GRIFFIN HOSSEINZADEH

Intro To Bayesian Statistics

Acknowledgements & Resources

- LSSTC Data Science Fellowship Program
  https://github.com/LSSTC-DSFP/LSSTC-DSFP-Sessions
- Jake VanderPlas
  https://github.com/jakevdp/BayesianAstronomy
- Dan Foreman-Mackey
  https://emcee.readthedocs.io
  https://george.readthedocs.io

Outline

- Interpretation of probability
- Bayes' Theorem
- When should I use Bayesian statistics?
- How should I choose priors?
- Sampling techniques

Error Bars: What Do They Mean?

Frequentist (Aristotle)

- Values you could get in *repeated trials*
- Data are uncertain; universe is constant

Bayesian (Thomas Bayes)

- *Degree of belief* in results of your experiment
- Data are constant; universe is uncertain

Bayes' Theorem

- We have some measurements, and we want to know what they tell us about the universe.
- Model with parameters θ. Data D. Calculate $P(\theta \mid D)$.
- Using $P(A \mid B)\, P(B) = P(A, B) = P(B \mid A)\, P(A)$ we have

$$P(\theta \mid D) = \frac{P(D \mid \theta) P(\theta)}{P(D)}$$

Bayes' Theorem

$$P(\theta \mid D) = \frac{P(D \mid \theta)P(\theta)}{P(D)}$$

- **Posterior**: $P(\theta \mid D)$ = probability of the parameters having certain values now that we have our data.
- **Likelihood**: $P(D \mid \theta)$ = probability of getting our observations given certain values of the model parameters. (Think χ^{-2}.)
- **Prior**: $P(\theta)$ = probability of those model parameters before taking data.
- $P(D)$ = probability of getting our observations.
  You can usually ignore this as a normalization constant.

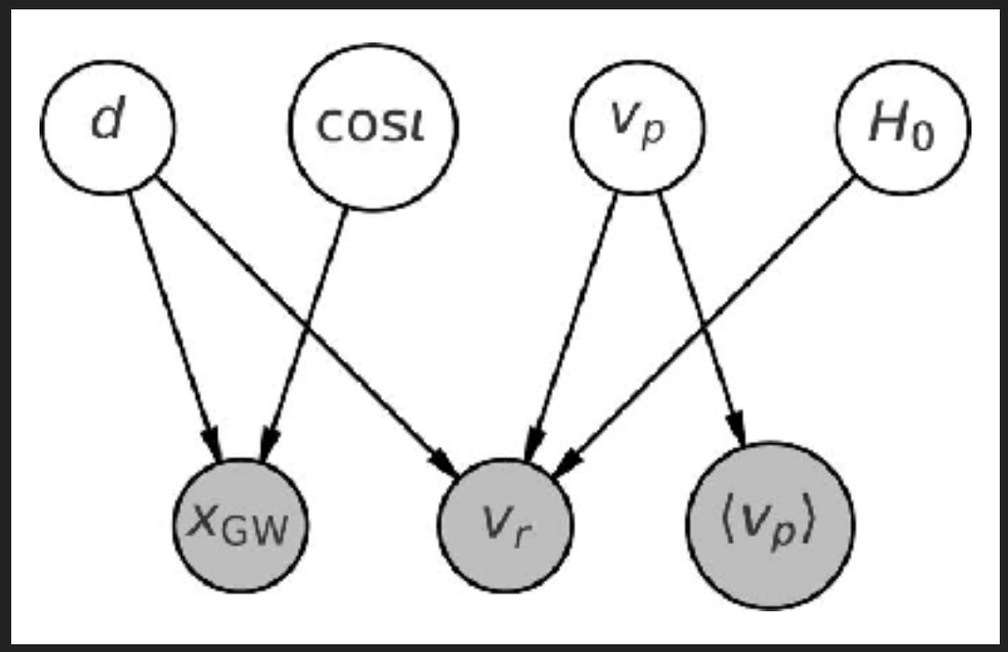
When Should I Use This?

- You have an analytical model and you want to find the parameters that best fit your data.
- You may not care about all the parameters but want to account for the fact that you do not know them exactly (“marginalization”).

$$P(\theta_{\text{interesting}}) = \int P(\theta_{\text{interesting}}, \theta_{\text{nuisance}})\, d\theta_{\text{nuisance}}$$

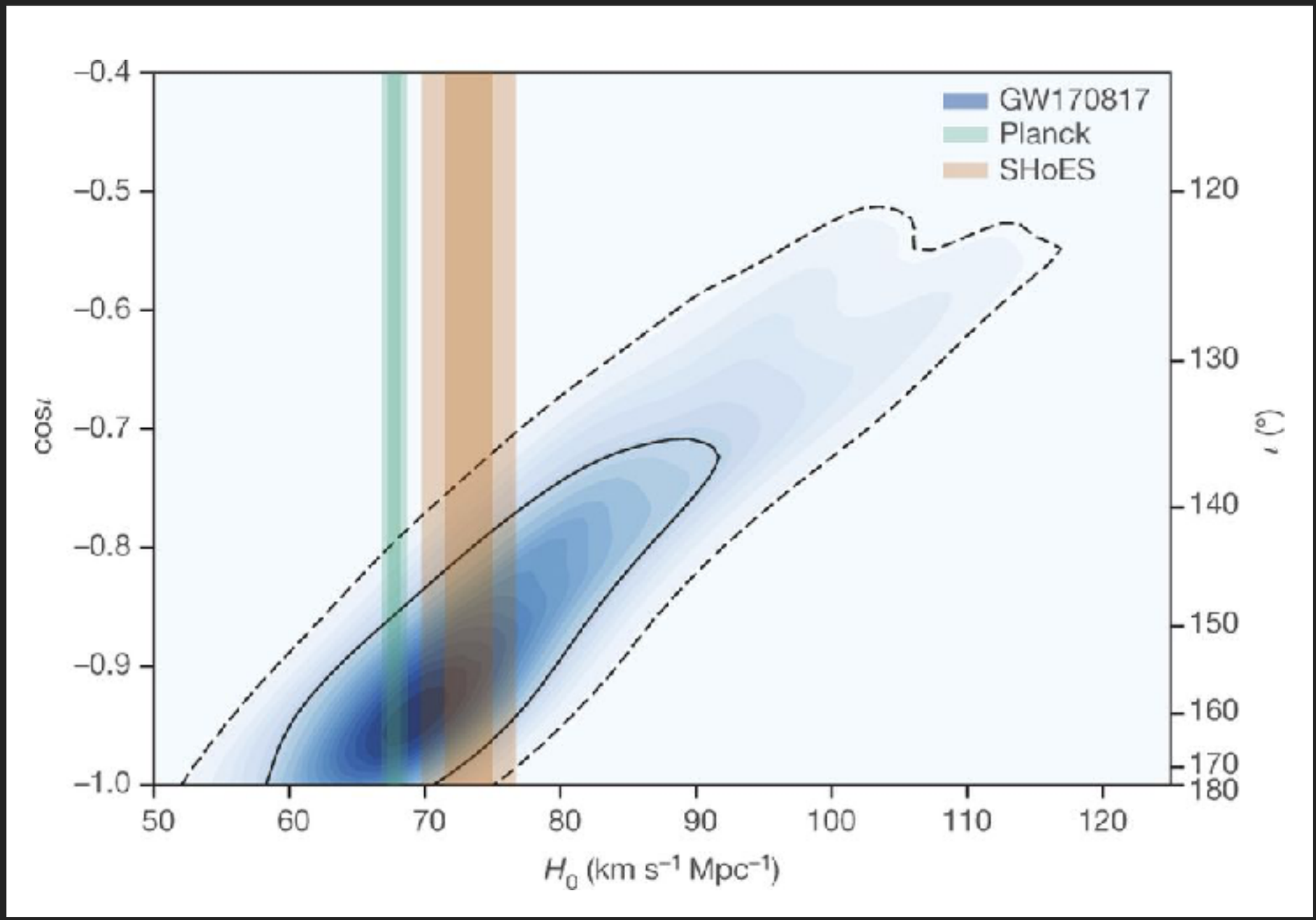
When Should I Use This? (Example)

- You want to measure the Hubble constant (H_0) based on data from a binary neutron star (NS) merger.
- Data: gravitational wave signal, recession velocity of host galaxy (redshift), average peculiar velocity of galaxies near host.
- Nuisance parameters: NS masses, NS spins, NS tidal deformability, binary inclination, distance to galaxy, peculiar velocity of galaxy, etc.



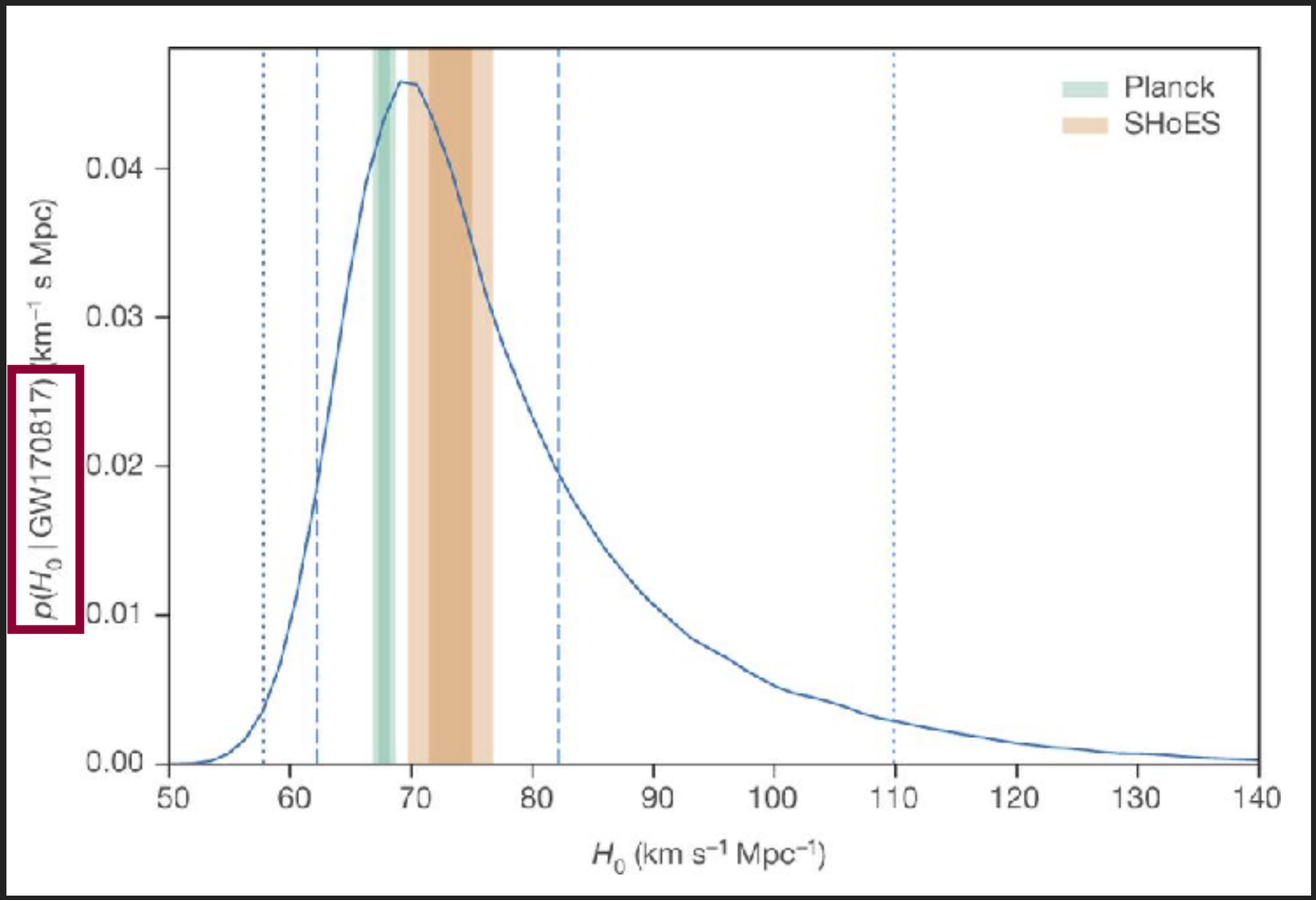
LIGO Scientific Collaboration & Virgo Collaboration et al. 2017

When Should I Use This? (Example)



LIGO Scientific Collaboration & Virgo Collaboration et al. 2017

When Should I Use This? (Example)



LIGO Scientific Collaboration & Virgo Collaboration et al. 2017

Priors: This Is the Tricky Part

- $P(\theta)$ = probability of model parameters having certain values before taking data.
- Options:
  - non-informative prior
  - posterior distribution from previous experiment

What Is a Non-Informative Prior?

- Ideally, using a different (but still reasonable) prior should not significantly affect your results.
- If you think you have a non-informative prior, but your prior still affects your results, it means your data are not very constraining.
- Simple options:
  - flat prior $P(\theta) \propto \begin{cases} 1, & \theta_{\min} < \theta < \theta_{\max} \\ 0, & \text{otherwise} \end{cases}$
  - log prior $P(\theta) \propto \begin{cases} \theta^{-1}, & \theta_{\min} < \theta < \theta_{\max} \\ 0, & \text{otherwise} \end{cases} \quad P(\log\theta) \propto \begin{cases} 1, & \theta_{\min} < \theta < \theta_{\max} \\ 0, & \text{otherwise} \end{cases}$
- For an insane amount of detail, look up Jeffreys Prior on Wikipedia.

Okay, We Wrote Down the Posterior. Now What?

- Remember we want the whole probability distribution, not just the best-fit parameters!
- Options:
  - I guess it could be analytically tractable…but that never happens with real data.
  - Calculate $P(\theta \mid D)$ over a grid of all parameters.
  - Randomly sample the parameter space (Monte Carlo).
  - Markov chains (will discuss this after the break).

Exercise: Fitting a Line to Some Noisy Data

Pretend we took some measurements (x_i, $y_i \pm dy_i$).
We think they should be modeled by a straight line: $y_i = mx_i + b$.
So we are fitting for two parameters: $\theta = [m, b]$.

1. Write down the posterior probability distribution.
2. Evaluate the posterior over a grid of parameters and plot the results. Also plot the marginalized posterior for each parameter.
3. Evaluate the posterior over a random sampling of parameters and plot the results.
4. How accurate was each method?

GRIFFIN HOSSEINZADEH

INTRO TO MCMC PACKAGES

MCMC = Markov Chain Monte Carlo

- "Monte Carlo" just means random sampling
- "Markov chain" means a process where each step only depends on the previous state ("memoryless" process)
- e.g., Brownian motion

Monte Carlo Casino, Monaco
Cristian Lorini

The goal of MCMC routines is to create Markov chains whose histories are drawn from the posterior probability distribution.

How Do You Do That?

Metropolis-Hastings algorithm (used by `emcee`)

- Pick a starting value
  - random sample from the prior, or
    Gaussian around the maximum-likelihood value
- Pick a proposed next value from the "proposal density"
  - usually Gaussian centered on θ_i
- Accept the proposal with probability $a = \min\left\{\frac{P(\theta_{i+1}\,|\,D)}{P(\theta_i\,|\,D)}, 1\right\}$

Assessing Convergence

- This algorithm guarantees the correct distribution as $i \to \infty$
- If you don't have infinite time, check if the chains are converged
  - visually: check if the chains are in a steady state
  - quantitatively:
    - autocorrelation time (long is bad)
    - acceptance fraction (0 and 1 are both bad)
    - Gelman-Rubin ratio (inter-chain variance ≈ intra-chain variance)

MCMC Packages in Python

- emcee: The MCMC Hammer (emcee.readthedocs.io)

```python
import numpy as np
import emcee

def lnprob(x, ivar):
    return -0.5 * np.sum(ivar * x ** 2)

ndim, nwalkers = 10, 100
ivar = 1. / np.random.rand(ndim)
p0 = [np.random.rand(ndim) for i in range(nwalkers)]

sampler = emcee.EnsembleSampler(nwalkers, ndim, lnprob, args=[ivar])
sampler.run_mcmc(p0, 1000)
```

- PyMC3 (docs.pymc.io)

```python
import pymc3 as pm

X, y = linear_training_data()
with pm.Model() as linear_model:
    weights = pm.Normal('weights', mu=0, sd=1)
    noise = pm.Gamma('noise', alpha=2, beta=1)
    y_observed = pm.Normal('y_observed',
                           mu=X.dot(weights),
                           sd=noise,
                           observed=y)

    prior = pm.sample_prior_predictive()
    posterior = pm.sample()
    posterior_pred = pm.sample_posterior_predictive(posterior)
```

Exercise: Fitting a Line to Some Noisy Data

Pretend we took some measurements (x_i, $y_i \pm dy_i$).
We think they should be modeled by a straight line: $y_i = mx_i + b$.
So we are fitting for two parameters: $\theta = [m, b]$.

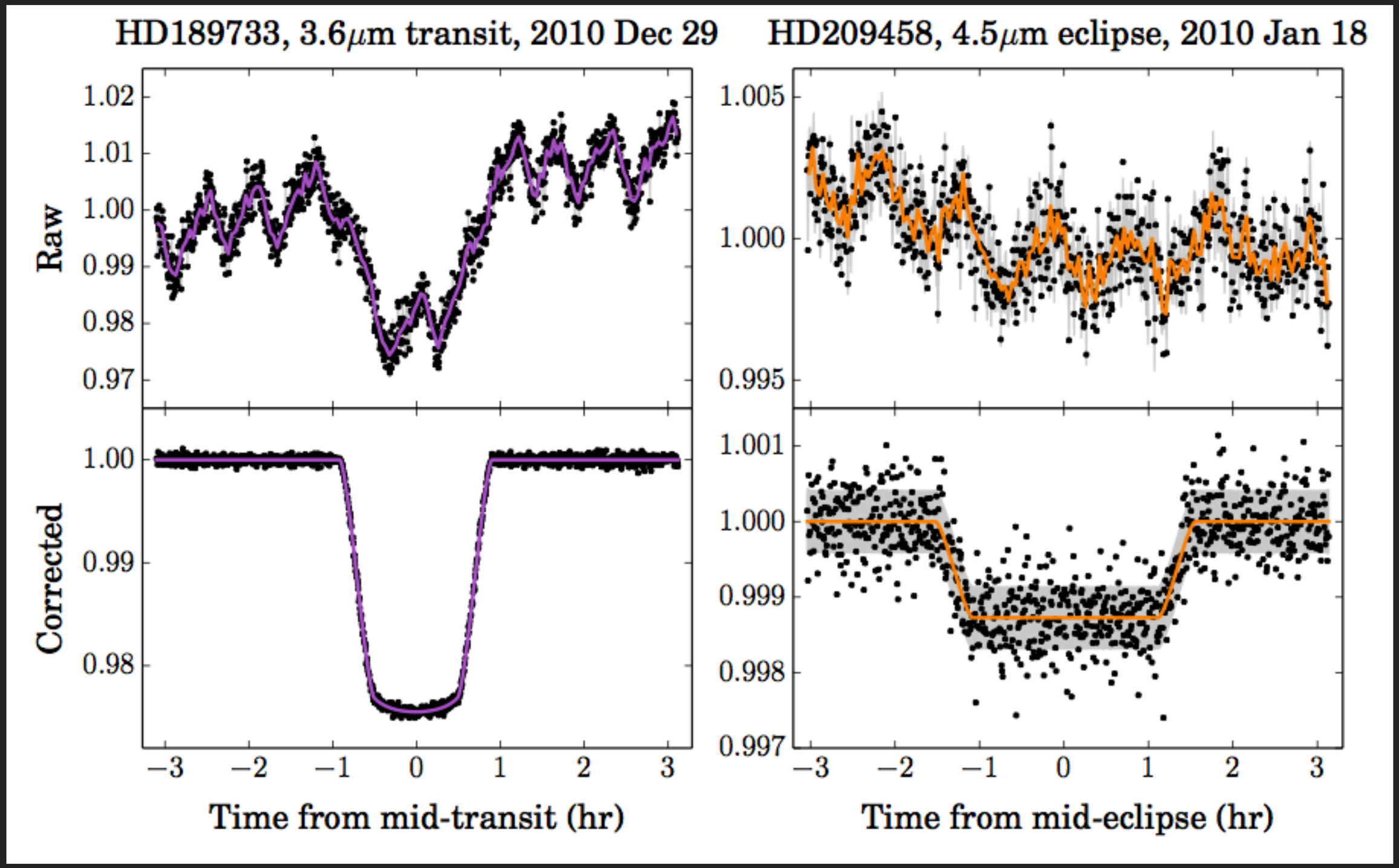
1. Write down the posterior probability distribution.
2. Initialize 100 MCMC walkers randomly from the prior.
3. Initialize and run the MCMC ensemble sampler. How do your results compare to the previous methods? How did the run time compare?
4. Check if your walkers are converged. How many steps did it take?

GRIFFIN HOSSEINZADEH

INTRO TO GAUSSIAN PROCESSES

What Problem Are We Trying To Solve?

- Model/interpolate a set of data points (observations) that come from an unknown (or overly complicated) function
- Deal correctly with correlated noise
  - time-series data **always** have correlated noise
- Want uncertainties on the interpolation (well constrained near the points, less constrained half way between points)



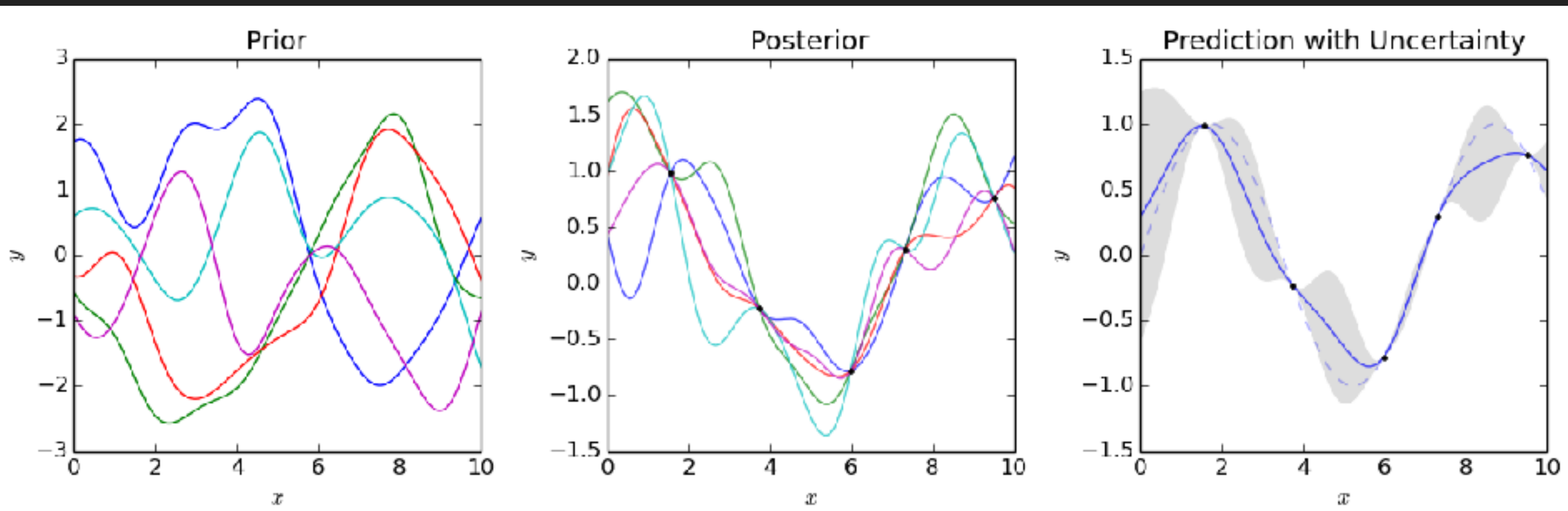
***Spitzer* Exoplanet Transits and Eclipses**

Evans et al. 2015

What Is a Gaussian Process?

- Completely specified by a "mean function" $m(x)$ and a "covariance function" or "kernel" $k(x_i, x_j)$
  - often use a mean of zero
- Each point is a Gaussian around the mean function with width determined by the covariance function
- "Prior" includes any function whose points are drawn from the probability distribution $P(y_i \mid x_i, x_j) = G(m(x_i), k(x_i, x_j))$
- "Posterior" includes all such functions that go through your points (within the error bars)

What Is a Gaussian Process?



Wikimedia User: Cdipaolo96

What Kernel Should I Use?

- Can be any symmetric, non-negative function
- Common choices:
  - Squared exponential $\exp\left(-\Gamma(x_i - x_j)^2\right)$
  - Matérn $\left(1 + \frac{\sqrt{2\nu}(x_i - x_j)}{l}\right)^{\nu} \exp\left(-\frac{\sqrt{2\nu}(x_i - x_j)}{l}\right)$
  - Periodic (trig functions, or exponentials of trig functions)
- Can multiply kernels together to make more complex kernels

What Kernel Should I Use?

- Kernels introduce “hyperparameters”
  - Squared exponential: Γ
  - Matérn: ν, ℓ
  - Periodic: period
- These are the “parameters” of your nonparametric regression function (still better than a polynomial)
- For bonus points, you can use MCMC to marginalize over them!

GP Packages in Python

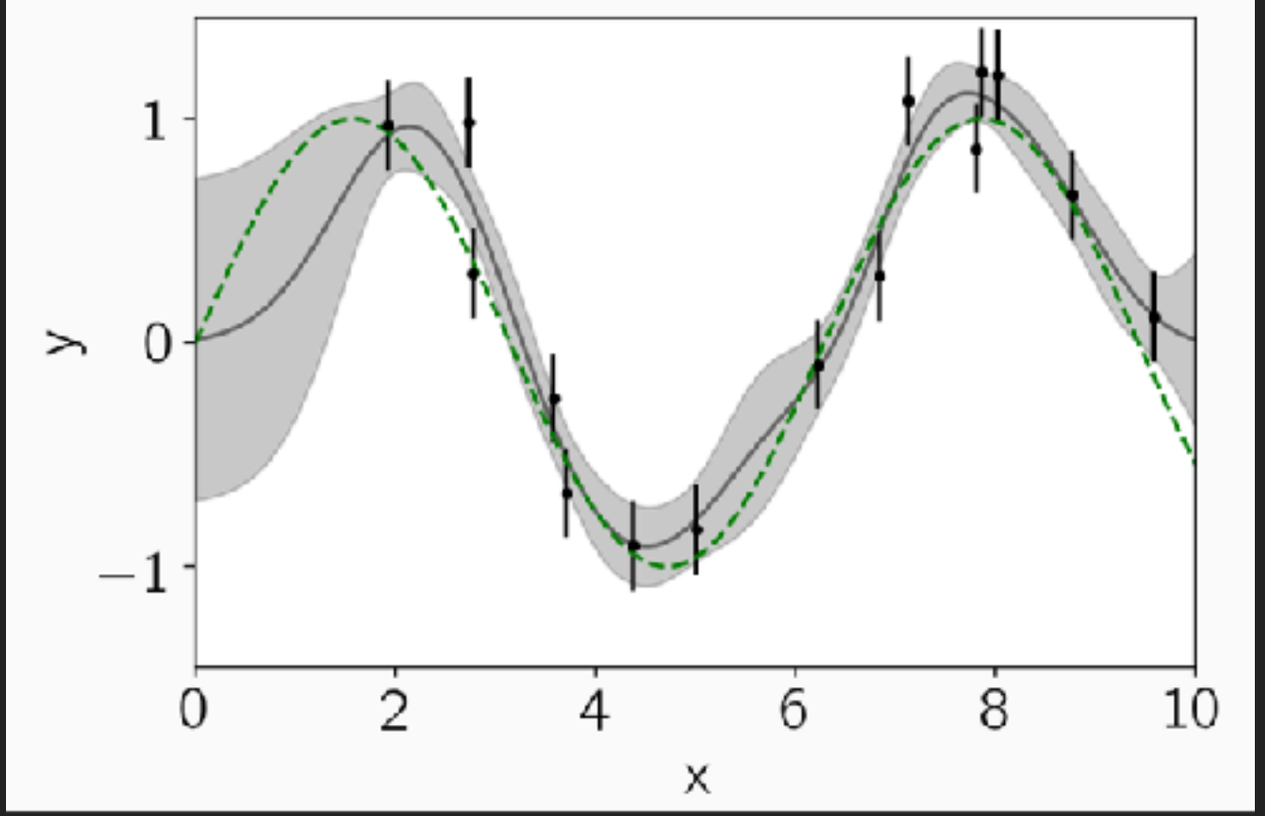
- George (george.readthedocs.io)
- scikit-learn (scikit-learn.org/stable/modules/gaussian_process.html)

```
from george import kernels

kernel = np.var(y) * kernels.ExpSquaredKernel(0.5)
gp = george.GP(kernel)
gp.compute(x, yerr)

x_pred = np.linspace(0, 10, 500)
pred, pred_var = gp.predict(y, x_pred, return_var=True)

pl.fill_between(x_pred, pred - np.sqrt(pred_var), pred + np.sqrt(pred_var),
                color="k", alpha=0.2)
pl.plot(x_pred, pred, "k", lw=1.5, alpha=0.5)
pl.errorbar(x, y, yerr=yerr, fmt=".k", capsize=0)
pl.plot(x_pred, np.sin(x_pred), "--g")
pl.xlim(0, 10)
pl.ylim(-1.45, 1.45)
pl.xlabel("x")
pl.ylabel("y");
```



Exercise: GP Regression of a Light Curve

Pretend we have a sparsely sampled supernova light curve (brightness measurements over time).
We want a smooth interpolation of the light curve to extract some measurements (peak brightness, decline rate, etc.).

1. Choose a kernel. Remember to normalize by the total variance!
2. Initialize the Gaussian process.
3. “Compute” the covariance matrix of the Gaussian process.
4. “Predict” the Gaussian process and its uncertainty over a regularly sampled range of times, given the observed data.