Lecture 21

Utility/Risk and Model Comparison

Previously

- posterior predictive checking
- glms
- oceanic tools example and centering

Today

- Decision theory
- point estimates for decisions
- model comparison and ensembling
- oceanic tools and other models model comparison
- poisson over-dispersion and hierarchical modeling
- prosocial chimps in lab

Decision Theory

Predictions (or actions based on predictions) are described by a utility or loss function, whose values can be computed given the observed data.

Indeed one can consider predictions itself as actions to be undertaken with respect to a particular utility.

Point Predictions: squared loss

Sometimes we want to make point predictions. In this case a is a single number.

squared error loss/utility: $l(a, y^*) = (a - y^*)^2$

The optimal point prediction that minimizes the expected loss (negative expected utility):

$$\bar{l}\,(a) = \int dy^*\,(a - y^*)^2\,p(y^*|D, M),$$

is the posterior predictive mean:

$$\hat{a} = E_p[y^*].$$

The expected loss then becomes:

$$\bar{l}(\hat{a}) = \int dy^* \, (\hat{a} - y^*)^2 \, p(y^*|D, M) = \int dy^* \, (E_p[y^*] - y^*)^2 \, p(y^*|D, M) = Var_p[y^*]$$

Squared loss $\implies$ we dont care about skewness or kurtosis

Components of Decision problem

1. $a \in A$, available actions
2. $\omega \in \Omega$, a state in the set of states of the world.
3. $p(\omega|D)$ which tells us out current beliefs about the world.
4. A utility function $u(a, \omega) : A \times \Omega \rightarrow R$ that awards a score/ utility/profit to each action a when the state of the universe is ω. This can be also formulated as a risk/loss.

2 Bayes distributions: posterior and posterior predictive

1. a parameter value or prediction, or action based on prediction
2. If $\Omega = \{y^*\}$, then $\omega = y^*$, a future y. If Ω is the posterior, then $\omega = \theta$ is a value of a parameter(s)
3. This is either the posterior distribution (for θ) or a predictive distribution like posterior predictive (for y^*)
4. A utility for an action based on θ, eg point prediction of median, or on y^*, same idea.

Process

First define the distribution-averaged utility:

$$\bar{u}(a) = \int d\omega \, u(a, \omega) \, p(\omega|D)$$

We then find the a that maximizes this utility:

$$\hat{a} = \arg\max_a \bar{u}(a)$$

This action is called the **bayes action**.

The resulting maximized expected utility is given by:

$$\bar{u}(\hat{a}, p) = \bar{u}(\hat{a}) = \int d\omega \, u(\hat{a}, \omega) \, p(\omega|D),$$

sometimes referred to as the entropy function, and an associate **divergence** can be defined:

$$d(a, p) = \bar{u}(p, p) - \bar{u}(a, p)$$

Then one can think of minimizing $d(a, p)$ with respect to a to get $\hat{a}$, so that this discrepancy can be thought of as a loss function.

Example: Bayes action for posterior predictive

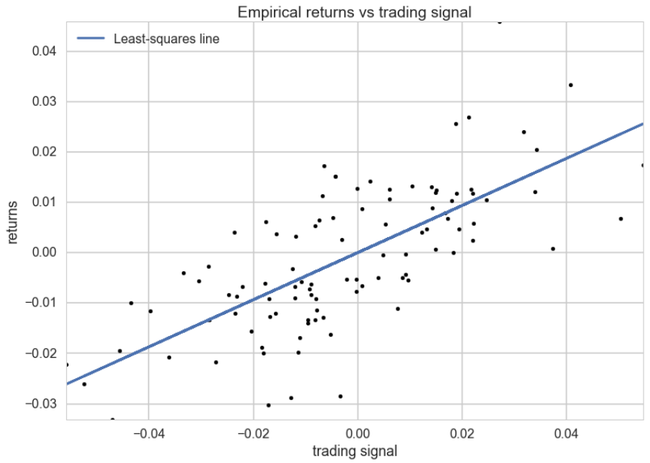
$$\bar{u}(a) = \int dy^* \, u(a, y^*) \, p(y^* | D, M) \text{ OR}$$

$$\bar{u}(a(x)) = \int dy^* \, u(a(x), y^*) \, p(y^* | x^*, D, M) \text{ (supervised)}$$

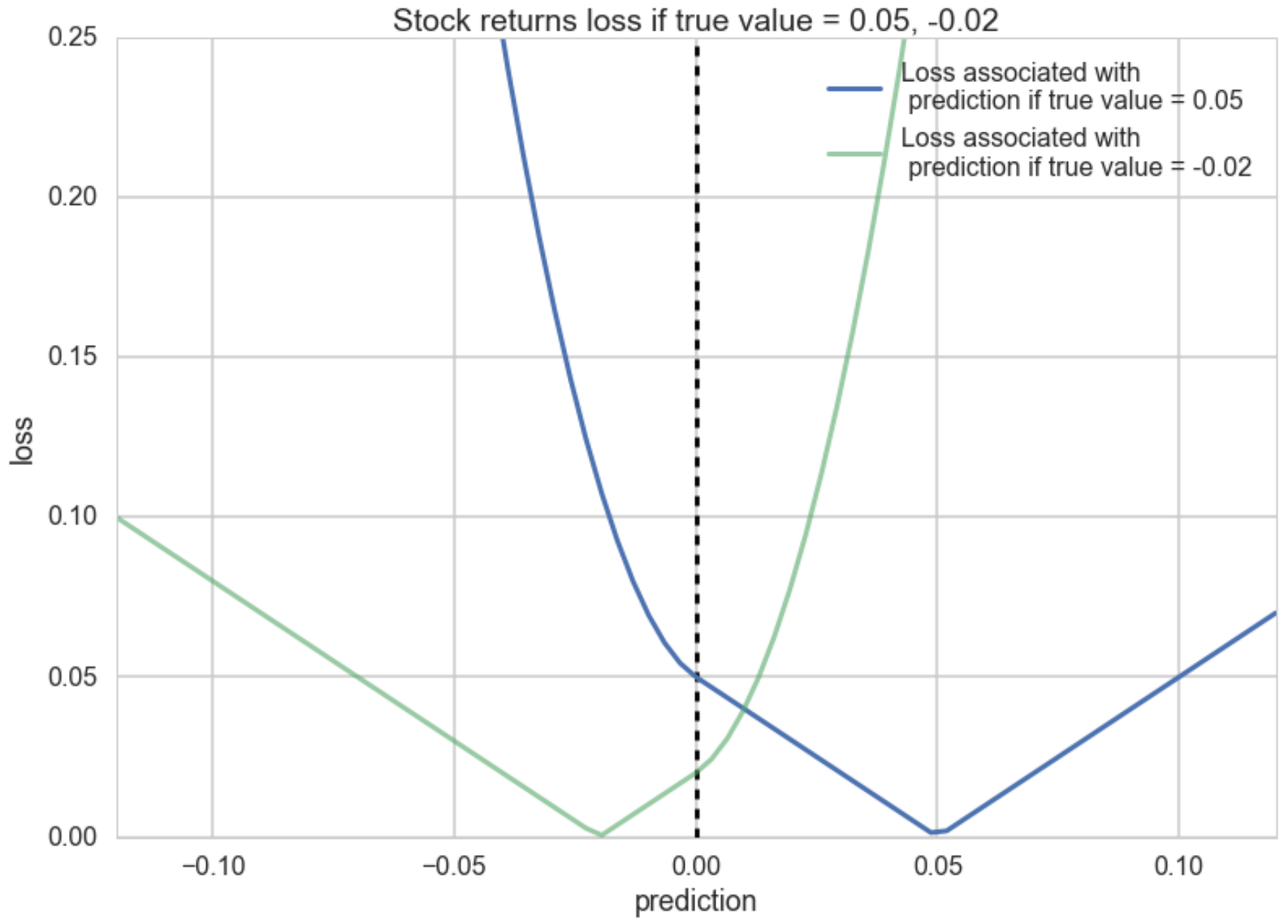
$$\bar{u}(\hat{a}(x^*)) = \int dy^* \, u(\hat{a}, y^*) \, p(y^* | x^*, D, M)$$

$$\hat{a}(x^*) = \arg\max_a \bar{u}(a(x^*))$$

Custom Loss: Stock Market Returns

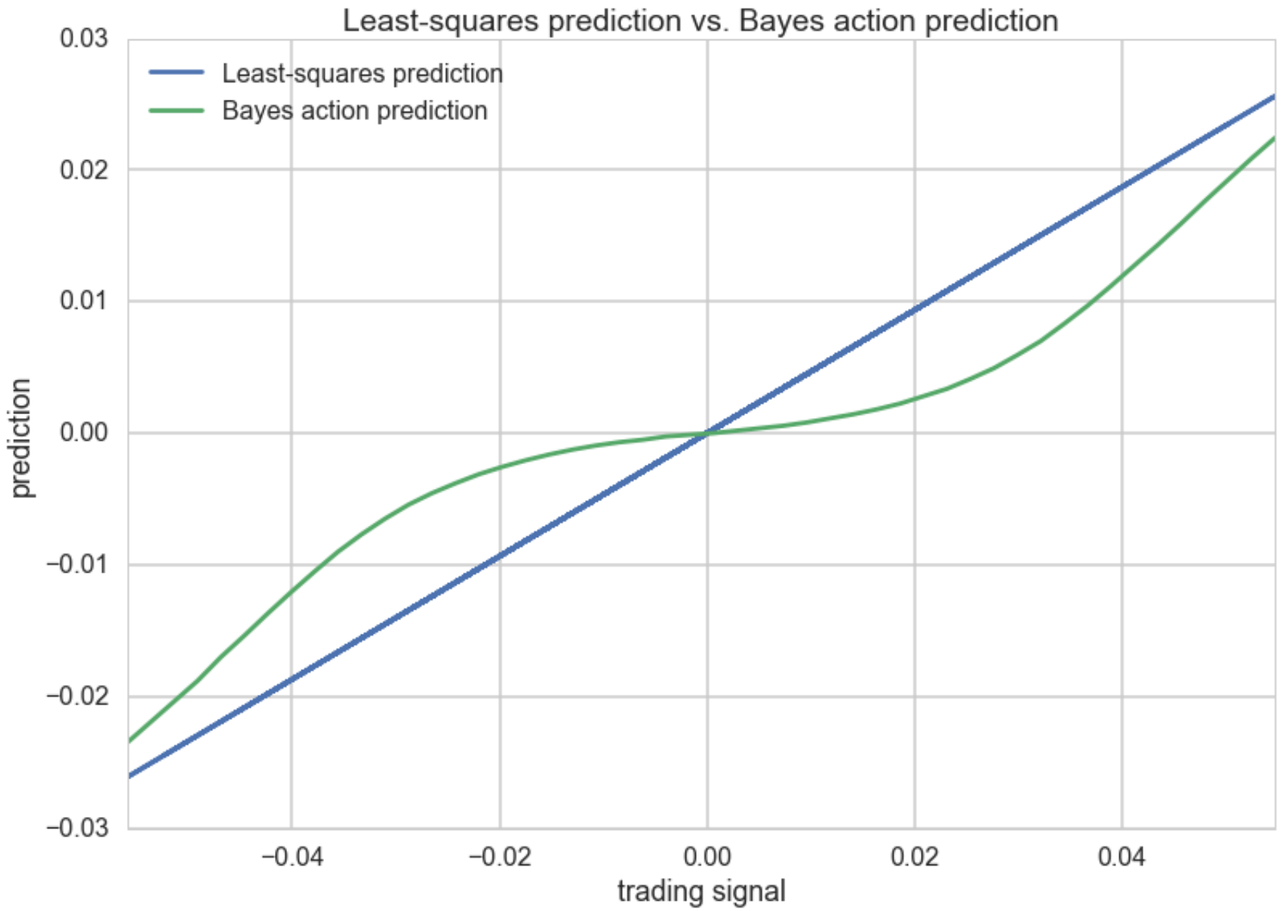



```
def stock_loss(stock_return, pred, alpha = 100.):
    if stock_return * pred < 0:
        #opposite signs, not good
        return alpha*pred**2 - np.sign(stock_return)*pred \
                        + abs(stock_return)
    else:
        return abs(stock_return - pred)
```



Loss at very x

```
with pm.Model() as model:
    std = pm.Uniform("std", 0, 100)

    beta = pm.Normal("beta", mu=0, sd=100)
    alpha = pm.Normal("alpha", mu=0, sd=100)

    mean = pm.Deterministic("mean", alpha + beta*X)

    obs = pm.Normal("obs", mu=mean, sd=std, observed=Y)

    trace = pm.sample(100000, step=pm.Metropolis())
    burned_trace = trace[20000:]
    ...
noise = std_samples*np.random.randn(N)

#posterior predictive samples at every x
possible_outcomes = lambda signal: alpha_samples + beta_samples*signal + noise


opt_predictions = np.zeros(50)
trading_signals =  np.linspace(X.min(), X.max(), 50)
for i, _signal in enumerate(trading_signals):
        _possible_outcomes = possible_outcomes(_signal)
        #expected loss over posterior predictive
        tomin = lambda pred: stock_loss(_possible_outcomes, pred).mean()
        #bayes action minimizes expected loss
        opt_predictions[i] = fmin(tomin, 0, disp = False)
```



The two risks

There are *two risks in learning* that we must consider, one to *estimate probabilities*, which we call **estimation risk**, and one to *make decisions*, which we call **decision risk**.

The **decision loss** $l(y, a)$ or **utility** $u(l, a)$ (profit, or benefit) in making a decision a when the predicted variable has value y. For example, we must provide all of the losses l(no-cancer, biopsy), l(cancer, biopsy), l(no-cancer, no-biopsy), and l(cancer, no-biopsy). One set of choices for these losses may be 20, 0, 0, 200

Classification Risk

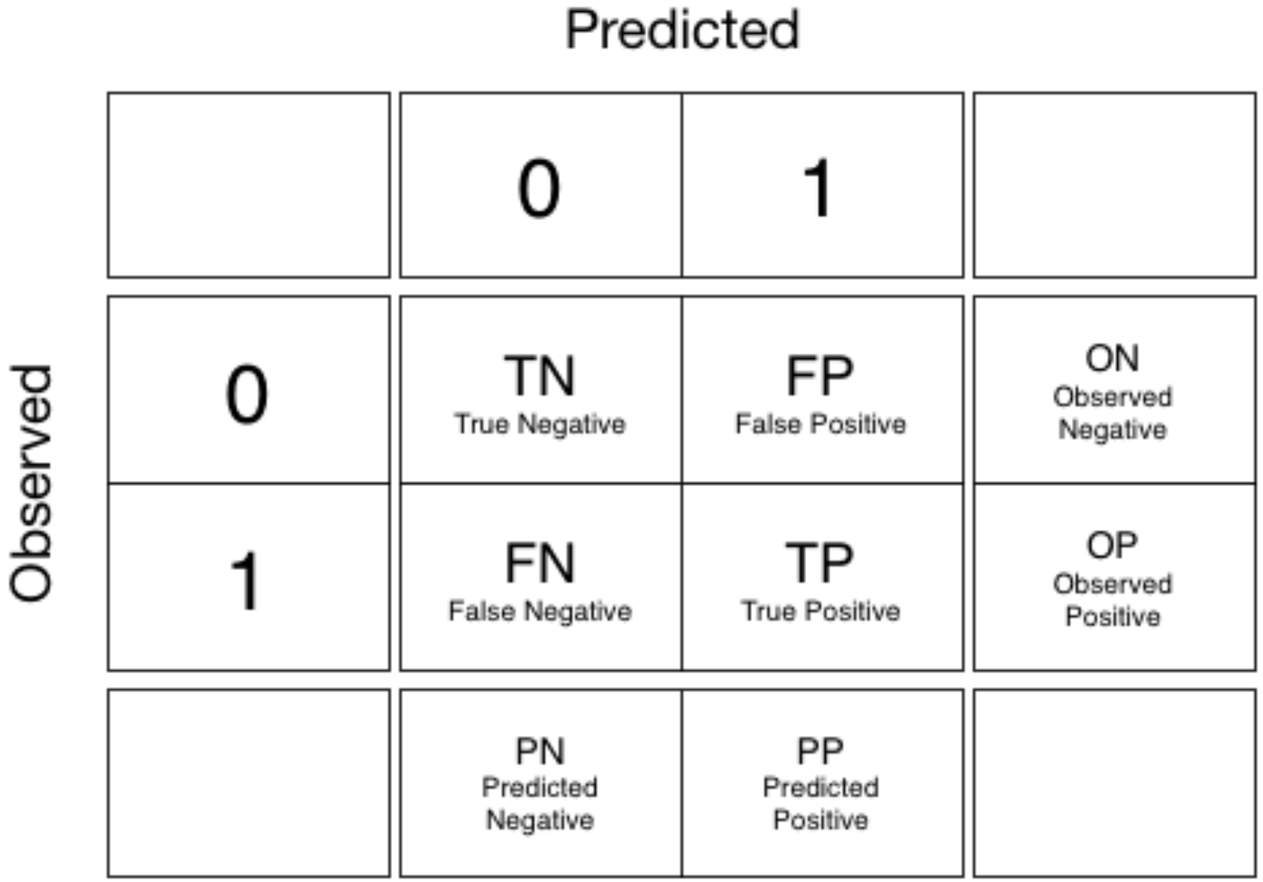
$$R_a(x) = \sum_y l(y, a(x))p(y|x)$$

That is, we calculate the **predictive averaged risk** over all choices y, of making choice a for a given data point.

Overall risk, given all the data points in our set:

$$R(a) = \int dx p(x) R_a(x)$$

Two class Classification



$$R_a(x) = l(1,g)p(1|x) + l(0,g)p(0|x).$$

Then for the "decision" $a = 1$ we have:

$$R_1(x) = l(1,1)p(1|x) + l(0,1)p(0|x),$$

and for the "decision" $a = 0$ we have:

$$R_0(x) = l(1,0)p(1|x) + l(0,0)p(0|x).$$

Now, we'd choose 1 for the data point at x if:

$$R_1(x) < R_0(x).$$

$$P(1|x)(l(1,1) - l(1,0)) < p(0|x)(l(0,0) - l(0,1))$$

So, to choose '1', the Bayes risk can be obtained by setting:

$$p(1|x) > rP(0|x) \implies r = \frac{l(0,1) - l(0,0)}{l(1,0) - l(1,1)}$$

$$P(1|x) > t = \frac{r}{1+r}.$$

One can use the prediction cost matrix corresponding to the consufion matrix

$$r == \frac{c_{FP} - c_{TN}}{c_{FN} - c_{TP}}$$

If you assume that True positives and True negatives have no cost, and the cost of a false positive is equal to that of a false positive, then $r = 1$ and the threshold is the usual intuitive $t = 0.5$.

		Predicted	
		0	1
Observed	0	TNC True Negative Cost	FPC False Positive Cost
	1	FNC False Negative Cost	TPC True Positive Cost

Log score: probabilistic prediction (estimation risk)

Here we want to find a distribution a.

The utility is defined as:

$$u(a, y^*) = log\, a(y^*),$$

The expected utility then is

$$\bar{u}(a) = \int dy^*\, log(a(y^*))\, p(y^*|D, M).$$

The a that maximizes this utility is the predictive itself (in the bayesian context, the posterior predictive)!

$$\hat{a}(y^*) = p(y^*|D, M)$$

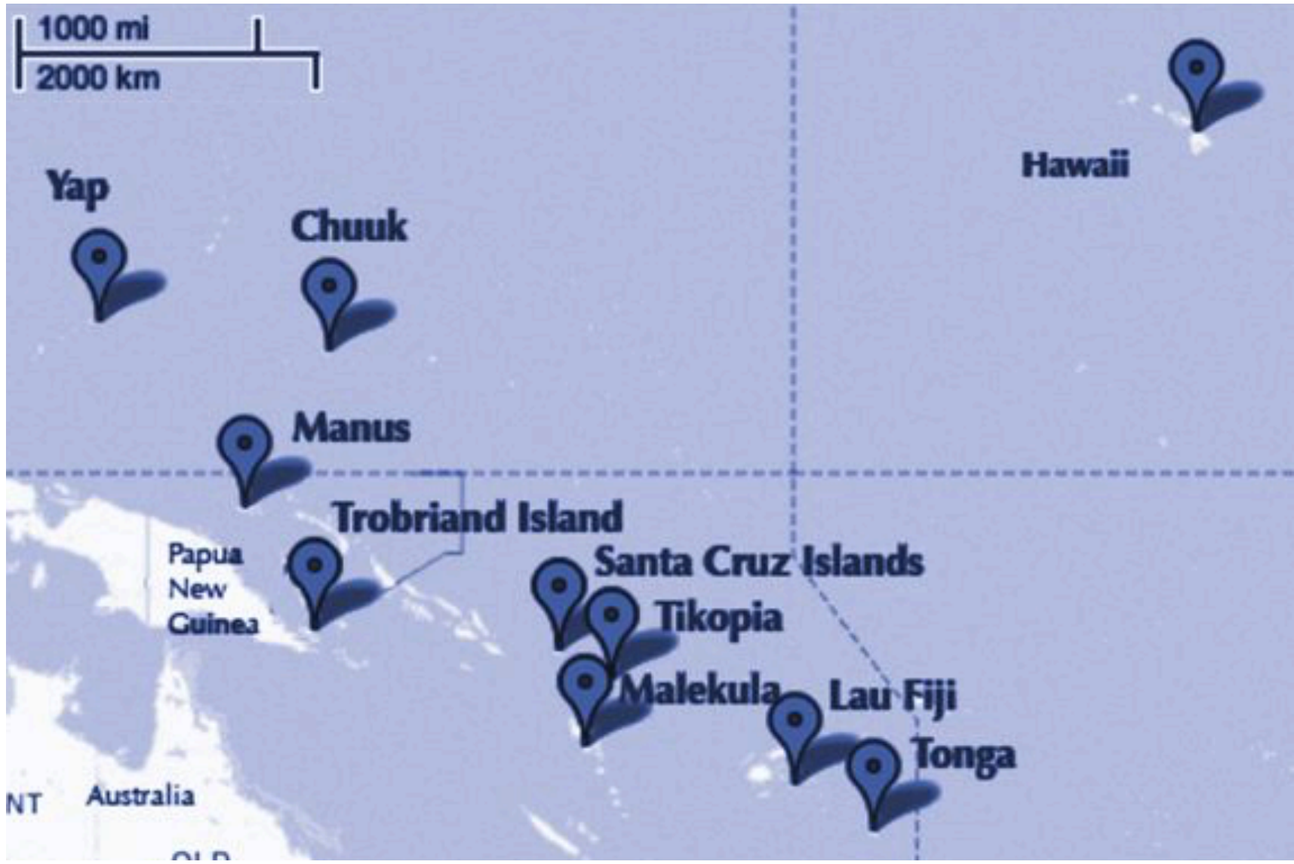
Maximized utility: $\bar{u}(a) = \int dy^* \, log(p(y^*|D, M)) \, p(y^*|D, M)$.

This is just the negative entropy of the predictive distribution, and the associated divergence is our old friend the KL-divergence.

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

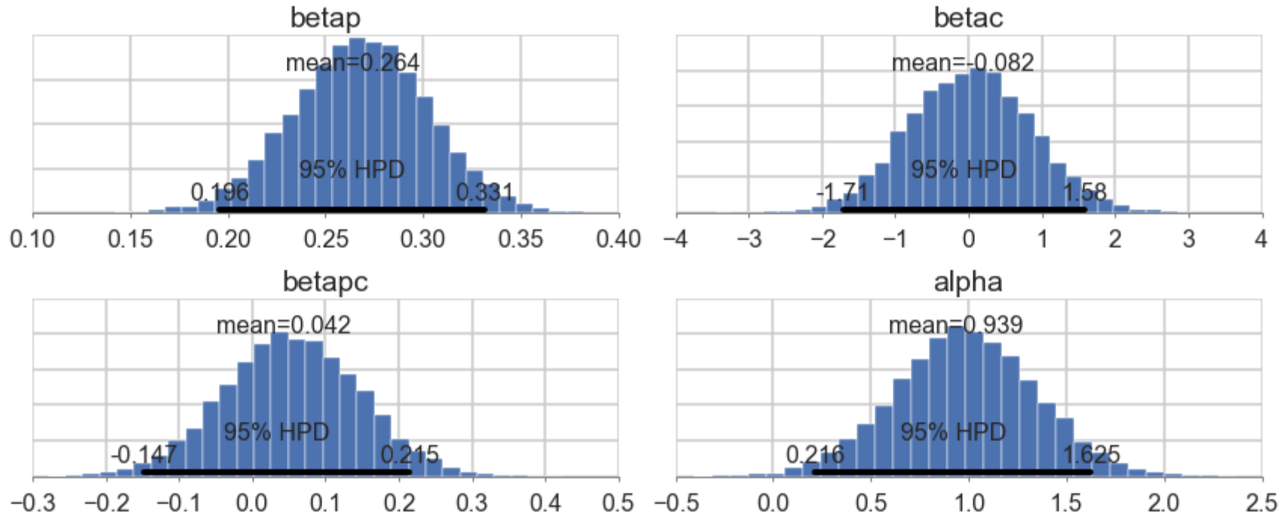
	culture	population	contact	total_tools	mean_TU	logpop	clevel
0	Malekula	1100	low	13	3.2	7.003065	0
1	Tikopia	1500	low	22	4.7	7.313220	0
2	Santa Cruz	3600	low	24	4.0	8.188689	0
3	Yap	4791	high	43	5.0	8.474494	1
4	Lau Fiji	7400	high	33	5.0	8.909235	1
5	Trobriand	8000	high	19	4.0	8.987197	1
6	Chuuk	9200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



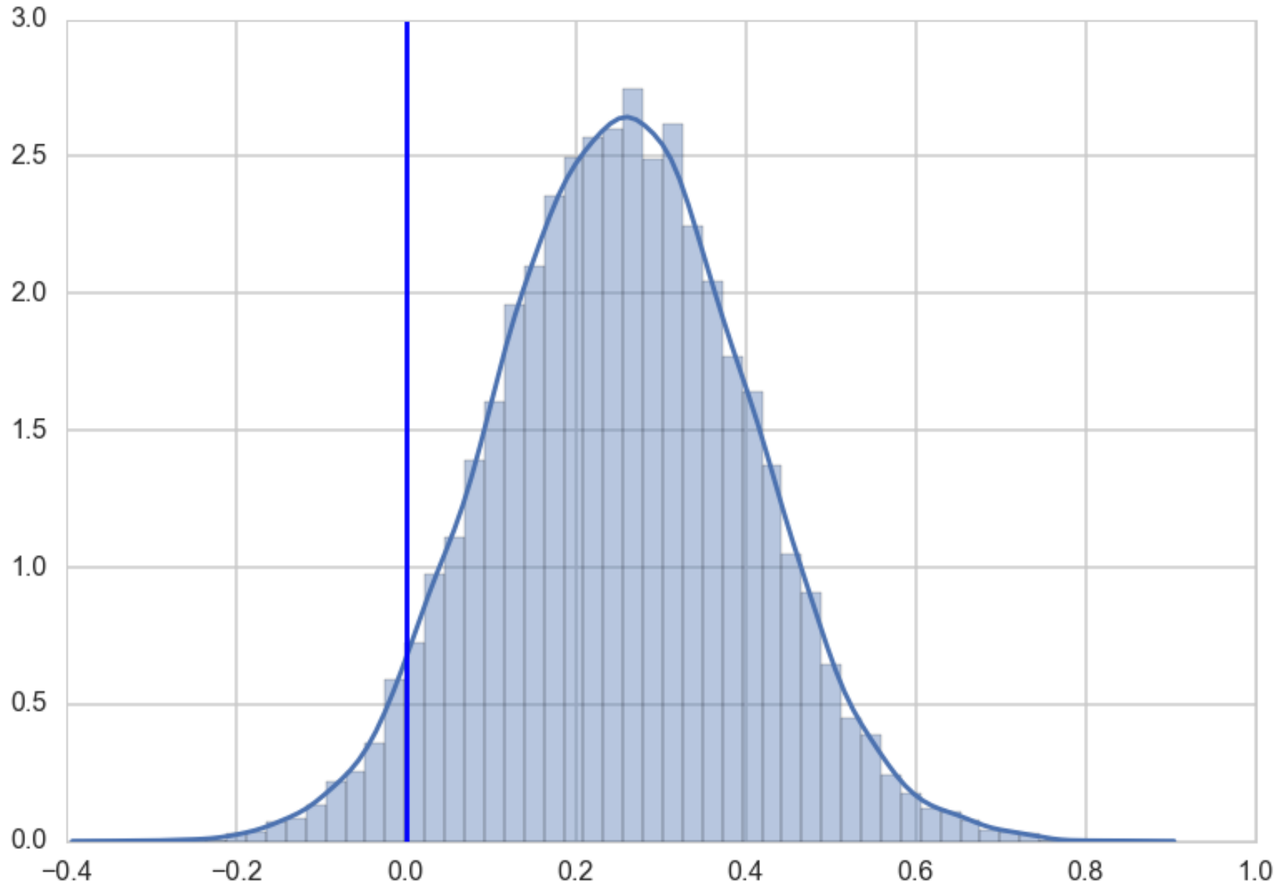
- traces and autocorrelations look good
- The posterior for β_p tightly constrained, and as expected from theory, shows a positive effect.
- The posteriors for β_c and β_{pc} both overlap 0 substantially, and seem comparatively poorly constrained.
- no substantial effect of contact rate, directly or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

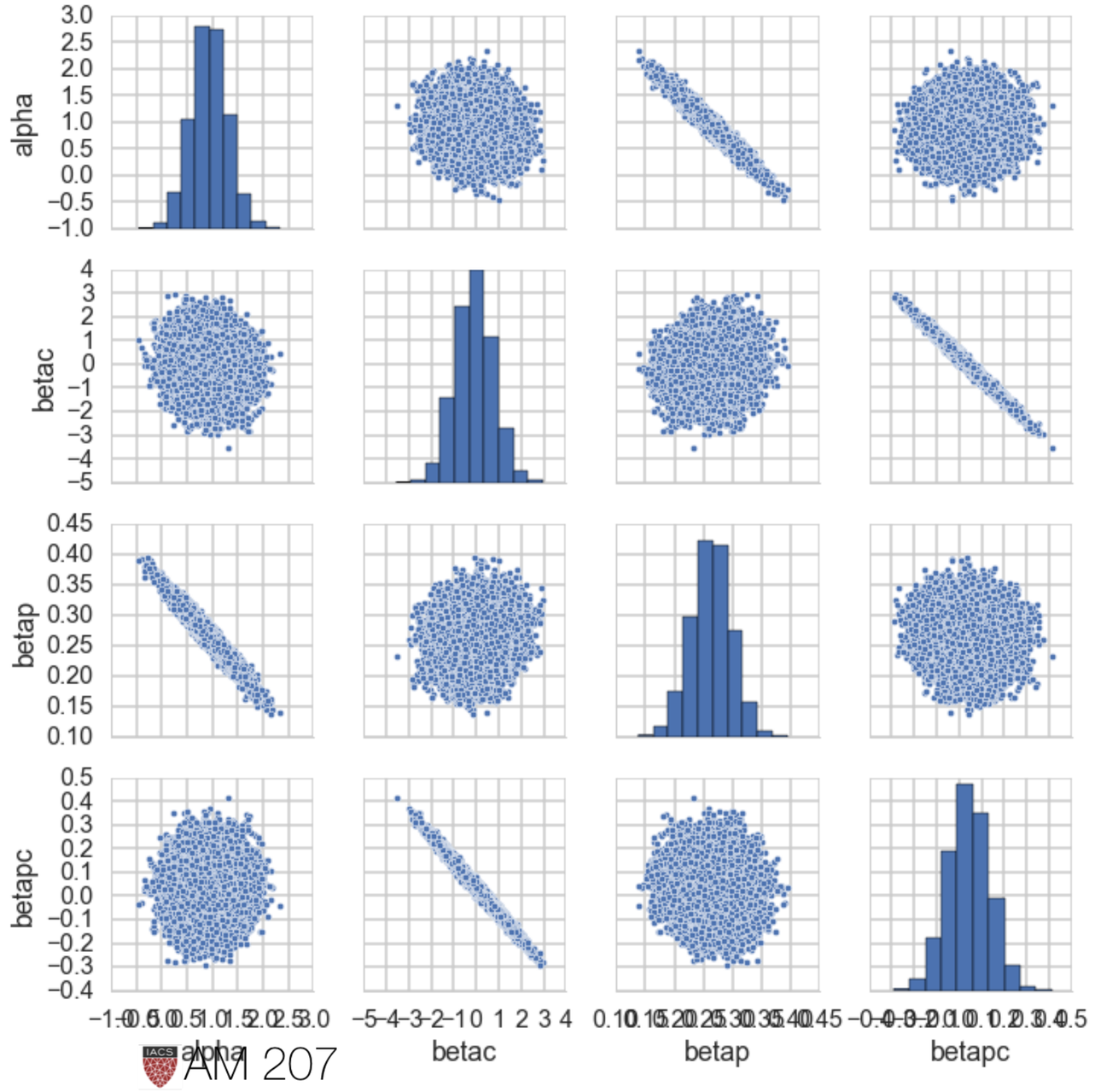
You would be wrong: counterfactual predictions

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

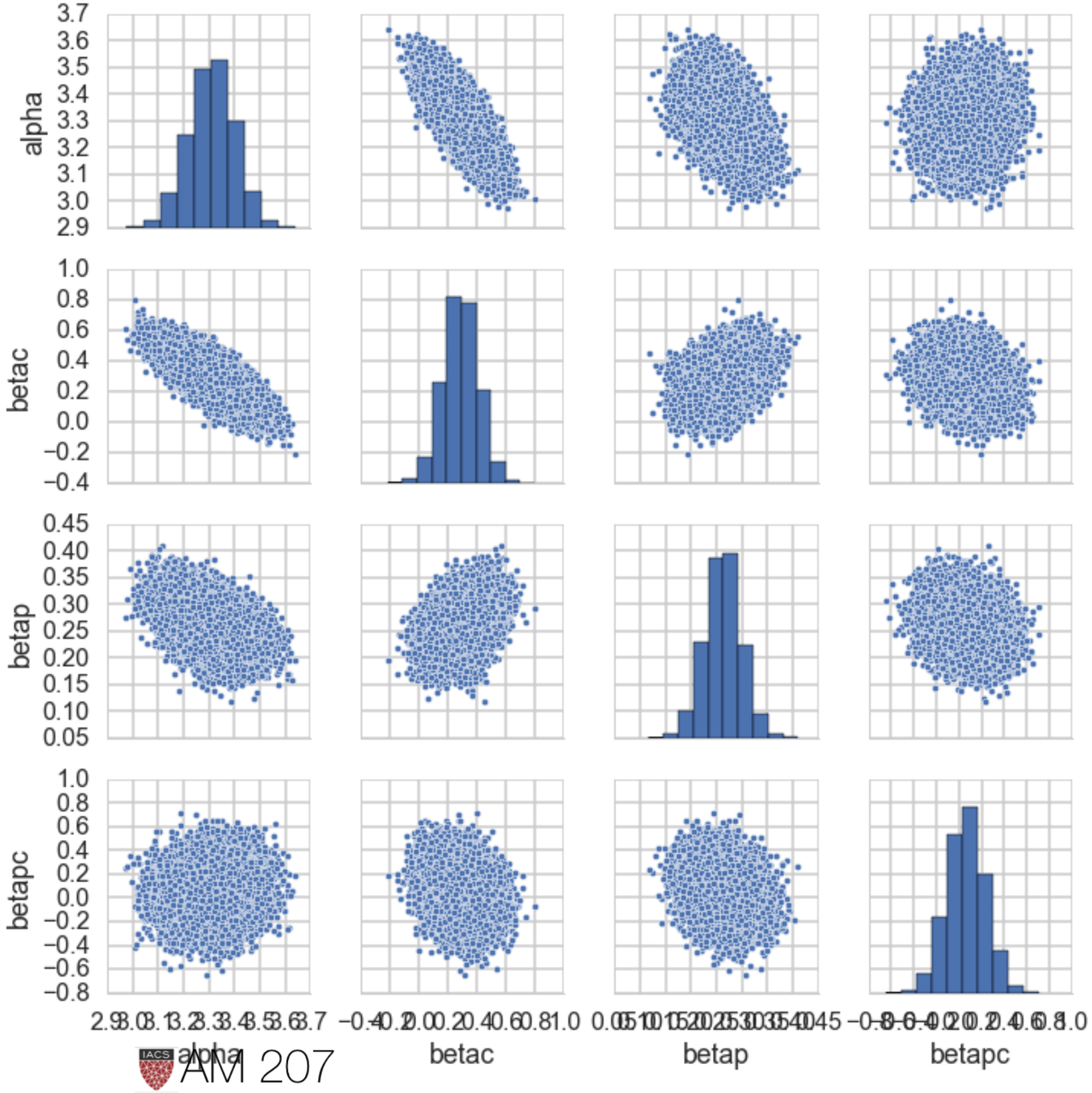
Fix by centering

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

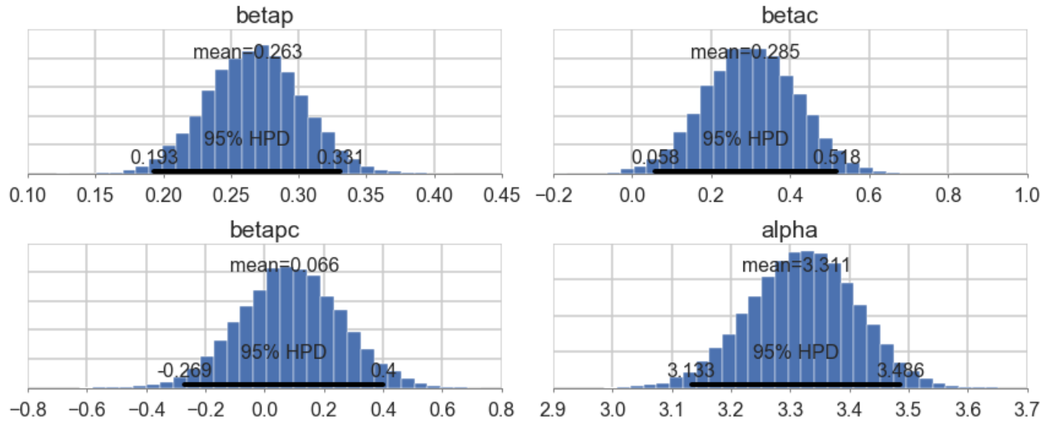
```
{'alpha': 8110.0, 'betac': 4600.0, 'betap': 8016.0, 'betapc': 4597.0}
```

```
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!



Were the contacts really needed?

Let us 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
```

Which Model to compare against?

- In model comparison scenario we might use the "true" distribution:

$$\bar{u}_t(\hat{a}) = \int dy^* u(\hat{a}, y^*) p_t(y^*)$$

Notice that we use $u(\hat{a}, y^*)$. The $\hat{a}$ has already been found by optimizing over our posterior predictive.

True-belief distribution

- the "p" we used in KL-divergence formulae eons ago
- model M_{tb} that has undergone posterior predictive checks and is very expressive, a model we can use as a reference model.
- often non-parametric or found via bayesian model averaging.
- if the true generating process is outside the hypothesis set of the models you are using, true belief model never = true. This is called misfit or bias.

Model comparison

The key idea in model comparison is that we will sort our average utilities in some order. The exact values are not important, and may be computed with respect to some true distribution or true-belief distribution M_{tb}.

Utility is maximized with respect to some model $M_k \in \mathcal{H}$ whereas the average of the utility is computed with respect to either the true, or true belief distribution.

$$\bar{u}(M_k, \hat{a}_k) = \int dy^* u(\hat{a}_k, y^*) p(y^* | D, M_{tb})$$

where a_k is the optimal prediction under the model M_k. Now we compare the actions, that is, we want:

$$\hat{M} = \arg\max_k \bar{u}(M_k, \hat{a}_k)$$

No calibration, but calculating the standard error of the difference can be used to see if the difference is significant, as we did with the WAIC score

We now maximize this over M_k .

For the squared loss the first step gives us $\hat{a}_k = E_{p(y^*|D,M_k)}[y^*]$.
Then:

$$\bar{l}\,(\hat{a_k}) = \int dy^*\,(\hat{a}_k - y^*)^2\,p(y^*|D, M_{tb})$$

$$= \int dy^*\,(E_{p_k}[y^*] - y^*)^2\,p(y^*|D, M_{tb}) = Var_{p_{tb}}[y^*] + (E_{p_{tb}}[y^*] - E_{p_k}[y^*])^2$$

We have bias if M_{tb} is not in our Hypothesis set $\mathcal{H}$.

Information criteria

- we dont want to go out-of-sample
- use information criteria to decide between models
- 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: $D(q) = -\frac{N}{2} E_p[log(q)]$

Key points

- Deviance of a predictive with respect to itself is the "action" that minimizes the loss = -utility: $-u(a, y^*) = -log\, a(y^*)$,. This is just the negative entropy.
- But once we have found the predictive that minimizes the loss, we use this "bayes action" for our model comparison: ie the deviance with respect to M_{tb} (notation: or p_{tb} or just p as we have introduced in the information theory lectures).

Deviance of a predictive

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

We want to estimate the "true-belief" average of a predictive:

$$E_p[log(pred(y^*))]$$

where $pred(y^)$ *is the predictive for points* $y^$ on the test set or future data.

Do it pointwise instead

Call the expected log predictive density at a "new" point:

$$elpd_i = E_p[log(pred(y_i^*)]$$

Then the "expected log pointwise predictive density" is

$$elppd = \sum_i E_p[log(pred(y_i^*))] = \sum_i elpd_i$$

What predictive distribution $pred$ do we use? We start from the frequentist scenario of using the likelihood at the MLE for the AIC, then move to using the likelihood at the posterior mean (a sort of plug in approximation) for the DIC, and finally to the fully Bayesian WAIC.

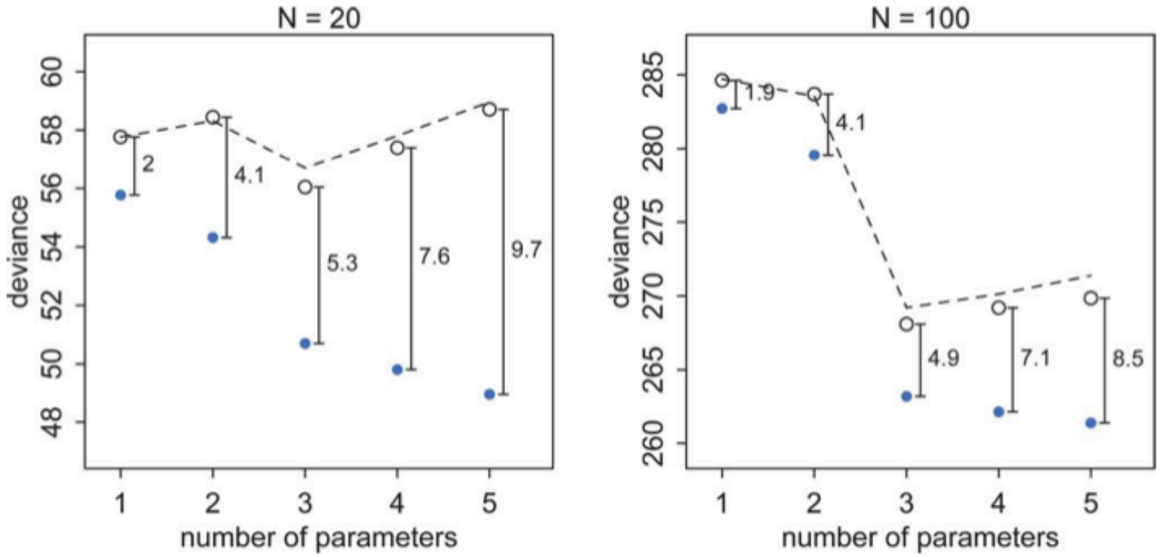
Specifically, in the first two cases, we are writing the predictive distribution conditioned on a point estimate from the posterior:

$$elpd_i = E_p[log(pred(y_i^* \mid \hat{\theta}))]$$

The game we will play in these first two cases is:

(1) Conditional on fixed θ, the full predictive splits into a product per point so the writing of elppd as a sum over pointwise elpd is exact
(2) However we dont know p_{tb} (or just p), so we use the empirical distribution on the training set
(3) this underestimates the test set deviance as we learnt in the case of the AIC, so we must apply a correction factor.

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, and assumes multivariate gaussian posterior distribution.

$$D_{train} = -2 * log(p(y|\theta_{postmean}), DIC = D_{train} + 2p_{DIC}$$ where

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

alternative fomulation for p_D, guaranteed to be positive, is

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

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 pp over new points y by product of marginals:

$$elpd_i = E_p[log(pp(y_i^*)]$$

$$elppd = \sum_i E_p[log(pp(y_i^*))] = \sum_i elpd_i$$

Game is to REPLACE

$$elppd = \sum_i E_p[log(pp(y_i^*))]$$ where y_i^* are new points

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

$$lppd = log\left(\prod_j pp(y_j)\right) = \sum_j log\langle p(y_j|\theta)\rangle_{post} = \sum_j log\left(\frac{1}{S}\sum_{s\sim post} p(y_j|\theta_s)\right)$$

- As we know now, is that **the $lppd$ of observed data y is an overestimate of the $elppd$ for future data**.
- Hence the plan is to like to start with the $llpd$ and then apply some sort of bias correction to get a reasonable estimate of $elppd$.

This gives us the WAIC (Widely Applicable Information Criterion or Watanable-Akaike Information Criterion)

WAIC

$$WAIC = lppd + 2p_W$$

where

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

Once again this can be estimated by

$$\sum_i Var_{post}[log(p(y_i|\theta))]$$

...it is tempting to use information criteria to compare models with different likelihood functions. Is a Gaussian or binomial better? Can't we just let WAIC sort it out?

Unfortunately, WAIC (or any other information criterion) cannot sort it out. The problem is that deviance is part normalizing constant. The constant affects the absolute magnitude of the deviance, but it doesn't affect fit to data.

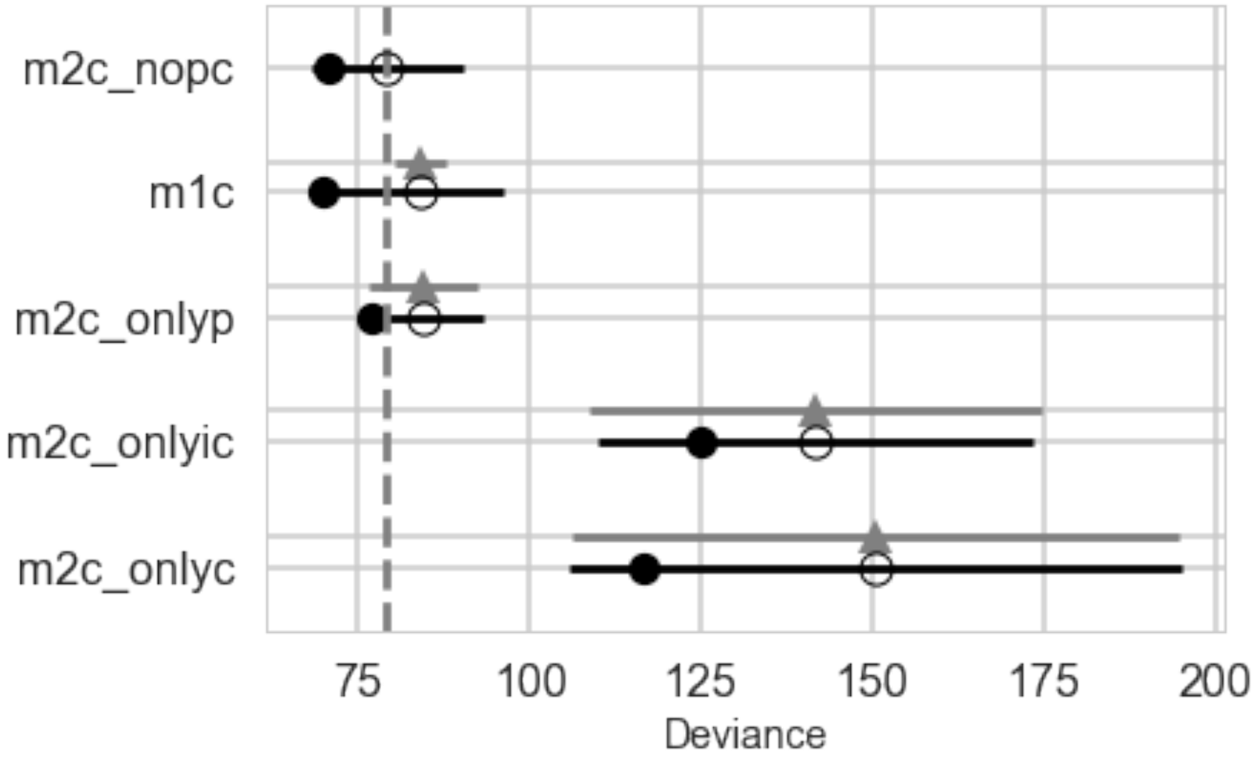
— McElreath

Oceanic tools

Lets use the 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



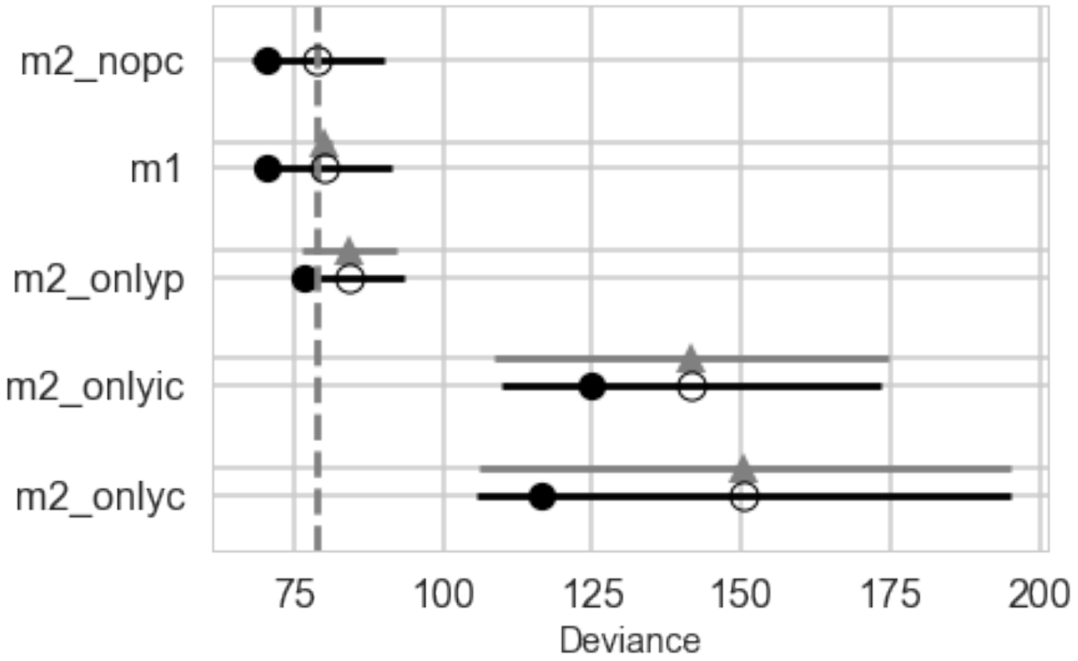
- dWAIC is the difference between each WAIC and the lowest WAIC.
- SE is the standard error of the WAIC estimate.
- dSE is the standard error of the difference in WAIC between each model and the top-ranked model.

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

read each weight as an estimated

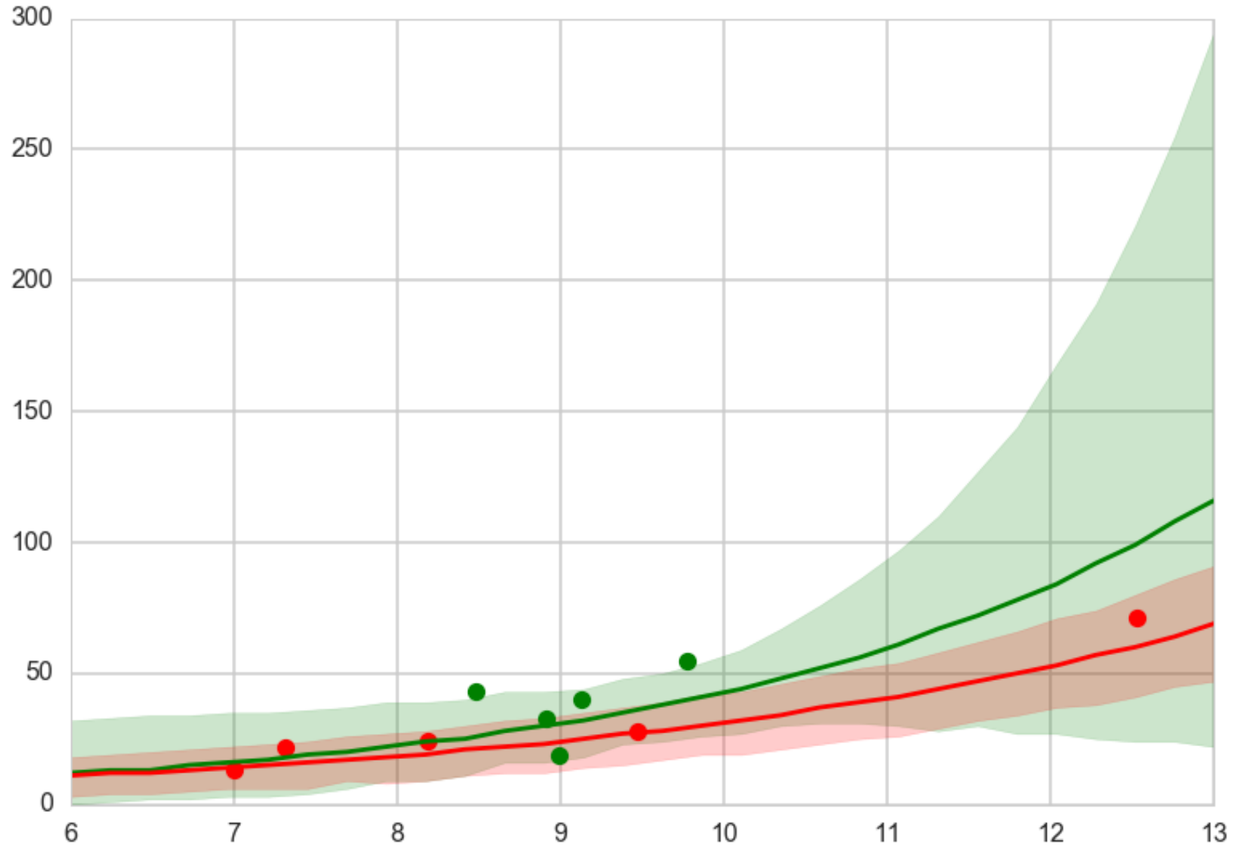
Uncentered

	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



Bayesian Model Averaging

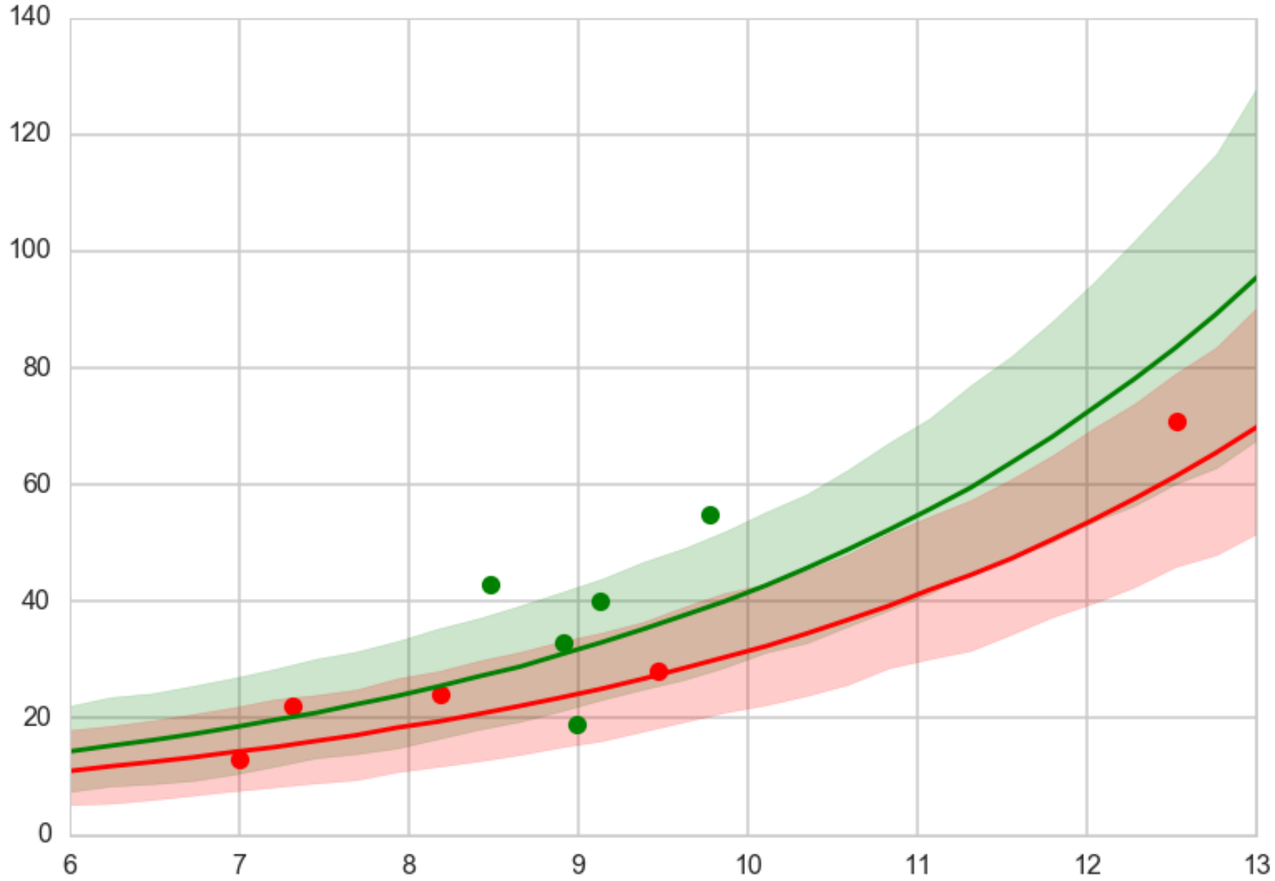
$$p_{BMA}(y^*|x^*, D) = \sum_k p(y^*|x^*, D, M_k)p(M_k|D)$$

where the averaging is with repect to weights $w_k = p(M_k|D)$, the posterior probabilities of the models M_k.

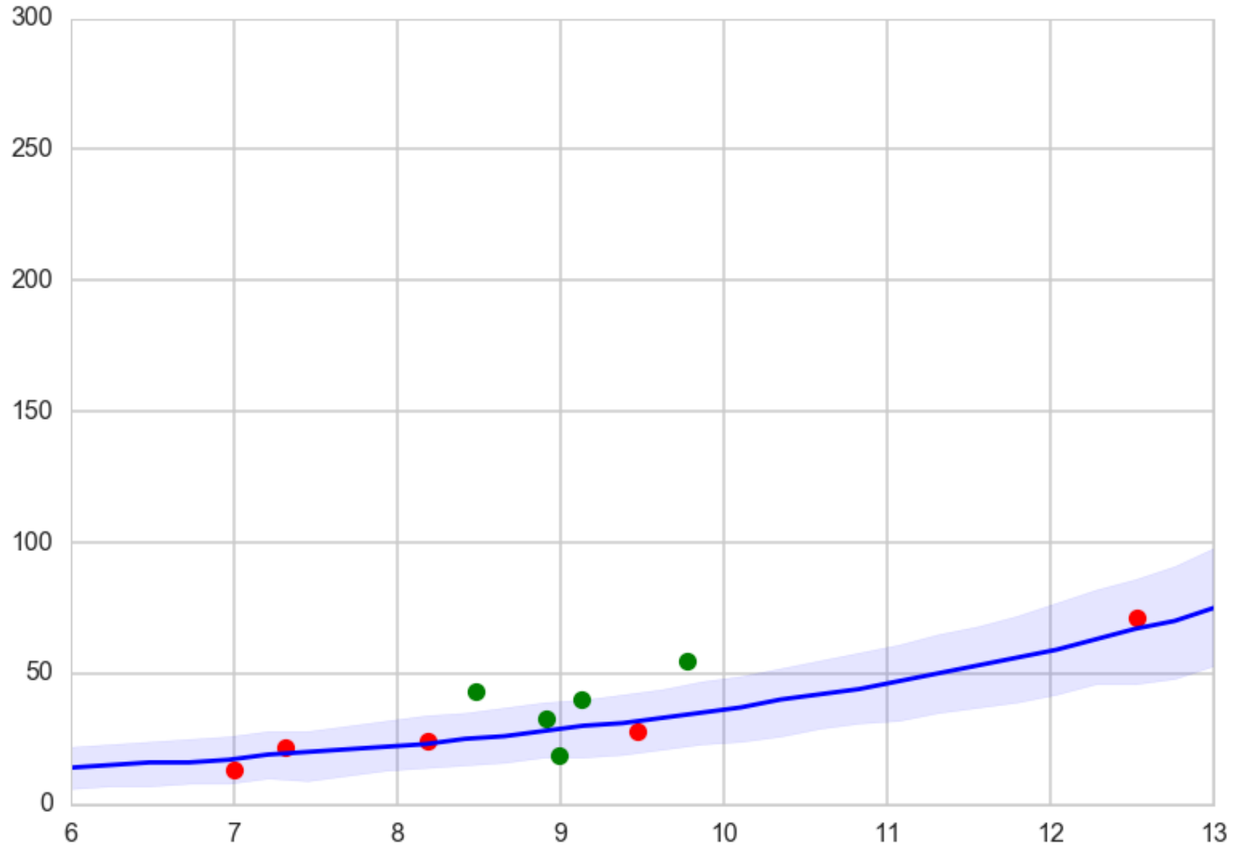
We will use the "Akaike" weights from the WAIC.

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.



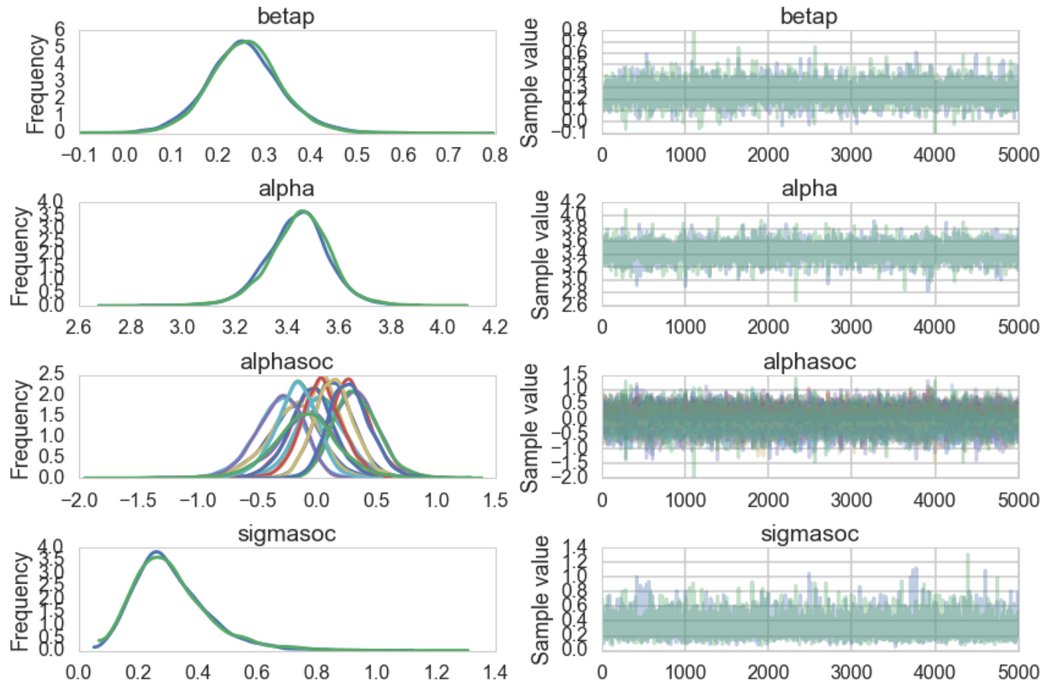
Overdispersion for only p

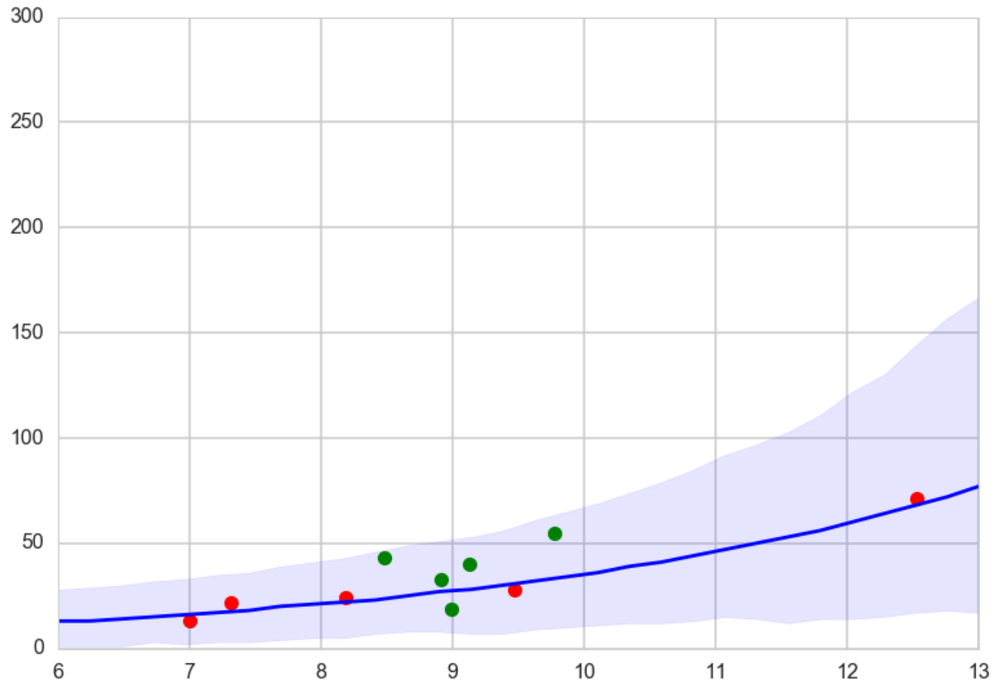


Varying hierarchical intercepts model

```
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