Lecture 20

Model Comparison and glms

Previously

- glms
- model Checking

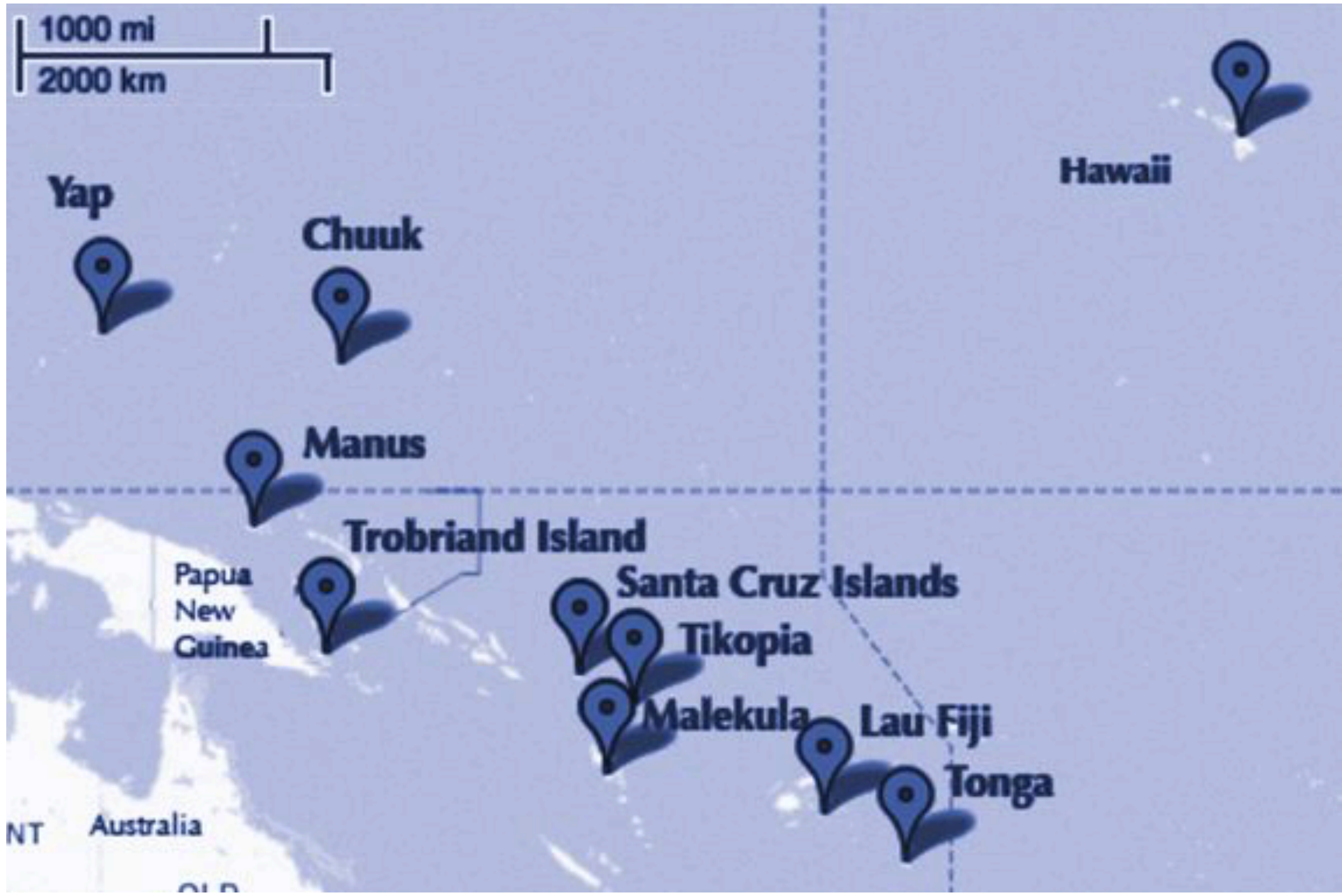
Today

- glms, contd
- oceanic tools example and centering
- model comparison
- oceanic tools and other models model comparison
- poisson over-dispersion and hierarchical modeling

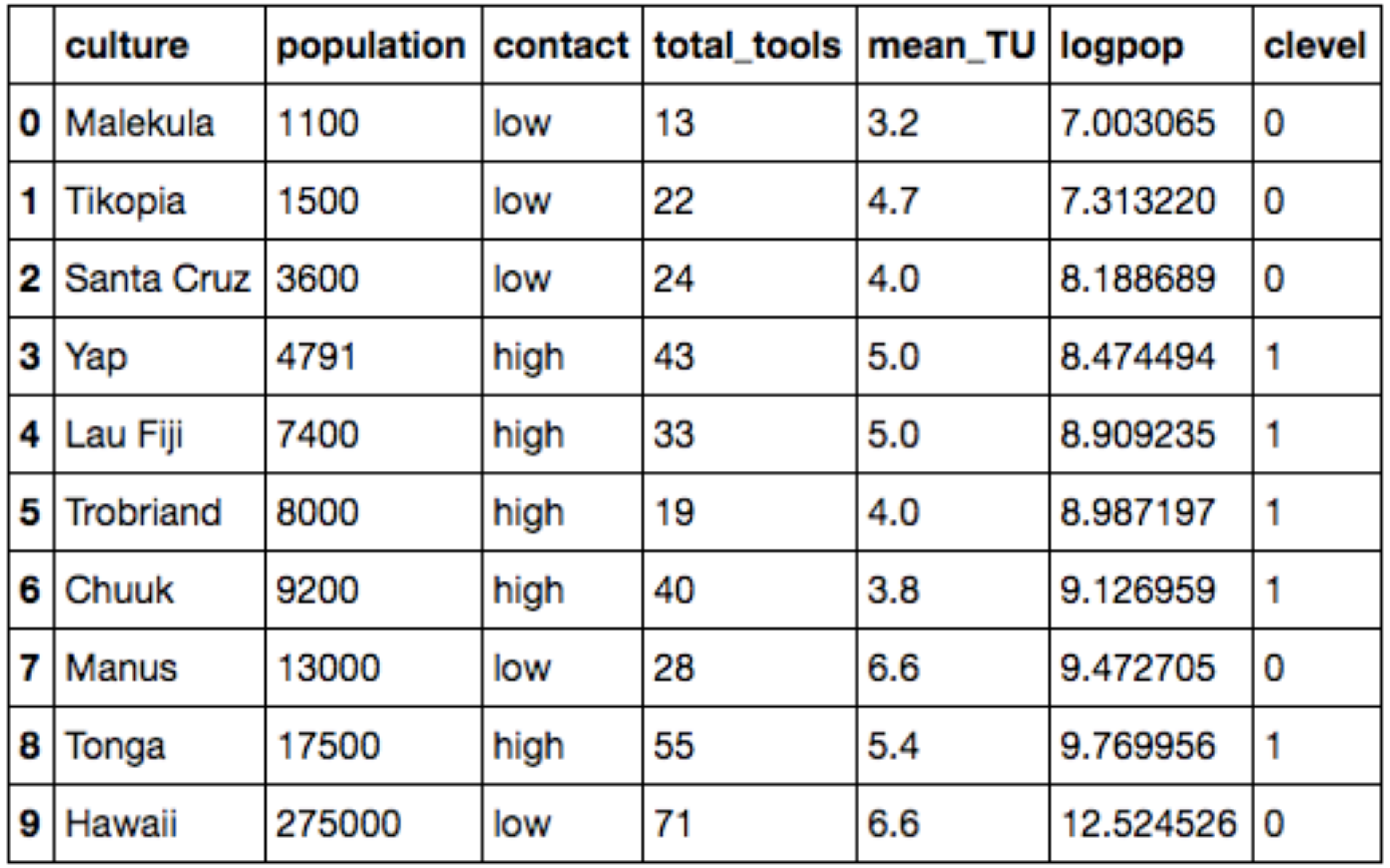
Back to Poisson GLMs

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



Model M1



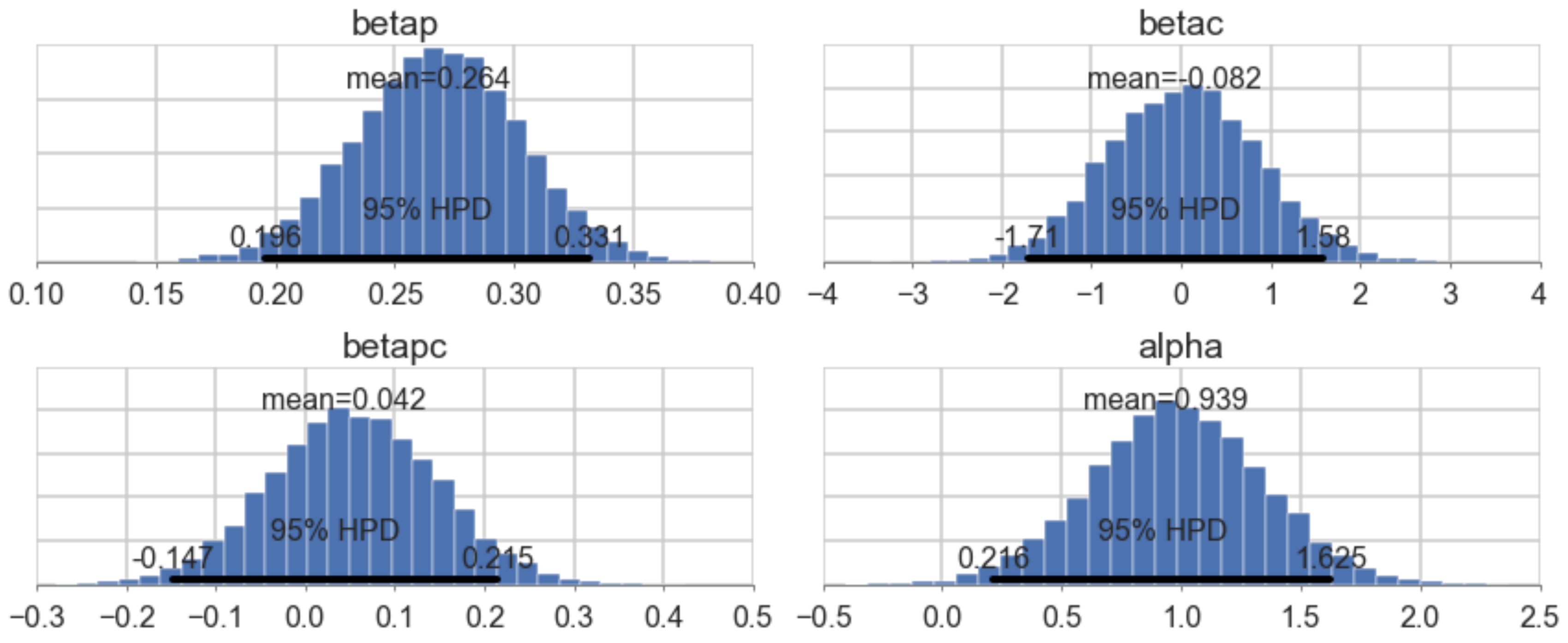
	culture	population	contact	total_tools	mean_TU	logpop	clevel
0	Malekula	1100	low	13	3.2	7.003065	0
1	Tikopia	1500	low	22	4.7	7.313220	0
2	Santa Cruz	3600	low	24	4.0	8.188689	0
3	Yap	4791	high	43	5.0	8.474494	1
4	Lau Fiji	7400	high	33	5.0	8.909235	1
5	Trobriand	8000	high	19	4.0	8.987197	1
6	Chuuk	9200	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



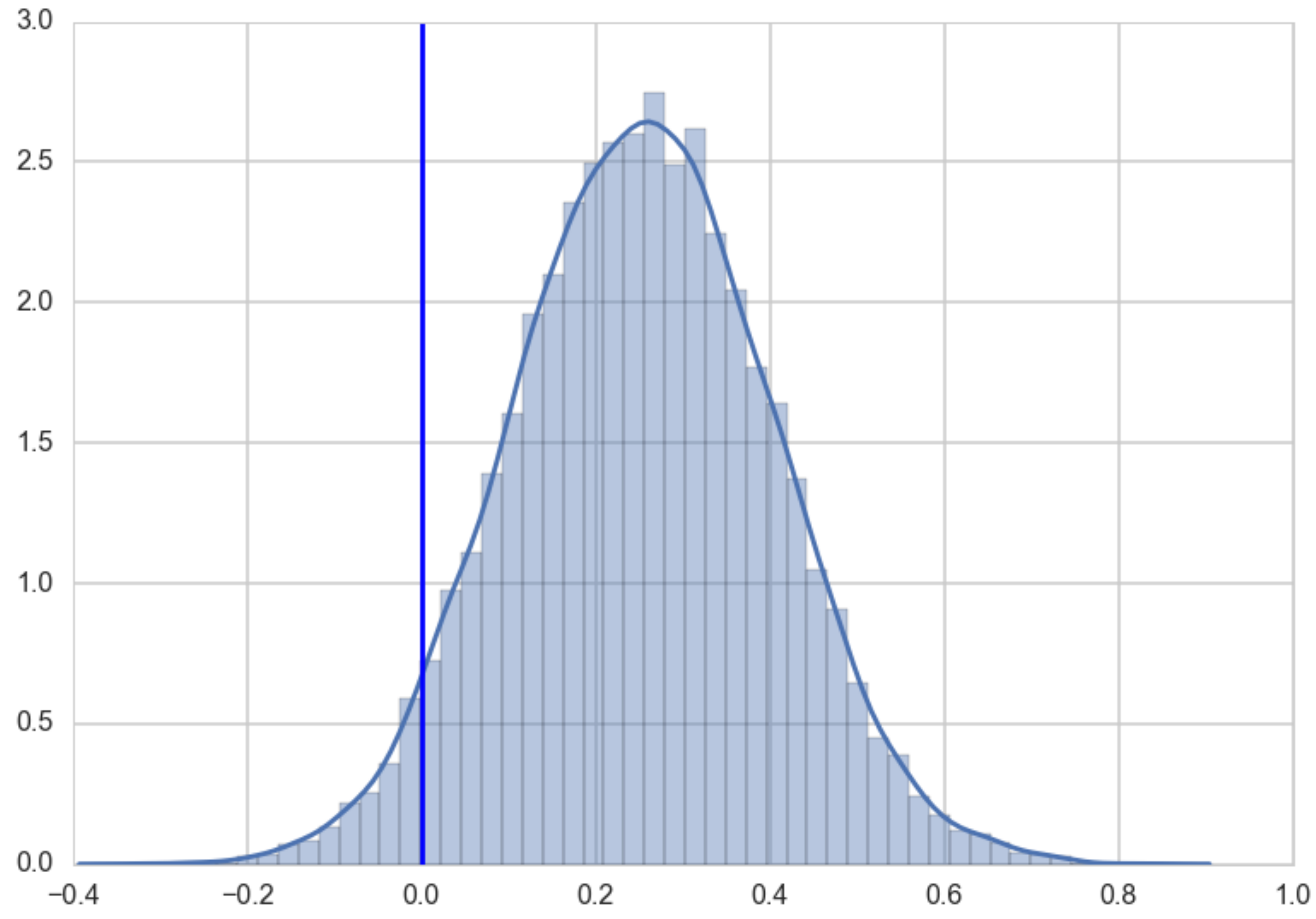
- traces and autocorrelations look good
- The posterior for β_p tightly constrained, and as expected from theory, shows a positive effect.
- The posteriors for β_c and β_{pc} both overlap 0 substantially, and seem comparatively poorly constrained.
- no substantial effect of contact rate, directly or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

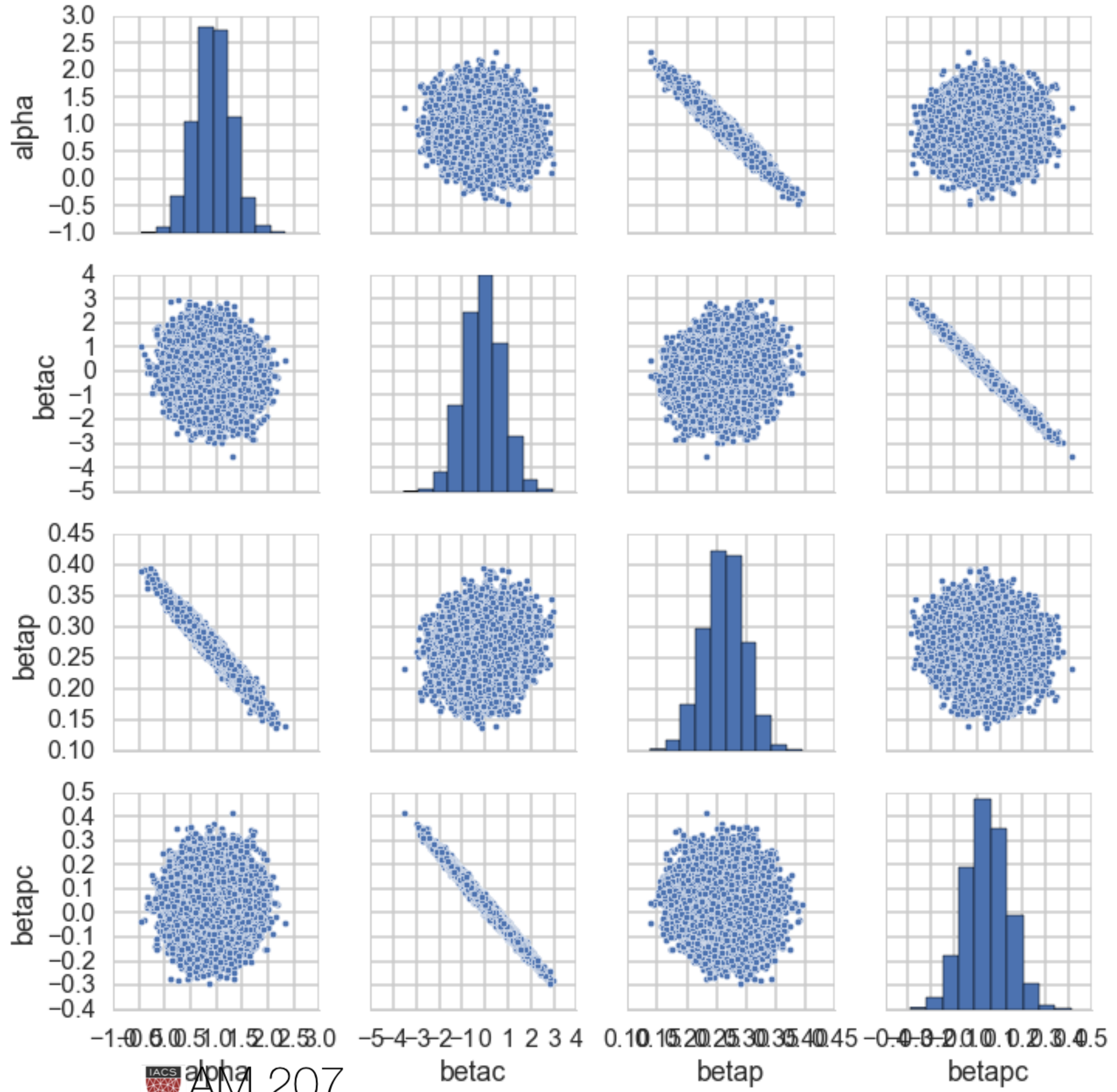
You would be wrong: counterfactual predictions

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

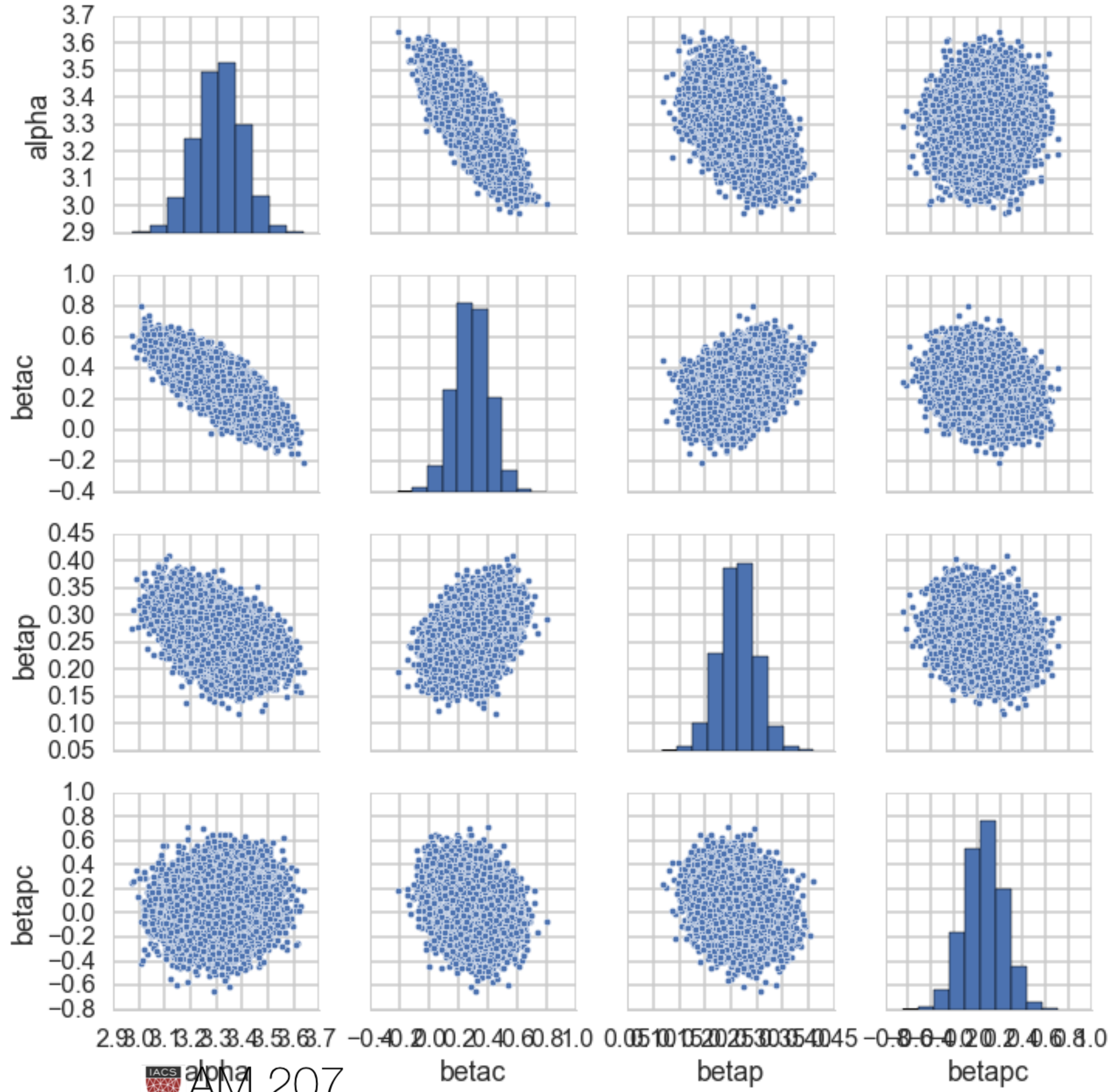
Fix by centering

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

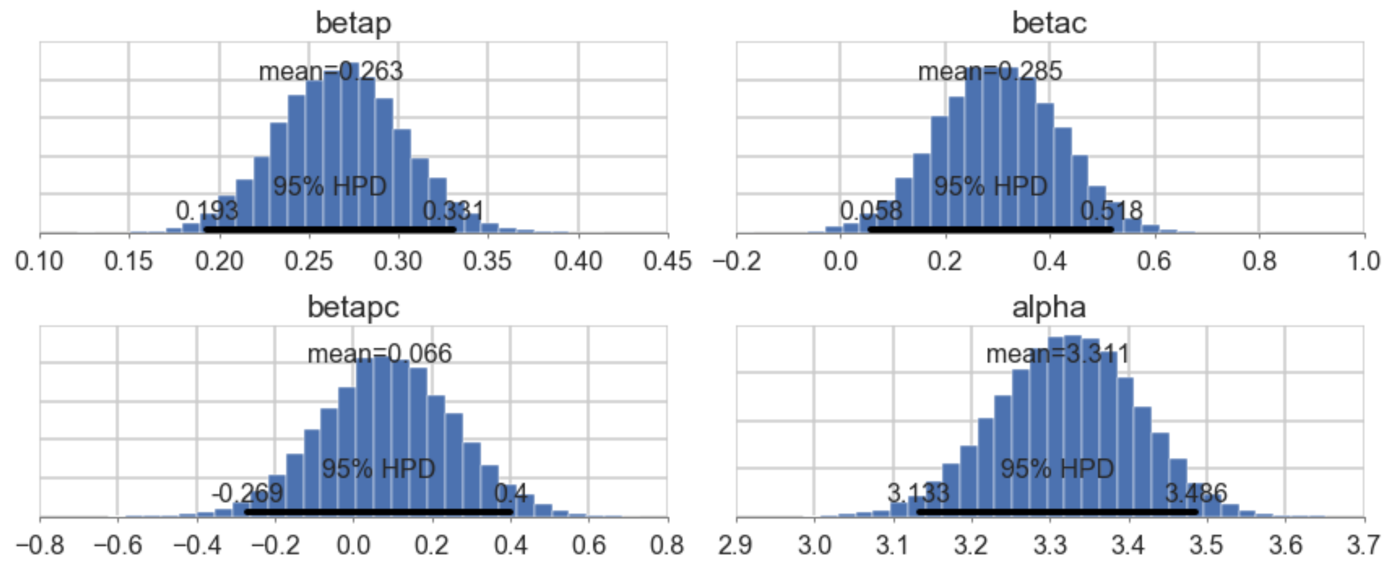
```
{'alpha': 8110.0, 'betac': 4600.0, 'betap': 8016.0, 'betapc': 4597.0}
```

```
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}
```

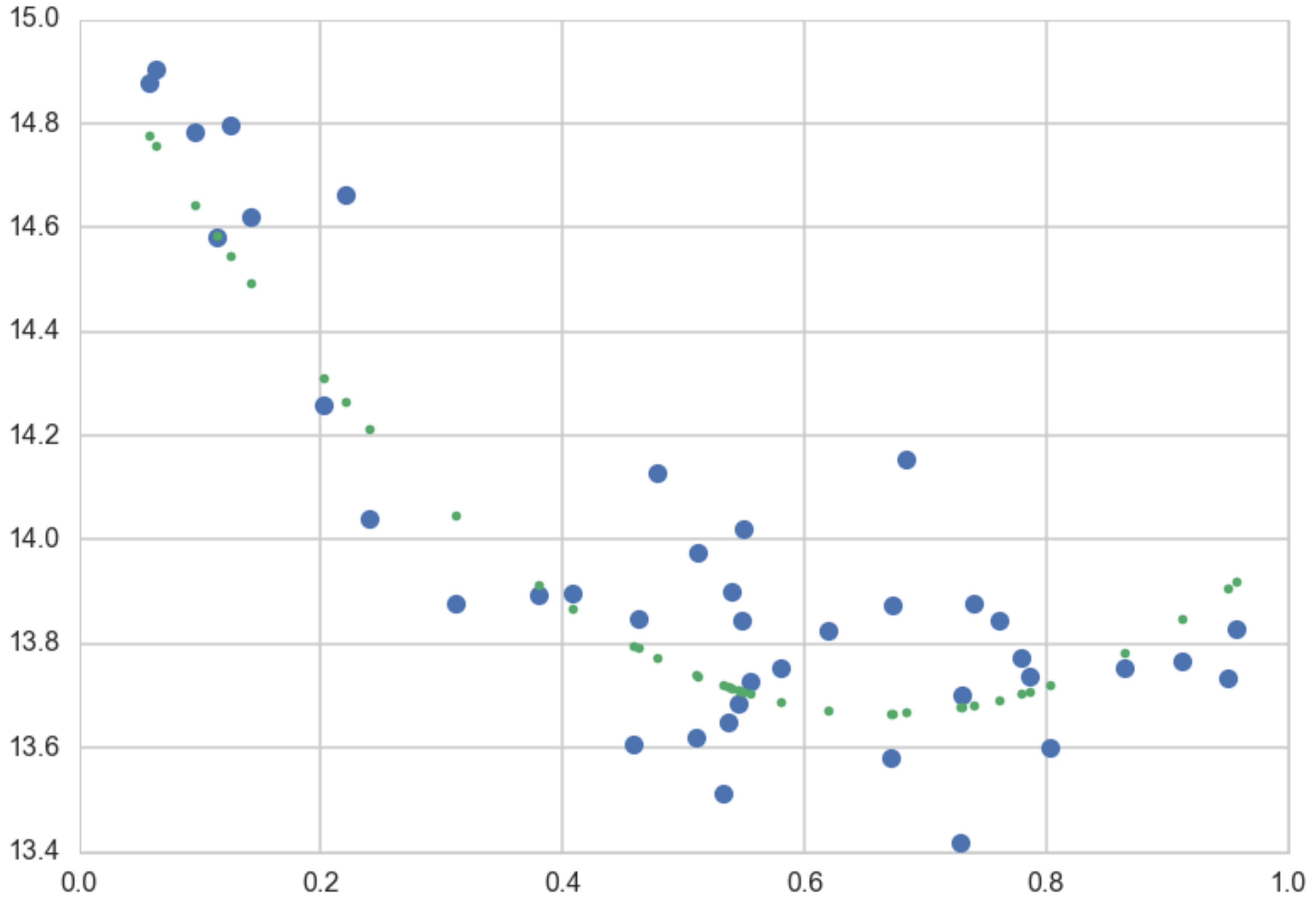


- better constrained, less correlated, sampling faster and better
- clear effect of contact, effect of interaction not clear yet
- will use model comparison next time for this!



Model comparison

Dataset



- 20 data points
- keep train and test, we'll see these later
- center x for speed of sampling
- fit multiple order polynomials
- The `logp` method of a node gives us the conditional (log) probability of the node given its parents.
- Conditional probability of the observed node, which is also the model node, ...

model.logp({paramdict})

gives us likelihood of data $p(y|etc)$

```
ridge=3
sigma=0.2
pf=PolynomialFeatures(msize, include_bias=False).fit_transform(xtrain.reshape(-1,1))
print(pf.shape)
with pm.Model() as m:
    alpha = pm.Normal('alpha', 0, 100)
    beta = pm.Normal('beta', mu=0, sd=ridge, shape=msize)
    mu = alpha + pm.math.dot(pf, beta)
    o = pm.Normal('o', mu, sigma, observed=ytrain)
    trace=pm.sample(5000, init='MAP')
```

```
meanpoint={'alpha': trace['alpha'].mean(), 'beta': trace['beta'].mean(0)}
{'alpha': 13.840500632702547, 'beta': array([-0.99418043,  2.31483997])}
m.logp(meanpoint), mtest.logp(meanpoint)
(array(-1.699085634529018), array(-5.251454815218393))
```

Information criteria

- simulate an ensemble of polynomials
- use information criteria to decide between them
- these come from the deviance

$$D_{KL}(p,q) = E_p[log(p) - log(q)] = E_p[log(p/q)] = \sum_i p_i log(\frac{p_i}{q_i}) \; or \int dP log(\frac{p}{q})$$

Use **law or large numbers** to replace the true distribution by its empirical estimate, then we have:

$$D_{KL}(p,q) = E_p[log(p/q)] = \frac{1}{N}\sum_i (log(p_i) - log(q_i))$$

Thus minimizing the KL-divergence involves maximizing $\sum_i log(q_i)$, justifies the maximum likelihood principle.

$$D_{KL}(p,q) - D_{KL}(p,r) = E_p[log(r) - log(q)] = E_p[log(\frac{r}{q})]$$

Deviance

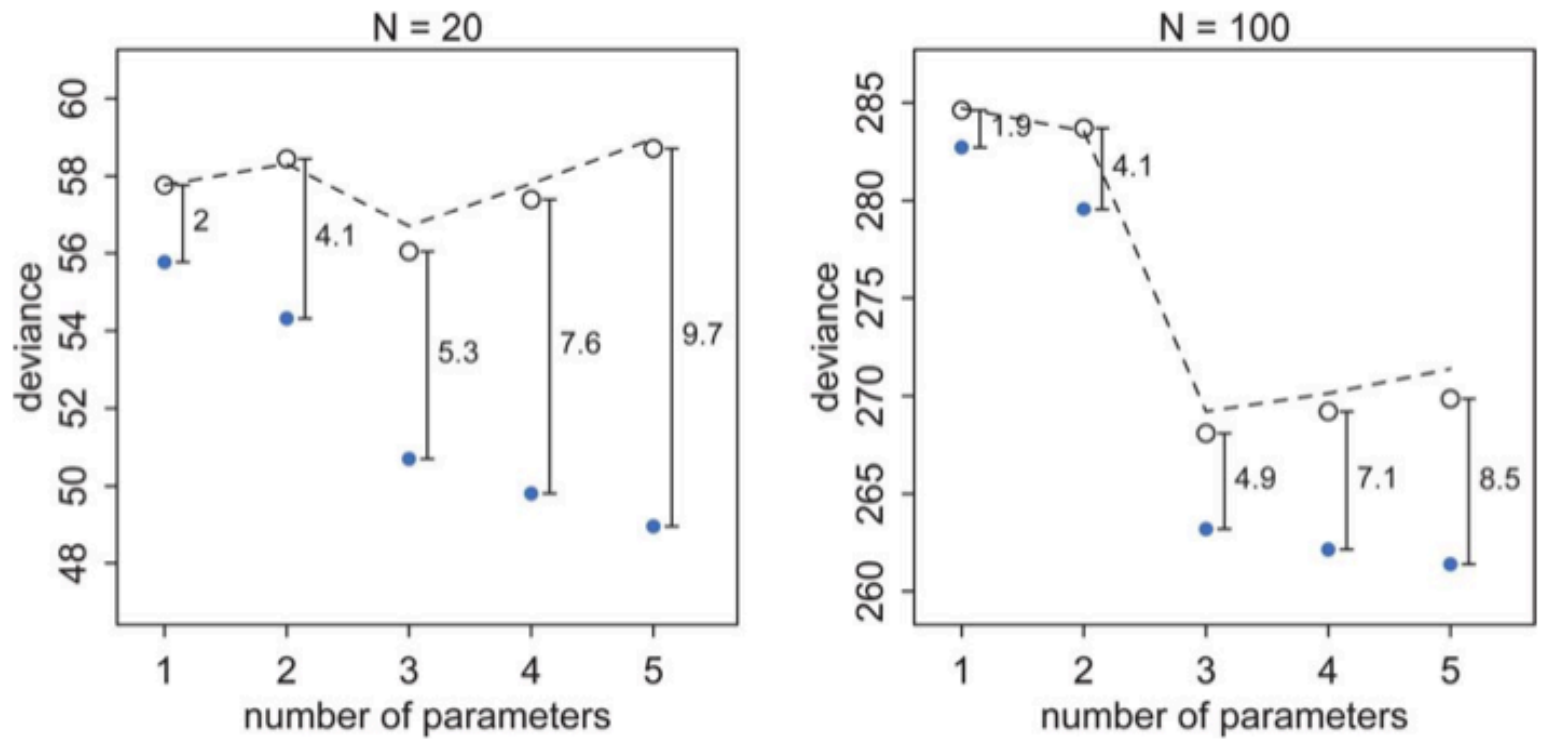
$$D(q) = -2 \sum_i log(q_i),$$

then

$$D_{KL}(p, q) - D_{KL}(p, r) = \frac{2}{N}(D(q) - D(r))$$

More generally: $D(q) = -\frac{N}{2} E_p[log(q)]$

AIC



Akaike Information Criterion, or AIC:

$$AIC = D_{train} + 2p$$

$$D_{train} = -2 * log(p(y|\theta_{mle}))$$

- multivariate gaussian posterior
- flat priors
- data >> parameters

DIC

- uses the posterior distribution, calculable from MCMC.
- multivariate gaussian posterior distribution.

$$D_{train} = -2 * log(p(y|\theta_{postmean}))$$

$$DIC = D_{train} + 2p_D \text{ where}$$

$$p_{DIC} = 2 * (log(p(y|\theta_{postmean})) - E_{post}[log(p(y|\theta)]]) \text{ (by monte carlo)}$$

alternative fomulation for p_D, guaranteed to be positive, is

$$p_D = 2 * Var_{post}[log(p(y|\theta_{postmean}))]$$

```
For Model with 19 slope, dic is 114.8922879773209
For Model with 1 slope, dic is 14.75488127758604
For Model with 2 slope, dic is 9.315532339708351
For Model with 3 slope, dic is 14.42962659691732
For Model with 4 slope, dic is 23.63445316371937
For Model with 5 slope, dic is 26.9986597360957
For Model with 10 slope, dic is 60.43633123874144
For Model with 15 slope, dic is 92.6822800518374
```

Bayesian deviance

$D(q) = -\frac{N}{2} E_p[log(pp(y))]$ posterior predictive for points y on the test set or future data

replace joint posterior predictive over new points y by product of marginals:

ELPD: $\sum_i E_p[log(pp(y_i))]$

Since we do not know the true distribution p,

replace elpd: $\sum_i E_p[log(pp(y_i))]$

by the computed "log pointwise predictive density" (lppd) **in-sample**

$$\sum_j log \langle p(y_j|\theta) \rangle = \sum_j log \left(\frac{1}{S} \sum_s p(y_j|\theta_s) \right)$$

WAIC

$$WAIC = lppd + 2p_W$$

where

$$p_W = 2\sum_i \left(log(E_{post}[p(y_i|\theta)] - E_{post}[log(p(y_i|\theta))]\right)$$

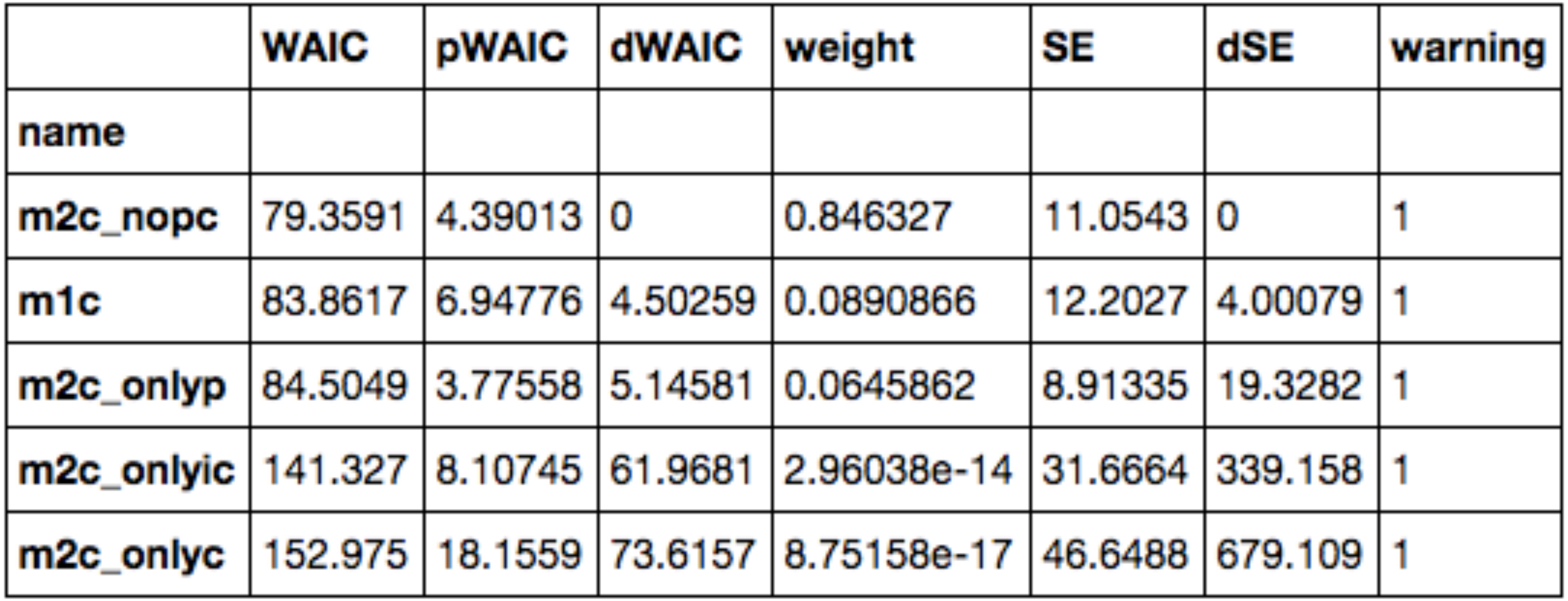
Once again this can be estimated by
$\sum_i Var_{post}[log(p(y_i|\theta))]$

Oceanic tools

Use WAIC to compare models

```
m2c_onlyic: loglam = alpha
m2c_onlyc: loglam = alpha +  betac*df.clevel
m2c_onlyp: loglam = alpha + betap*df.logpop_c
m2c_nopc: loglam = alpha + betap*df.logpop_c + betac*df.clevel
m1c: loglam = alpha + betap*df.logpop_c + betac*df.clevel + betapc*df.clevel*df.logpop_c
```

Centered

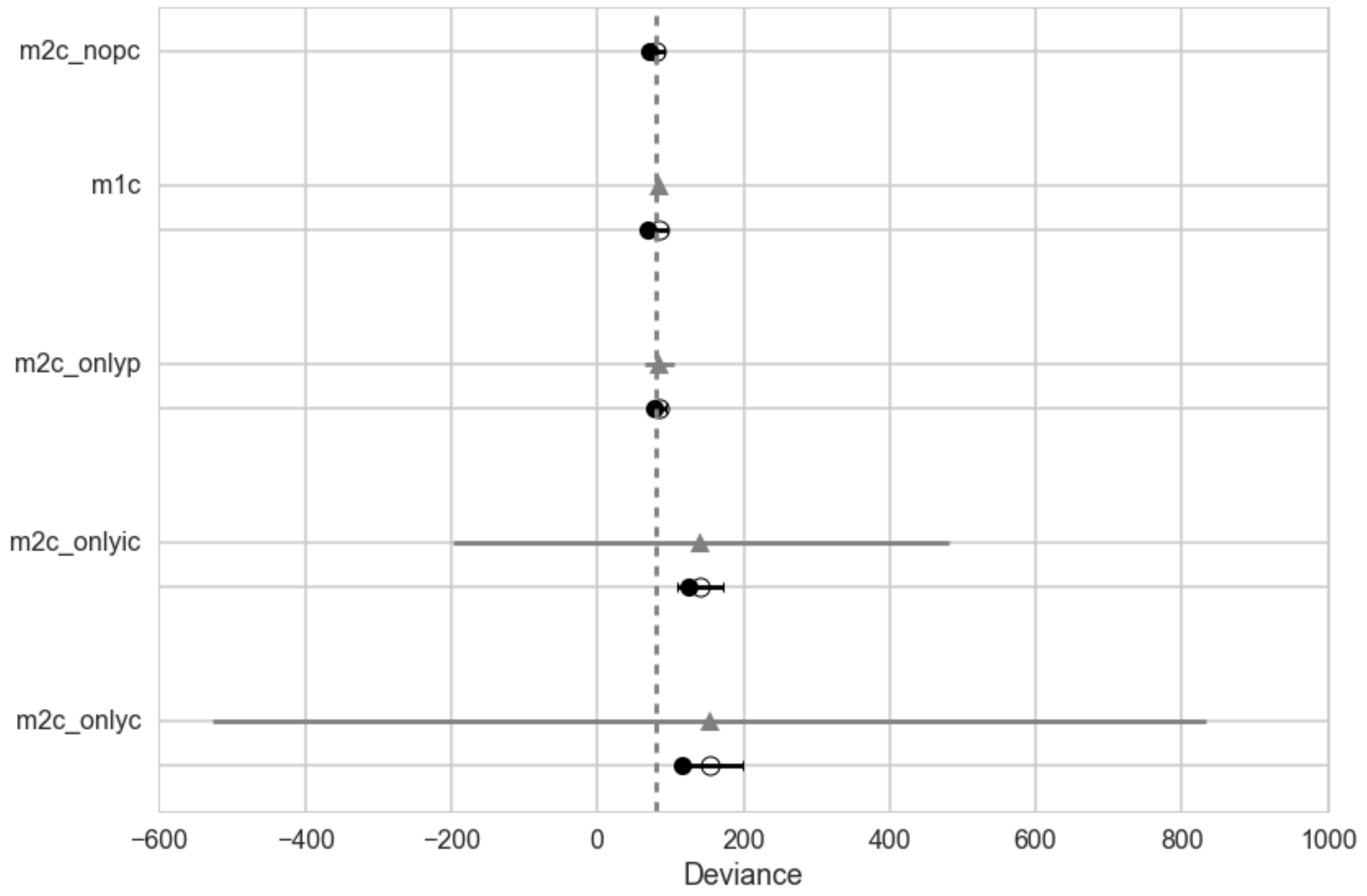


	WAIC	pWAIC	dWAIC	weight	SE	dSE	warning
name							
m2c_nopc	79.3591	4.39013	0	0.846327	11.0543	0	1
m1c	83.8617	6.94776	4.50259	0.0890866	12.2027	4.00079	1
m2c_onlyp	84.5049	3.77558	5.14581	0.0645862	8.91335	19.3282	1
m2c_onlyic	141.327	8.10745	61.9681	2.96038e-14	31.6664	339.158	1
m2c_onlyc	152.975	18.1559	73.6157	8.75158e-17	46.6488	679.109	1

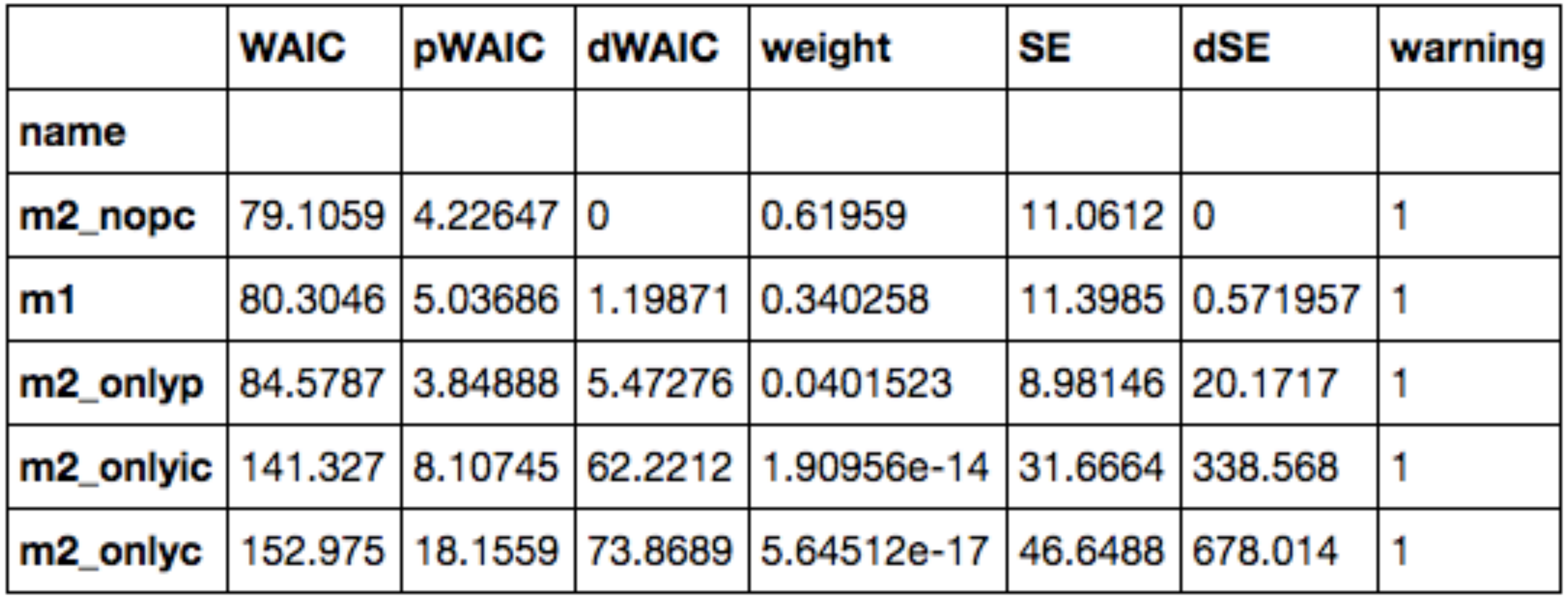
- dWAIC is the difference between each WAIC and the lowest WAIC.
- SE is the standard error of the WAIC estimate.
- dSE is the standard error of the difference in WAIC between each model and the top-ranked model.

$$w_i = \frac{exp(-\frac{1}{2}dWAIC_i)}{\sum_j exp(-\frac{1}{2}dWAIC_j)}$$

read each weight as an estimated probability that each model will perform best on future data.



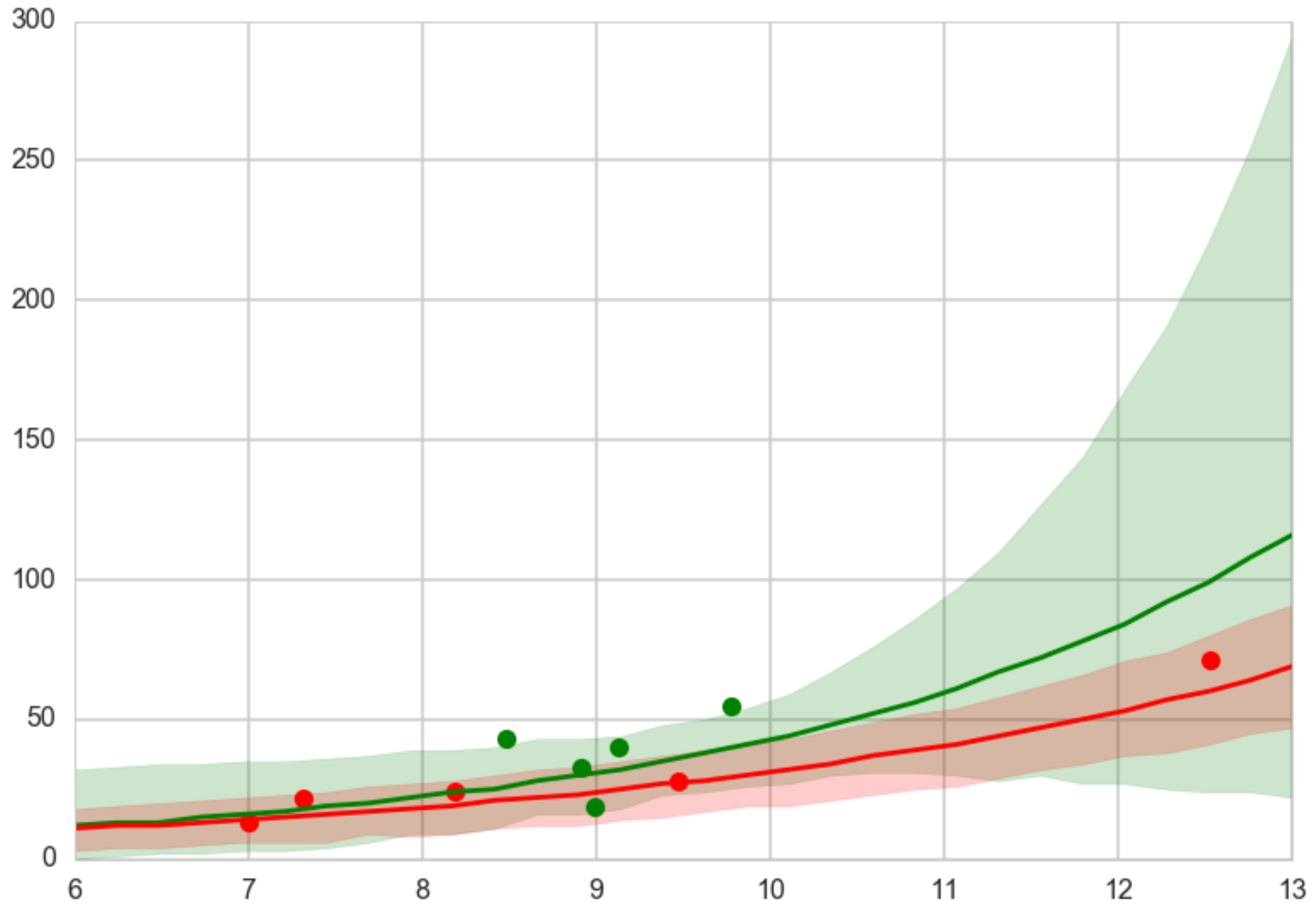
Uncentered



	WAIC	pWAIC	dWAIC	weight	SE	dSE	warning
name							
m2_nopc	79.1059	4.22647	0	0.61959	11.0612	0	1
m1	80.3046	5.03686	1.19871	0.340258	11.3985	0.571957	1
m2_onlyp	84.5787	3.84888	5.47276	0.0401523	8.98146	20.1717	1
m2_onlyic	141.327	8.10745	62.2212	1.90956e-14	31.6664	338.568	1
m2_onlyc	152.975	18.1559	73.8689	5.64512e-17	46.6488	678.014	1

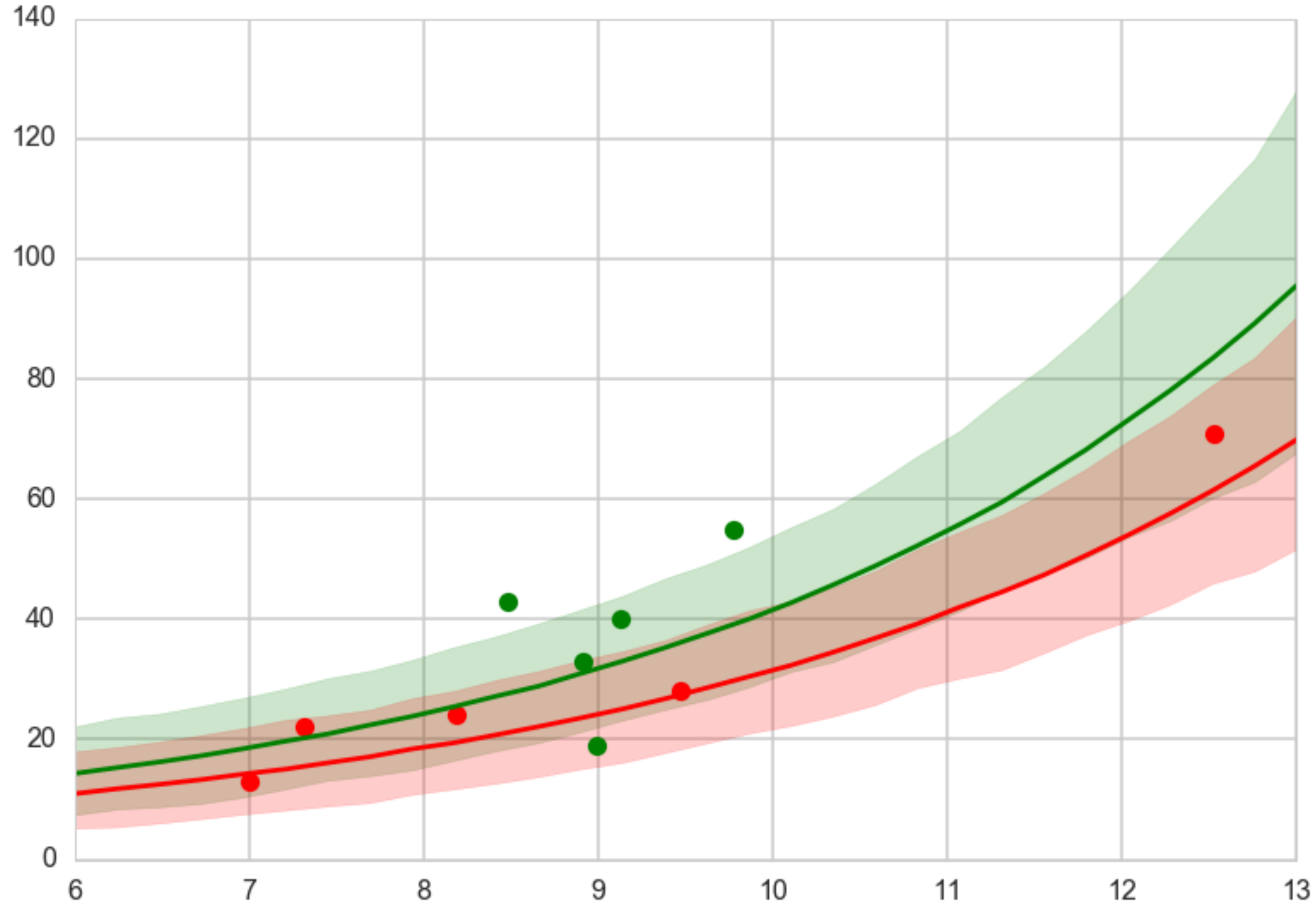
interaction is overfit. centering decorrelates

Counterfactual Posterior predictive

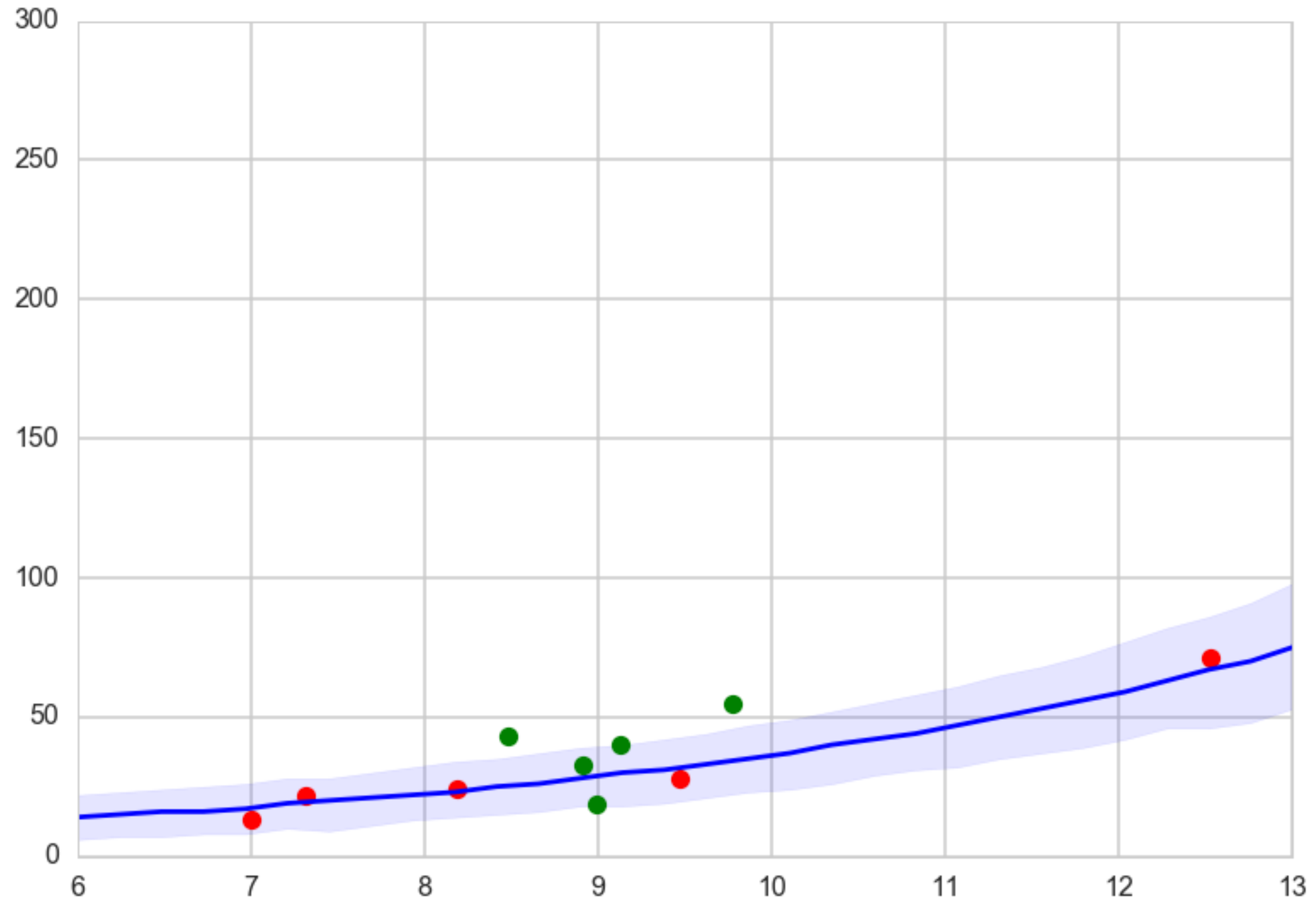


Ensembling

- use WAIC based akaike weights for top 3
- regularizes down the green band at high population by giving more weight to the no-interaction model.



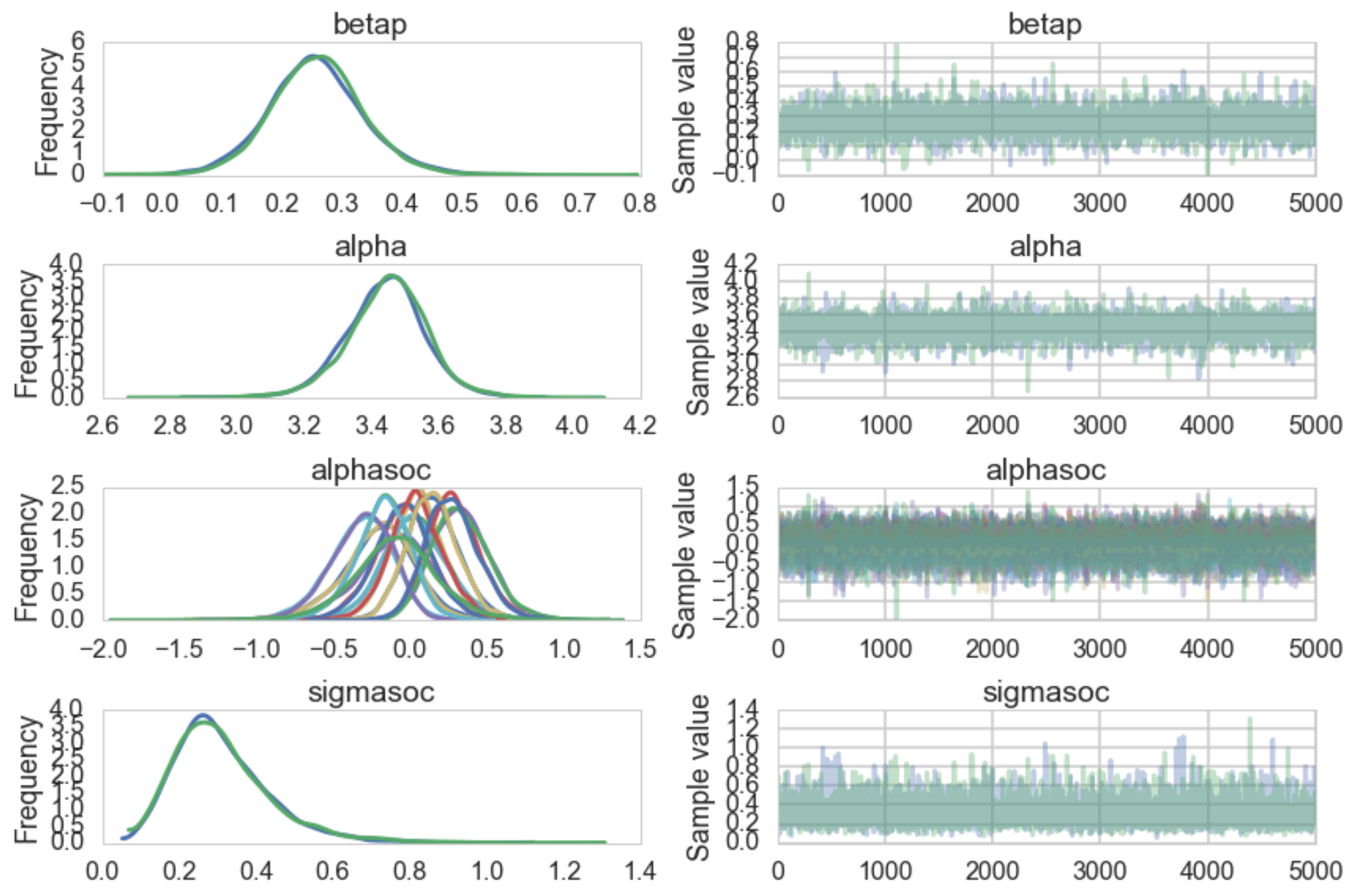
Overdispersion for only p

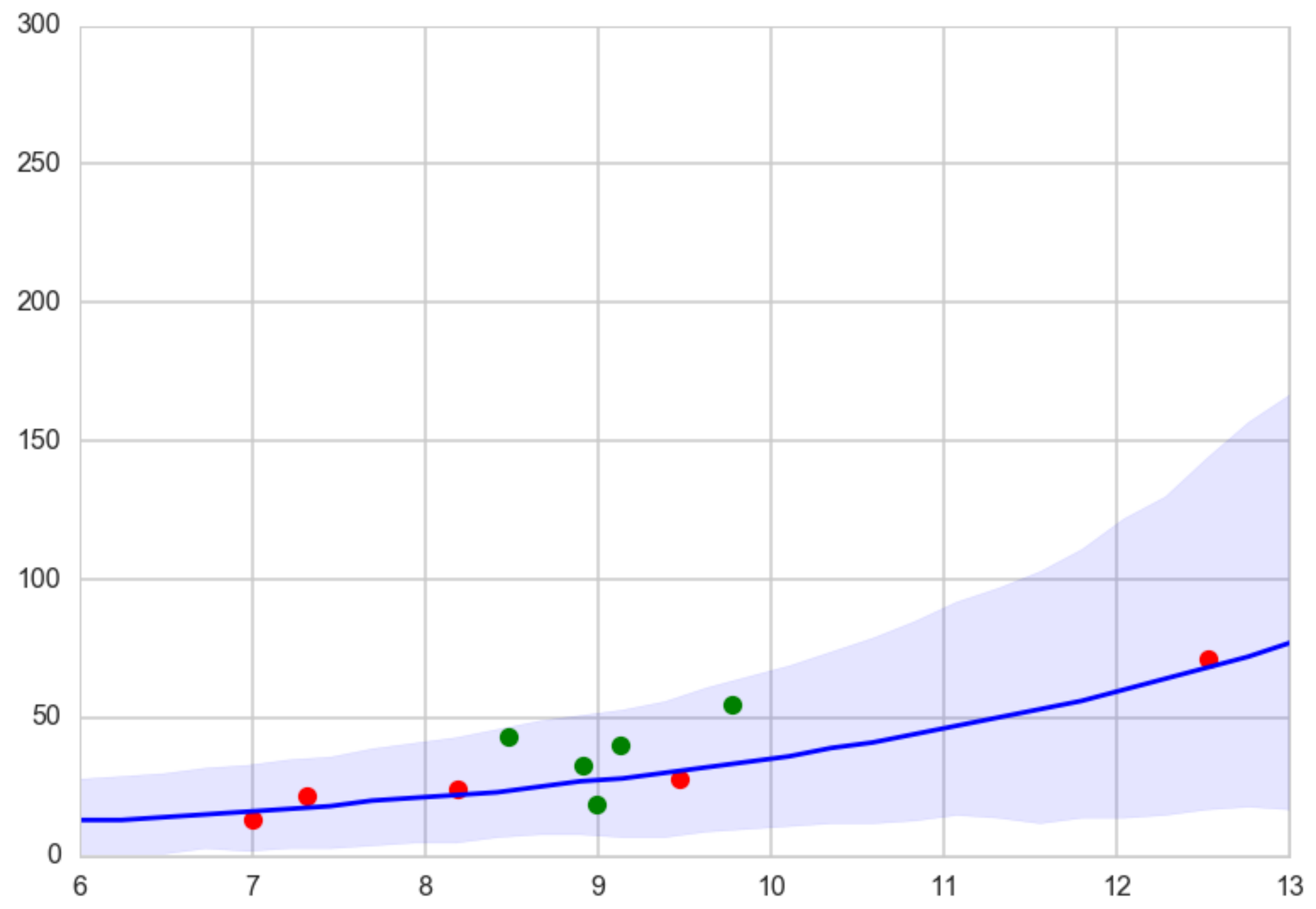


Varying hierarchical intercepts model

```
with pm.Model() as m3c:
    betap = pm.Normal("betap", 0, 1)
    alpha = pm.Normal("alpha", 0, 100)
    sigmasoc = pm.HalfCauchy("sigmasoc", 1)
    alphasoc = pm.Normal("alphasoc", 0, sigmasoc, shape=df.shape[0])
    loglam = alpha + alphasoc + betap*df.logpop_c
    y = pm.Poisson("ntools", mu=t.exp(loglam), observed=df.total_tools)
```

Hierarchical Model Posterior predictive





much wider, includes data areas