Lecture 19

Model Checking and glms

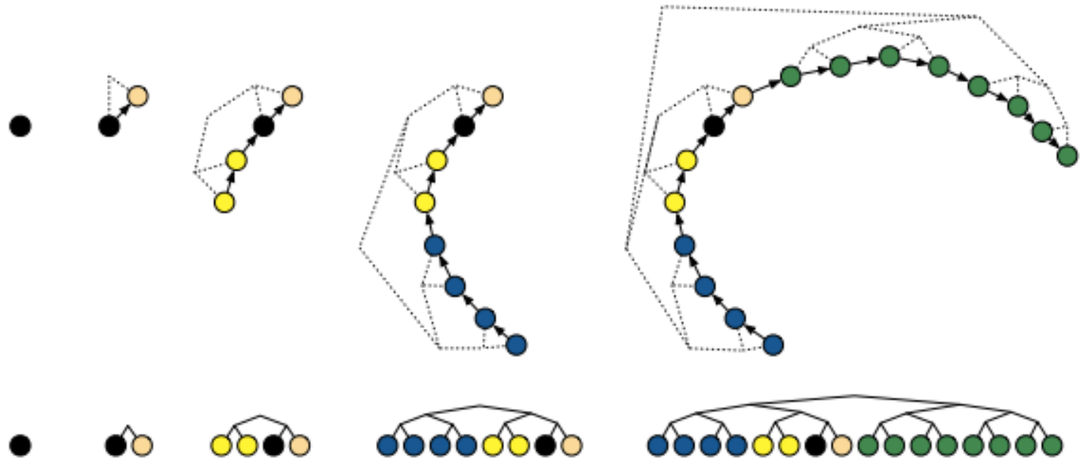
Previously

- HMC
- Hierarchical modelling, divergences
- step sizes, tuning L, NUTS
- intro to glms
- model Checking

Today and Thursday

- glms, contd
- model checking, contd
- oceanic tools example and centering
- model comparison
- oceanic tools and other models model comparison
- Theory and practice of NUTS

NUTS in a nutshell



- termination criterion destroys detailed balance, must rebuild
- sample from trajectory not just endpoint
- sample backwards and forwards in time until u-turn
- choose a sample with boltzmann weights over the trajectory usining multinomial or slice sampling

glms: MAXENT and LINK

- MAXENT: use all the information we have about the constraints on an outcome variable to choose a likelihood, typically in the exponential family, that is a maxent distribution.
- LINK: $f(p_i) = \alpha + \beta x_i$ where p_i is the parameter at the ith data point.
- common links we use are the *logit* link and the *log* link.

	days	monastery	y
0	1	0	0
1	1	0	1
2	1	0	1
3	1	0	2
4	1	0	0
5	1	0	1
6	1	0	2
7	1	0	1
8	1	0	1
9	1	0	0
10	1	0	4
11	1	0	1
12	1	0	4
13	1	0	3
14	1	0	1
15	1	0	0
16	1	0	2
17	1	0	1
18	1	0	1
19	1	0	1
20	1	0	1
21	1	0	1
22	1	0	1
23	1	0	1
24	1	0	0
25	1	0	1
26	1	0	1
27	1	0	1
28	1	0	2
29	1	0	2
30	7	1	6
31	7	1	2
32	7	1	7
33	7	1	3

$$y_i \sim Poisson(\lambda_i)$$

$$log(\lambda_i) = log(\frac{\mu_i}{\tau_i}) = \alpha + \beta x_i$$

λ_i is rate, μ_i is counts, τ_i is exposure.

μ_i or λ_i constrained to be positive.

```
import theano.tensor as t
with pm.Model() as model1:
    alpha=pm.Normal("alpha", 0,100)
    beta=pm.Normal("beta", 0,1)
    logmu = t.log(df.days)+alpha+beta*df.monastery
    y = pm.Poisson("obsv", mu=t.exp(logmu), observed=df.y)
    lambda0 = pm.Deterministic("lambda0", t.exp(alpha))
    lambda1 = pm.Deterministic("lambda1", t.exp(alpha + beta))
```

lambda0:

Mean	SD	MC Error	95% HPD interval
1.243	0.199	0.004	[0.869, 1.635]

Posterior quantiles:

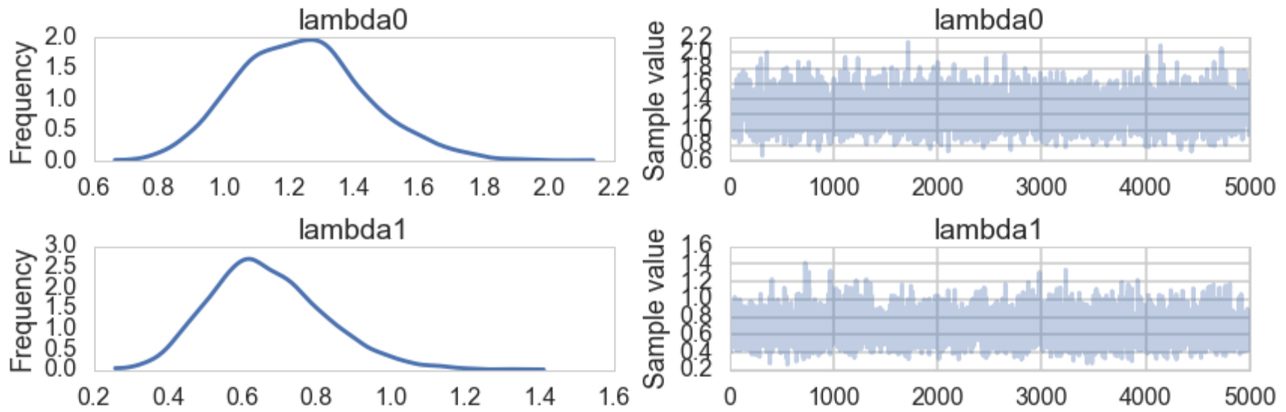
2.5	25	50	75	97.5
0.889	1.100	1.234	1.365	1.671

lambda1:

Mean	SD	MC Error	95% HPD interval
0.669	0.155	0.003	[0.394, 0.988]

Posterior quantiles:

2.5	25	50	75	97.5
0.407	0.561	0.655	0.765	1.008



Zero Inflated Poisson Mixture model

- (A) Monks take a break on some days, drink, produce no manuscripts
- (B) looks the same like other unproductive days
- (B) some days are productive and produce manuscripts
- a mixture of (A) and (B)

Likelihood

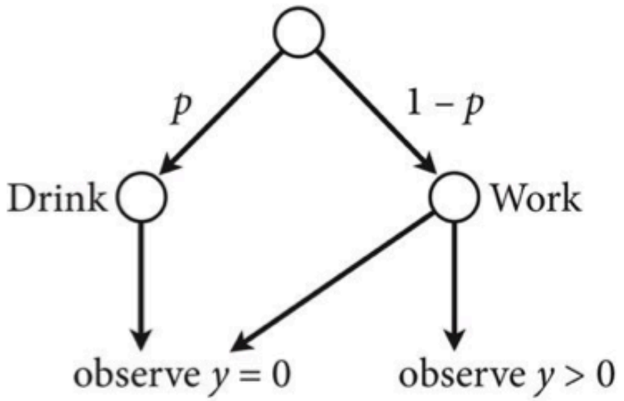
(A) : p

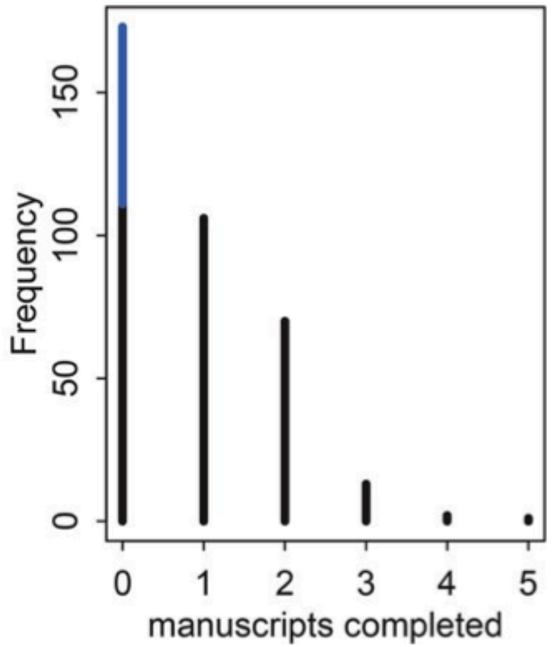
(B) : $(1-p)e^{-\lambda} + (1-p)\frac{\lambda^y e^{-\lambda}}{y!}$

Can also split this as

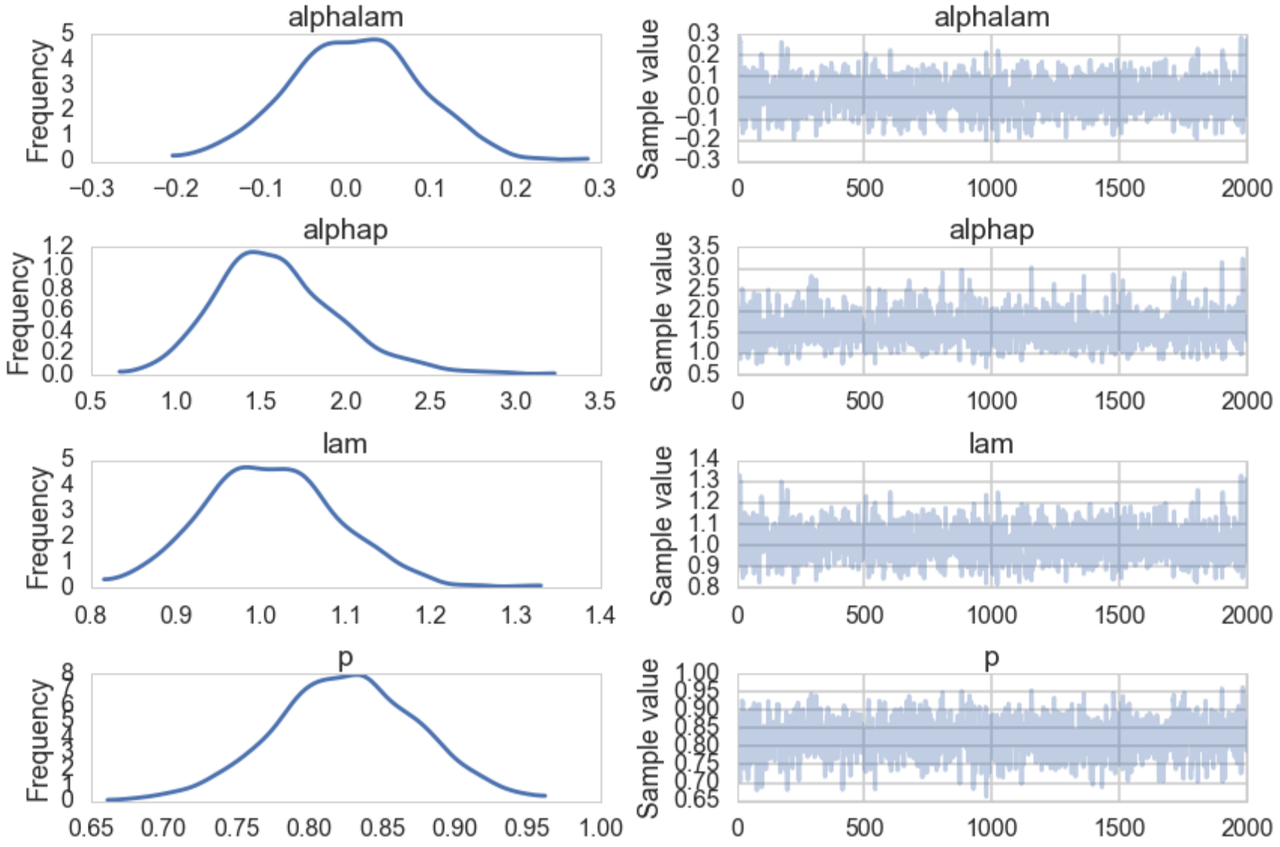
$$\mathcal{L}(y=0) = p + (1-p)e^{-\lambda}$$

$$\mathcal{L}(y \neq 0) = (1-p)\frac{\lambda^y e^{-\lambda}}{y!}$$





Fit the model




```
with pm.Model() as model2:
    alphalam=pm.Normal("alphalam", 0,10)
    alphap=pm.Normal("alphap", 0,1)
    #regression models with intercept only
    logmu = alphalam
    logitp = alphap
    y = pm.ZeroInflatedPoisson("obsv", theta=t.exp(logmu),
        psi=tinvlogit(logitp), observed=y)
    lam = pm.Deterministic("lam", t.exp(logmu))
    p = pm.Deterministic("p", tinvlogit(logitp))
```

notice one level of indirection to introduce intercepts

```
with model2:
    trace2=pm.sample(2000)
```

```
8%|▊         | 16735/200000 [00:01<00:16, 11183.71it/s]| 1103/200000 [00:00<00:18, 11026.25it/s]
100%|██████████| 2000/2000 [00:01<00:00, 1256.31it/s]
```

Posterior Predictive Checking

Speed of light experiment

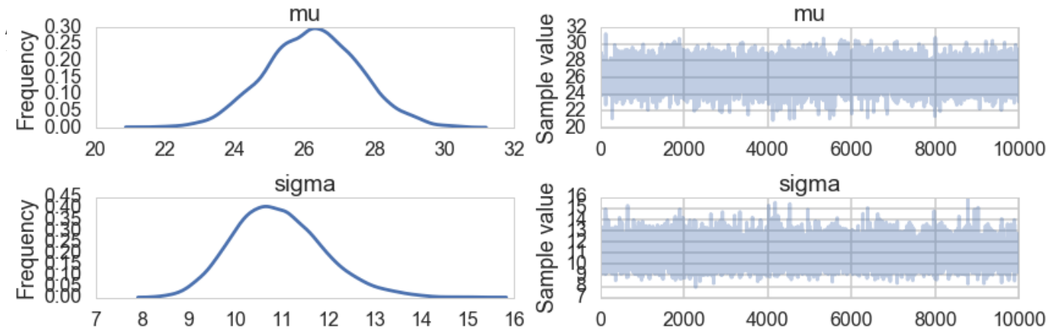
- Simon Newcomb, 1882, times required for light to travel 7442 metres, recorded as deviations from 24,800 nanoseconds

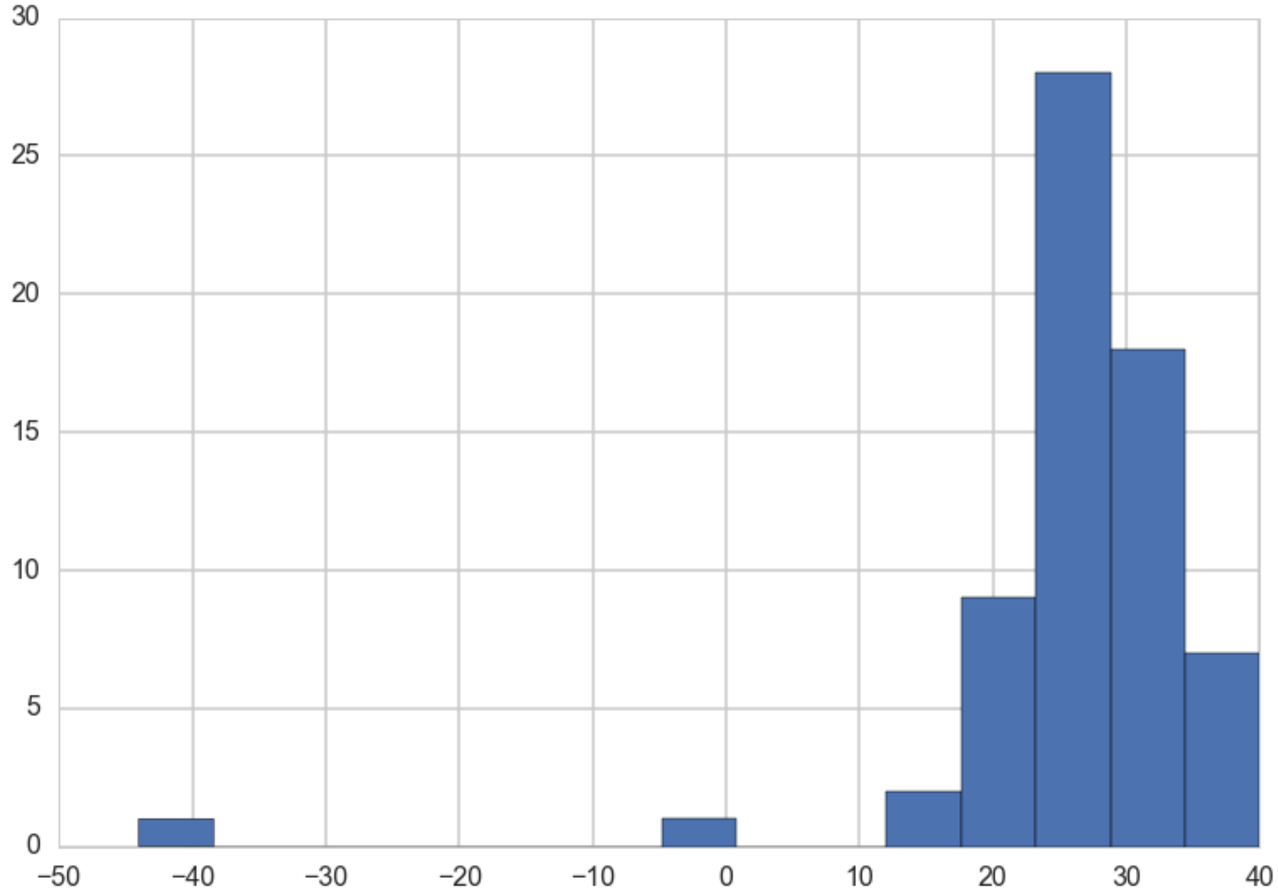
```
light_speed = np.array([28, 26, 33, 24, 34, -44, 27, 16, 40, -2, 29, 22, 24, 21, 25,
                        30, 23, 29, 31, 19, 24, 20, 36, 32, 36, 28, 25, 21, 28, 29,
                        37, 25, 28, 26, 30, 32, 36, 26, 30, 22, 36, 23, 27, 27, 28,
                        27, 31, 27, 26, 33, 26, 32, 32, 24, 39, 28, 24, 25, 32, 25,
                        29, 27, 28, 29, 16, 23])
```

Use Normal model with weakly informative priors to model

```
with pm.Model() as light_model:
    mu = pm.Uniform('mu', lower=-1000,
                    upper=1000.0)
    sigma = pm.Uniform('sigma', lower=0.1, upper=1000.0)
    obsv = pm.Normal('obsv', mu=mu, sd=sigma, observed=light_speed)
```

```
with light_model:
    trace = pm.sample(10000)
```





Some big outliers in data

Multiple replications of the posterior predictive

$$p(\{y^*\}) = \int p(\{y^*\}|\theta)p(\theta|\mathcal{D})d\theta, \text{ observed data: } \mathcal{D} = \{y\}$$

Replicated Data: $\{y_r\}$: data seen tomorrow if experiment replicated with same model and value of θ producing todays data $\{y\}$.

$\{y_r\}$ comes from posterior predictive, and if there are covariates $\{x^*\}$, then $\{y_r\}$ is calculated at those covariates only (`sample_ppc`).

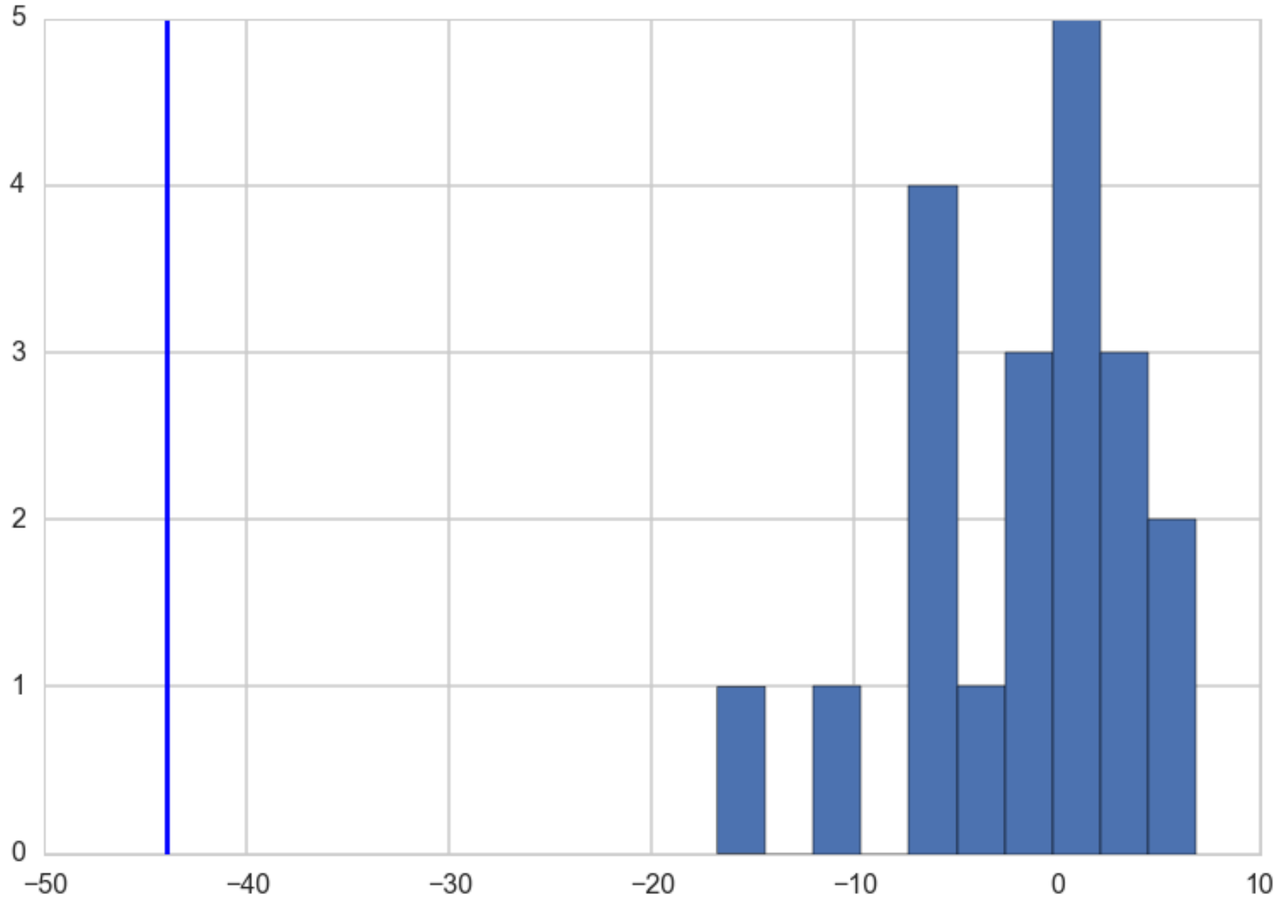
Departure from usual predictive sampling

Sample an entire $\{y_r\}$ at each θ from trace.

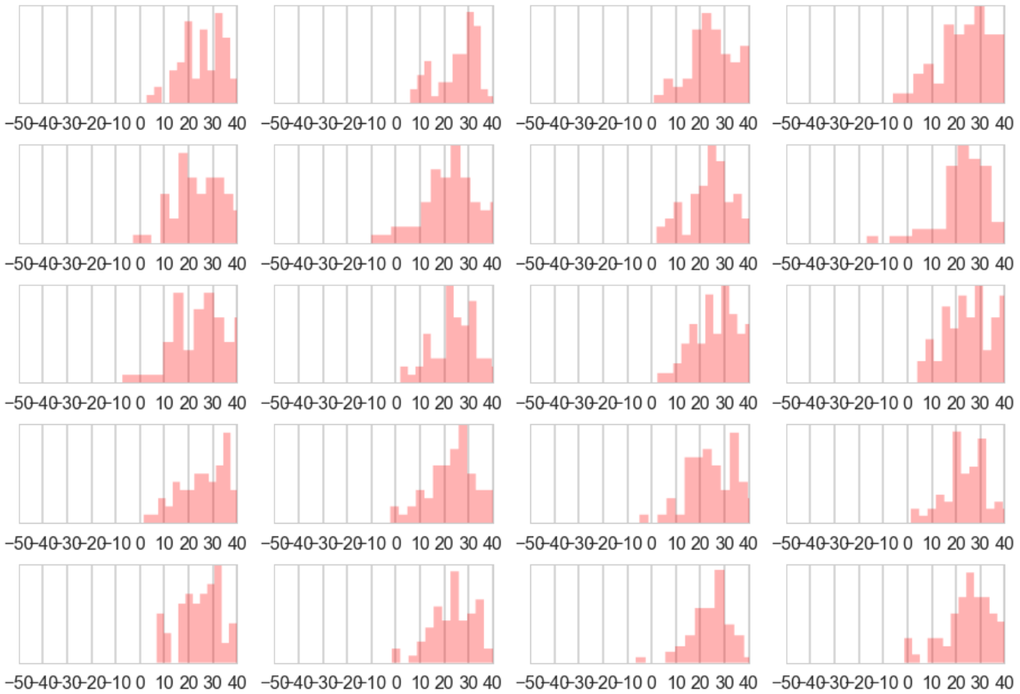
This allows to compute distributions from the posterior predictive replications.

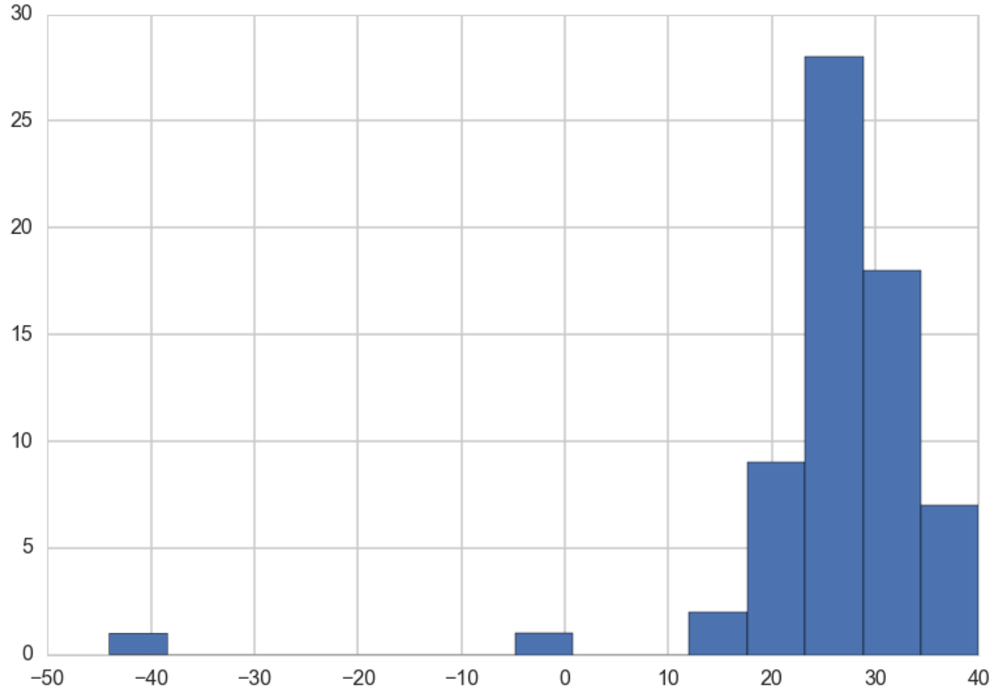
For example the minimum value of speed of light in 20 predictive replications.

An informal test statistic.



Visual Checking





Do these even look similar??

Discrepancy

Gelman: *A test quantity, or discrepancy measure, $T(\{y\}, \theta)$, is a scalar summary of parameters and data that is used as a standard when comparing data to predictive simulations.*

The classical p-value for the test statistic $T(\{y\})$ is given by

$$p_C = P(T(\{y_r\}) \geq T(\{y\})|\theta)$$

where probability is over distrib of $\{y_r\}$ with θ fixed (bootstrap).

Bayesian p-values

$$p_B = Pr(T(\{y_r\}, \theta) \geq T(\{y\}, \theta)|\{y\}),$$

probability over the posterior and posterior predictive (that is, the joint distribution, $p(\theta, \{y_r\}|\{y\})$).

$$p_B = \int d\theta \, d\{y_r\} \, I(T(\{y_r\}, \theta) \geq T(\{y\}, \theta)) \, p(\{y_r\}|\theta) p(\theta|\{y\})$$

using $p(\{y_r\}|\theta, \{y\}) = p(\{y_r\}|\theta)$.

Appropriate usage

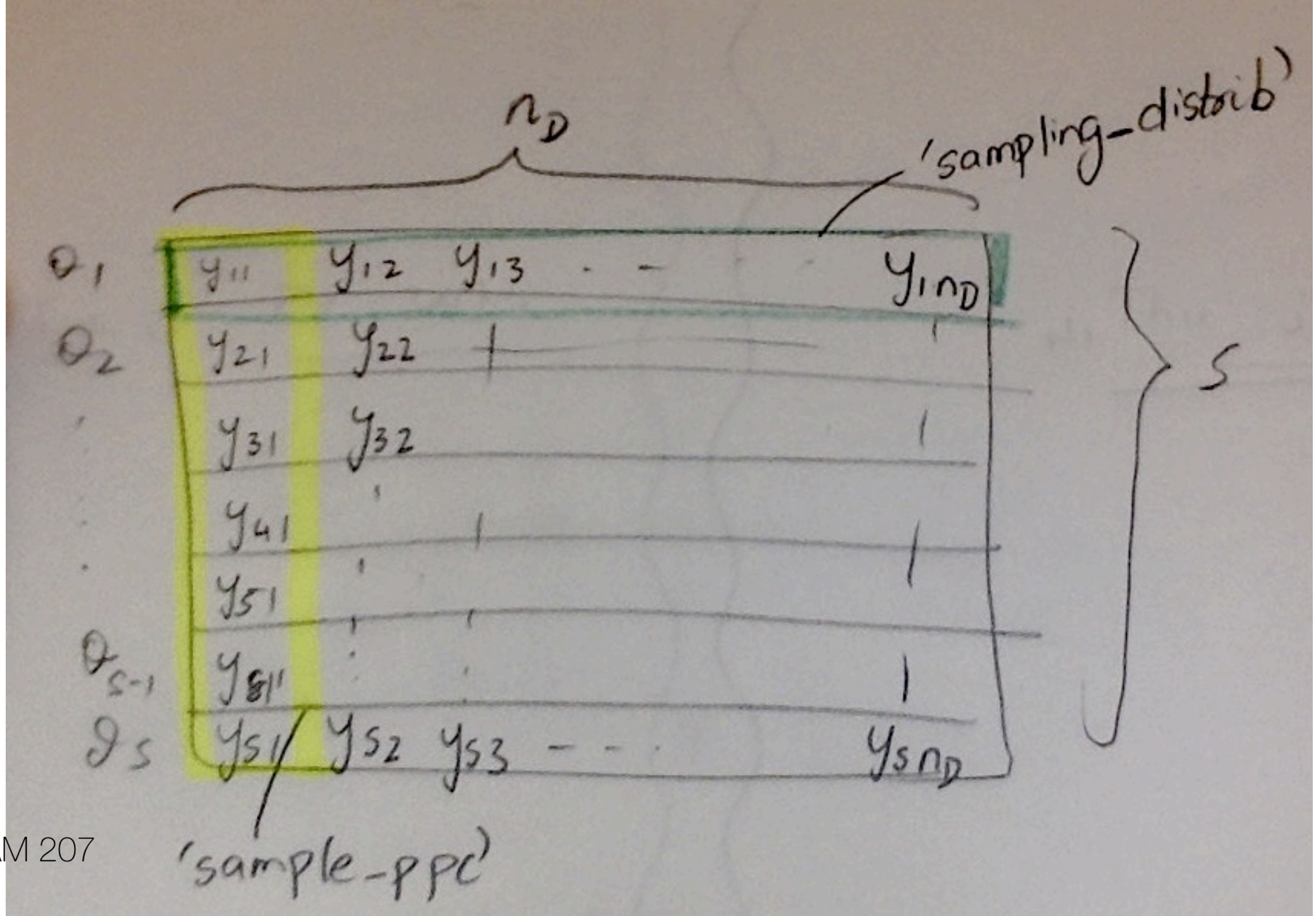
Gelman: *Finding an extreme p-value and thus 'rejecting' a model is never the end of an analysis; the departures of the test quantity in question from its posterior predictive distribution will often suggest improvements of the model or places to check the data, as in the speed of light example. Moreover, even when the current model seems appropriate for drawing inferences (in that no unusual deviations between the model and the data are found), the next scientific step will often be a more rigorous experiment incorporating additional factors, thereby providing better data.*

Another way to sample

```
indices=np.random.choice(range(len(trace)), size=200, replace=True)
mus = trace['mu'][indices]
sigmas = trace['sigma'][indices]
ppc2=np.empty((66,200))
for i in range(66):
    ppc2[i,:] = np.random.normal(loc=mus, scale=sigmas)
```

For each data point, sample using the likelihood(sampling distribution) from S samples of the posterior. Gives an S sized posterior predictive at each "data point".

You can then slice the other way to get a dataset sized posterior-predictive


n_D
'sampling-distrib'
θ_1
θ_2
θ_{S-1}
θ_S
y_{11} y_{12} y_{13} y_{1n_D}
y_{21} y_{22}
y_{31} y_{32}
y_{41}
y_{51}
y_{81}
y_{S1} y_{S2} y_{S3} y_{Sn_D}
S
'sample-ppc'

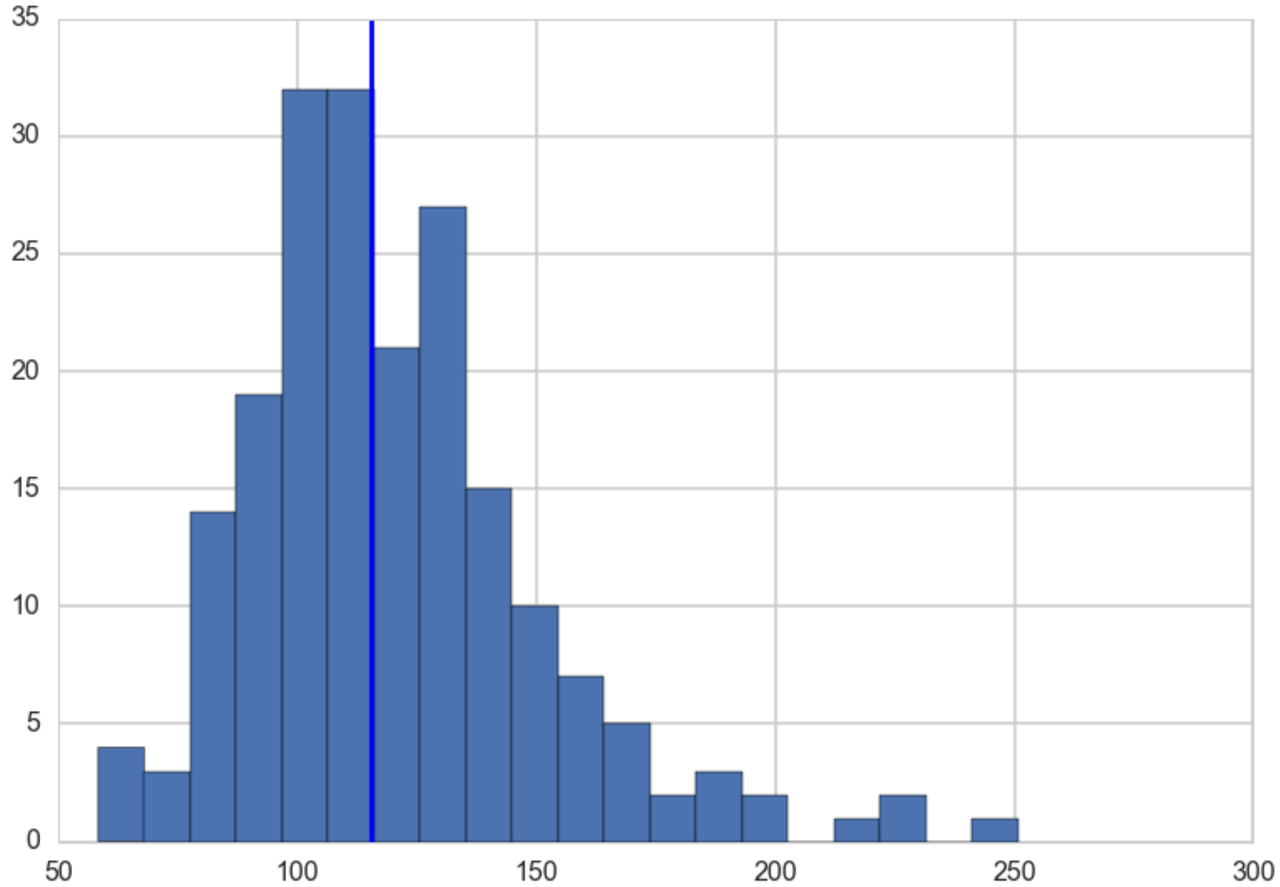
p-value of sampling variance

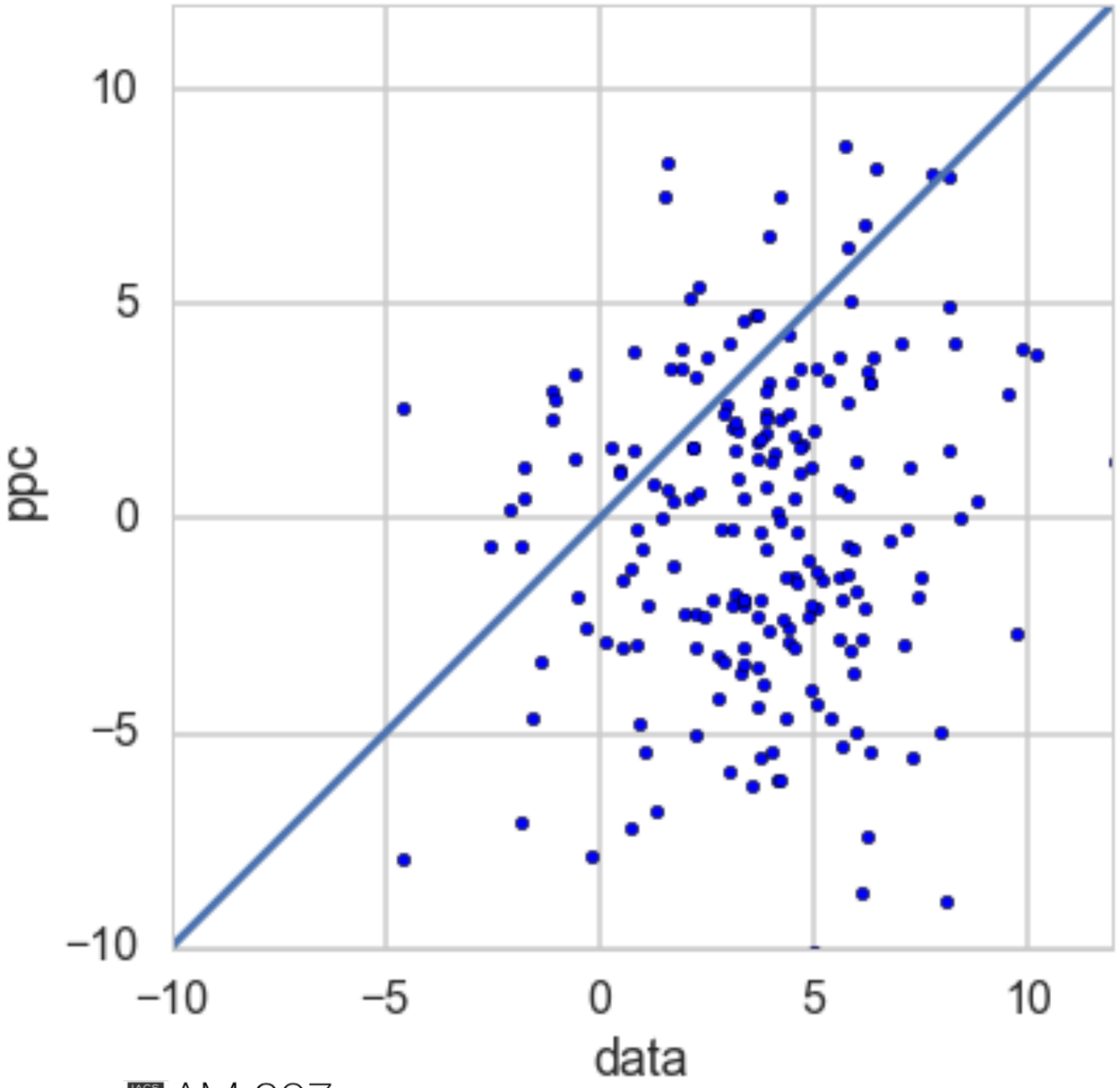
```
ppvars2=np.var(ppc2, ddof=1, axis=0)
plt.hist(ppvars2, bins=20);
plt.axvline(np.var(light_speed, ddof=1));
np.mean(ppvars2>=np.var(light_speed, ddof=1))
```

```
0.4899999999999999
```

Gelman:

The sample variance…is a sufficient statistic of the model and thus the posterior distribution will automatically be centered near the observed value.





p-value of a measure of symmetry

Is the model adequate but for extreme tails?

$$T(\{y\},\theta) = |y(61) - \theta| - |y(6) - \theta|.$$

Reflects the middle 80% of the mass.

```
tee_ppc=[]
tee_data=[]
data_sort=np.sort(light_speed)
for i in range(200):
    sortarray = np.sort(ppc2[:,i])
    tee_data.append(np.abs(data_sort[60] - mus[i]) - np.abs(data_sort[5] - mus[i]))
    tee_ppc.append(np.abs(sortarray[60] - mus[i]) - np.abs(sortarray[5] - mus[i]))
np.mean(np.array(tee_ppc) >= np.array(tee_data))
0.17999999999999999
```

Any asymmetry can be simply explained by sampling variation.

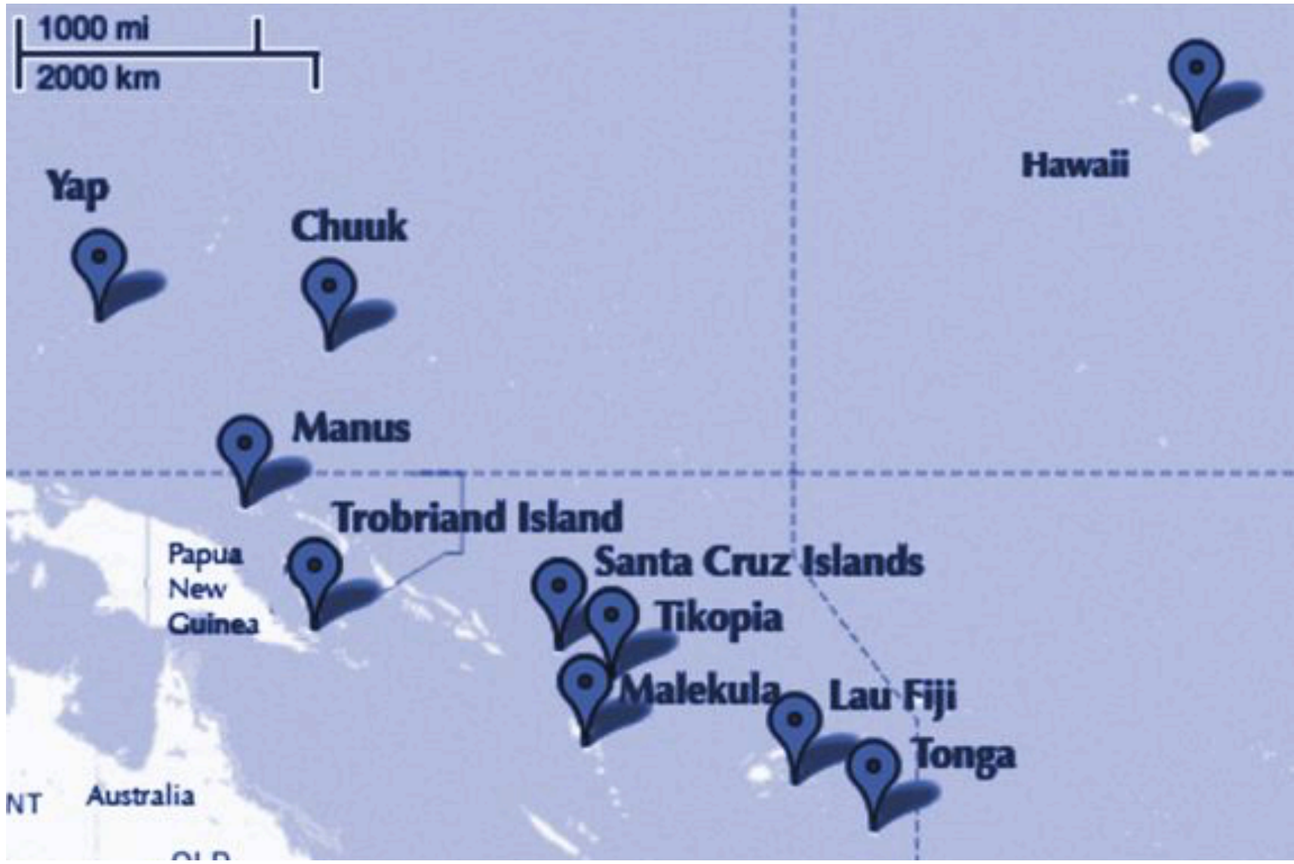
Conclusion from Checking

Model is adequate for some purposes but not others.

Back to Poisson GLMs

From Mcelreath:

The island societies of Oceania provide a natural experiment in technological evolution. Different historical island populations possessed tool kits of different size. These kits include fish hooks, axes, boats, hand plows, and many other types of tools. A number of theories predict that larger populations will both develop and sustain more complex tool kits. So the natural variation in population size induced by natural variation in island size in Oceania provides a natural experiment to test these ideas. It's also suggested that contact rates among populations effectively increase population size, as it's relevant to technological evolution. So variation in contact rates among Oceanic societies is also relevant. (McElreath 313)



Model M1

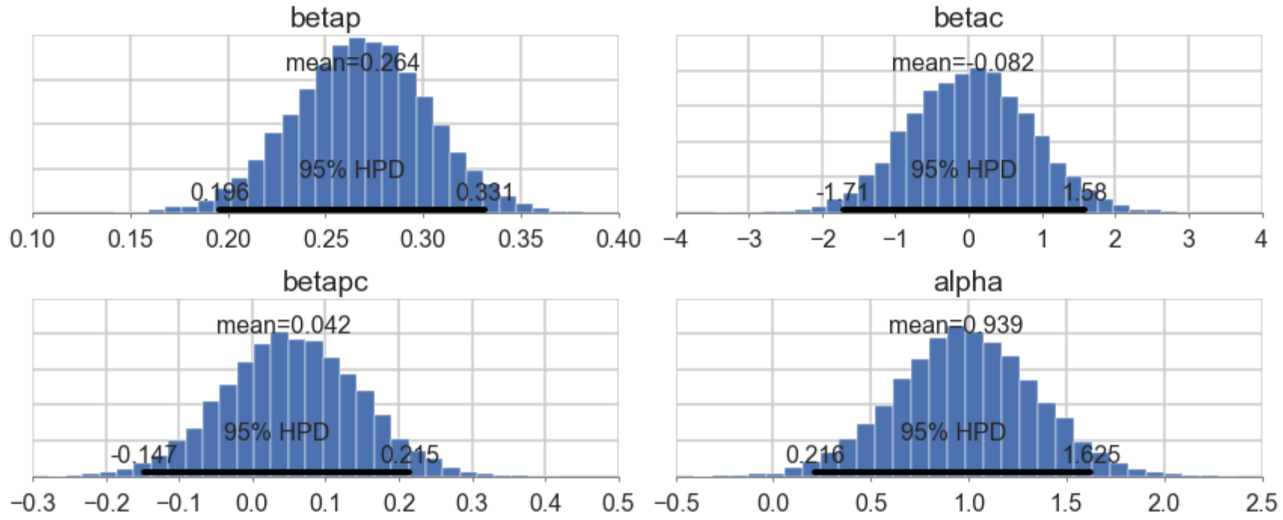
	culture	population	contact	total_tools	mean_TU	logpop	clevel
0	Malekula	1100	low	13	3.2	7.003065	0
1	Tikopia	1500	low	22	4.7	7.313220	0
2	Santa Cruz	3600	low	24	4.0	8.188689	0
3	Yap	4791	high	43	5.0	8.474494	1
4	Lau Fiji	7400	high	33	5.0	8.909235	1
5	Trobriand	8000	high	19	4.0	8.987197	1
6	Chuuk	9200	high	40	3.8	9.126959	1
7	Manus	13000	low	28	6.6	9.472705	0
8	Tonga	17500	high	55	5.4	9.769956	1
9	Hawaii	275000	low	71	6.6	12.524526	0

$$
\begin{aligned}
T_i &\sim Poisson(\lambda_i) \\
log(\lambda_i) &= \alpha + \beta_P log(P_i) + \beta_C C_i + \beta_{PC} C_i log(P_i) \\
\alpha &\sim N(0, 100) \\
\beta_P &\sim N(0, 1) \\
\beta_C &\sim N(0, 1) \\
\beta_{PC} &\sim N(0, 1)
\end{aligned}
$$

```
with pm.Model() as m1:
    betap = pm.Normal("betap", 0, 1)
    betac = pm.Normal("betac", 0, 1)
    betapc = pm.Normal("betapc", 0, 1)
    alpha = pm.Normal("alpha", 0, 100)
    loglam = alpha + betap*df.logpop +
        betac*df.clevel + betapc*df.clevel*df.logpop
    y = pm.Poisson("ntools", mu=t.exp(loglam), observed=df.total_tools)
```

```
with m1:
    trace=pm.sample(10000, njobs=2)
Average ELBO = -55.784:
100%|██████████| 200000/200000 [00:15<00:00, 13019.16it/s]   12683.03it/s]
100%|██████████| 10000/10000 [01:59<00:00, 83.80it/s]
```

Posteriors for M1



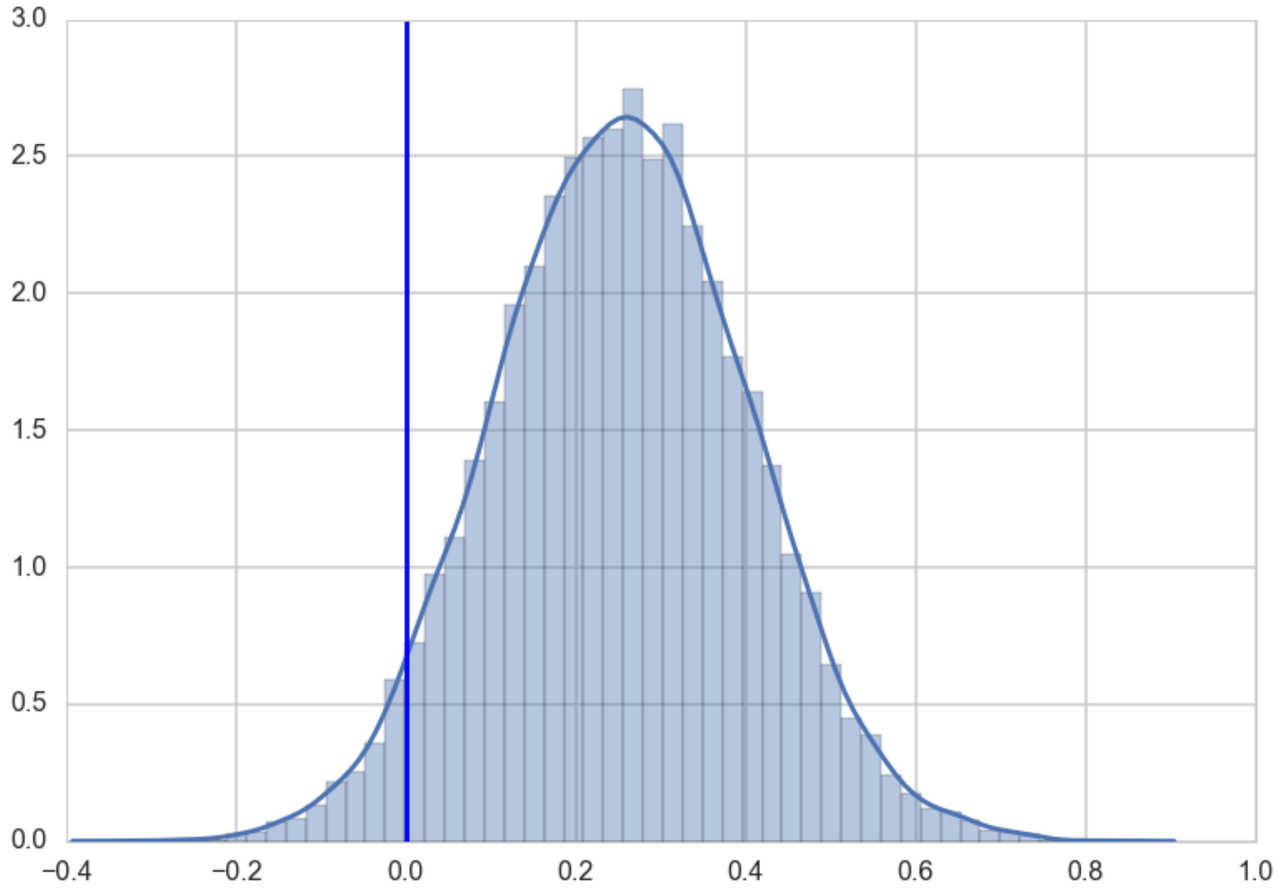
- traces and autocorrelations look good
- The posterior for β_p tightly constrained, and as expected from theory, shows a positive effect.
- The posteriors for β_c and β_{pc} both overlap 0 substantially, and seem comparatively poorly constrained.
- no substantial effect of contact rate, directly or through the interaction?

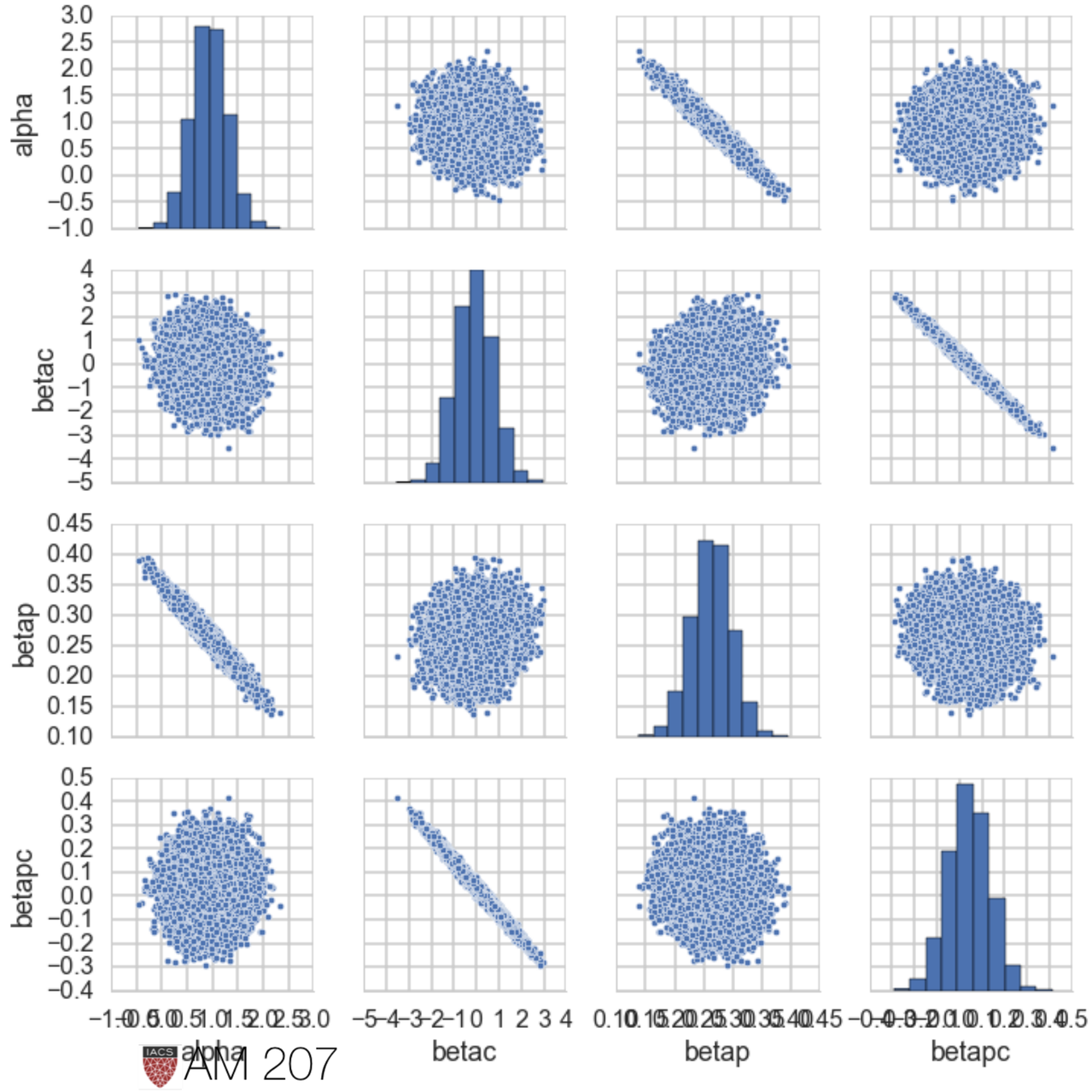
You would be wrong: counterfactual predictions

λ traces for high-contact and low contact, log(population) of 8.

```
lamlow = lambda logpop: trace['alpha']+trace['betap']*logpop
lamhigh = lambda logpop: trace['alpha']+(trace['betap'] +
    trace['betapc'])*logpop + trace['betac']
sns.distplot(lamhigh(8) - lamlow(8));
```

A new kind of model checking.





What happened?

- very strong negative correlations between α and β_p
- very strong negative correlations between β_c and β_{pc}.
- The latter is the cause for the 0-overlaps.
- When β_c is high, β_{pc} must be low, and vice-versa. Look at the joint uncertainty of the correlated variables rather than just marginals

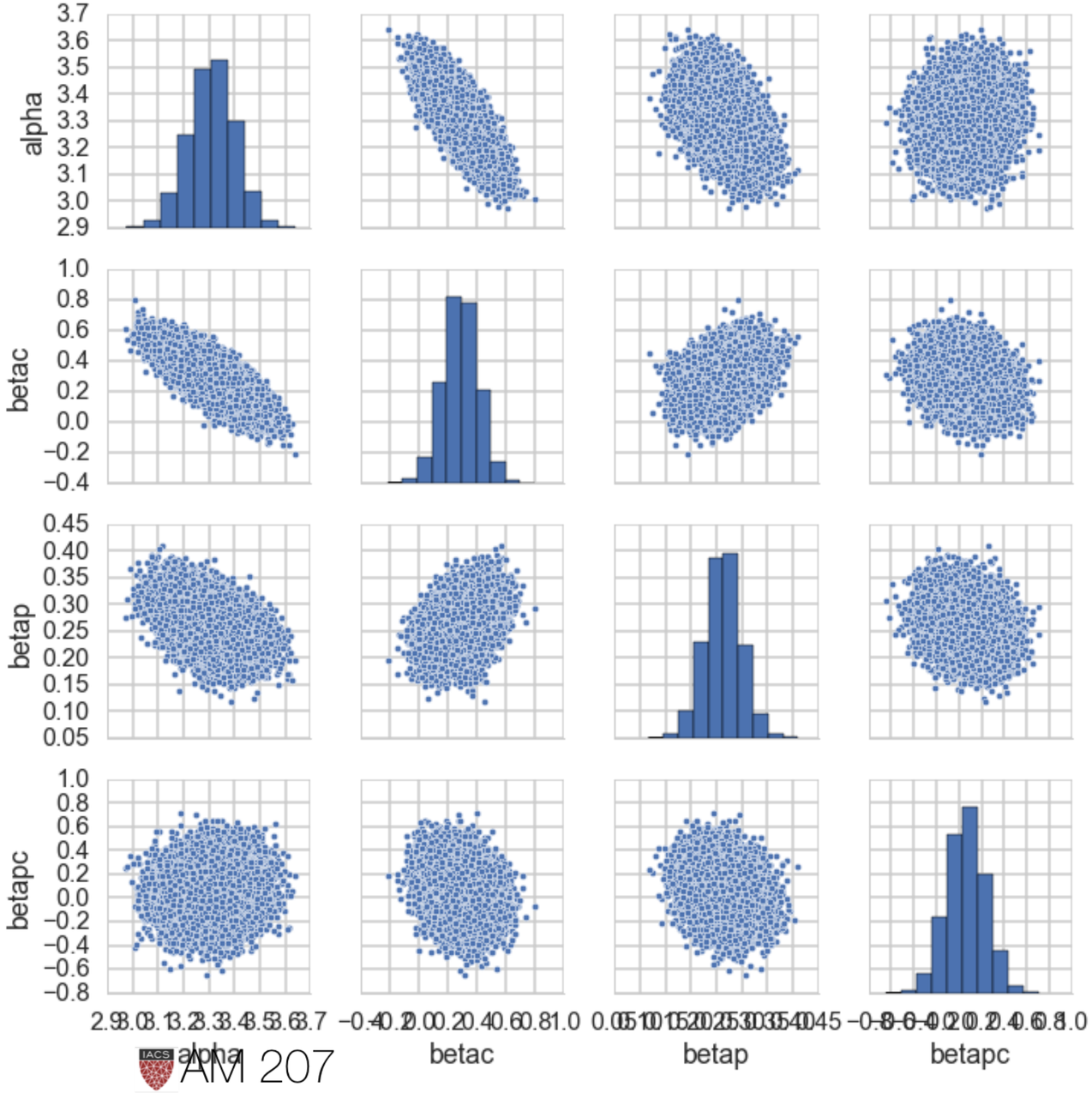
Fix by centering

- you would have seen the problem in n_{eff}:

```
{'alpha': 8110.0, 'betac': 4600.0, 'betap': 8016.0, 'betapc': 4597.0}
```

```
with pm.Model() as m1c:
    betap = pm.Normal("betap", 0, 1)
    betac = pm.Normal("betac", 0, 1)
    betapc = pm.Normal("betapc", 0, 1)
    alpha = pm.Normal("alpha", 0, 100)
    loglam = alpha + betap*df.logpop_c + betac*df.clevel + betapc*df.clevel*df.logpop_c
    y = pm.Poisson("ntools", mu=t.exp(loglam), observed=df.total_tools)
```

```
{'alpha': 7978.0, 'betac': 7898.0, 'betap': 13621.0, 'betapc': 17703.0}
```



- better constrained, less correlated, sampling faster and better
- clear effect of contact, effect of interaction not clear yet
- will use model comparison next time for this!