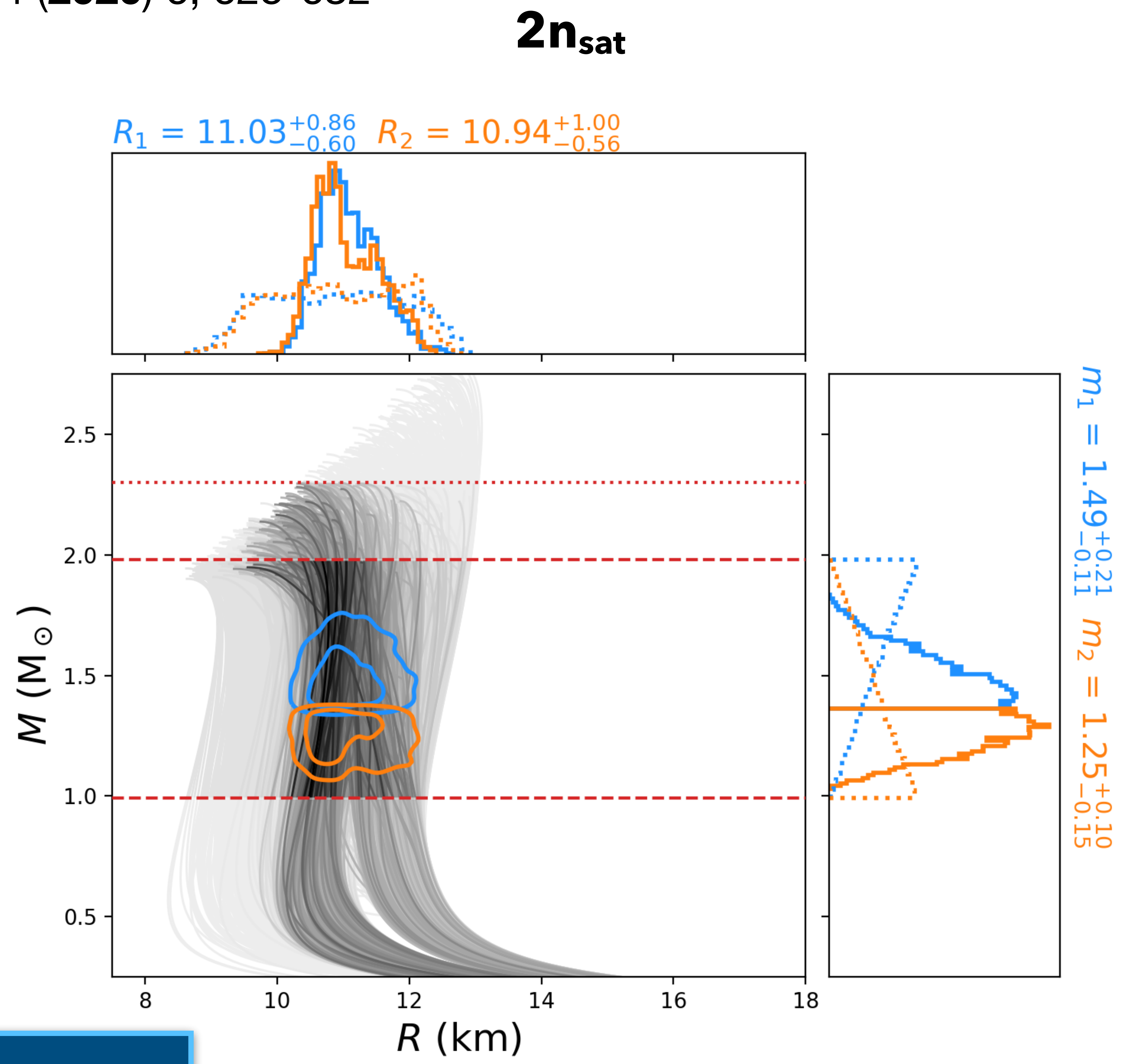
# Overview

- Combining results from NICER and GW data provides best constraints on neutron-star equation of state (EOS)
- NICER results are dependent on hotspot configuration
- Our goal: reanalyze NICER/XMM Newton data jointly with GW data to see if we get a different preferred hotspot configuration & EOS constraint
- Caution: work-in-progress! Today demonstrating proof of concept
- Collaborators: Chaitanya Afle (Syracuse University), Duncan Brown (Syracuse University), Ingo Tews (LANL), Rahul Somasundaram (IPNL)

# Motivation

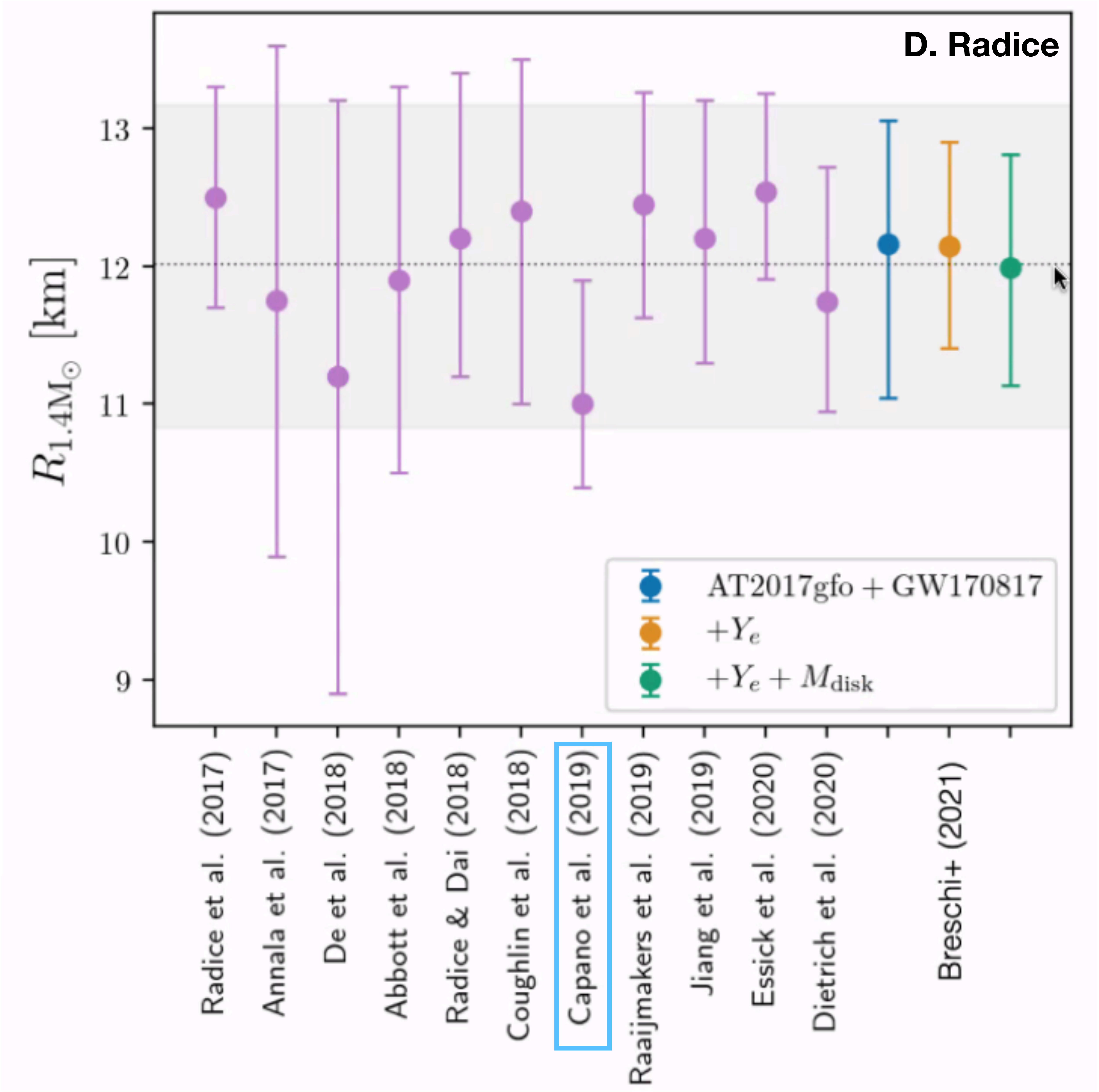
Capano et al., Nature Astron. 4 (**2020**) 6, 625-632



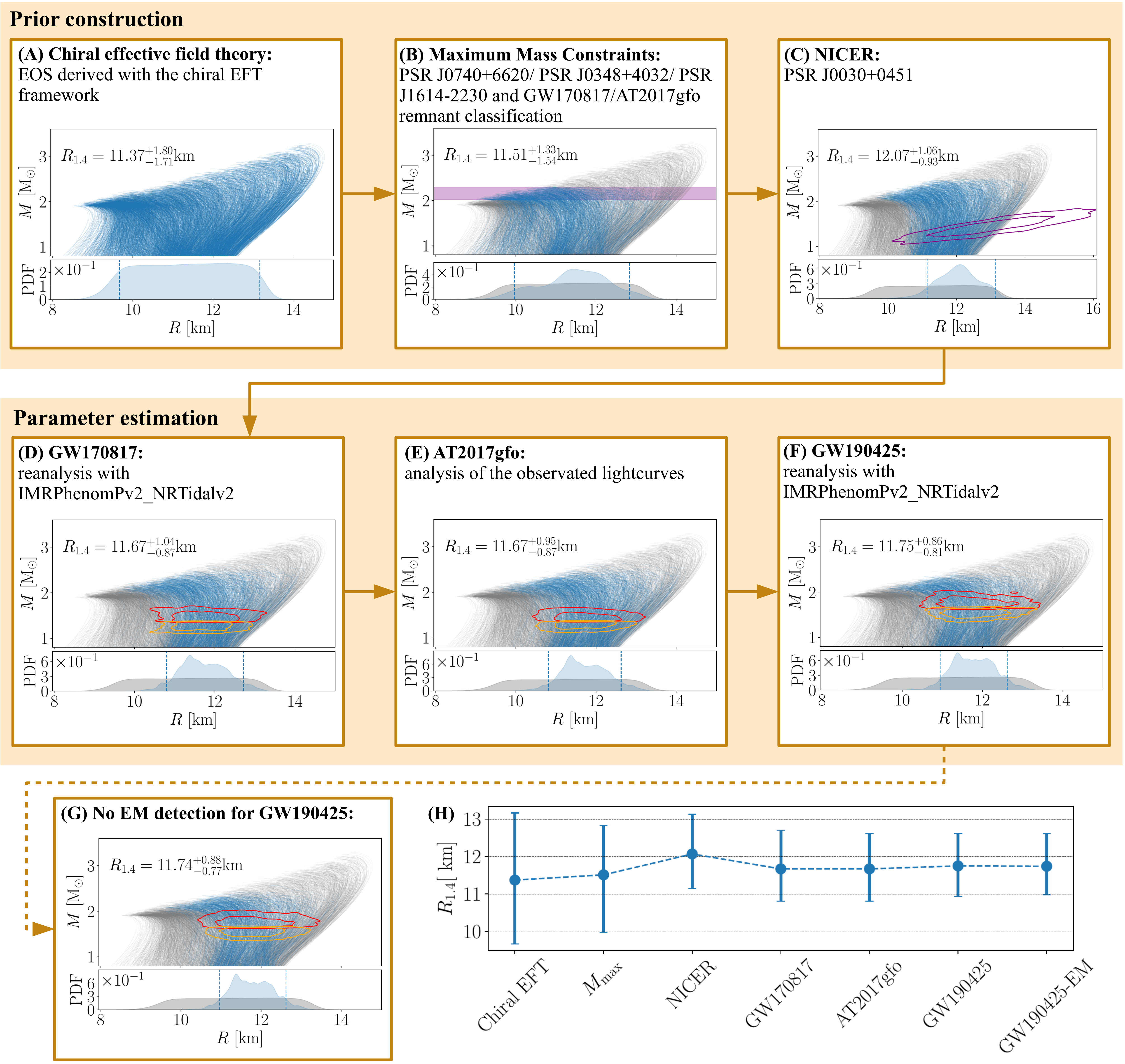




$$R_{1.4\,\mathrm{M}_\odot} = 11.0^{+0.9}_{-0.6}\,\mathrm{km}$$

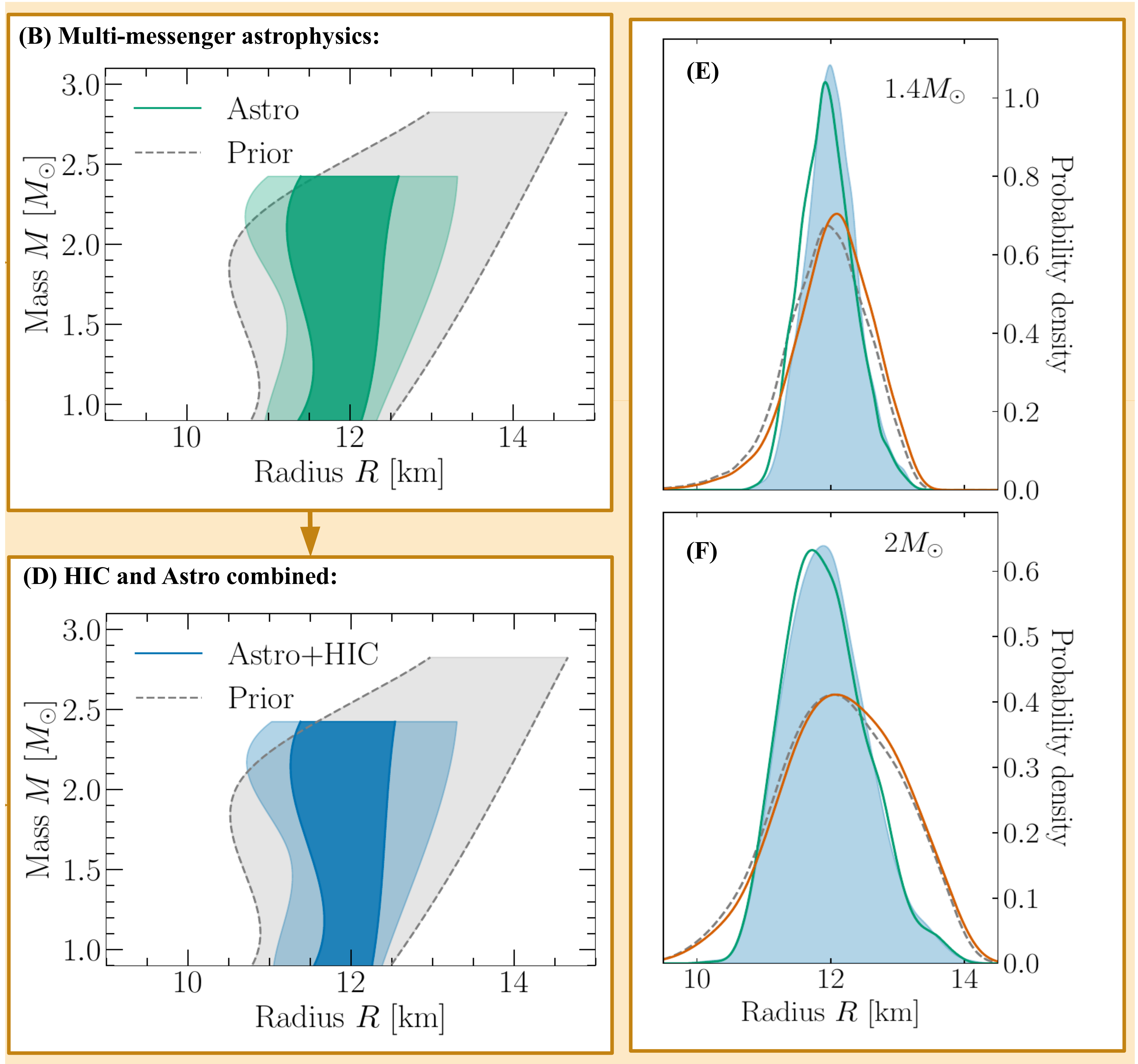
# One of these is not like the others…




NICER J0030+0451 posterior (green) from Miller et al., ApJL 887, L24 (2019)

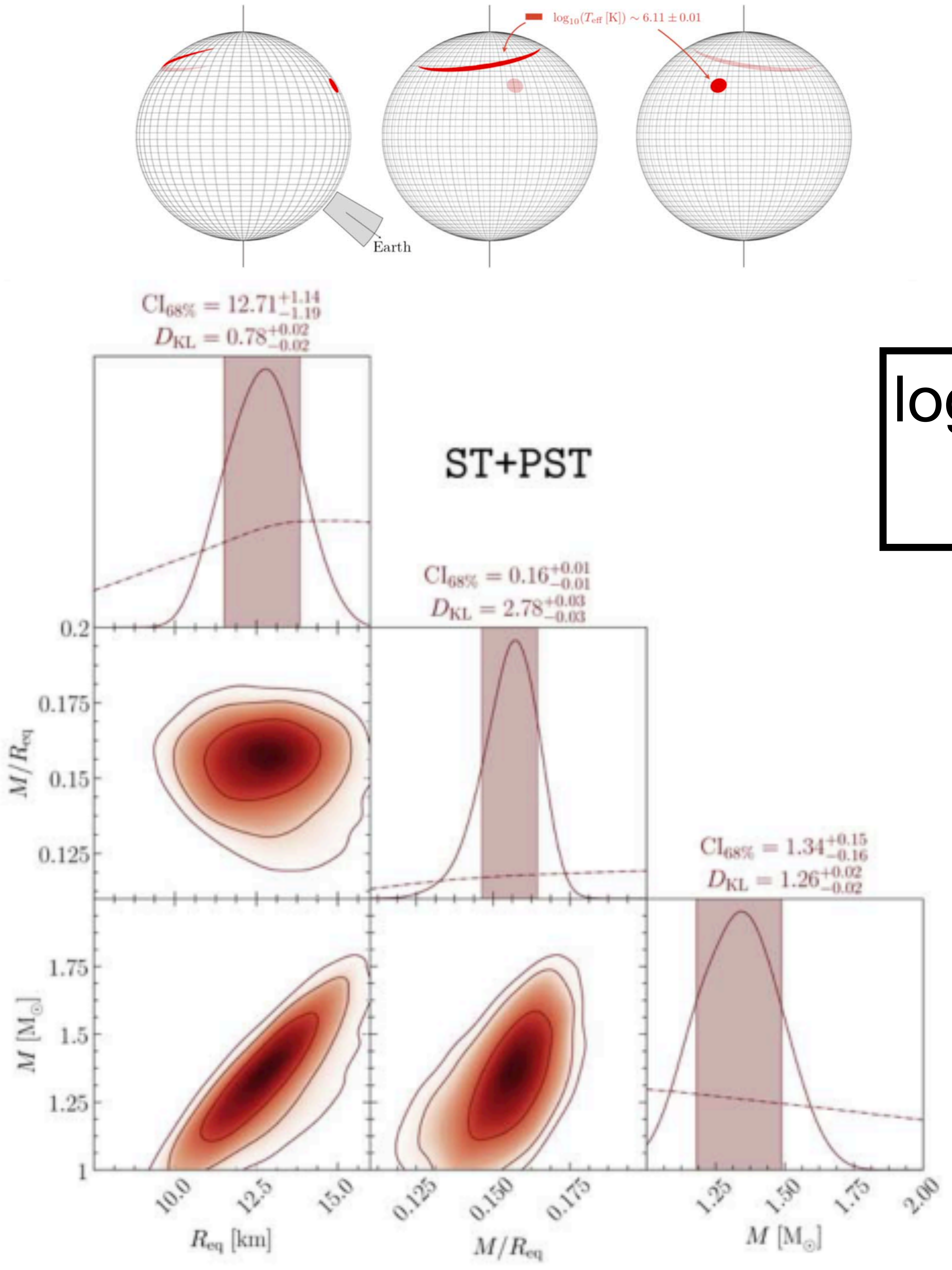



Dietrich et al., Science, Vol. 370, Issue 6523, pp. 1450-1453 (2020)



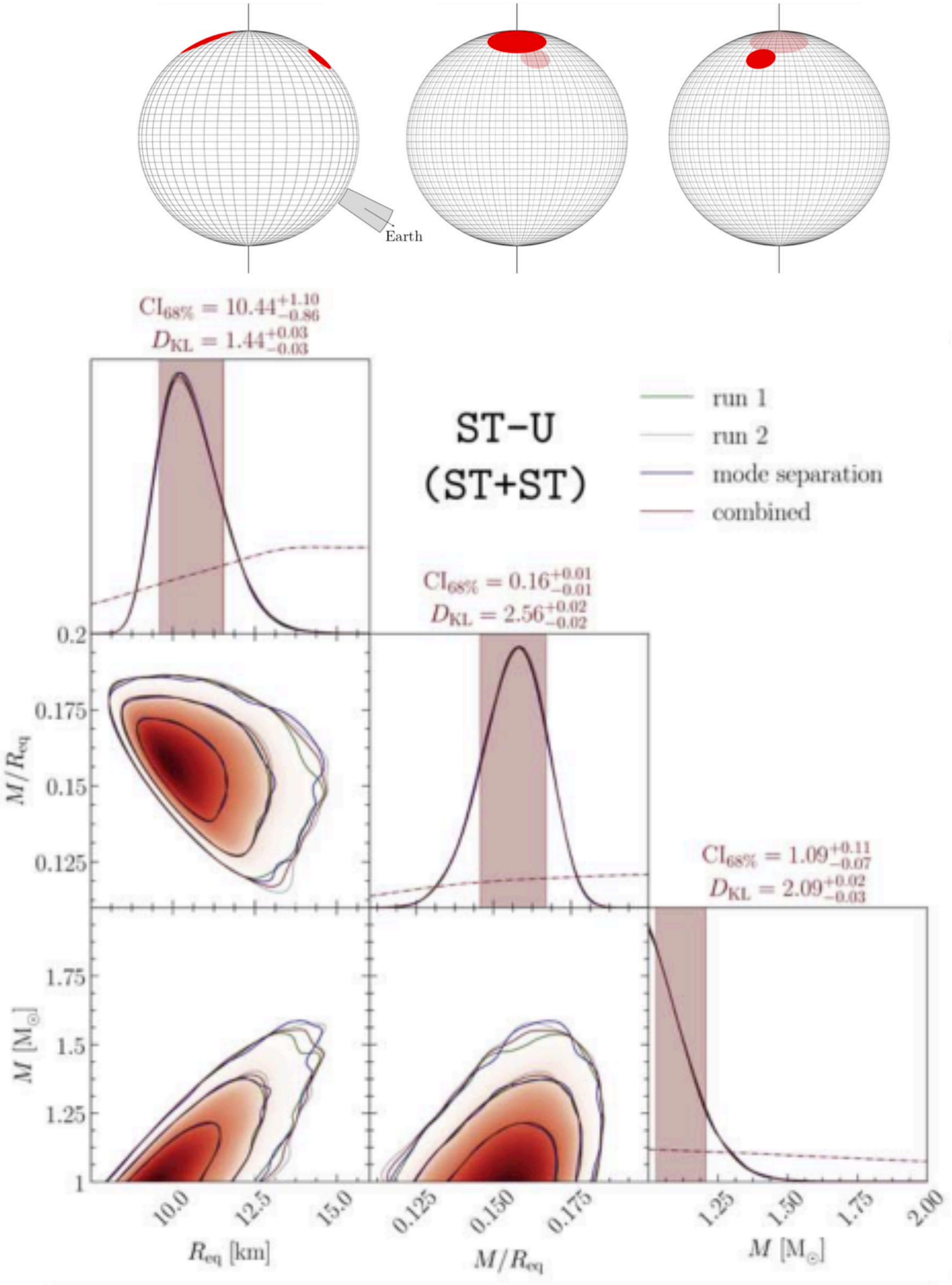

Huth et al., Nature 606, 276-280 (2022)

# NICER J0030+0451



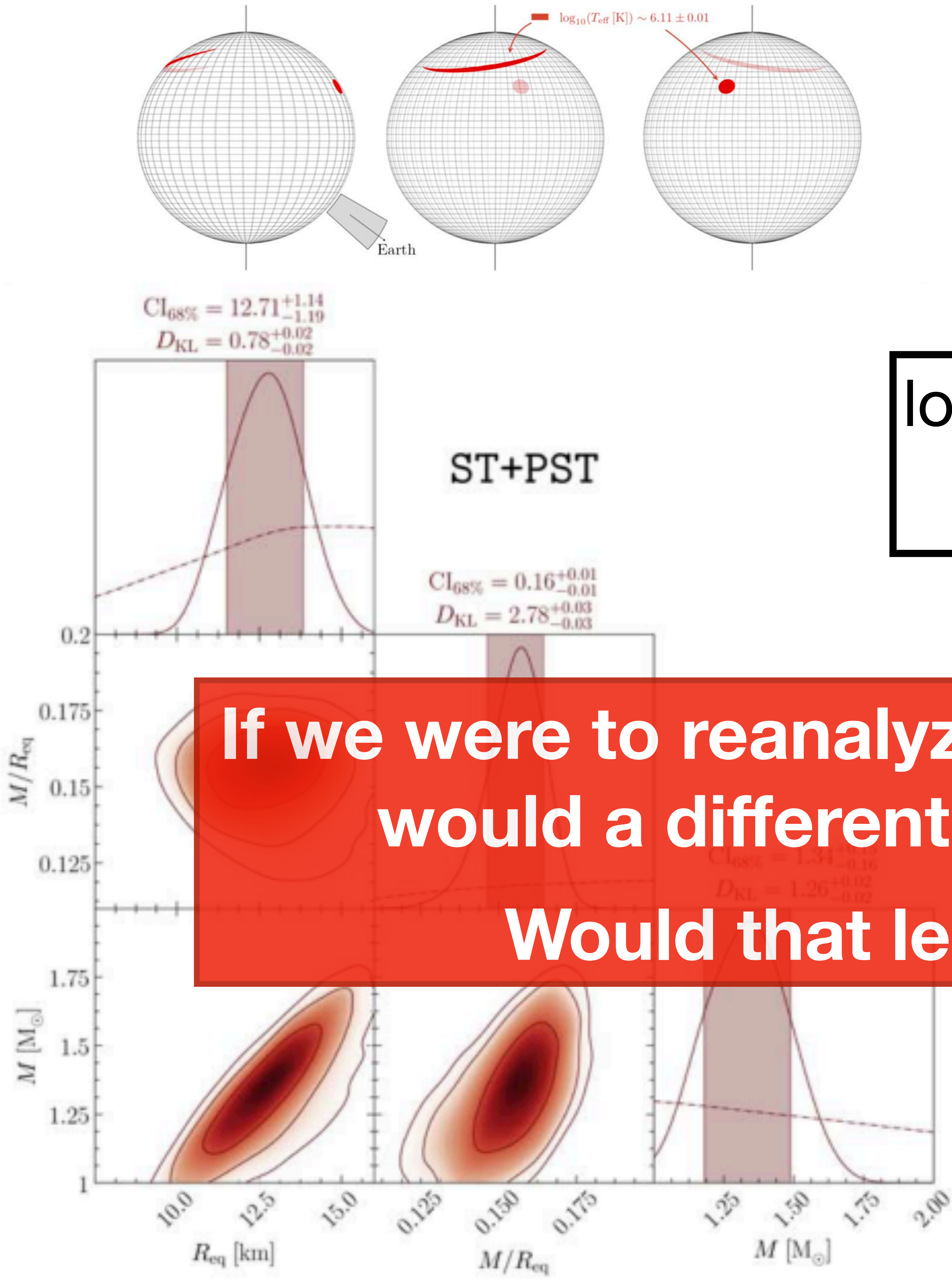

$\log Z_{\mathrm{ST+PST}} - \log Z_{\mathrm{ST\text{-}U}}$

$= 9.4$



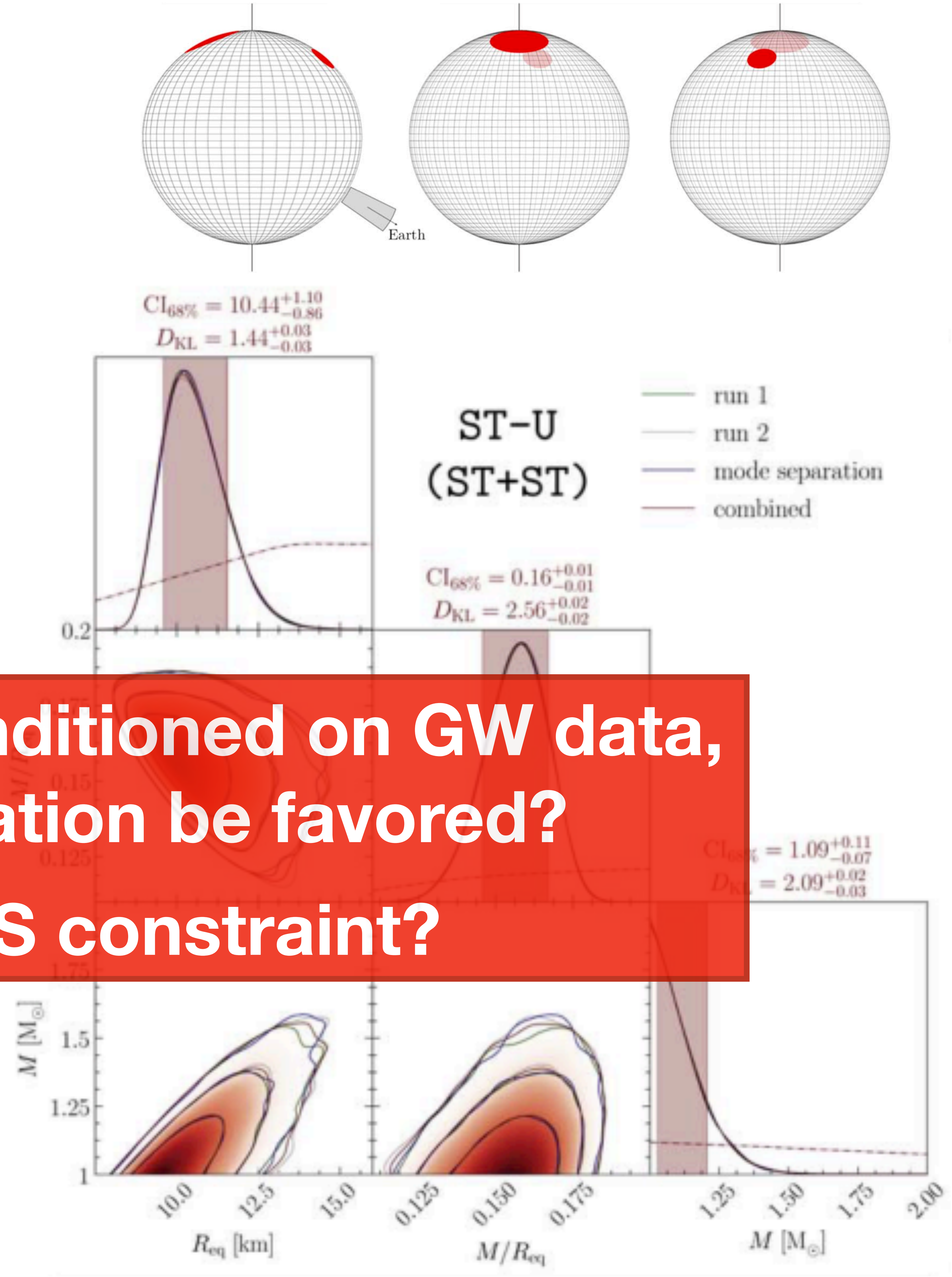

Riley et al., ApJL 887, L21 (2019)

# NICER J0030+0451




log $Z_{ST+PST}$ - log $Z_{ST\text{-}U}$

= 9.4




**If we were to reanalyze NICER data conditioned on GW data, would a different hotspot configuration be favored?**

**Would that lead to different EOS constraint?**

Riley et al., ApJL 887, L21 (2019)

# How to input GW results into NICER/XMM analysis

**Option 1: Use the GW170817 EOS posterior as a prior for NICER analyses**

- Problem: difficult to get Bayesian evidence this way.
  - Most nested samplers (e.g., MultiNest, dynesty) cannot take arbitrary prior distributions.
  - MCMC solutions require parallel tempering: computationally expensive and hard to get reliable evidence estimate.

# How to input GW results into NICER/XMM analysis

**Option 2: Jointly analyze GW & NICER data together** (i.e., do one giant Bayesian inference analysis over GW170817, PSR J0030 and/or PSR J0740)

- Advantages: can use nested samplers; get better constraints on all parameters; potentially learn about correlations between unexpected parameters
- Challenges:
  - Not scalable — but doable for a few events at a time
  - Requires combining code & analysis methods from very different disciplines* (here, GW and X-ray astronomy)

*but see Pang et al., arXiv:2205.08513

# How to input GW results into NICER/XMM analysis

**Option 2: Jointly analyze GW & NICER data together** (i.e., do one giant Bayesian inference analysis over GW170817, PSR J0030 and/or PSR J0740)

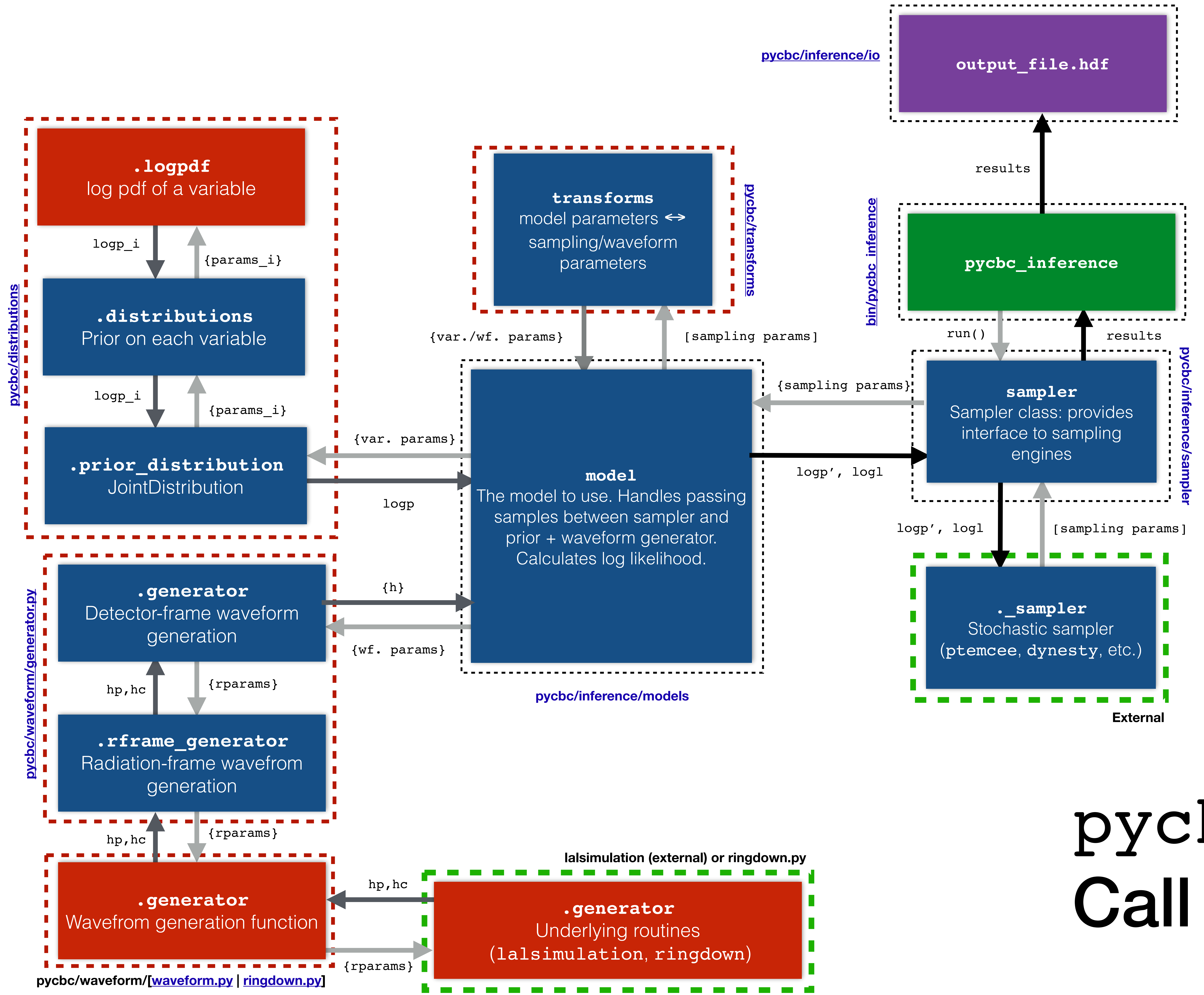
- Advantages: can use nested samplers; get better constraints on all parameters; potentially learn about correlations between unexpected parameters
- Challenges:
  - Not scalable — but doable for a few events at a time
  - Requires combining code & analysis methods from very different disciplines* (here, GW and X-ray astronomy)
  - **Problem: No one likes using other people's code.**

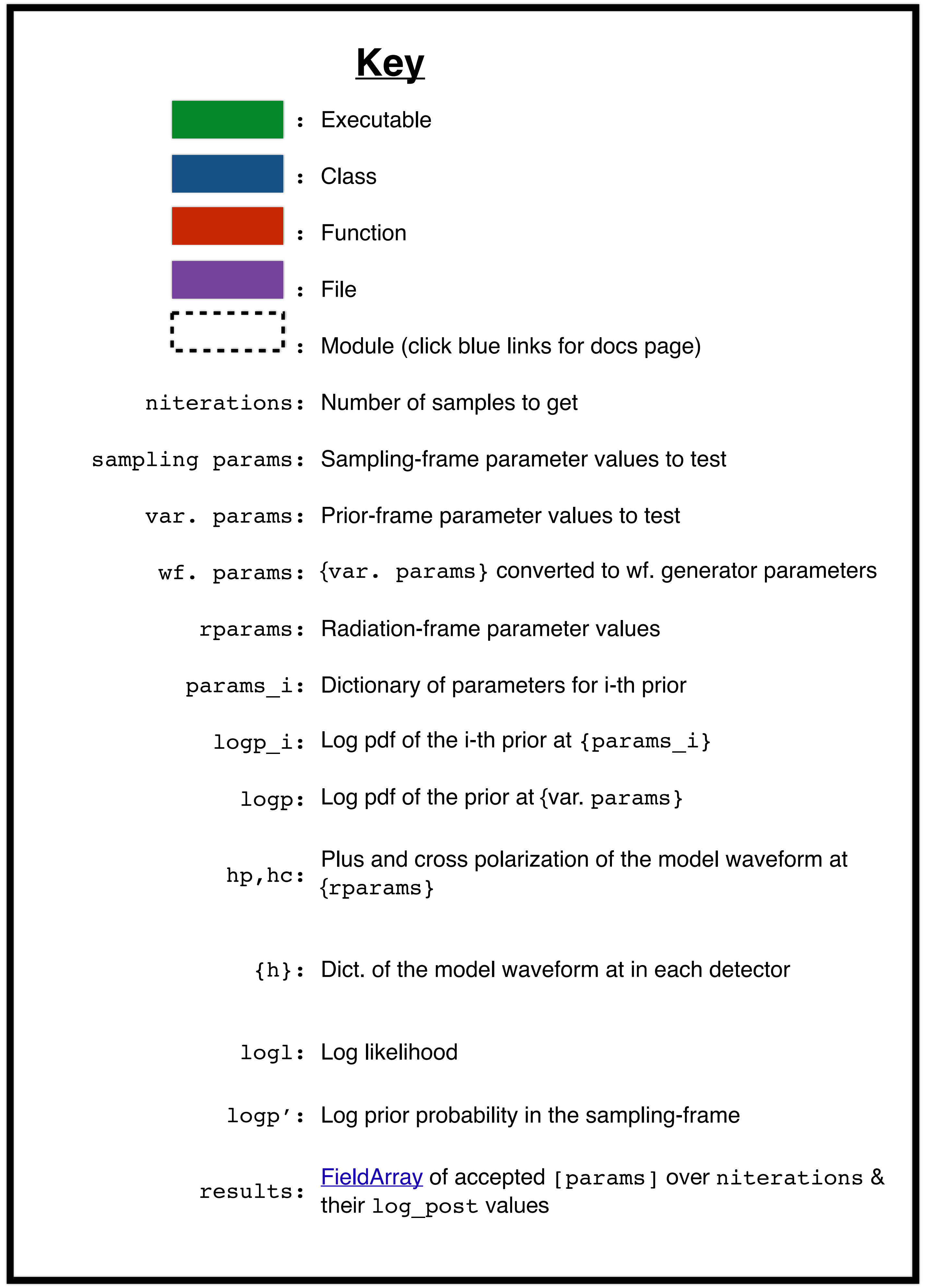
*but see Pang et al., arXiv:2205.08513

# PyCBC Inference

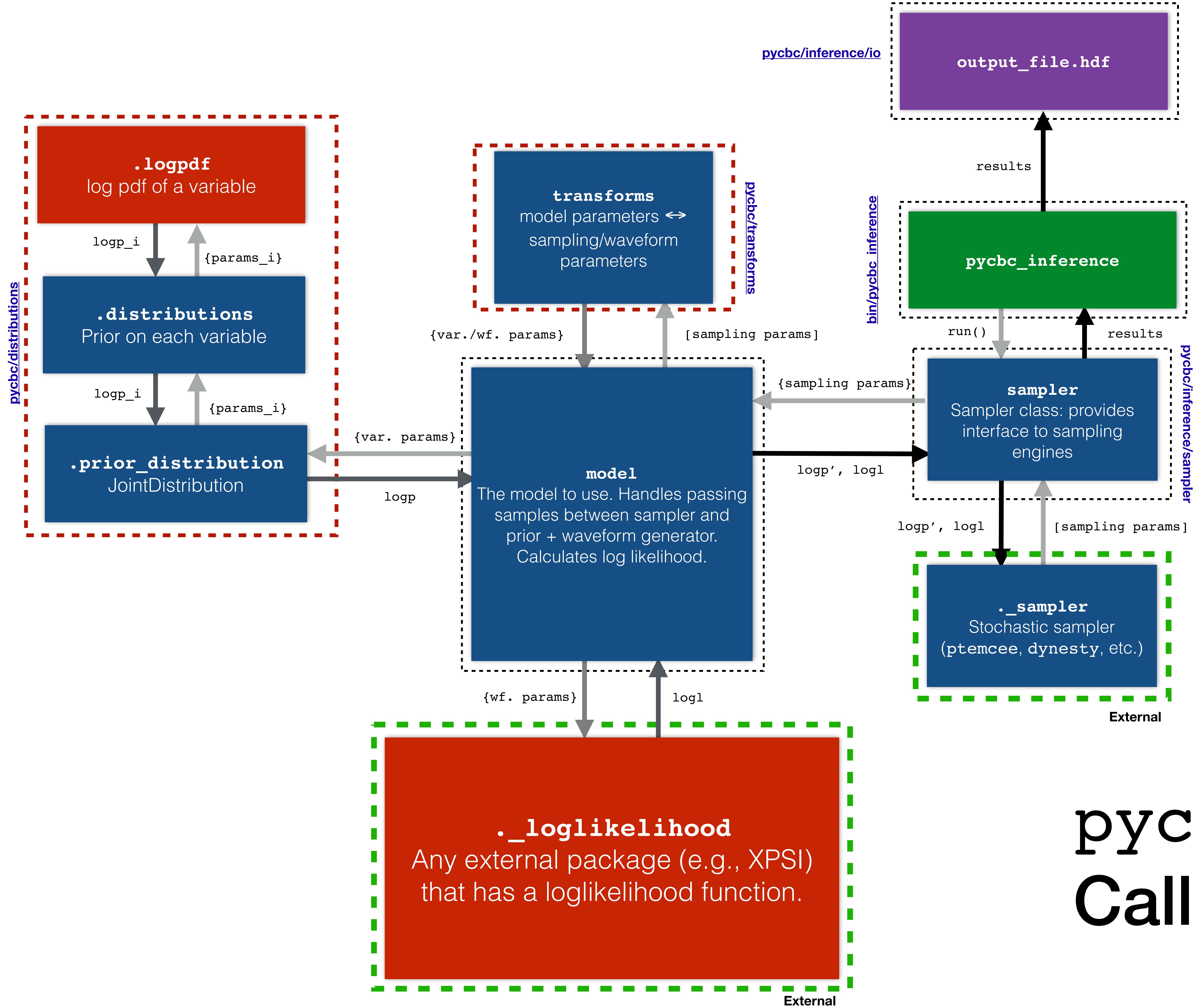
- Python package for doing Bayesian inference with gravitational waves
- Part of the larger PyCBC package
- Nature of GW problems — different model waveforms, different types of analyses (regular inference, testing GR problems) — forced us to make PyCBC Inference very modular
    - i.e., various steps involved in general Bayesian inference are abstracted into smaller component modules.
- What data to analyze and signal/noise model to use set by config text files

PyCBC




# pycbc_inference Call Graph




## Key

| | |
|---|---|
| (green) | Executable |
| (blue) | Class |
| (red) | Function |
| (purple) | File |
| (dotted box) | Module (click blue links for docs page) |
| `niterations`: | Number of samples to get |
| `sampling params`: | Sampling-frame parameter values to test |
| `var. params`: | Prior-frame parameter values to test |
| `wf. params`: | `{var. params}` converted to wf. generator parameters |
| `rparams`: | Radiation-frame parameter values |
| `params_i`: | Dictionary of parameters for i-th prior |
| `logp_i`: | Log pdf of the i-th prior at `{params_i}` |
| `logp`: | Log pdf of the prior at {var. `params`} |
| `hp,hc`: | Plus and cross polarization of the model waveform at `{rparams}` |
| `{h}`: | Dict. of the model waveform at in each detector |
| `logl`: | Log likelihood |
| `logp'`: | Log prior probability in the sampling-frame |
| `results`: | FieldArray of accepted `[params]` over `niterations` & their `log_post` values |

# pycbc_inference Call Graph

# Plug-in models

- Python has the ability to create “plug-in” packages
  - we make use of this to support “plug-in models” in PyCBC
- Say you have your own code that analyzes some EM data (or anything else)
- You can use it with PyCBC without needing to modify/interact with/curse at the PyCBC source code.
- Wrap your likelihood function with the appropriate API and add the appropriate lines to your installation file.
- PyCBC will automatically detect it at run-time, and can use it in `pycbc_inference`

# An XPSI plug-in for PyCBC

- **XPSI**[1]: developed by Riley et al. to analyze NICER & XMM Newton data [see Serena Vinciguerra's talk]
- Used in analysis of PSR J0030+0451[2] & PSR J0740+6620[3]
- Here, we create a XPSI plugin for PyCBC

model.py + (~/mnt/atlas/projects...c-xpsi-plugin/pycbc_xpsi) - VIM3

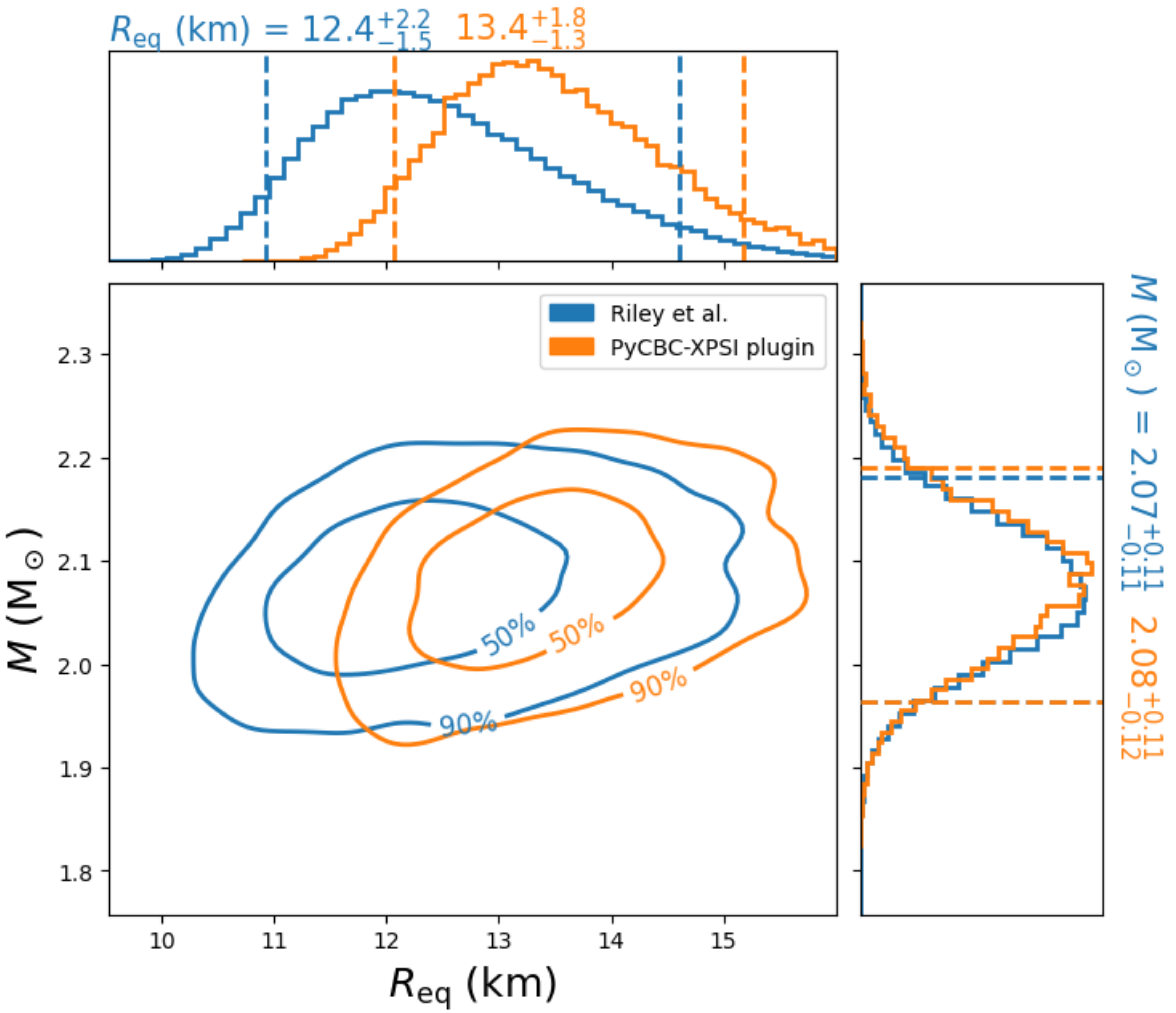
```python
import xpsi
from pycbc.inference.models import BaseModel

class XPSIModel(BaseModel):
    """Model wrapper around XPSI likelihood function."""
    name = 'xpsi'

    # we need to alias some of the parameter names to be compliant with
    # pycbc config file sections
    _param_aliases = {
        'XTI__alpha': 'nicer_alpha',
        'PN__alpha': 'xmm_alpha'
        }

    def __init__(self, variable_params, star, signals, num_energies, **kwargs):
        super().__init__(variable_params, **kwargs)
        # set up the xpsi likelihood
        self._xpsi_likelihood = xpsi.Likelihood(star=star, signals=signals,
                                                num_energies=num_energies,
                                                externally_updated=True)
        # store a dictionary of param aliases
        self.param_aliases = {p: p for p in self._xpsi_likelihood.names}
        self.param_aliases.update(self._param_aliases)

    def _loglikelihood(self):
        # map the current parameters to the ordered list
        params = self.current_params
        # update the underlying likelihood
        for p in self._xpsi_likelihood.names:
            try:
                self._xpsi_likelihood[p] = params[self.param_aliases[p]]
            except StrictBoundsError:
                return -numpy.inf
        # check addtional constraints
        if not self.apply_additional_constraints():
            return -numpy.inf
        logl = self._xpsi_likelihood()
        # FIXME: for some reason, this occasionally returns arrays of len 1
        # if so, force to float
        if isinstance(logl, numpy.ndarray):
            logl = logl.item()
        return logl

    @classmethod
    def from_config(cls, cp, **kwargs):
        args = cls.xpsi_args_from_config(cls, cp, **kwargs)
        return cls(**args)
                                                                59,19          2%
```

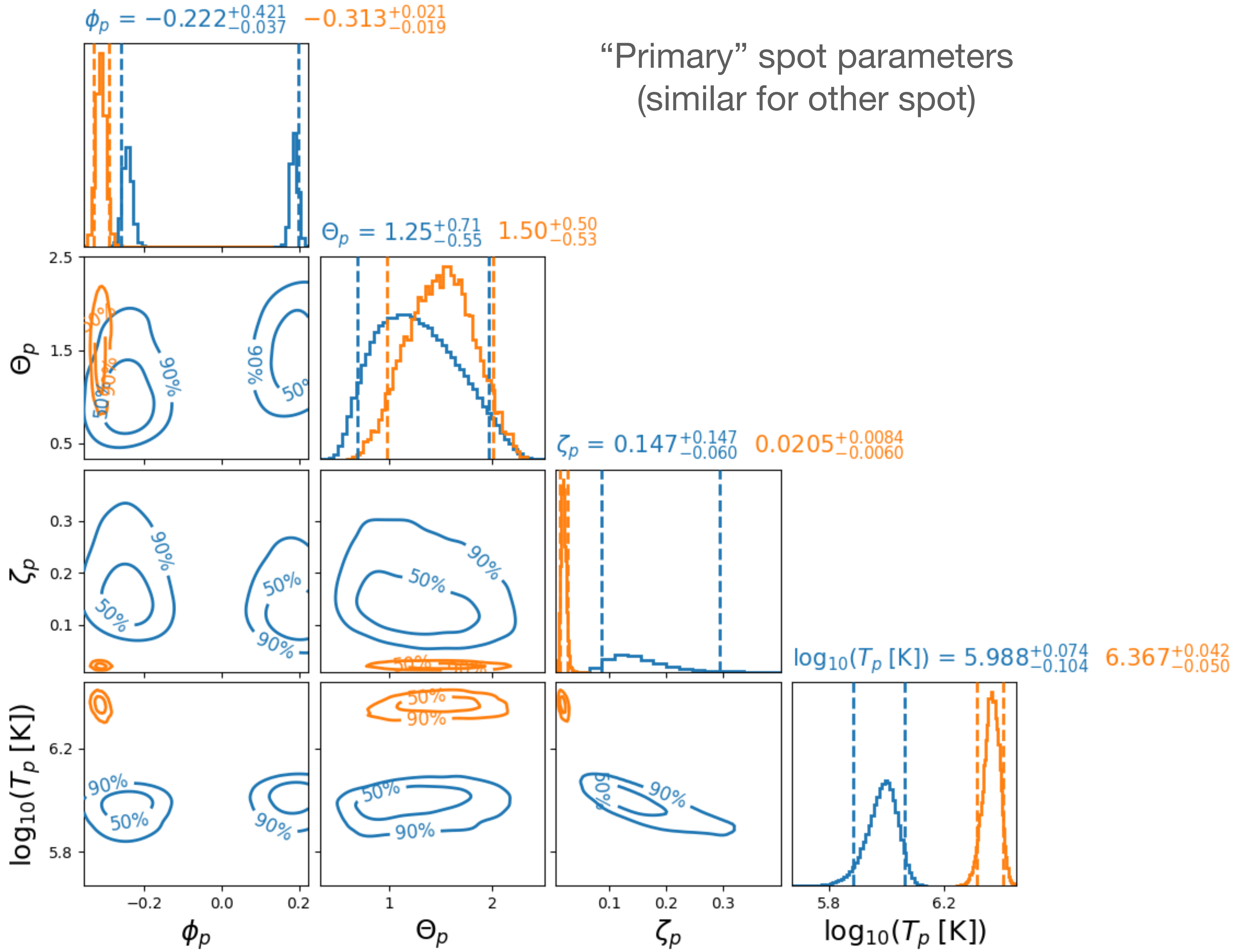
setup.py (~/mnt/atlas/projects/pycbc_xpsi/pycbc-xpsi-plugin) - VIM4

```python
#!/usr/bin/env python
"""
Setup file for XPSI plugin model for PyCBC.
"""

from setuptools import (setup, find_packages)

VERSION = '0.1.dev0'
NAME = 'pycbc_xpsi'
URL = 'https://github.com/cdcapano/pycbc-xpsi-plugin'

setup(
    name=NAME,
    version=VERSION,
    description='XPSI plugin model for PyCBC',
    author='Collin Capano',
    author_email='cdcapano@gmail.com',
    url=URL,
    download_url='{}/tarball/v{}'.format(URL, VERSION),
    keywords = ['pycbc', 'xpsi', 'nicer', 'bayesian inference',
                'gravitational waves', 'x-ray astronomy',
                'multimessenger astronomy'],
    install_requires = ['pycbc', 'xpsi'],
    packages=find_packages(),
    entry_points = {
        "pycbc.inference.models": "pycbc_xpsi = pycbc_xpsi:XPSIModel",
        },
    classifiers=[
        'Programming Language :: Python',
        'Programming Language :: Python :: 3',
        'Intended Audience :: Science/Research',
        'Natural Language :: English',
        'Topic :: Scientific/Engineering',
        'Topic :: Scientific/Engineering :: Astronomy',
        'Topic :: Scientific/Engineering :: Physics',
        'License :: OSI Approved :: GNU General Public License v3 (GPLv3)',
        ],
)
                                                                6,45           All
```


1. https://github.com/xpsi-group/xpsi
2. Riley et al., ApJL 887 L21 (2019)
3. Riley et al., ApJL 918 L27 (2021)

# Test: replicate J0740 analysis with PyCBC (NICER+XMM only)




"Primary" spot parameters
(similar for other spot)

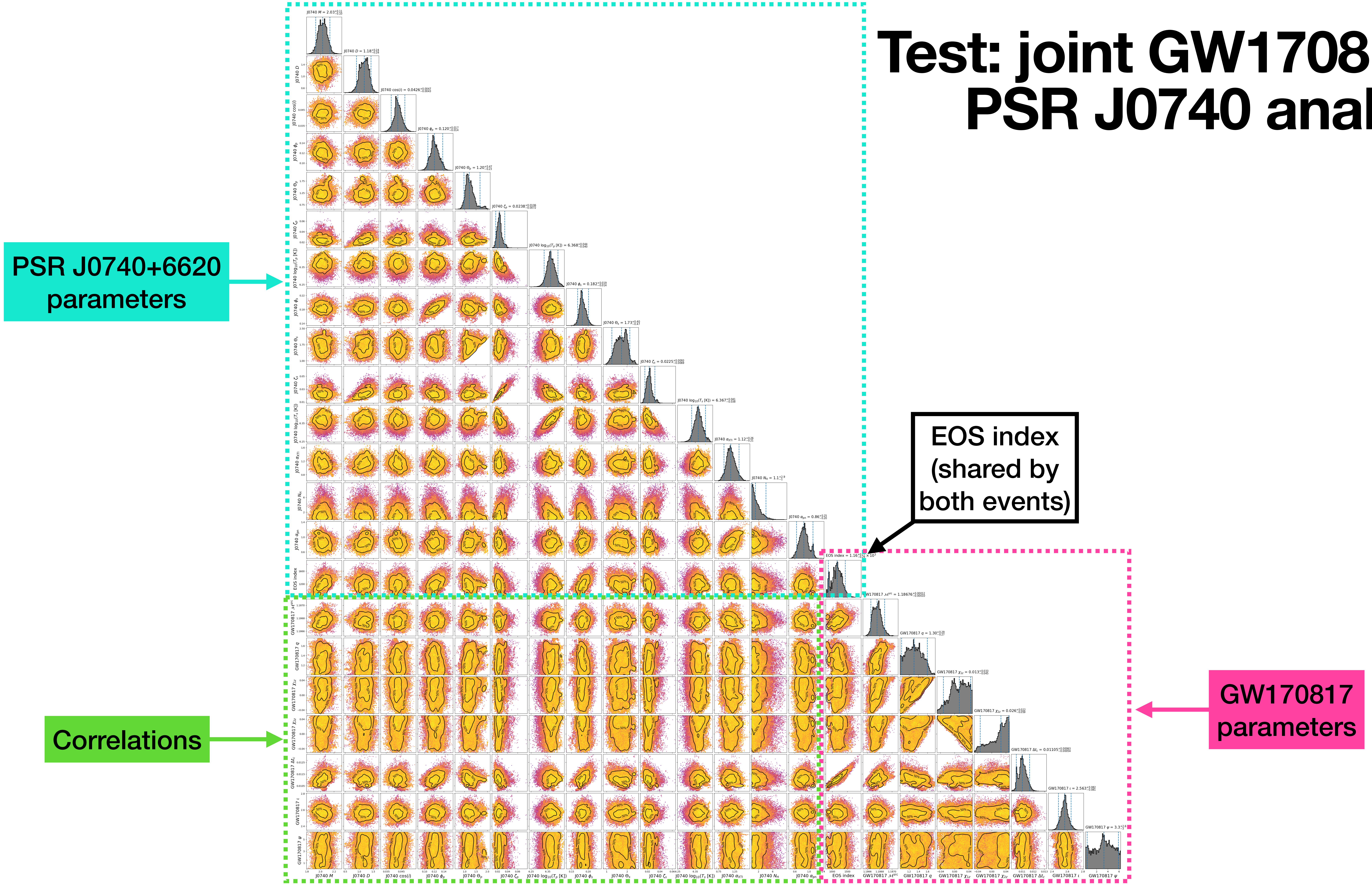
- Close, but some bugs to work out, especially with hotspot orientation.
- Differences possibly due to mismatch between XPSI version used in publication & version used here.

# Hierarchical model

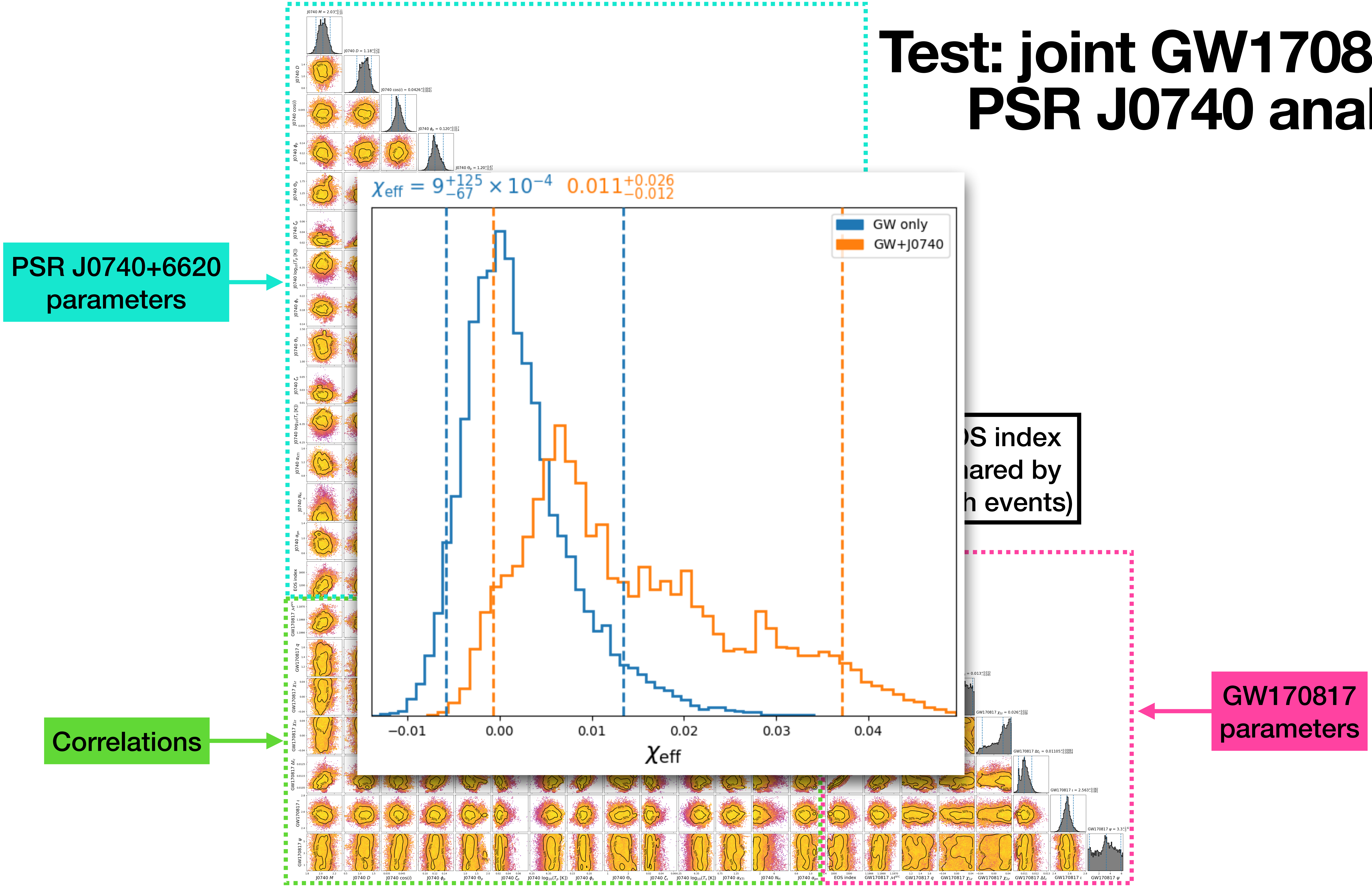
- The hierarchical model in PyCBC is a model of models
  - Takes product of likelihoods from constituent models
- Use hierarchical model to combine standard GW models in PyCBC with plugin models

  ➡ can perform joint GW + anything-else Bayesian inference
- We use the hierarchical model with our XPSI plugin to jointly analyze GW170817with PSR J0740 (and eventually PSR J0030)

# Test: joint GW170817 + PSR J0740 analysis



# Test: joint GW170817 + PSR J0740 analysis



# Summary

- We are able to jointly analyze NICER/XMM and GW data using XPSI plugin model for PyCBC.
- Work in progress!
- Plan: test different hotspot configurations; analyze PSR J0030; sample over EOS parameters directly.
- More generally, using Python plugin modules (whether with PyCBC or something else) to do Bayesian inference over multi-messenger data very promising.
- For more on PyCBC — including documentation, tutorials, and help on building your own plugins — go to: https://pycbc.org

**Thank you!**

# Extras

# Test: replicate J0740 analysis (NICER+XMM only) Full corner plot