Lecture 21

Model Comparison, Ensembling and Bayesian Workflow

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.

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

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

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

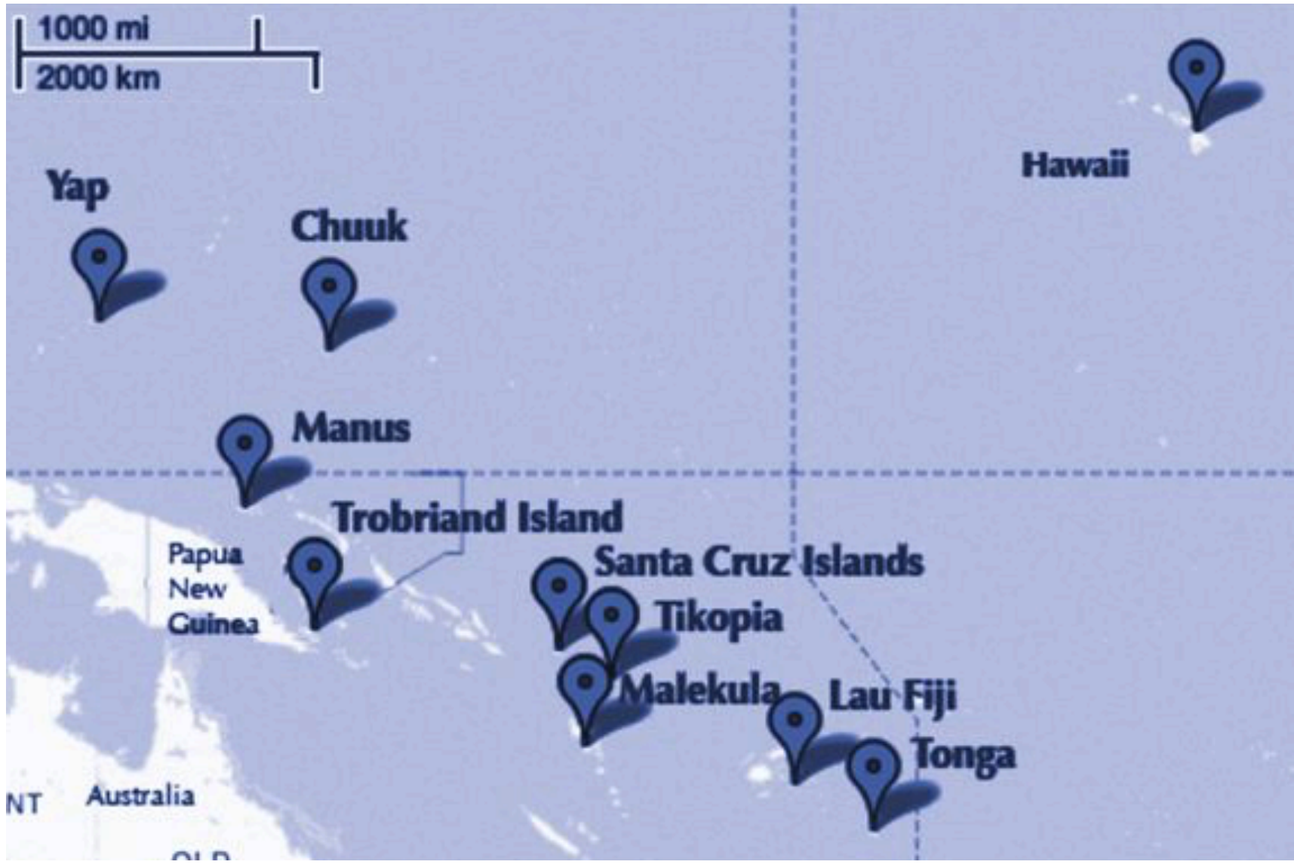
$$\bar{u}(\hat{a}(x^*)) = \int dy^* \, u(\hat{a}, y^*) \, p(y^* | x^*, D, M)$$

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

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

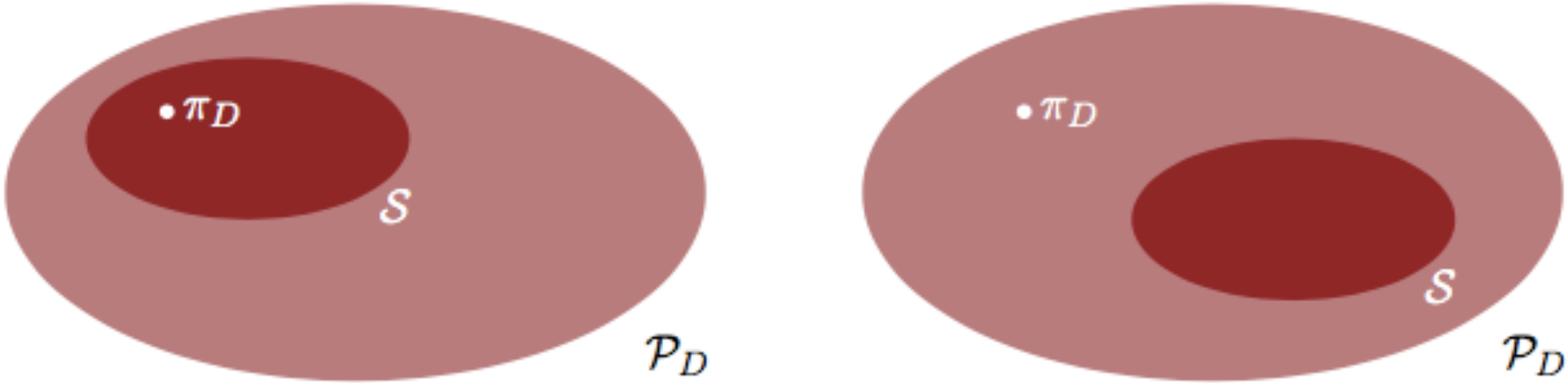


Were the contacts really needed?

Let us compare models:

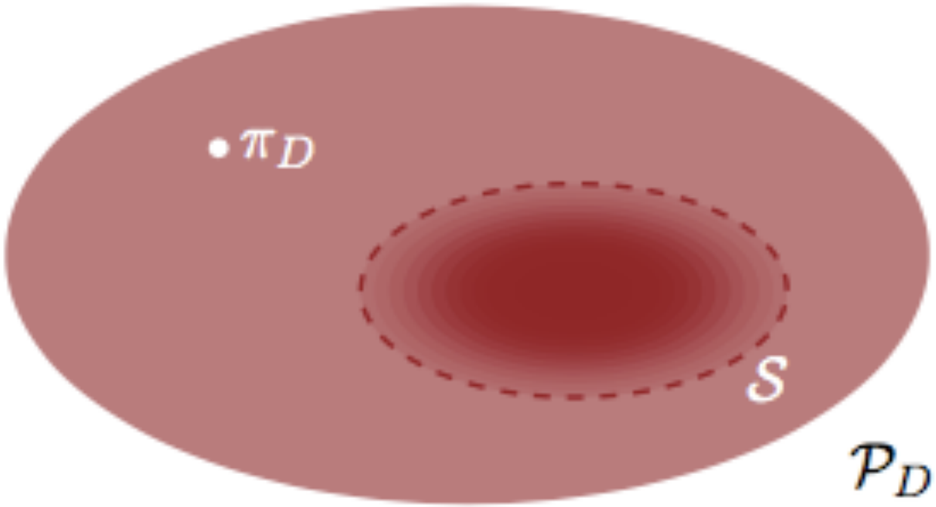
```
m2c_onlyic: loglam = alpha
m2c_onlyc: loglam = alpha +  betac*df.clevel
m2c_onlyp: loglam = alpha + betap*df.logpop_c
m2c_nopc: loglam = alpha + betap*df.logpop_c + betac*df.clevel
m1c: loglam = alpha + betap*df.logpop_c + betac*df.clevel + betapc*df.clevel*df.logpop_c
```

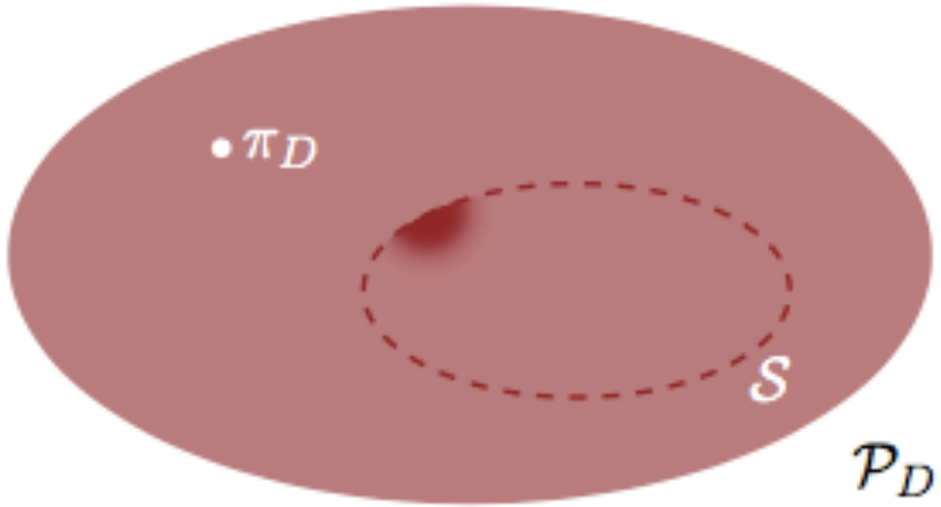
Bayesian Inference works in the small world



(some times includes the true generating process p_{tb})

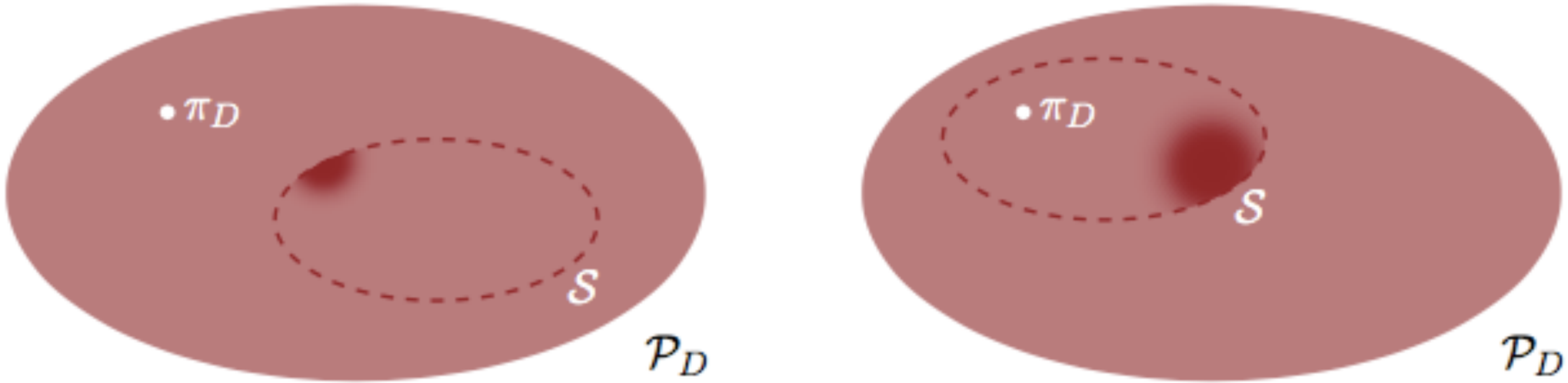
Inference in the small world





we go from prior to posterior

Bias and Variance: Overfitting



Overfitting can occur even if the small world includes the true data generating process p_{tb}.

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 empirical estimate

Deviance

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

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

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

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:

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

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.

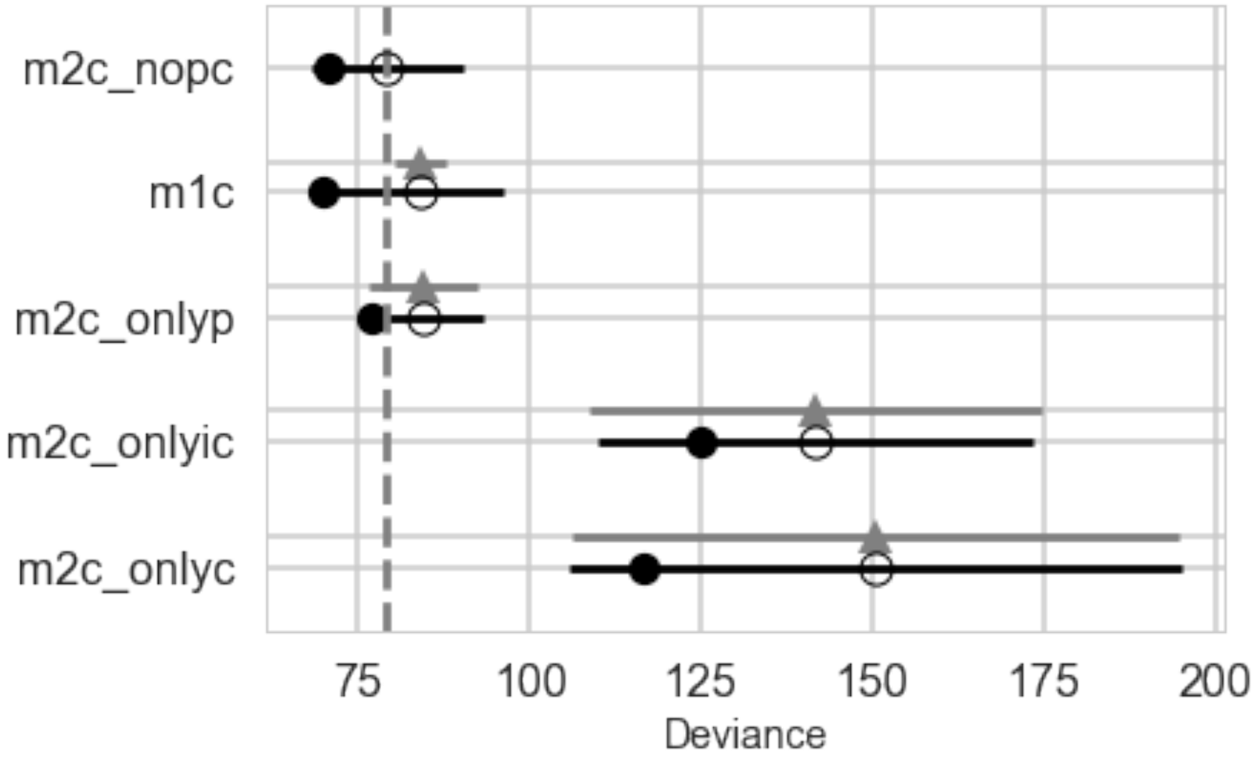
— McElreath

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



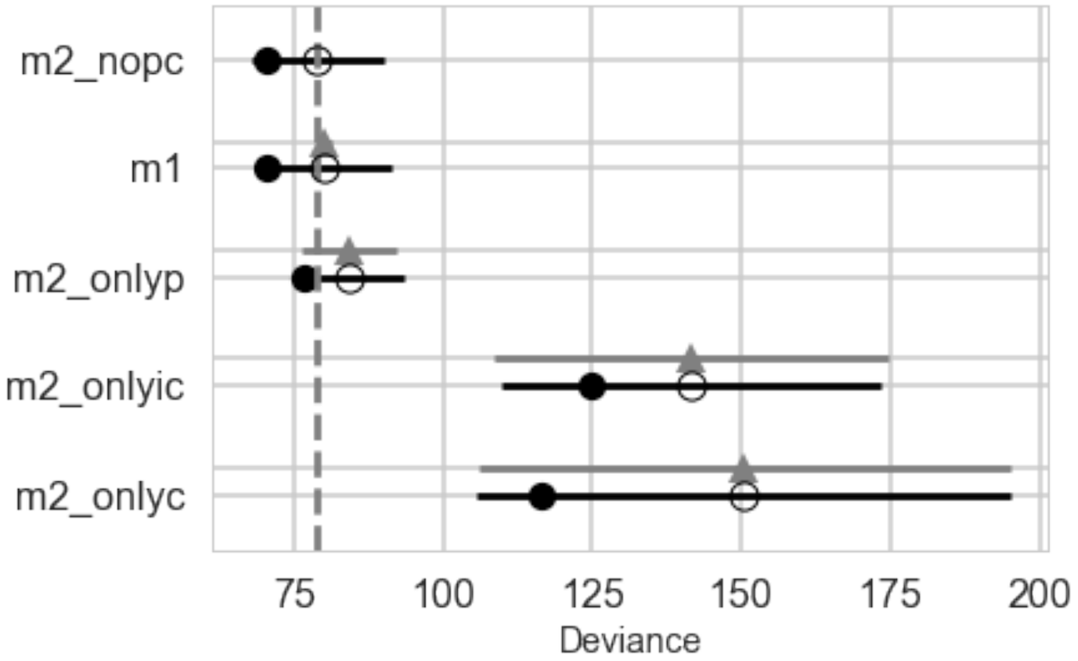
- dWAIC is the difference between each WAIC and the lowest WAIC.
- SE is the standard error of the WAIC estimate.
- dSE is the standard error of the difference in WAIC between each model and the top-ranked model.

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

read each weight as an estimated

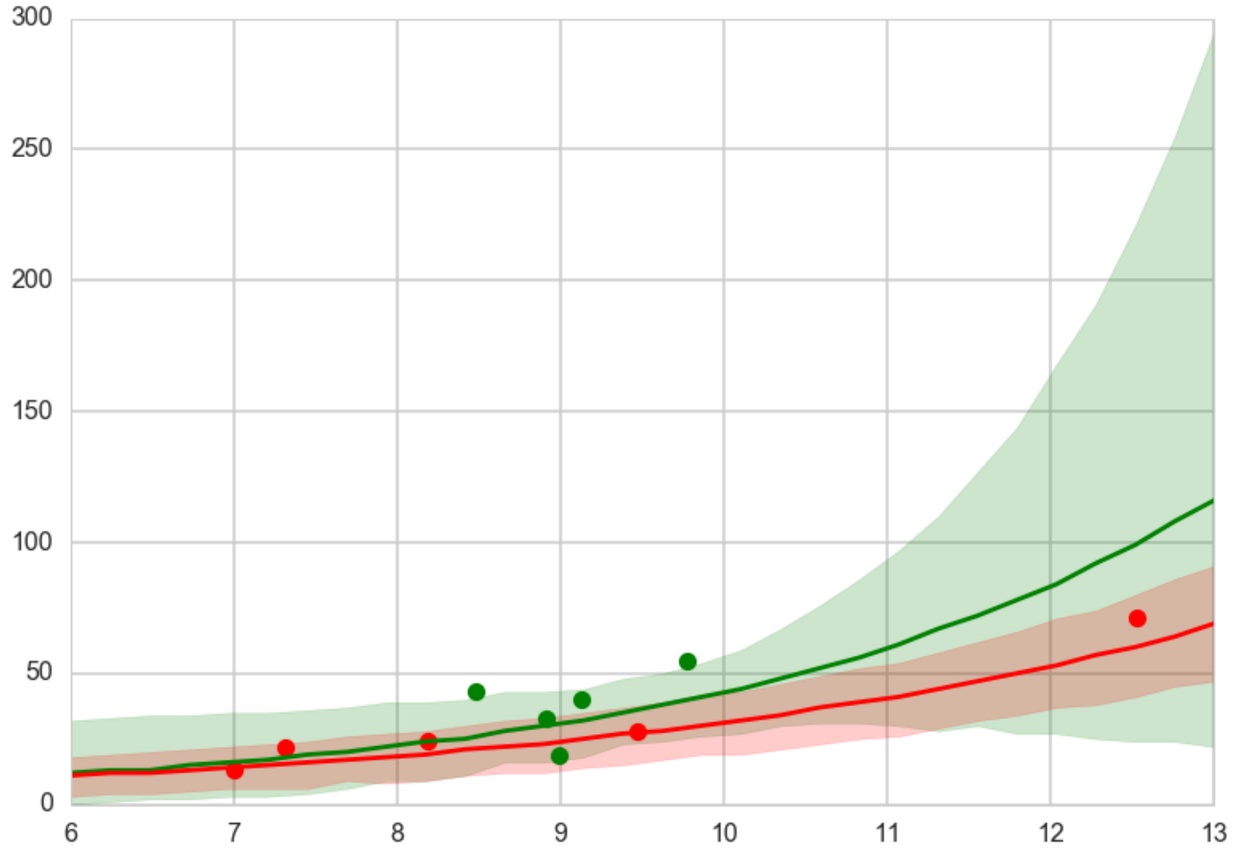
Uncentered

	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



Bayes Theorem in model space

$$p(M_k|D) \propto p(D|M_k)p(M_k)$$

where:

$$p(D|M_k) = \int d\theta_k \, p(y|\theta_k, M_k)p(\theta_k|M_k)$$

is the marginal likelihood under each model. Can use these "Bayes Factors" to compare but high sensitivity to prior.

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 respect 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. This is called `pseudo-BMA`

- BMA is appropriate in the M-closed case, which is when the true generating process is one of the models
- what we will use here is to estimate weights by the WAIC, following McElreath (pseudo-BMA)
- But see Yao et. al. which claims log-score stacking is better. Implemented in pymc3

$$\max_w \frac{1}{n}\sum_{i=1}^{n}\log\sum_{k=1}^{K} w_k p(y_i|y_{-i}, M_k), \quad s.t. \quad w_k \geq 0, \quad \sum_{k=1}^{K} w_k = 1.$$

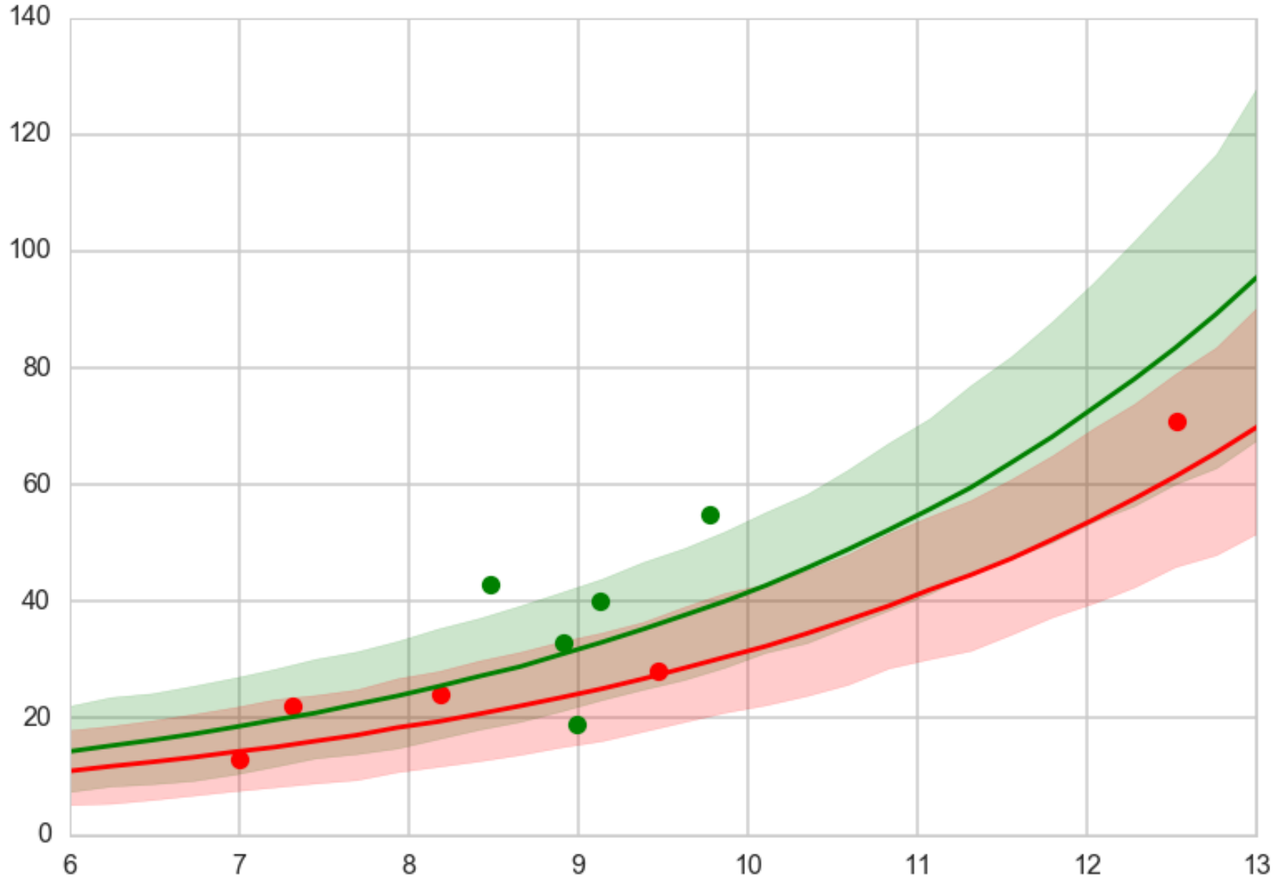
Pseudo BMA vs stacking

name	WAIC	pWAIC	dWAIC	weight	SE	dSE	warning
m2c_nopc	79.06	4.24	0	0.87	11.06	0	1
m1c	84.09	7.05	5.04	0.07	12.19	3.77	1
m2c_onlyp	84.43	3.75	5.37	0.06	8.94	7.93	1
m2c_onlyic	141.65	8.38	62.6	0	31.7	32.84	1
m2c_onlyc	150.44	16.94	71.38	0	44.67	44.44	1

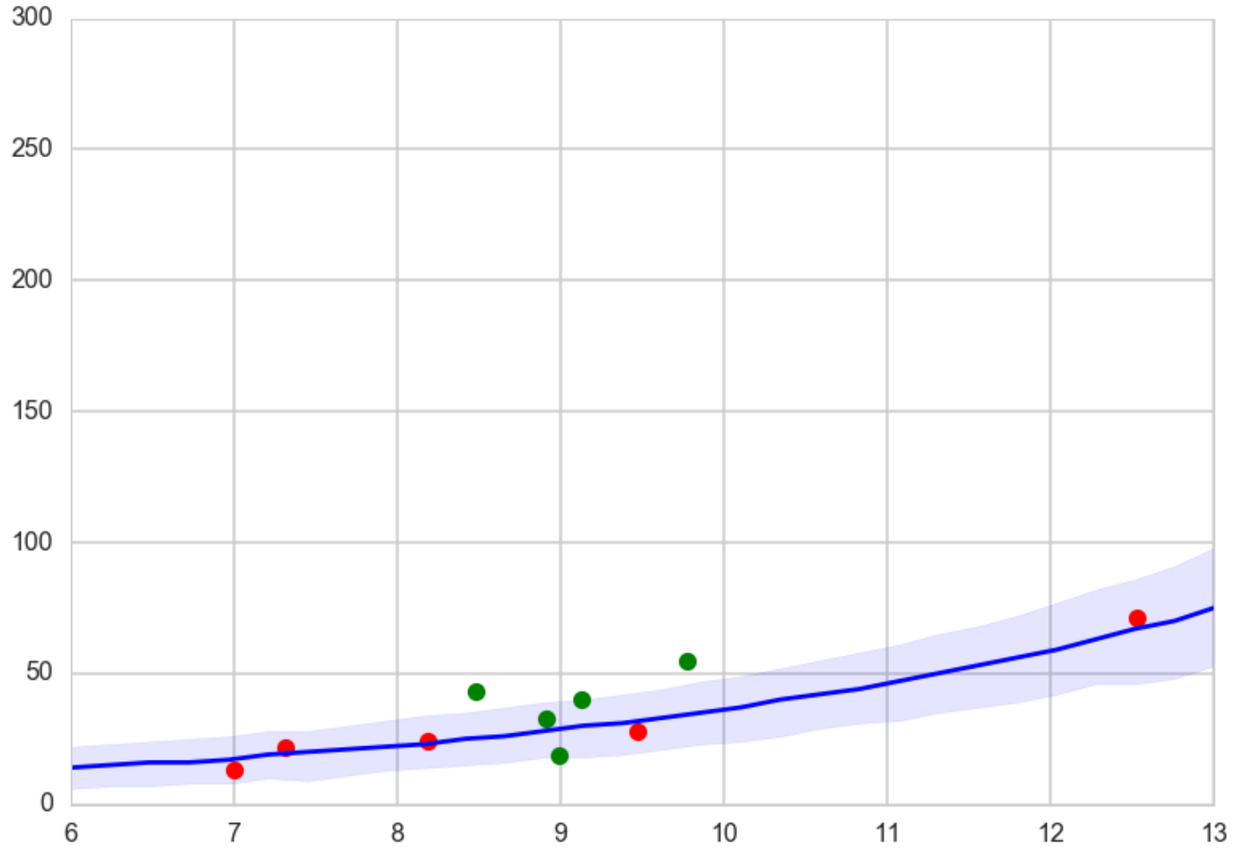
name	WAIC	pWAIC	dWAIC	weight	SE	dSE	warning
m2c_nopc	79.06	4.24	0	0.76	11.06	0	1
m1c	84.09	7.05	5.04	0	12.19	3.77	1
m2c_onlyp	84.43	3.75	5.37	0.24	8.94	7.93	1
m2c_onlyic	141.65	8.38	62.6	0	31.7	32.84	1
m2c_onlyc	150.44	16.94	71.38	0	44.67	44.44	1

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.



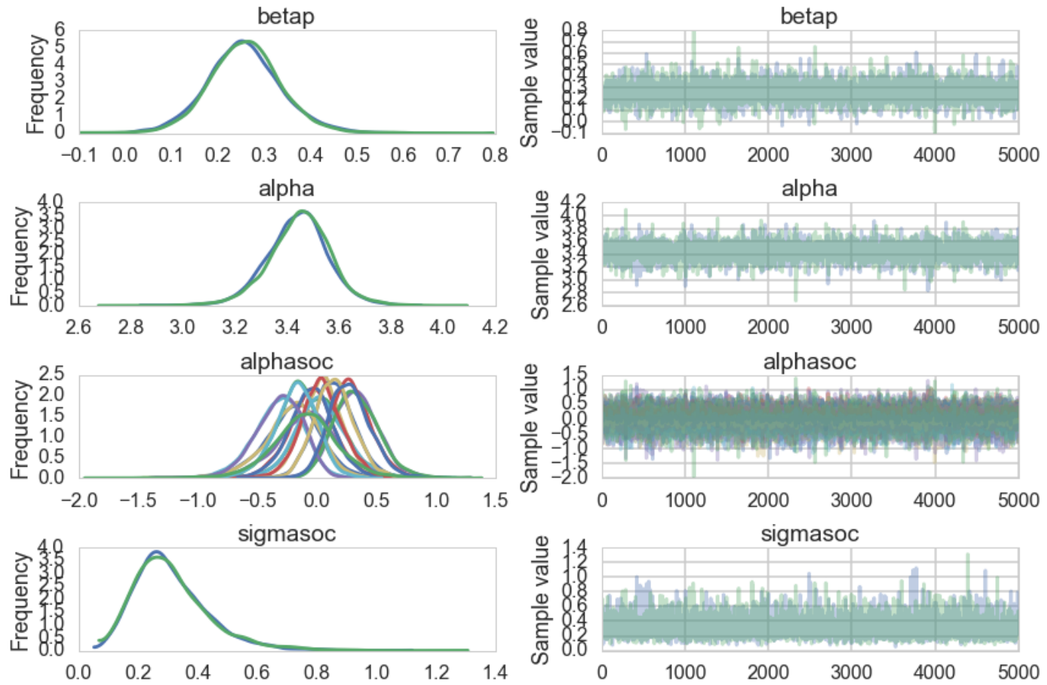
Overdispersion for only p

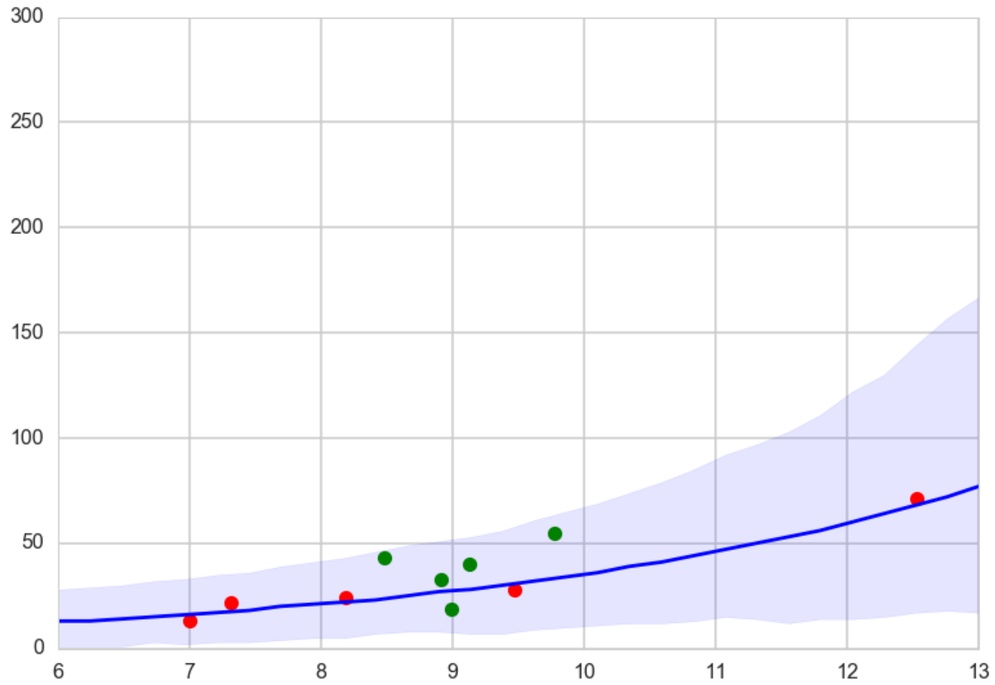


Varying hierarchical intercepts model

```
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

cross-validation

- estimate the out-of-sample risk as an average, thus gaining robustness to odd validation sets
- providing some measure of uncertainty on the out-of-sample performance.
- less data to fit so biased models
- we are not talking here about cross-validation to do hyperparameter optimization

hyperparameter fitting

- part of the prior specification, uses entire data set
- or we can use empirical bayes, and use entire data set.
- faster than cross-val but prone to model mis-specification
- but EB is not a model selection procedure

LOOCV

- The idea here is that you fit a model on N-1 data points, and use the Nth point as a validation point. Clearly this can be done in N ways.
- the N-point and N-1 point posteriors are likely to be quite similar, and one can sample one from the other by using importance sampling.

$$E_f[h] = \frac{\sum_s w_s h_s}{\sum_s w_s} \text{ where } w_s = f_s/g_s.$$

An aside: Importance sampling

The basic idea behind importance sampling is that we want to draw more samples where $h(x)$, a function whose integral or expectation we desire, is large. In the case we are doing an expectation, it would indeed be even better to draw more samples where $h(x)f(x)$ is large, where $f(x)$ is the pdf we are calculating the integral with respect to.

Unlike rejection sampling we use all samples!!

$$E_f[h] = \int_V f(x)h(x)dx.$$

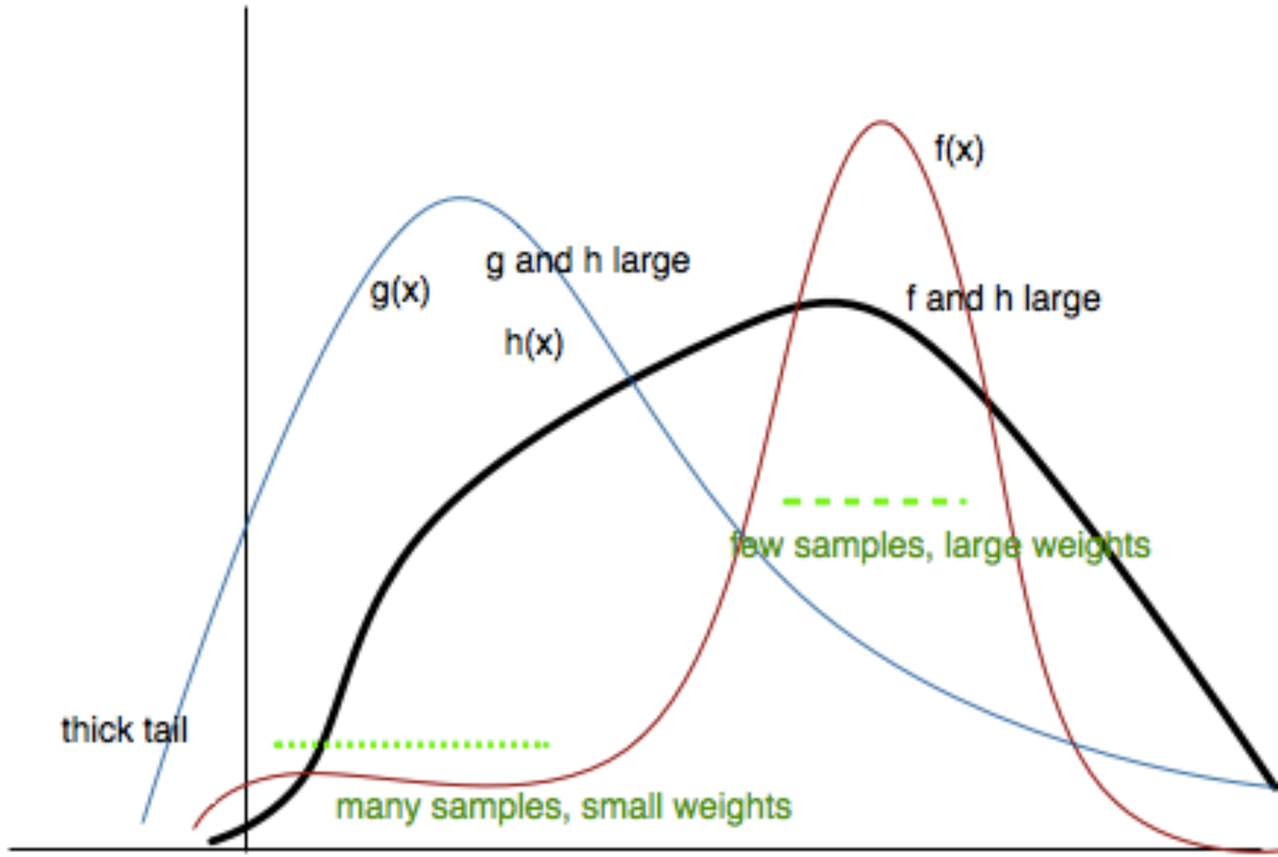
Choosing a proposal distribution $g(x)$:

$$E_f[h] = \int h(x)g(x)\frac{f(x)}{g(x)}dV$$

$$E_f[h] = \lim_{N\to\infty}\frac{1}{N}\sum_{x_i\sim g(.)} h(x_i)\frac{f(x_i)}{g(x_i)}$$

If $w(x_i) = f(x_i)/g(x_i)$:

$$E_f[h] = \lim_{N\to\infty}\frac{1}{N}\sum_{x_i\sim g(.)} w(x_i)h(x_i)$$



Variance reduction

Usually: $\hat{V} = \frac{V_f[h(x)]}{N}$

Importance Sampling: $\hat{V} = \frac{V_g[w(x)h(x)]}{N}$

Minimize $V_g[w(x)h(x)]$ (make 0), if:

$$w(x)h(x) = C \implies f(x)h(x) = Cg(x), ...$$

Gives us $g(x) = \dfrac{f(x)h(x)}{E_f[h(x)]}$

To get low variance, we must have $g(x)$ **large where the product** $f(x)h(x)$ **is large**.

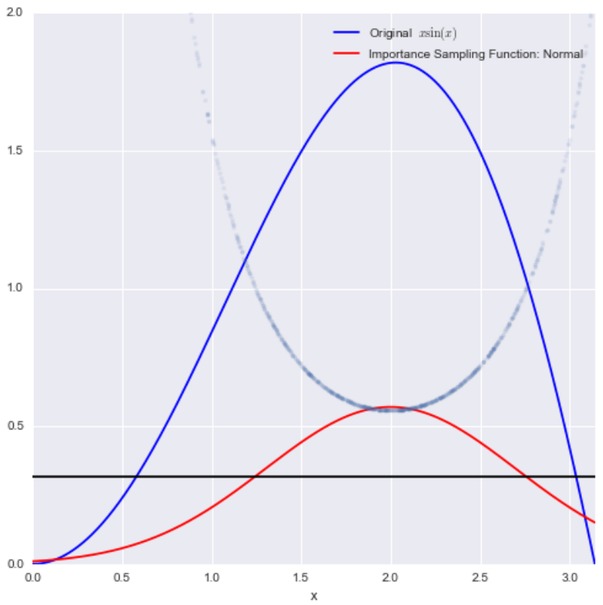
Or, $\dfrac{g(x)}{f(x)}$ ought to be large where $h(x)$ is large. This means that, as we said earlier, choose more samples near the peak.

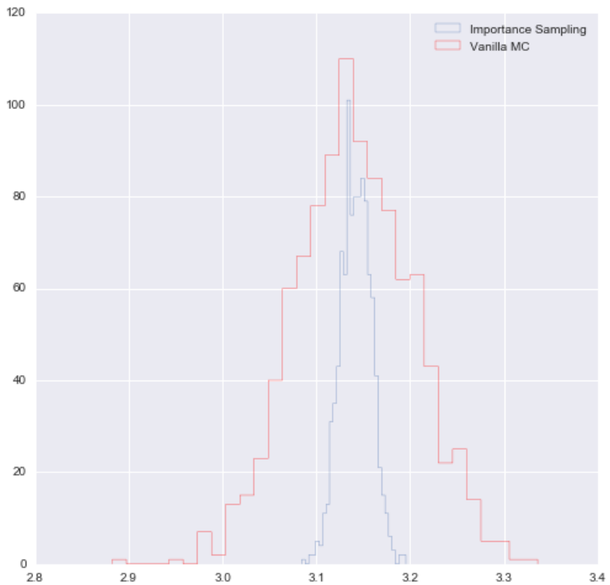
Example: integral of x sin(x)

```
mu = 2;
sig =.7;
f = lambda x: np.sin(x)*x
infun = lambda x: np.sin(x)-x*np.cos(x)
p = lambda x: (1/np.sqrt(2*np.pi*sig**2))*np.exp(-(x-mu)**2/(2.0*sig**2))
normfun = lambda x:  norm.cdf(x-mu, scale=sig)
# range of integraion
xmax =np.pi
xmin =0
N =1000 # Number of draws

# Just want to plot the function
x=np.linspace(xmin, xmax, 1000)
plt.plot(x, f(x), 'b', label=u'Original  $x\sin(x)$')
plt.plot( x, p(x), 'r', label=u'Importance Sampling Function: Normal')
plt.plot(x, np.ones(1000)/np.pi,'k')
xis = mu + sig*np.random.randn(N,1);
plt.plot(xis, 1/(np.pi*p(xis)),'.', alpha=0.1)

# IMPORTANCE SAMPLING
Iis = np.zeros(1000)
for k in np.arange(0,1000):
    # DRAW FROM THE GAUSSIAN mean =2 std = sqrt(0.4)
    xis = mu + sig*np.random.randn(N,1);
    xis = xis[ (xis<xmax) & (xis>xmin)] ;
    # normalization for gaussian from 0..pi
    normal = normfun(np.pi)-normfun(0);
    Iis[k] =np.mean(f(xis)/p(xis))*normal;
```






```
Exact solution is:  3.14159265359
Mean basic MC estimate:  3.14068341144
Standard deviation of our estimates:  0.0617743877206
Mean importance sampling MC estimate:  3.14197268362
Standard deviation of our estimates:  0.0161935244302
```

Fit the full posterior once. Then we have

$$w_s = \frac{p(\theta_s|y_{-i})}{p(\theta_s|y)} \propto \frac{1}{p(y_i|\theta_s, y_{-i})}$$

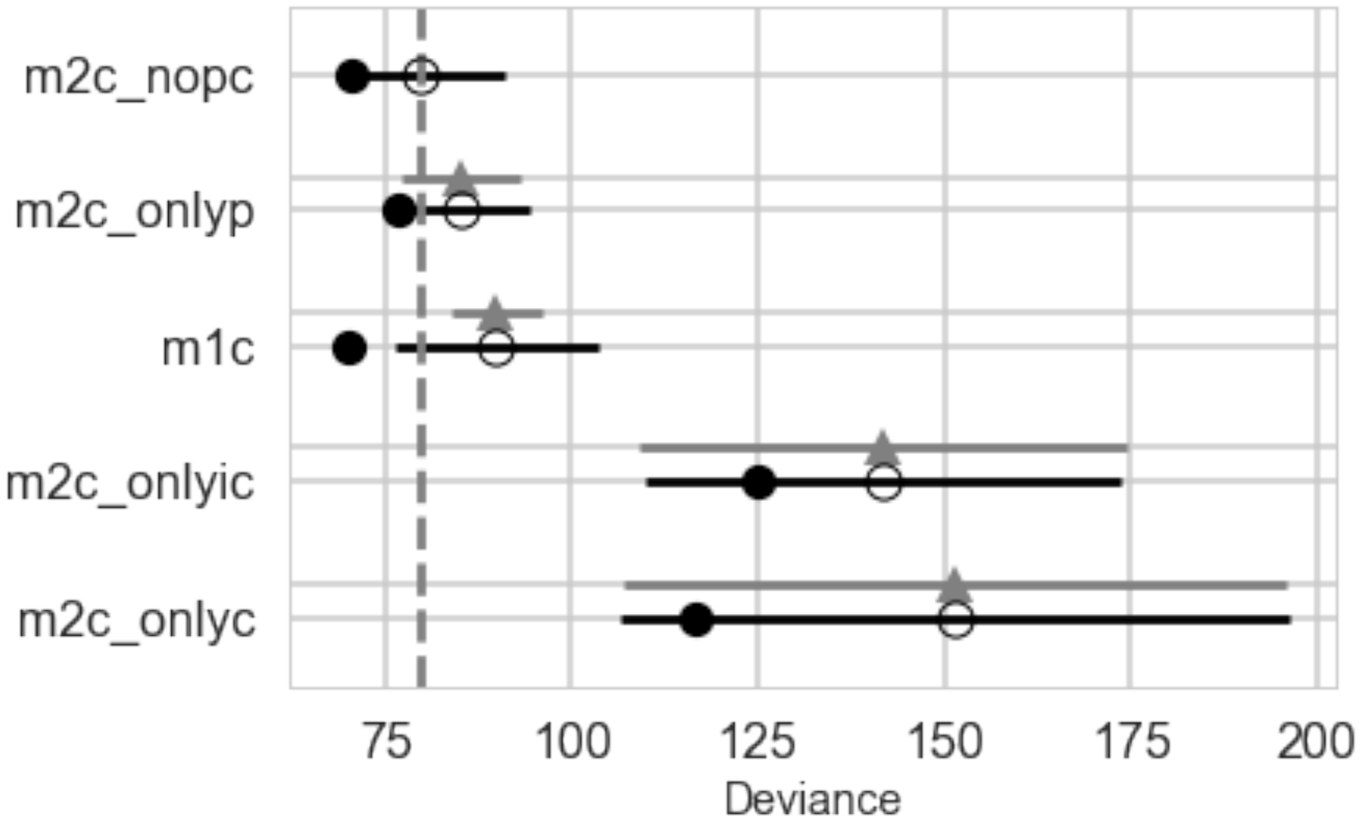
- the importance sampling weights can be unstable out in the tails.
- importance weights have a long right tail, pymc (`pm.loo`) fits a generalized pareto to the tail (largest 20% importance ratios) for each held out data point i (a MLE fit). This smooths out any large variations.

$$elpd_{loo} = \sum_i log(p(y_i|y_{-i}))$$

$$= \sum_i log\left(\frac{\sum_s w_{is} p(y_i|\theta_s)}{\sum_s w_{is}}\right)$$

over the training sample.

Oceanic tools LOOCV



What should you use?

1. LOOCV and WAIC are fine. The former can be used for models not having the same likelihood, the latter can be used with models having the same likelihood.

2. WAIC is fast and computationally less intensive, so for same-likelihood models (especially nested models where you are really performing feature selection), it is the first line of attack

3. One does not always have to do model selection. Sometimes just do posterior predictive checks to see how the predictions are, and you might deem it fine.

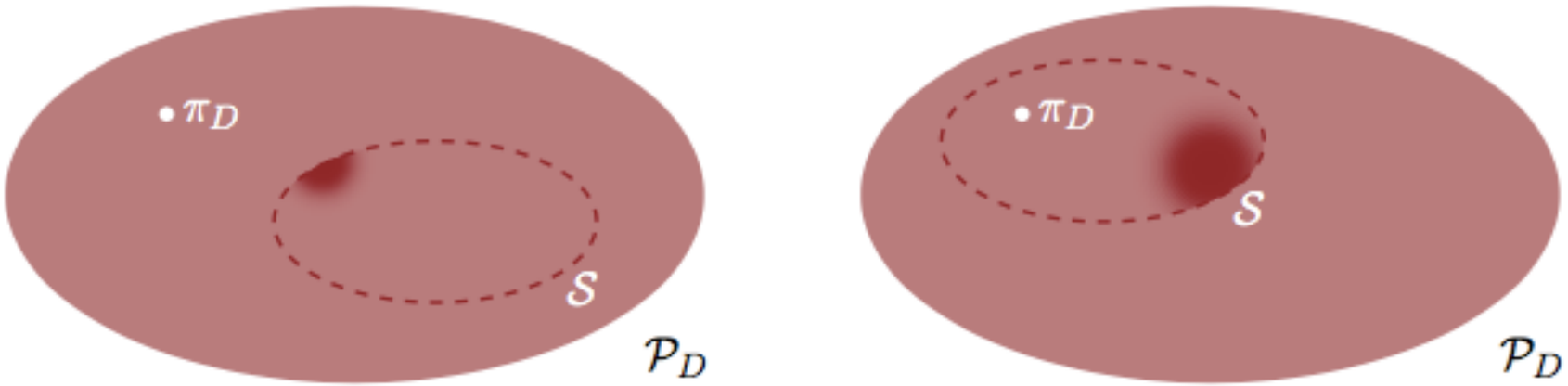
4. For hierarchical models, WAIC is best for predictive performance within an existing cluster or group. Cross validation is best for new observations from new groups

Bayesian Workflow

$$p(\theta|y) = \frac{p(\theta, y)}{p(y)}$$

$$= \frac{p(y \mid \theta)p(\theta)}{p(y)}$$

Think of the prior generatively AND predictively



Bias can come from a prior, but do not construct a prior to allow for overfitting (draws far away from good place). Too many heavy tails caqn be bad.

Model Calibration

Think about the **Bayesian Joint distribution**.

$$p(\theta, y) = p(y \mid \theta)p(\theta)$$

The prior predictive:

$$p(y) = \int d\theta \, p(\theta, y) = \int d\theta \, p(y \mid \theta)p(\theta)$$

How to choose priors?

- mild regularization
- un-informativity
- sensible parameter space
- should correspond to scales and units of process being modeled
- we should calibrate to them

Generate Artificial data sets

- from fixed params, but even better, from priors
- $\tilde{\theta} \sim p(\theta)$
- $\tilde{y} \sim p(y \mid \tilde{\theta})$
- callibrate inferences or decisions by analysing this data
- $U(a) = \int d\tilde{\theta} d\tilde{y} p(\tilde{y}, \tilde{\theta}) U(a(\tilde{y}), \tilde{\theta})$

Now fit a posterior to each generated dataset

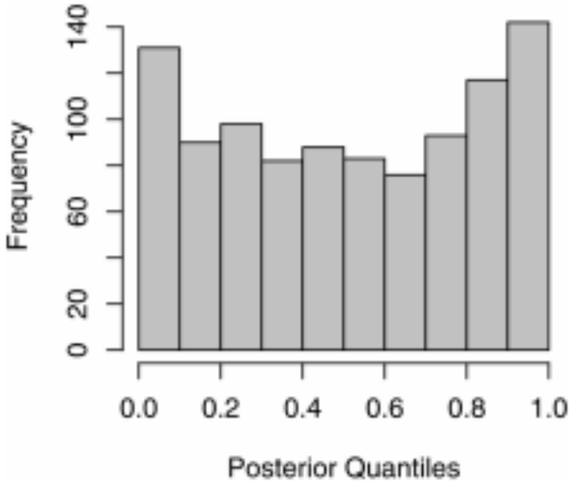
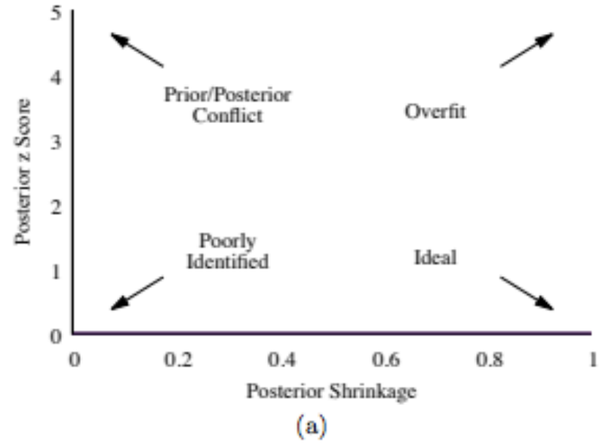


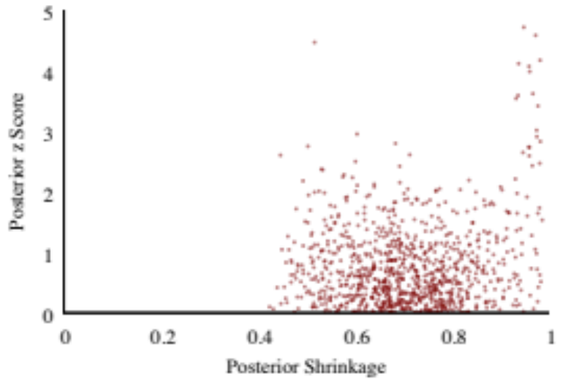
Figure 3. An example of posterior quantiles q from software with error. An effective summary for detecting the error should emphasize quantiles near 0 or 1, such as $h(q) = (\Phi^{-1}(q))^2$.

- see Cook et al
- take each $\tilde{y}$
- get a $\theta \mid \tilde{y}$ posterior
- find the rank of $\tilde{\theta}$ in "its" posterior
- a histogram of ranks should be uniform- this tests our sampling software

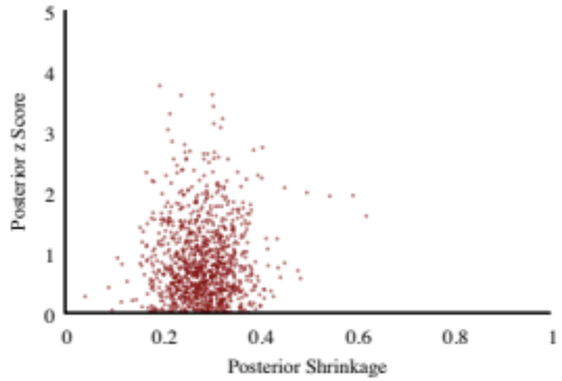
Sensitivity of posterior to range allowed by prior



(a)



(b)



(c)

$$z_n = \left| \frac{\mu_n(\theta_n|\tilde{y}) - \tilde{\theta}_n}{\sigma_n(\theta_n|\tilde{y})} \right|$$

$$s_n = 1 - \frac{\sigma_n(\theta_n|\tilde{y})^2}{\tau_n(\tilde{y})^2}$$

where μ and σ are generated-posterior quantities and τ is a prior one, and n indexes the parameters

Then move to the REAL DATA posterior

- now we do posterior predictive checks
- the prior checks have specified possible data distributions that can be generated
- the posterior predictive ought to be a subset of these. If not our model is mis-specified
- this may seem strange as we didnt think priors are data generating
- they are not but are defined with respect to the likelihood

The Workflow (from Betancourt, and Savage)

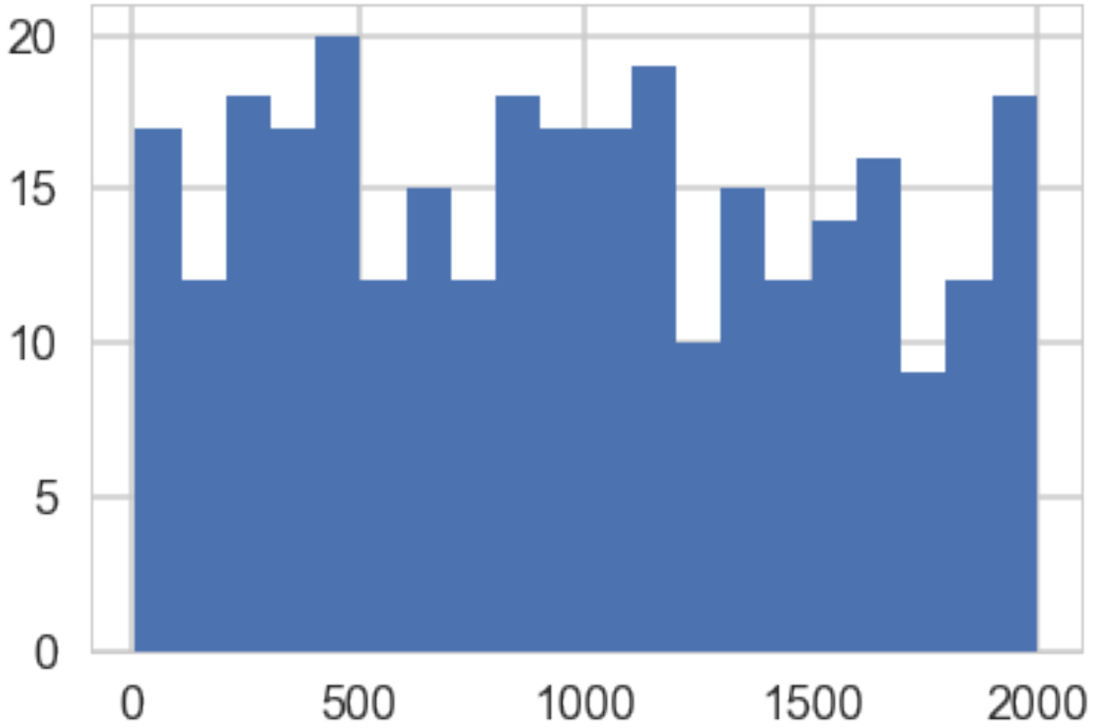
Prior to Observation

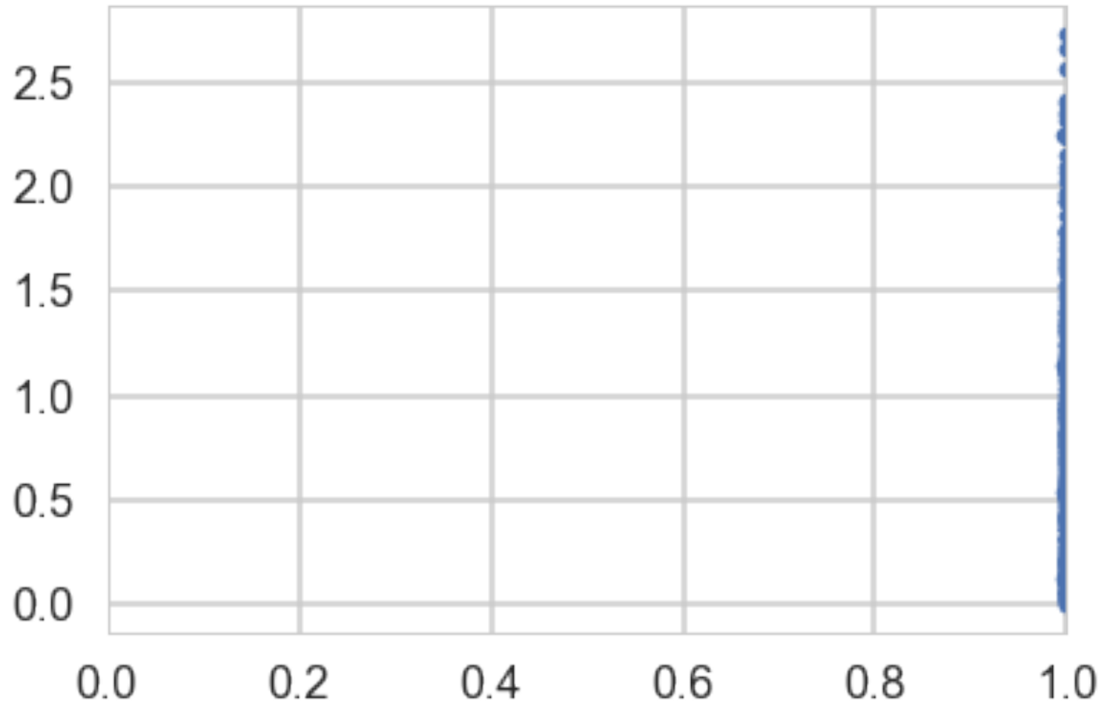
1. Define Data and interesting statistics
2. Build Model
3. Analyze the joint, and its data marginal (prior predictive) and its summary statistics
4. fit posteriors to simulated data to calibrate
    - check sampler diagnostics, and correlate with simulated data
    - use rank statistics to evaluate prior-posterior consistency
    - check posterior behaviors and behaviors of decisions

Posterior to Observation

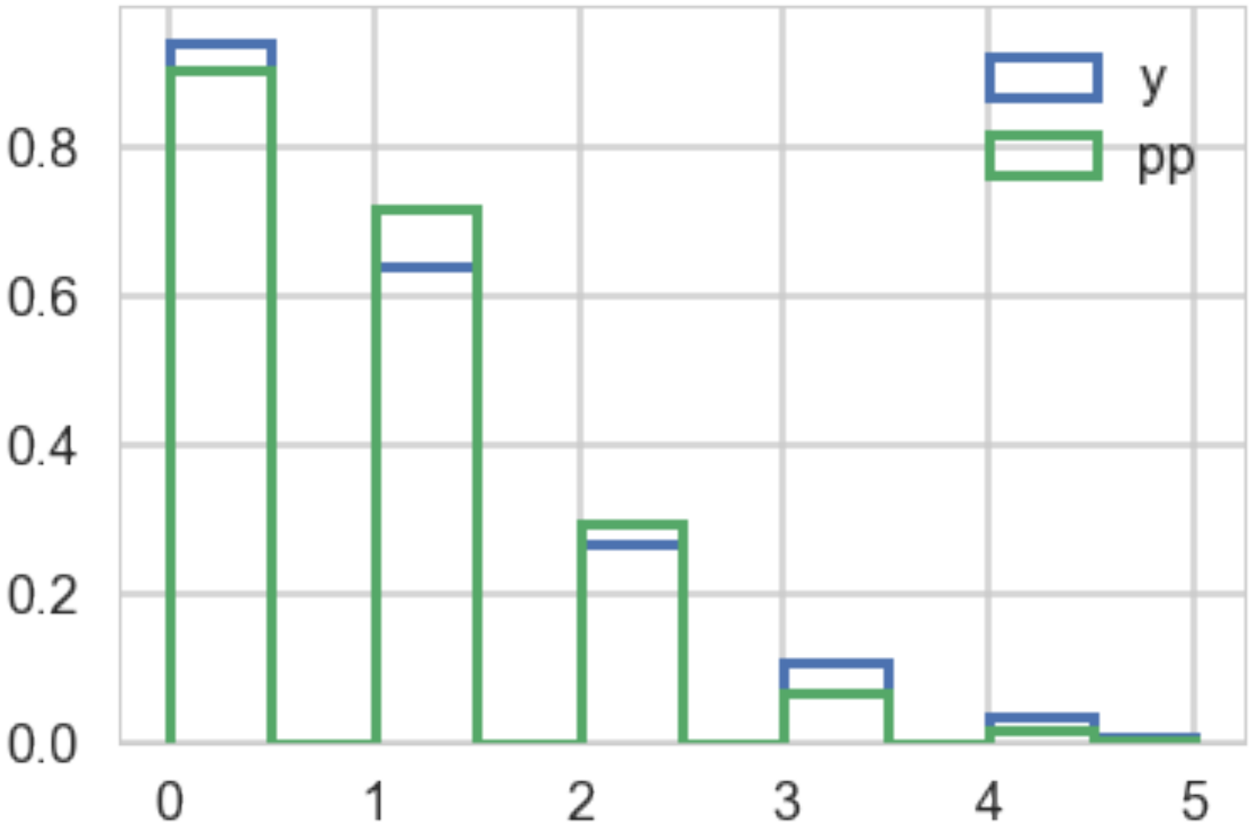
1. Fit the Observed Data and Evaluate the fit
    - check sampler diagnostics, poor performance means generative model not consistent with actual data
2. Analyze the Posterior Predictive Distribution
    - do posterior predictive checks, now comparing actual data with posterior-predictive simulations
    - consider expanding the model
3. Do model comparison
    - usually within a nested model, but you might want to apply a different modeling scheme, in which case use loo
    - you might want to ensemble instead

Drunk Monks example part 1, pre-obs





Drunk Monks, post-obs



- specify $\lambda \sim HalfN(0, 4)$ instead of the crazy $N(0, e^{100})$ we had earlier
- domain knowledge: *A survey of Abbey Heads has told us, that the most a monk could produce, ever, was 10 manuscripts in a day.*
- $max(\lambda + 3\sqrt{\lambda}) < 10$, `5+3*np.sqrt(5)=11.7`
- `halfnorm.ppf(0.99, loc=0,scale=4)=10.3`
- pp check shows need for 0 inflation

WHEN BAYES

from Jim Savage

Jim Savage
@jim_savage_

Following

A test for whether a problem requires Bayesian methods:
1. Is there information that is not in your data about population-level unknowns?
2. Do you need coherent uncertainty?
3. Are you combining complex models and want uncertainty to percolate through?

Yes to any? Bayes it.

11:49 AM - 9 Apr 2018

11 Retweets **90** Likes

3 11 90

Tweet your reply

Jake Mortenson @jm0rt · 20h
Replying to @jim_savage_
Have been looking for an excuse to do Bayesian stuff in a tax policy research setting. But isn't there also 4, do you have some sparsely populated (and interesting) bins? The answer to 1 and 2 are virtually always yes, but have avoided so far because our data are typically yuge.

1

Jim Savage @jim_savage_ · 20h
I couldn't add 4) You want to generalize to new populations (post-strat) & so want to estimate sub-group effects, but your sample has small N in those sub-groups, there's a lot of value in hierarchical priors.

Are we saying the same thing?

1 3

Jake Mortenson @jm0rt · 19h
That was part of my point, the other part being (perhaps out of my depth): with large data the benefits from incorporating priors may not be large (fixed effects may be sufficient, depending on parameters of interest), and also computation might be time-expensive. Sound right?

1 1

Jim Savage @jim_savage_ · 19h
See rule 1 though: if there is information your enormous data doesn't contain about the unknown of interest (in the population--which for most purposes is a future population) then there might still be value in having priors. Turkey before thanksgiving story.

1

Noah Motion @statmodcitizen · 22h
Replying to @jim_savage_
My intuition is that the answer to (2) is always "yes", but I may be misunderstanding what you mean by the question...

1

Jim Savage @jim_savage_ · 22h
Strictly yes, if computation and analyst time has no cost. Business maximize profit, not correctness.

4

Frank Harrell @f2harrell · 18h
Replying to @jim_savage_
Nice. I'd simply say "Does your problem require statistical inference?". If yes, Bayes it. Among other things this solves is that inference is exact. Most frequentist analyses are approximations, other than the ordinary linear model and a few others.

11