Lecture 7: advanced methods – hierarchical models, ODEs, and model selection

Ben Lambert[1]
`ben.lambert@some.ox.ac.uk`

[1]Somerville College
University of Oxford

Wednesday 7th December, 2016

Lecture outcomes

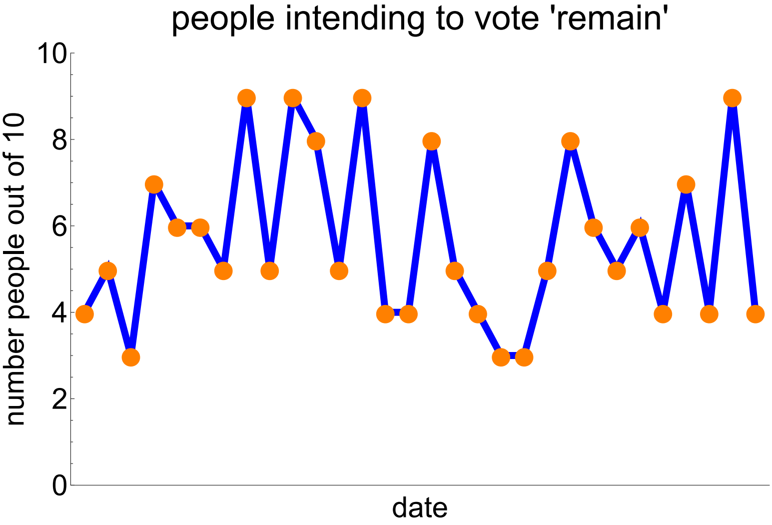
By the end of this lecture you should:

1. Understand what is meant by a hierarchical model.
2. Appreciate some of the benefits of using hierarchical models.
3. Know how to think Bayesianly about ordinary differential equation models (ODEs), and use Stan to code up simple examples.
4. Understand how to think Bayesianly about model selection.

Example: EU referendum

- Imagine a dystopian future where the UK decides to hold a referendum on its membership of the EU.
- Have data from 20 polls on EU membership carried out over the past month (fake data.)
- Polls conducted by a range of different agencies.
- The sample size of each poll is 10.
- Ultimately want to use model to forecast the result of the EU referendum.

EU referendum: data



EU referendum: complete pooling model

Suppose that the we assume that the data across all polls are:

- Independent.
- Identically-distributed.

Sample size is fixed and data are discrete $\implies$ binomial likelihood:

$$Pr(X = X_i|\theta) = \theta^{X_i}(1-\theta)^{10-X_i} \tag{1}$$

where X_i is the number of people voting 'remain' in poll i, and θ is the probability that a randomly-chosen person will vote ‘remain’.

Important: we are assuming that θ is the same across all polls.

EU referendum: complete pooling model in Stan

```
data {
  int<lower=1> K; ## number of polls
  int<lower=0> X[K]; ## numbers voting 'remain'
  int<lower=1> N; ## sample size
}
parameters {
  real<lower=0,upper=1> theta;
}
model {
  for (i in 1:K){
    X[i] ~ binomial(N,theta);
  }
}
```

Implicitly $\implies$ that we are using a uniform prior on (0,1) for theta.

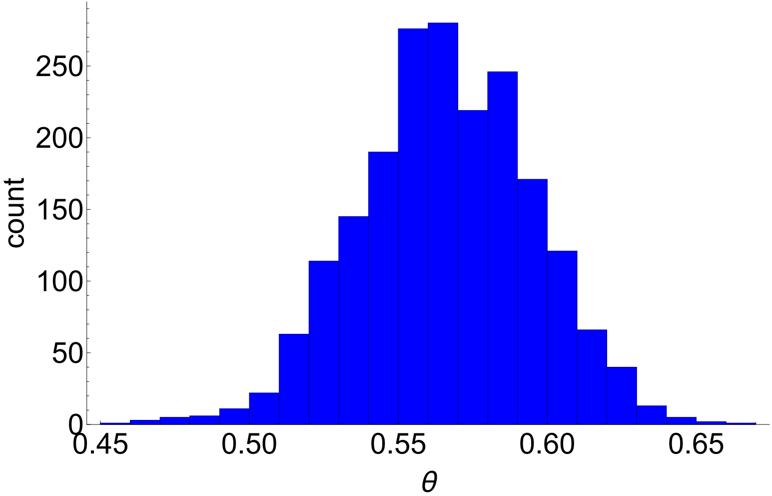
EU referendum: complete pooling model results

```
print(fit,probs = c(0.25, 0.5, 0.75))

## Inference for Stan model: lec6_euBinomialHomgeneousSimple.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500..
##
##            mean se_mean   sd     25%     50%     75% n_eff Rhat
## theta      0.57    0.00 0.03    0.55    0.57    0.59   656 1.01
## lp__    -206.59    0.02 0.65 -206.75 -206.32 -206.17   774 1.00
```

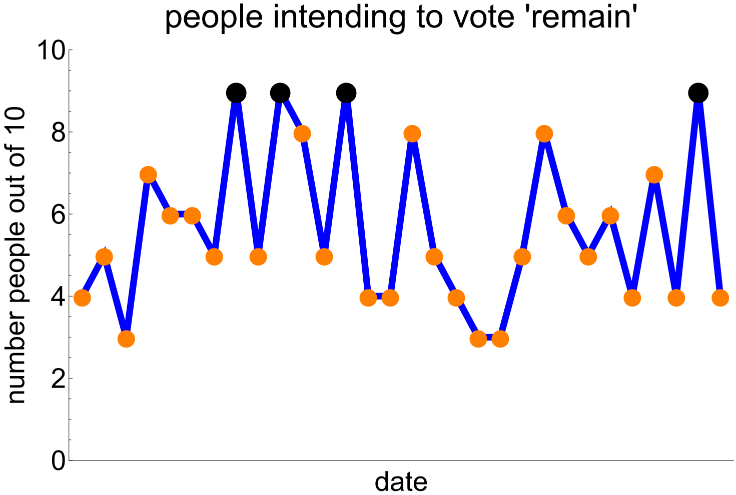
$\Longrightarrow$ all looks ok.

EU referendum: complete pooling model results



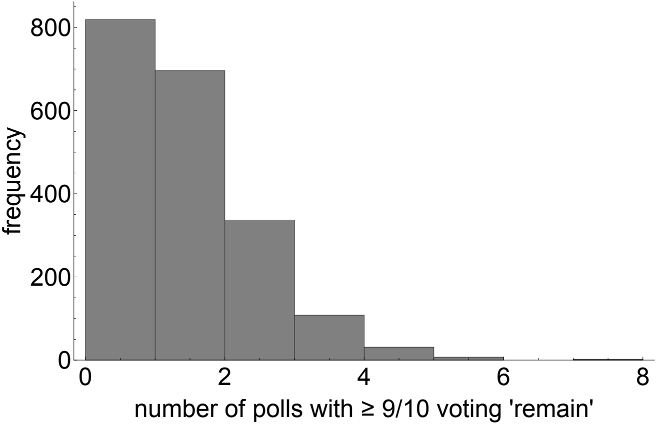
EU referendum: complete pooling model posterior predictive checks

Count number of times that 9 or more people vote 'remain', and find 4/30 cases.



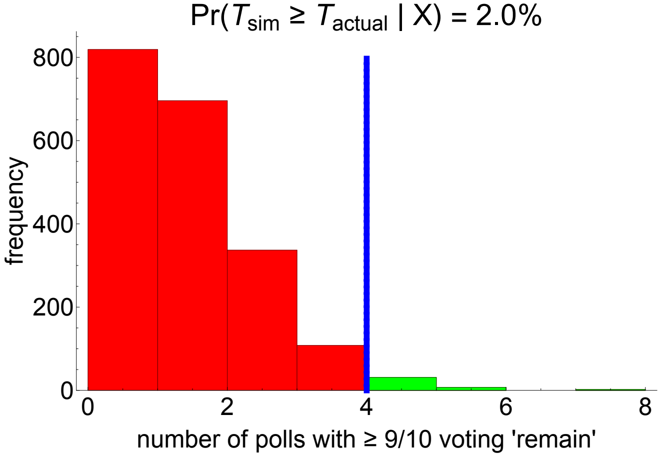
EU referendum: complete pooling model posterior predictive checks

Repeat for 2000 simulated data series; for each simulated dataset counting the number of $X_i \geq 9$.



EU referendum: complete pooling model posterior predictive checks

Low probability of replicating this aspect of real data.



EU referendum: complete pooling model summary

- Assumed a model where the probability of a polled individual intending to vote 'remain' is **identical** across all polls.
- However polls were conducted over a range of time by a range of agencies, each with their own methodology $\implies$ sampling method, exact interview process etc vary.
- Therefore assuming a common θ across polls is too strong an assumption.
- $\implies$ model will **understate** true uncertainty.

$\implies$ try the opposite extreme where we allow a different θ_i for each poll.

EU referendum: heterogeneous model

For each poll i we assume a binomial likelihood:

$$Pr(X = X_i|\theta_i) = \theta_i^{X_i}(1-\theta_i)^{10-X_i} \tag{2}$$

where θ_i is the probability of a interviewee indicating they intend to vote 'remain' in the referendum.

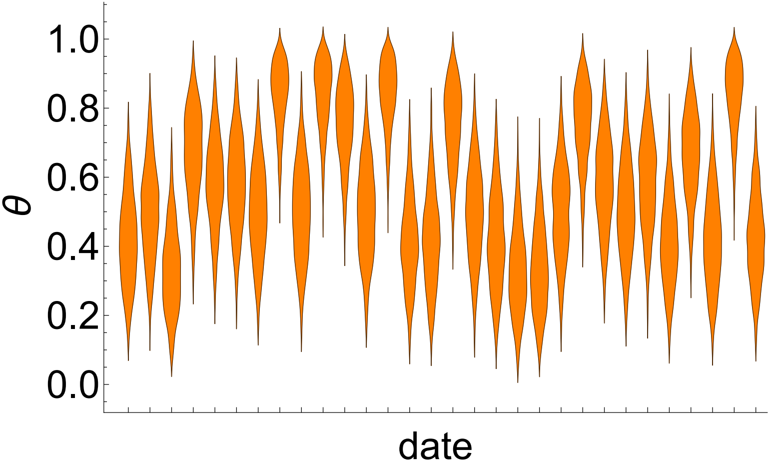
By allowing θ_i to vary across polls $\implies$ **different** data generating processes for each poll!

EU referendum: heterogeneous model

```
data {
  int<lower=1> K; ## number of polls
  int<lower=0> X[K]; ## numbers voting 'remain'
  int<lower=1> N; ## sample size
}
parameters {
  real<lower=0,upper=1> theta[K]; ## now array
}
model {
  for (i in 1:K){
    X[i] ~ binomial(N,theta[i]);## select element
  }
}
```

$\Longrightarrow$ only two changes to homogeneous model necessary.

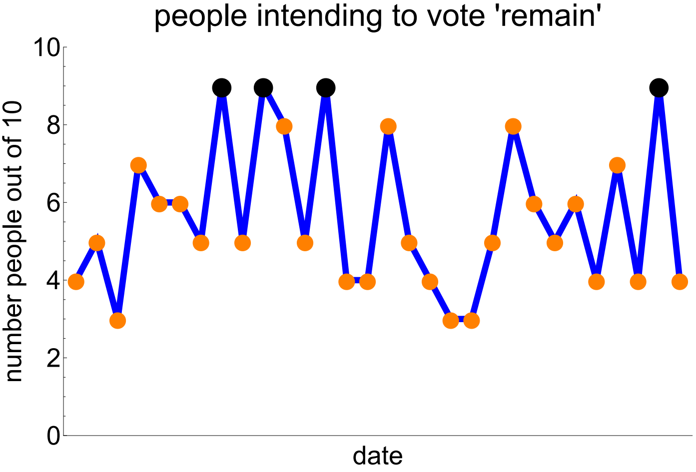
Heterogeneous model results



$\implies$ large uncertainty associated with each θ_i.

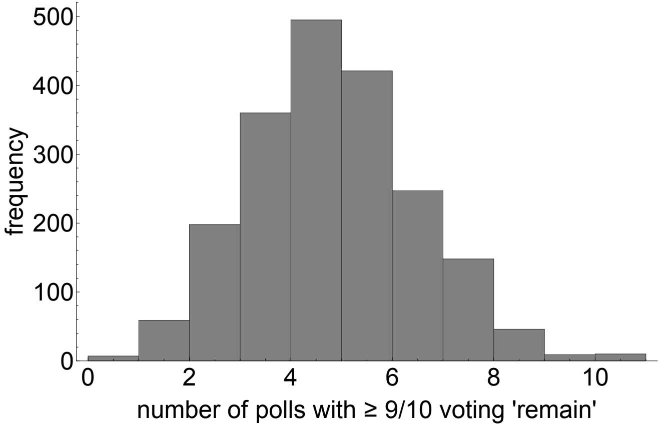
Heterogeneous model posterior predictive checks

Compare again occurrence of 9+/10 'remain' voters with real data; where 4/30.



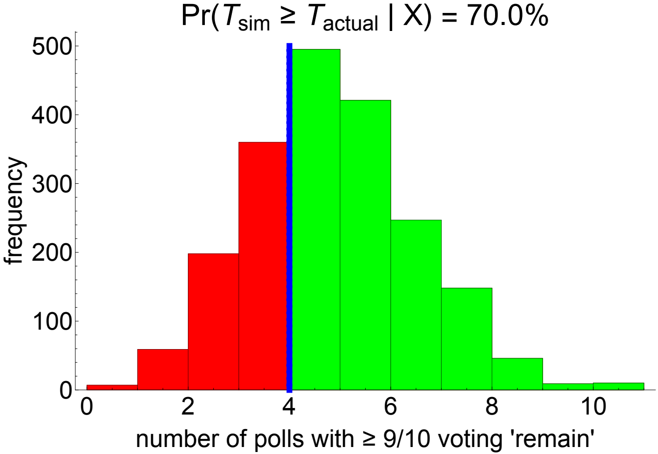
Heterogeneous model posterior predictive checks

For 2000 posterior predictive samples.



Heterogeneous model posterior predictive checks

$\Longrightarrow$ better than complete pooling model.



Heterogeneous model problems

- Heterogeneous model is a better fit to the data than fully-pooled $\Longleftarrow$ is overfitting the data?
- However not clear what we should now forecast for the polls? Should we use:
    - Estimate from one poll.
    - Average across point estimates for all polls.
- Further how should we quantify our uncertainty?

Heterogeneous model: summary

- Same binomial likelihood as before but allowed θ to vary across polls.
- $\implies$ obtained separate estimates of θ across each poll.
- Found considerable variability and uncertainty in estimates of θ_i.
- Heterogeneous θ model better captured the extremes seen in the data $\implies$ fully-pooled model was too strong.
- However key question: **what do we forecast will be the result of the EU referendum, and with what uncertainty?**

Introducing a hierarchical model

- In fully-pooled model case we assumed that data from all the polls was the same; i.e. θ was constant.
- However there are differences between polling methodologies, and time when polls were taken $\implies$ data generated by different processes.
- $\implies$ we estimated a separate θ_i for each poll; i.e. assumed data from different polls was completely unrelated.

Introducing a hierarchical model

Question 1: do we really think that data from different polls is completely unrelated? **Answer 1:** no! After all the polls measure the same thing, at around the same point in time.

Question 2: do we really think that the polls are exactly the same? **Answer 2** no! We rejected this initially because of differences between polling agencies and time over which the polls were done.

Introducing a hierarchical model

Question: isn't there somewhere between the extremes of complete-separation and fully-pooled?
Answer:

Completely separate

Hierarchical model

Fully pooled

Hierarchical model for EU referendum polls

As for heterogeneous model we allow θ to vary by poll:

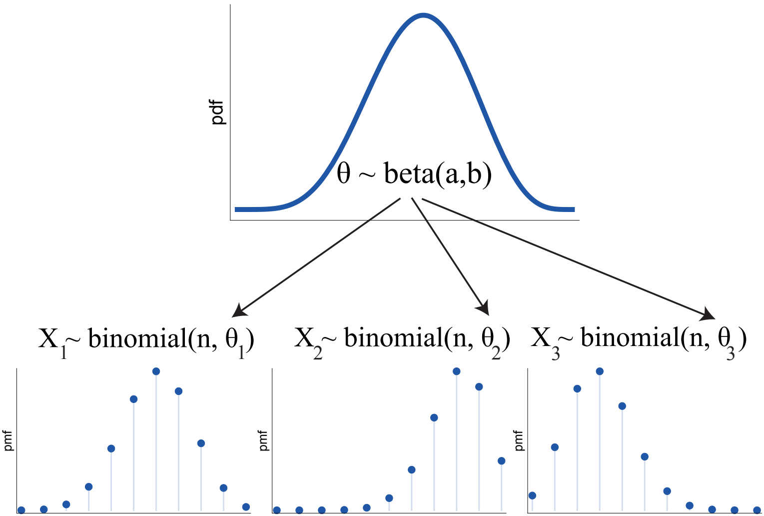
$$Pr(X = X_i|\theta_i) = {\theta_i}^{X_i}(1 - \theta_i)^{10-X_i} \tag{3}$$

However now we assume that the θ_i are drawn from a common "population" distribution:

$$\theta_i \sim beta(a, b) \tag{4}$$

where a and b are parameters that define the population-level beta distribution.

Hierarchical model for EU referendum polls



Hierarchical model for EU referendum polls

$$\theta_i \sim beta(a, b) \tag{5}$$

(a, b) are parameters just like any other in Bayesian inference
$\implies$ assign them **priors**!
Actually easier to set priors for transformed parameters:

$$\begin{aligned} a &= \alpha \times \kappa \\ b &= (1 - \alpha) \times \kappa \end{aligned}$$

where α represents the "population" chance of voting 'remain' and κ measures the concentration $\implies$

$$\theta_i \sim beta(\alpha \times \kappa, (1 - \alpha) \times \kappa) \tag{6}$$

Hierarchical model for EU referendum polls

$$\theta_i \sim beta(\alpha \times \kappa, (1-\alpha) \times \kappa) \tag{7}$$

Set independent priors:

$$p(\alpha, \kappa) = p(\alpha) \times p(\kappa) \tag{8}$$

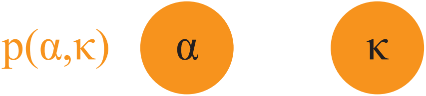
Question: what is the numerator of Bayes' rule for this problem?
Answer: the joint distribution of the data X and parameters; i.e. $p(X, \theta, \alpha, \kappa)$
Another question: how do we find this here? **Another answer:** exploit conditional independence of the problem!

Hierarchical model for EU referendum polls

Start with “population” level parameters.

Hierarchical model for EU referendum polls

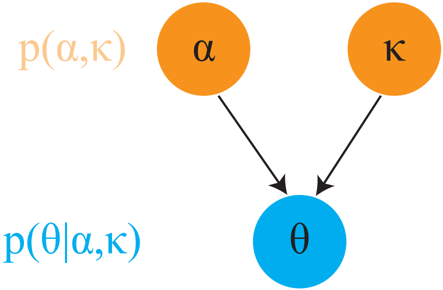
And determine their joint probability.



$p(\alpha,\kappa)$

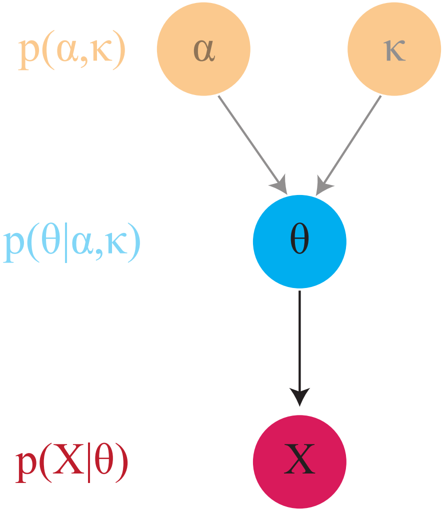
Hierarchical model for EU referendum polls

Find the probability of θ **conditional** on α and κ.



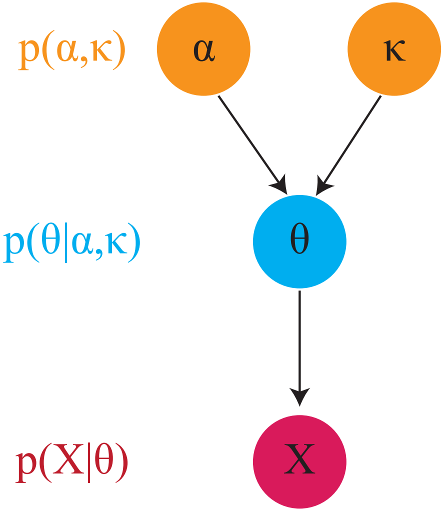
Hierarchical model for EU referendum polls

Find the probability of X **conditional** on θ .



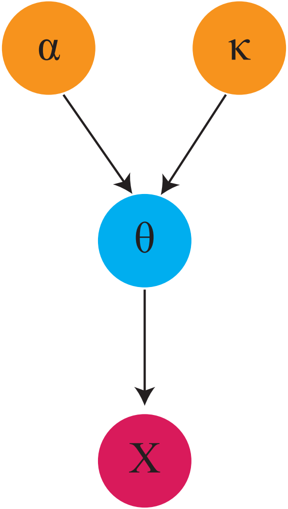
Hierarchical model for EU referendum polls

Finally to obtain the overall probability...



Hierarchical model for EU referendum polls

...multiply together all the terms.



$p(\alpha,\kappa)$

$\times$

$p(\theta|\alpha,\kappa)$

$\times$

$p(X|\theta)$

Hierarchical model for EU referendum polls

So the posterior is found as:

$$\begin{aligned} p(\theta, \alpha, \kappa | X) &\propto p(X, \theta, \alpha, \kappa) \\ &= \underbrace{p(X|\theta)}_{\text{likelihood}} \times \underbrace{p(\theta|\alpha, \kappa)}_{\text{prior}} \times \underbrace{p(\alpha, \kappa)}_{\text{hyper-prior}} \end{aligned}$$

- $p(X|\theta)$ is just the **likelihood**.
- $p(\theta|\alpha, \kappa)$ is the **prior** on θ.
- $p(\alpha, \kappa)$ is the **hyper-prior** on the **hyper-parameters** α and κ.
- However the word "hyper" is really just a fancy word we use to represent priors on "population" level parameters.
- In hierarchical models there is a blurring of the likelihood/prior boundary.

Hierarchical model for EU referendum polls: back to the problem

We set the following independent (hyper-)priors on α and κ:

$$\alpha \sim \textit{beta}(5,5)$$
$$\kappa \sim \textit{pareto}(1,0.3)$$

where:

- beta(5,5) has a mean of 0.5, and only has support for $0 \leq \alpha \leq 1$.
- pareto(1,0.3) is a distribution only with support for values of $\kappa \geq 1$.

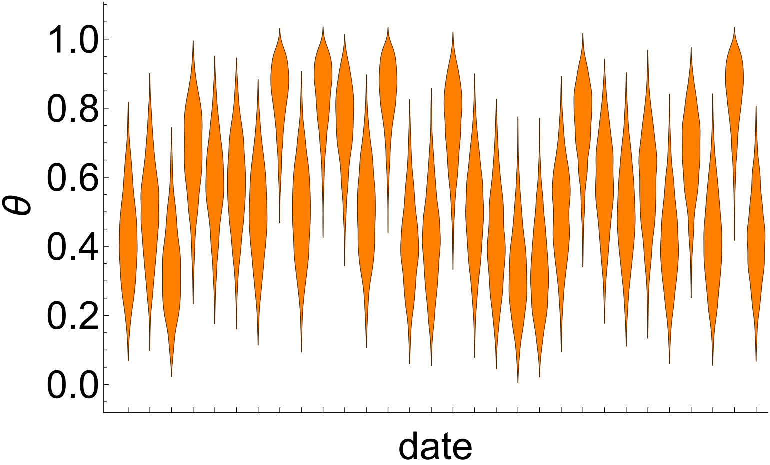
Hierarchical model for EU referendum polls: coding up model in Stan

```
parameters {
  real<lower=0, upper=1> alpha;
  real<lower=1> kappa;
  vector<lower=0, upper=1>[K] theta;
}
model {
  for (i in 1:K){
    Y[i] ~ binomial(N[i],theta[i]);## Likelihood
  }
  ## prior
  theta ~ beta(alpha * kappa, (1 - alpha) * kappa);

  ## hyper-priors
  kappa ~ pareto(1, 0.3);
  alpha ~ beta(5,5);
}
```

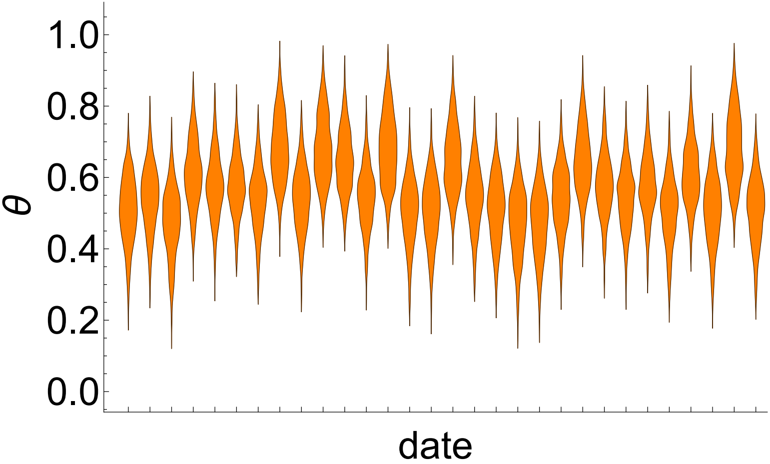
Hierarchical model for EU referendum polls: heterogeneous model estimates

Remember the posterior from the heterogeneous model?



Hierarchical model for EU referendum polls: hierarchical model estimates

Hierarchical model estimates.



Hierarchical model for EU referendum polls: hierarchical model estimates

Two effects evident:

- Shrinkage to **grand** mean.
- Shrinkage in variance.

Shrinkage to grand mean because hierarchical models lie on a spectrum between completely heterogeneous estimates and fully pooled.
Important: the data determines where on the spectrum we exactly end up! No choice in analysis.
Important: shrinkage helps reduce the variance of estimates $\implies$ outliers have less impact.

Also "partially-pooling" information across groups $\implies$ essentially a larger sample size and lower variance.

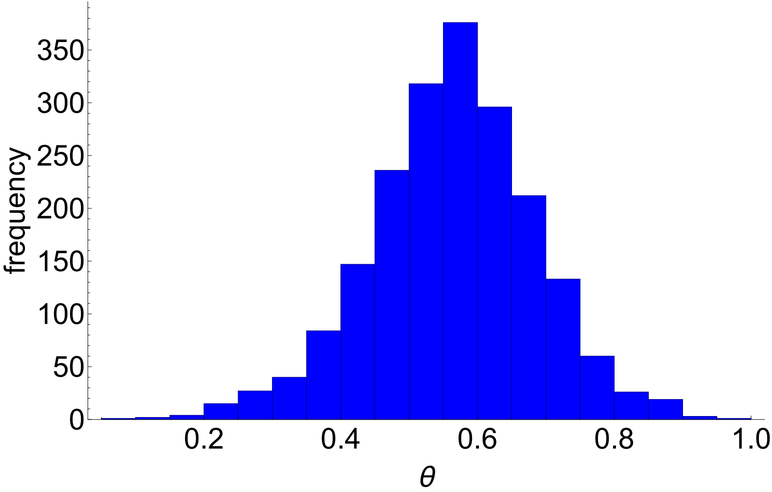
Hierarchical model: forecasting outcome of overall elections

Want to estimate the value of θ for the last poll: the referendum itself!
To do this do the following:

1. Sample (α, κ) from their posteriors.
2. Sample $\theta \sim \textit{beta}(\alpha\kappa, (1-\alpha)\kappa)$.

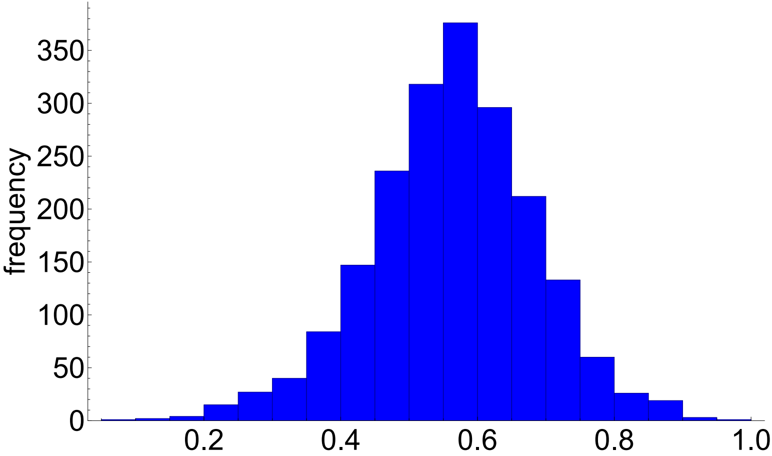
Hierarchical model: forecasting outcome of overall elections

Yields the posterior below (gulp):



Hierarchical model: forecasting outcome of overall elections

Conclusion: the result could go either way (I did a similar analysis for real poll data and it produced a posterior very much like the one below!)

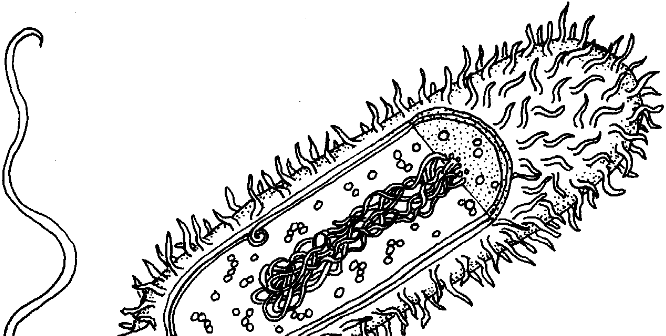


Hierarchical models: summary

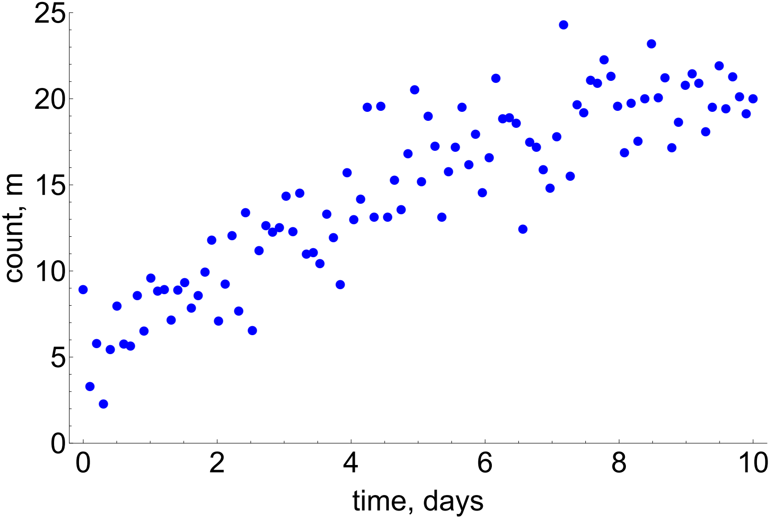
- Model with **same** parameters for all groups $\implies$ data generating process is the **same**.
- Model with separately estimated group-level parameters $\implies$ data generating process is completely **different**.
- Frequently neither of the aforementioned models are appropriate; i.e. we want some dependence between parameters but not 100%.
- $\implies$ use hierarchical model where the data determines parameter dependence across groups.
- In hierarchical models $\implies$ **shrinkage** of **group** means towards the **grand** mean.
- Also **shrinkage** of group variance $\impliedby$ sample size$\uparrow$ by **partial-pooling** information across groups.

Example: bacterial growth

- We carry out experiments where we inoculate agar plates with bacteria at time 0.
- At pre-defined time intervals we count the number of bacteria on each plate, $N(t)$.
- Suppose we want to model bacterial population growth over time.



Example: bacteria growth data



Example: bacterial growth model

- Assume the following model for bacterial population growth:

$$\frac{\mathrm{d}N}{\mathrm{d}t} = \alpha N(1 - \beta N) \tag{9}$$

where $\alpha > 0$ is the rate of growth due to bacterial cell division, and $\beta > 0$ measures the reduction in growth rate due to “crowding”.

Question: how should we infer the parameters of this model?

Example: bacterial growth model

Answer: assume measurement error around true value:

$$N^*(t) \sim \text{normal}(N(t), \sigma) \tag{10}$$

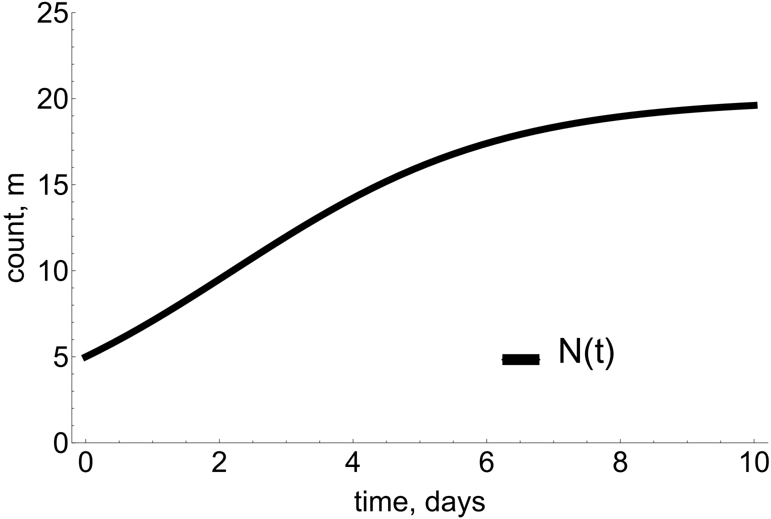
where

- $N^*(t)$ is the **measured** count of bacteria at time t.
- $N(t)$ is the solution to the ODE at time t (true number of bacteria on plate.)
- $\sigma > 0$ measures the magnitude of the measurement error about the true value.

Question: how does this model work?

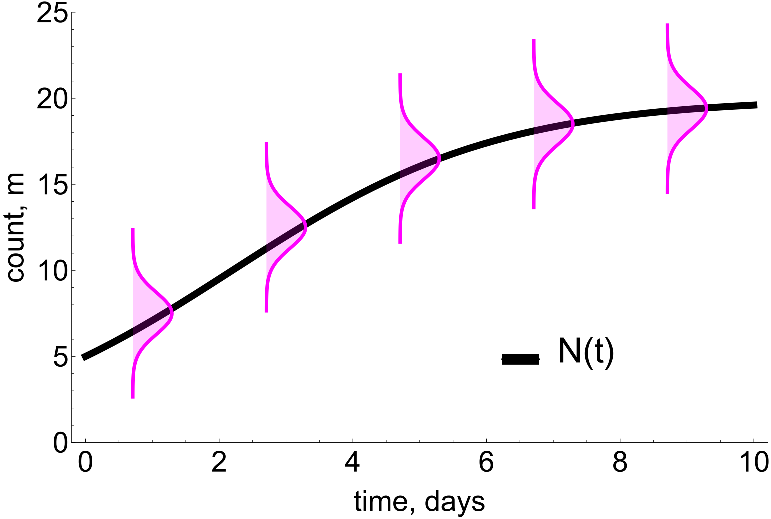
Example: bacterial growth model

Start with true number of bacterial cells, $N(t)$.



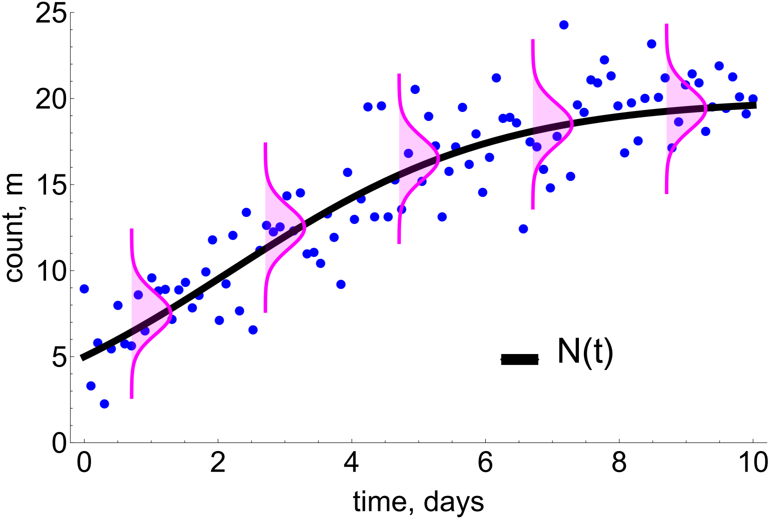
Example: bacterial growth model

Overlay sampling distribution representing measurement error.



Example: bacterial growth model

And data generated from this process.



Example: bacteria growth model inference

Remember we are using a normal likelihood:

$$N^*(t) \sim \text{normal}(N(t), \sigma) \tag{11}$$

$\implies$ likelihood for all observations:

$$L(N(t), \sigma) = \prod_{t=t_1}^{T} \frac{1}{\sqrt{2\pi\sigma^2}} exp\left[\frac{-(N^*(t) - N(t))^2}{2\sigma}\right] \tag{12}$$

Question: how do we calculate $N(t)$?

Example: bacteria growth model inference

$$\frac{\mathrm{d}N}{\mathrm{d}t} = \alpha N(1 - \beta N) \tag{13}$$

- In most ODE models, the mean $N(t)$ cannot be solved for exactly so we **can't write down a "closed-form" expression for the likelihood.**
- $\implies$ approximate answer using a numerical method.
- However any solution for $N(t)$ - exact or numerical - depends on the parameters of the ODE model. For our example:

$$N(t) = f(t, \alpha, \beta) \tag{14}$$

Question: how do we do MCMC in this setting?

Example: bacteria growth model inference

For example, in Random Walk Metropolis:

- Start at random location in (α, β, σ) space.
- For t=1,...,T do:
  1. Propose a new location $(\alpha', \beta', \sigma')$ using a jumping distribution.
  2. Numerically (or analytically) integrate ODE to solve for $N(t, \alpha', \beta')$.
  3. Calculate un-normalised posterior at proposed location $\implies$ calculate r.
  4. Based on r move to new location or stay at original.

$\implies$ at every step we must solve ODE for $N(t)$; can be computationally expensive!

Example: bacteria growth model in Stan

- ODE models can be slow to converge - particularly when accounting for numerical solution of equations.
- Fortunately Stan has an inbuilt ODE integrator $\Longrightarrow$ can leverage the speed of HMC (NUTS) for ODE systems!
- Write a function that returns the derivative (RHS of ODE):

```
real[] bacteria_diff(real t,real[] N,
              real[] theta,real[] x_r,int[] x_i){
  real dNdt[1];
  dNdt[1] <- theta[1] * N[1] *
                         (1 - theta[2] * N[1]);
  return dNdt;
}
```

Example: bacteria growth model in Stan

```
model {
    sigma ~ cauchy(0,1); ## Prior
    theta ~ normal(0,2); ## Prior for alpha, beta
    N0 ~ normal(5,2); ## Prior

    ## Solve ODE at current parameter values
    real N_hat[T,1];
    N_hat <- integrate_ode(bacteria_diff, N0, t0, ts,
                          theta, x_r, x_i);
    ## Likelihood
    for (t in 1:T) {
        N[t] ~ normal(N_hat[t,1],sigma);
     }
}
```

$\Longrightarrow$ `integrate_ode` takes the derivative function as its first input argument.

Issues with inference for ODEs and PDEs

Whilst Stan's integrator makes it quite easy to use HMC to do MCMC, there are still problems that arise:

- ODE models are very often non-identifiable $\implies$ need to reparameterise model.
- (Linked) ODE models can be slower to converge than simpler models $\implies$ need to run MCMC for longer before $\hat{R} < 1.1$ achieved.

$\implies$ important that we "know" our model well before we start to do inference explicitly.
Worth putting energy into mathematical analysis + non-MCMC Bayesian methods (NLS or ABC) before trying MCMC.

Inference for ODEs: summary

- ODE models are no harder to formulate than “traditional” problems.
- However for ODE models we cannot typically write down a “closed-form” expression for the likelihood.
- $\implies$ use integrator to numerically solve for mean for each set of parameters.
- Stan has inbuilt integrator meaning we can do MCMC using Stan’s fast HMC implementation.
- ODE models are often under-identified $\implies$ do non-linear least squares, or approximate Bayesian computation *before* MCMC.

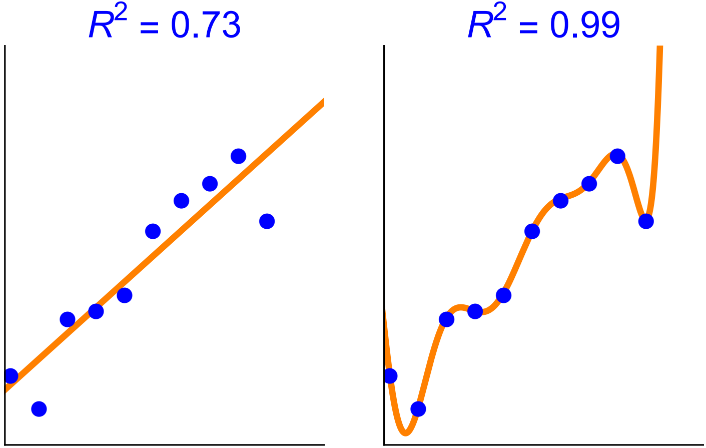
Why do we need a measure of a model's fit?

Often in modelling we are required to choose between a number of models, in order to:

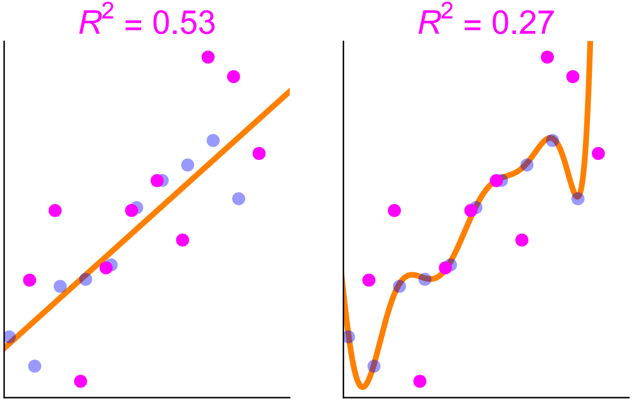
- Test between rival hypotheses about the real world.
- Choose the most predictive model to make forecasts about the future.
- Avoid the computational expense of using a number of models for some future use.

Note: model selection can sometimes be avoided by **a.** using a parameter to specify model choice or **b.** using a general model that allows all models as special cases.

Which of these models is more predictive?



Simple models generalise better to out-of-sample data



$\Longrightarrow$ right model is **overfit** to data; it fits the **noise** not the **signal**.

Selection bias in model selection

The problem:

- Want to measure a model's predictive capability on an **independent** data set; i.e. one that has not been used to fit our model.
- But don't have out-of-sample data! (If we did it would form part of the "sample"!)
- If use in-sample data to gauge model fitness $\implies$ predictive performance is upwardly-biased due to overfit.

Note: this selection bias effect $\implies$ posterior predictive checks will fail to detect model overfit.

Selection bias in model selection

Solutions:

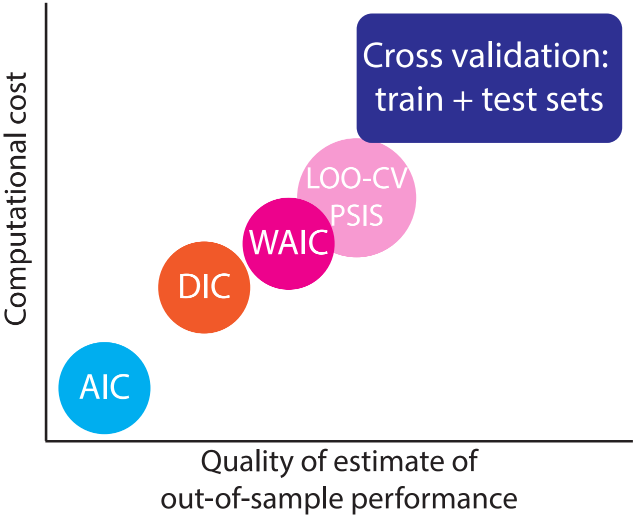
- Use **heuristics** to correct in-sample measures of fit to account for overfit.
- (Better) Use **cross validation** where data is partitioned into:
  - "Training" sets: used to fit model.
  - "Testing" sets: used to evaluate model.

Using heuristics to estimate out-of-sample predictive performance

Correct measures of fit to account for overfitting:

- $R^2 \rightarrow \overline{R^2}$; i.e. a metric that measures proportion of variance explained by model.
- Log-likelihood → AIC, DIC, or WAIC; more appropriate metric for Bayesian models due to their basis in likelihood/probability.

Ranking heuristics



Evaluating heuristics: AIC

“Akaike Information Criterion” computed by (ignoring -2 at front):

$$\text{AIC} = log\ p(X|\hat{\theta}_{MLE}) - k \tag{15}$$

where k is the number of parameters estimated in model fitting, and $\hat{\theta}_{MLE}$ is maximum likelihood (point) estimate.

- Based on asymptotic approximation to normal linear models with uniform priors.
- $\implies$ not reasonable correction for more general models; particularly hierarchical ones.
- Ignores uncertainty in parameter estimates, and hence log-likelihood.

Evaluating heuristics: DIC

"Deviance Information Criterion" computed by (ignoring -2 at front):

$$\text{DIC} = log\ p(X|\hat{\theta}_{Bayes}) - p_{DIC} \tag{16}$$

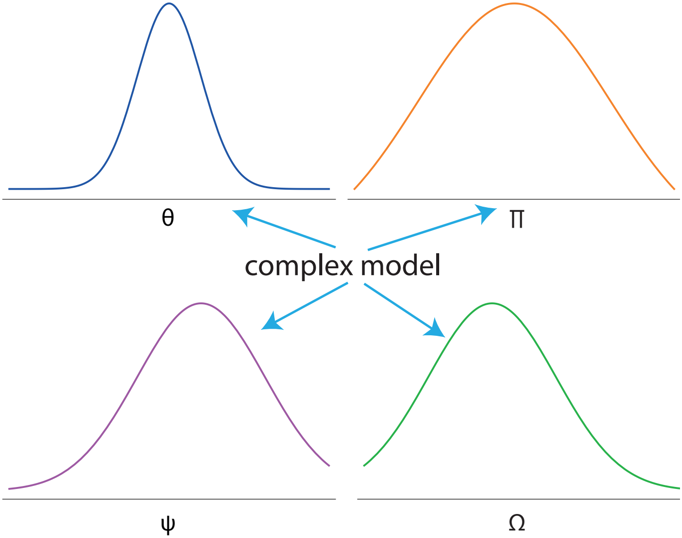
where $\hat{\theta}_{Bayes}$ is posterior mean (point) estimate and p_{DIC} is the "effective number of parameters" defined by:

$$p_{DIC} = 2var_{post}\left[log\ p(X|\theta)\right] \tag{17}$$

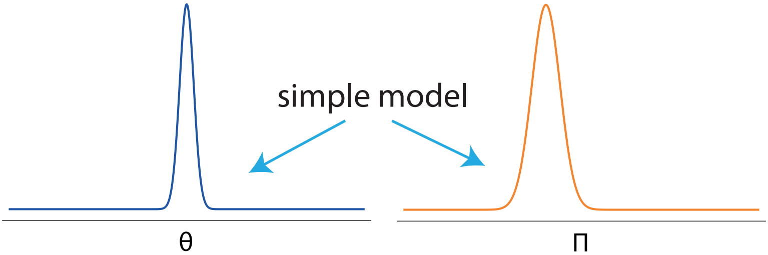
which is typically estimated across all posterior samples.

$$p_{DIC} \approx 2V_{s=1}^{S} log\ p(X|\theta_s) \tag{18}$$

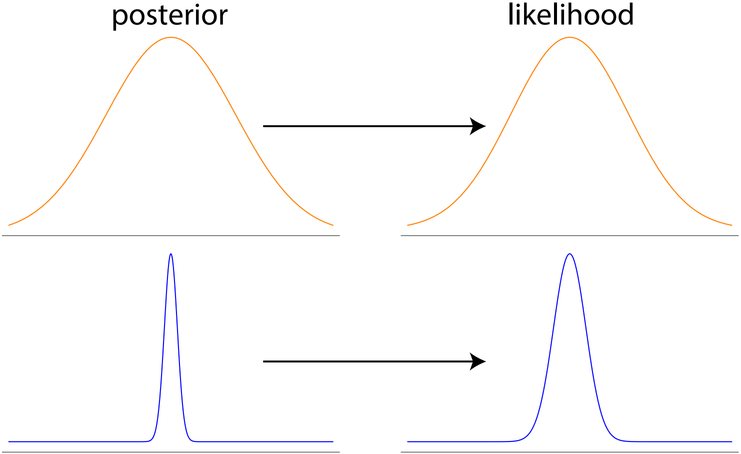
DIC complexity correction intuition: complex models have high parameter uncertainty



DIC complexity correction intuition: simpler models have lower uncertainty



DIC complexity correction intuition: lower posterior uncertainty $\implies$ lower uncertainty in fit

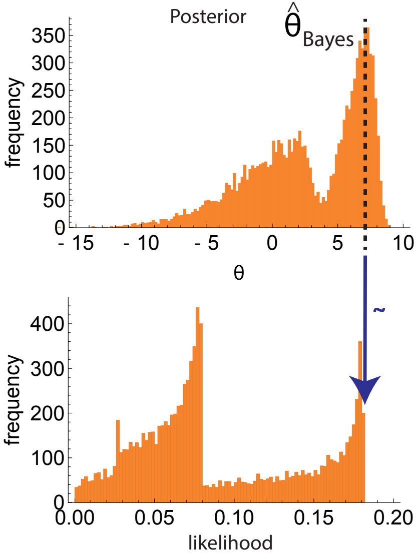


Evaluating heuristics: DIC

$$p_{DIC} = 2var_{post}\left[log\ p(X|\theta)\right] \tag{19}$$

- Spiegalhalter (2002) introduced P_{DIC} as a measure of model dimensionality.
- Overfit models have many parameters, each with high uncertainty.
- A high variance in $\theta \implies$ high variance in $log(X|\theta)$.
- $\implies p_{DIC}$ is big, so large correction.

DIC issue: uses a point estimate to determine fit



Evaluating heuristics: DIC

$$\text{DIC} = log\ p(X|\hat{\theta}_{Bayes}) - p_{DIC} \tag{20}$$

In summary:

- Uses less arbitrary notion of model dimensionality to correct measure of fit than AIC.
- $\implies$ better for a wider class of models.
- Still does not fully address uncertainty in fit since calculates first term above with a point estimate $\hat{\theta}_{Bayes}$.

Evaluating heuristics: WAIC

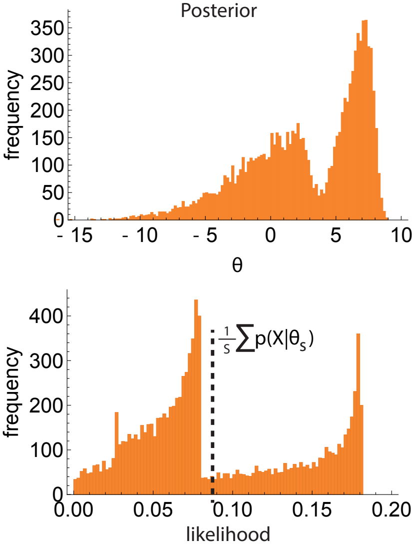
“Watanabe-Akaike Information Criterion” (Watanabe, 2010) computed by (ignoring -2 at front):

$$WAIC = \underbrace{\sum_{i=1}^{N} log \left(\frac{1}{S} \sum_{s=1}^{S} p(X_i|\theta_s) \right)}_{\text{log pointwise predictive density}} - p_{WAIC} \tag{21}$$

Since allows for uncertainty in fit $\implies$ fully-Bayesian estimate. Where p_{WAIC} is another correction factor calculated by:

$$p_{WAIC} = \sum_{i=1}^{N} var_{post} \left[log \ p(X_i|\theta) \right] \tag{22}$$

WAIC: averages likelihood over posterior to determine fit



Evaluating heuristics: WAIC

$$WAIC = \underbrace{\sum_{i=1}^{N} log \left(\frac{1}{S} \sum_{s=1}^{S} p(X_i|\theta_s) \right)}_{\text{log pointwise predictive density}} - \sum_{i=1}^{N} var_{post} \left[log \;\; p(X_i|\theta) \right] \tag{23}$$

In summary:

- Fully Bayesian since estimates fit across all posterior samples.
- Evaluates the log predictive density using pointwise sums.
- $\implies$ for structured models can be problematic if model not easily split into parts (can always use groups rather than individual points, though.)
- (Still an **approximation** $\implies$ ideally use proper cross validation.)

WAIC using Stan + ‘loo’ package

Use “generated quantities” block to store log-likelihood across all data points. For example:

```
model {
    ...
   for (i in 1:N){
      y[i] ~ normal(mu,sigma);
    }
}
generated quantities{
    vector logLikelihood[N];
    for (i in 1:N){
       logLikelihood[i] <- normal_log(y[i],mu,sigma);
    }
}
```

WAIC using Stan + ‘loo’ package

```r
library(loo)

## Extract log-likelihood
logLikelihood <- extract_log_lik(fit,'logLikelihood')

## Calculate WAIC
aWAIC <- waic(logLikelihood)

## Print answer
print(aWAIC)

## Computed from 12000 by 5000 log-likelihood matrix
##
##           Estimate   SE
## elpd_waic -10543.1 49.9
## p_waic         4.0  0.1
## waic       21086.3 99.7
```

“loo” also estimates leave-one-out-cross-validation errors using importance sampling

```
## Estimate LOO-CV
aLOO <- loo(logLikelihood)

## Print answer
print(aLOO)

## Computed from 12000 by 5000 log-likelihood matrix
##
##           Estimate   SE
## elpd_loo -10543.1 49.9
## p_loo         4.0  0.1
## looic     21086.3 99.7
##
## All Pareto k estimates OK (k < 0.5)
```

Note: this is a preferred measure to WAIC, but be careful $\Longrightarrow$ if get Pareto k estimates warning do manual cross validation.

Important: use pairwise comparison to do model selection with WAIC or LOO-CV

```
## Extract log-likelihood for both models
logLikelihood_1 <- extract_log_lik(fit_1,'logLikelihood')
logLikelihood_2 <- extract_log_lik(fit_2,'logLikelihood')

## Estimate LOO-CV for eac
aLOO_1 <- loo(logLikelihood_1)
aLOO_2 <- loo(logLikelihood_2)

## Print answer
compare(aLOO_1,aLOO_2)

## elpd_diff        se
##      -0.7       0.7
```

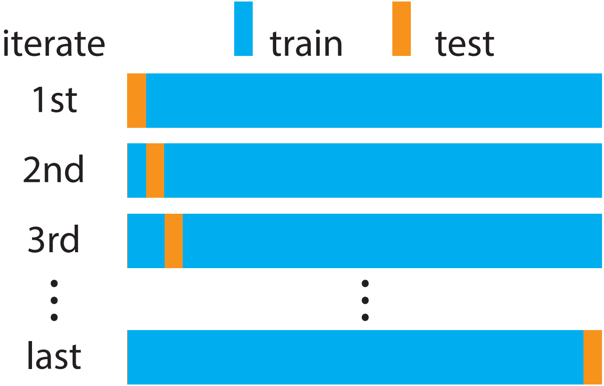
$\Longrightarrow$ pairwise comparison takes a proper account of uncertainty in fit of each model **relative** to the other.

Manual cross-validation

- The aforementioned methods **approximate** out-of-sample predictive capability by use of post-hoc corrections to account for overfitting.
- A better class of approximations is to partition the dataset and do proper **cross-validation.**

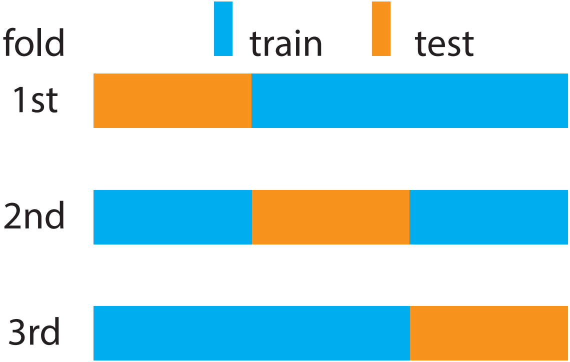
Question: how should we choose our training and test sets?

Manual cross-validation: leave-one-out cross-validation


iterate
🟦 train
🟧 test
1st
2nd
3rd
⋮
last

$\Longrightarrow$ average log-likelihood across all posterior θ across all partitions. **Note:** no need for overfitting correction!

Manual cross-validation: k-fold cross-validation



$\Longrightarrow$ average log-likelihood across all posterior θ across all folds. **Note:** no need for overfitting correction!

Cross validation: issues for consideration

- Cross-validation method should reflect eventual use of model. For example, if predicting individual data points is most important use leave-one-out.
- Manual-refitting the model for each train/set can be costly $\Longrightarrow$ k-folds is less time-consuming than leave-one-out.
- Leave-one-out may be difficult for hierarchical models where data is naturally grouped $\Longrightarrow$ use k-folds where one fold = one group.
- Remember cross-validation is still an approximation to out-of-sample estimation $\Longrightarrow$ best to get new and independent data set!

Bayes factors

An alternative approach is to use Bayes factors to choose between models:

$$\frac{p(\text{model } 1|X)}{p(\text{model } 2|X)} = \frac{p(X|\text{model } 1)}{p(X|\text{model } 2)} \times \frac{p(\text{model } 1)}{p(\text{model } 2)} \tag{24}$$

where blue expression is known as the **Bayes factor.**

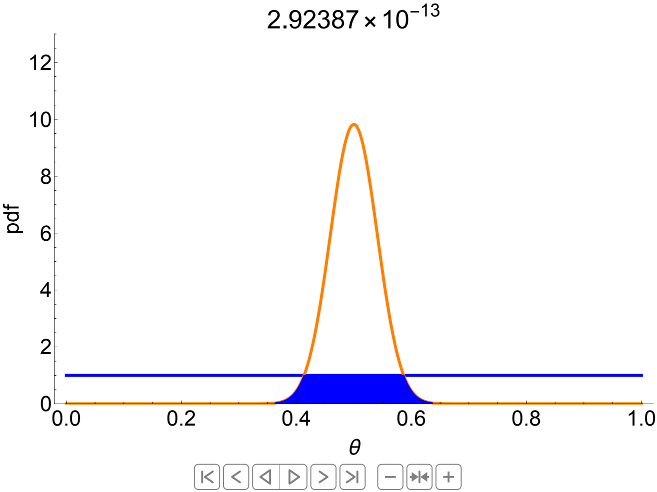
$\implies$ requires:

- *a priori* specification of our preferences over models (not simple for models with different dimensions.)
- Calculate $p(X|\text{model})$ – known as the **marginal likelihood** for a model. Calculate by,

$$p(X|\text{model}) = \int \underbrace{p(X|\theta, \text{model})}_{\text{likelihood}} \times \underbrace{p(\theta|\text{model})}_{\text{prior}} \, \mathrm{d}\theta \tag{25}$$

Problem with Bayes factors 1: marginal likelihood's sensitivity to priors

Suppose $X \sim \text{binomial}(10, \theta)$ likelihood, and $\theta \sim \text{beta}(a, a)$ prior. **Question:** how does the marginal likelihood vary as $a \uparrow$?



Problem with Bayes factors 2: difficulty estimating marginal likelihood

For most problems $\theta = \theta_1, ..., \theta_p$,

$$p(X|\text{model}) = \int ... \int p(X|\theta_1, ..., \theta_p, \text{model}) \times \tag{26}$$

$$p(\theta_1, ..., \theta_p|\text{model}) \, \mathrm{d}\theta_1 ... \mathrm{d}\theta_p \tag{27}$$

$\implies$ integral too difficult to compute in practice. But can approximate using Monte Carlo integration,

$$p(X|\text{model}) \approx \frac{1}{S} \sum_{s=1}^{S} p(X|\theta_{1s}, ..., \theta_{ps}, \text{model}) \tag{28}$$

where $\theta_{1s}, ..., \theta_{ps} \sim p(\theta_1, ..., \theta_p|\text{model})$, i.e. priors.

Problem with Bayes factors 2: difficulty estimating marginal likelihood

$$p(X|\text{model}) \approx \frac{1}{S}\sum_{s=1}^{S} p(X|\theta_{1s}, ..., \theta_{ps}, \text{model}) \tag{29}$$

However simple Monte Carlo integration is very slow to converge, particularly when there is a mismatch between the position of the prior and likelihood. Solutions exist, for example:

- Bayesian Monte Carlo.
- Importance Sampling Annealing.
- Adiabatic Monte Carlo.

But still early days in the research.

Bayes factors: summary

- Require us to specify priors over model choice and calculate marginal likelihood.
- Marginal likelihood is highly sensitive to priors, even for nuisance parameters.
- Marginal likelihood hard to calculate exactly (multi-dimensional integral), and so approximate using sampling.
- $\implies$ simple Monte Carlo is slow to converge.

Model selection: summary

- Model selection inevitable part of scientific process $\Longrightarrow$ need a method for choosing between rival models.
- Posterior predictive checks useful in model building process but use predictive performance to select between hypotheses.
- Out-of-sample predictive performance is our aim.
- Two methods:
  - Post-hoc correction: Approximate LOO-CV using "loo" R package, or failing that use WAIC.
  - (Better) cross-validation: leave-one-out or k-folds.
- Bayes factors can be useful for model selection although need to be careful!

Full course modelling tips

- Aim to graph data in the most enlightening way for eventual model use; this is difficult and requires going through a lot of useless visualisations.
- Start simple with modelling and build up complexity as needed.
- Use creative posterior predictive checks to test models assumptions.
- Read books on inference $\implies$ build up knowledge of models/frameworks.
- Present your work to colleagues; the process of preparing for presentation as well as the feedback itself is always useful.
- When in doubt use hierarchical models.

Thanks

- David Gavaghan.
- Kevin Judd and Susanne Vogel.
- Sam, Nina, and Francesca at the DTC.
- Simon, Simon, and Soohee.
- Stan development team.
- Everyone for coming to the lectures and classes!

Not sure I understand