AMS 207: Intermediate Bayesian Modeling

6: Model Comparison and Model-Checking

David Draper

Department of Applied Mathematics and Statistics
University of California, Santa Cruz

draper@ams.ucsc.edu
http://www.ams.ucsc.edu/∼draper

The Big Picture, Again

The **ingredients** in a **Bayesian statistical modeling problem** are as follows:

• θ, something **unknown** to me (often this is a **vector** of **real numbers** of length $k \geq 1$ — in which case I'm doing **Bayesian parametric modeling**, which is the focus of this class — but it could be **just about anything**);

• y, some **information (data)** that I judge **relevant** to **decreasing my uncertainty** about θ (often this is a **vector** $y = (y_1, \ldots, y_n)$ of length n, in which each y_i is itself a **vector** of **real numbers** of length $d \geq 1$ — which is the focus of this class — but again it could be **just about anything**); and

• a desire to **summarize my uncertainty** about θ, on the basis of **information** both **internal to** and **external to** y, in a way that's **internally (logically) consistent (coherent)**.

Then it's not just a **choice**, it's a **theorem**, that

• I need to **quantify my uncertainty** about θ through three **conditional probability distributions** (i.e., to use the **machinery** that arises by treating θ as a **random variable** even though its **logical status** is that of a **fixed unknown constant**):

— the **prior distribution** $p(\theta|\mathcal{B})$, which **summarizes my information** about θ **external** to the data set y, on the basis of my **background assumptions and judgments** $\mathcal{B}$ about how the world works as far as θ is concerned;

— the **likelihood distribution** $l(\theta|y, \mathcal{B}) = c\, p(y|\theta, \mathcal{B})$, obtained by **density-normalizing**, with the **positive constant** c, my **prior conditional predictive distribution** $p(y|\theta, \mathcal{B})$ (also known as my **sampling distribution**) for y given θ and $\mathcal{B}$, and

— the **posterior distribution** $p(\theta|y, \mathcal{B})$, which **summarizes my information** about θ both **external** and **internal** to the **data set** y, given $\mathcal{B}$; and

The Specification Problem, Revisited

• To avoid **internal logical inconsistencies (incoherence)**, these **distributions** must be **related**, via **Bayes's Theorem**, as follows:

$$p(\theta|y,\mathcal{B}) = c\, p(\theta|\mathcal{B})\, l(\theta|y,\mathcal{B}). \quad (1)$$

Let's agree to call $\{p(\theta|\mathcal{B}), l(\theta|y,\mathcal{B})\}$ (together) the **Bayesian (parametric) model** M; and from now on I'll usually **suppress** $\mathcal{B}$ for **notational convenience** (but we need to remember that it's **still there**).

OK, so now I **know what to do**, but:

• How do I **specify** $p(\theta)$ **"well"**?

• How do I **specify** $l(\theta|y)$ (or equivalently $p(y|\theta)$) **"well"**?

• What does **"well"** mean?

Believe it or not, choosing a **reasonable meaning of "well"** and **making it operational** are still **active areas of Bayesian research**.

As we've seen, **judgments of conditional exchangeability** (based on the **science** of the problem, i.e., part of $\mathcal{B}$) help a lot with the **likelihood/sampling distribution**, but (as we've also seen) these judgments **don't uniquely pin down a single sampling model** except with categorical **multinomial** data (of which **Bernoulli outcomes** are a **special case**) or **quantitative data** (if you're willing to go directly to **Bayesian nonparametric modeling**, and this is beyond the scope of this class: **AMS 241**).

In my view this is where the **other** of the **two basic principles** governing **good Bayesian modeling** (**coherence** + _____) comes in:

I also want my **Bayesian answers** to be **externally consistent** (in the usual **calibration** sense of comparing **how often I get the right answer** with **how often I say I'll get the right answer**).

Coherence + Calibration

As I've discussed in my **7 Dec 2009 talk** (on the course website), I see only **three ways** to **pay the right price**, in a **calibration** sense, for my **uncertainty** about how to **specify my Bayesian model**:

- **Bayesian nonparametrics** (BNP; in my view the **most satisfying solution of all**, but beyond the scope of this class);

- **Bayesian model averaging** (BMA): if I'm unsure about how to specify M, I collect all of the **reasonable possibilities** together into a **set** of models $\mathcal{M}$ and **sum** or **integrate hierarchically** over $\mathcal{M}$ to **quantify my specification uncertainty**: for example, with finite $\mathcal{M} = \{M_1, \ldots, M_m\}$ and a quantity Δ whose **meaning is common to all these models** (e.g., the **next data value** y_{n+1}),

$$p(\Delta|y, \mathcal{M}) = \sum_{j=1}^{m} p(\Delta|y, M_j, \mathcal{M})\, p(M_j|y, \mathcal{M}); \qquad (2)$$

here what this (sensibly) says to do is to take a **weighted average** of my **conditional predictive distributions** $p(\Delta|y, M_j, \mathcal{M})$, weighted by their **posterior probabilities** $p(M_j|y, \mathcal{M})$.

I view **BMA** as a **parametric approximation to BNP**:

- 3CV: As described in my **7 Dec 2009 talk**, if I **pay the right price** for **using the data** to **guide a search** for **"good" models**, I can still achieve **good calibration**.

The Two Questions

Let's postpone **details of how to do 3CV** til later and focus now on **how Bayesians might search for good models**.

Such a **search** would need **four ingredients**:

(1) A **starting point**, say $M_0 = M_{old}$;

(2) A way to **suggest another model** M_{new} that **might be better**;

(3) A way to **answer** the question **"Is M_{new} better than M_{old}?"** (this is **model comparison**); and

(4) A **stopping rule**.

With these **ingredients** I can **start** at (1), go to (2) and then (3); if M_{new} is **better**, set the **current best model** to M_{new} and **go back** to (2); if M_{new} is **not better**, keep the **current best model** at M_{old} and **go back** to (2); when my **reservoir** of {**time, money, ingenuity**} runs out, take the **current best model** M^* and make (4) **operational** by answering the question "Is M^* **good enough?**"

(When I'm **done** with this search, nothing says I have to **move forward just with the best model I've found**; I could **keep track of all of the good models discovered** and **use BMA**.)

As I argue in my **7 Dec 2009 talk**, the two questions **"Is M_{new} better than M_{old}?"** and "Is M^* **good enough?**" are **not yet well-posed**:

better than/good enough for what purpose?

Specifying the **purpose** of my modeling **transforms** the problem from **inference** to **decision-making**: in my view **fully satisfying answers** to these questions require (a) specifying a **utility function** that **quantifies my value judgments** among **good and bad possibilities** and (b) **maximizing expected utility** to choose a model (this is also beyond the scope of this course: **AMS 221**).

M_2 Versus M_1

(See the example with **Fouskakis** in my **7 Dec 2009 talk** for a fully-worked-out **case study** of **model specification via decision theory**.)

However, the **decision-theoretic approach** is **hard work**, and it's **not always clear** what the **end-use of the modeling exercise will be**; it would be good to have a rather **general-purpose utility-based** way to decide if M_2 is **better** than M_1; here are **two ideas** along these lines.

Idea 1: Why not base the **choice** on **posterior model probabilities**?

By **Bayes's Theorem in odds form**,

$$\frac{p(M_2|y)}{p(M_1|y)} = \left[\frac{p(M_2)}{p(M_1)}\right] \cdot \left[\frac{p(y|M_2)}{p(y|M_1)}\right]; \tag{3}$$

the **first term** on the right is just the **prior odds** in favor of M_2 over M_1, and the **second term** on the right is called the **Bayes factor**, so in words equation (3) says

$$\left(\begin{array}{c}\textbf{posterior}\\ \textbf{odds}\\ \text{for } M_2\\ \text{over } M_1\end{array}\right) = \left(\begin{array}{c}\textbf{prior odds}\\ \text{for } M_2\\ \text{over } M_1\end{array}\right) \cdot \left(\begin{array}{c}\textbf{Bayes factor}\\ \text{for } M_2\\ \text{over } M_1\end{array}\right). \tag{4}$$

Odds o are related to **probabilities** p via $o = \frac{p}{1-p}$ and $p = \frac{o}{1+o}$; these are **monotone increasing transformations**, so the **decision rules** {choose M_2 over M_1 if the **posterior odds** for M_2 are greater} and {choose M_2 over M_1 if $P(M_2|y) > P(M_1|y)$} are **equivalent**.

This approach does have a **decision-theoretic basis**, but it's rather **odd**: if you pretend that the **only possible data-generating mechanisms** are $\mathcal{M} = \{M_1, \ldots, M_m\}$ for finite m, and you pretend that one of the models in $\mathcal{M}$ must

Bayes Factors

be the **true data-generating mechanism**, and you pretend that the **utility function**

$$U(a) = \left\{ \begin{array}{cc} 1 & \text{if your choice of model } a \text{ is } \textbf{correct} \\ 0 & \text{otherwise} \end{array} \right\} \quad (5)$$

reflects your **real-world values**, then it's **decision-theoretically optimal** to choose the model in $\mathcal{M}$ with the **highest posterior probability** (i.e., that choice **maximizes expected utility**).

If it's **scientifically appropriate** to take the **prior model probabilities** $p(M_j)$ to be **equal**, this rule reduces to **choosing the model with the highest Bayes factor in favor of it**; this can be found by (a) **computing the Bayes factor** in favor of M_2 over M_1,

$$BF(M_2 \text{ over } M_1|y) = BF(M_2|y) = \frac{p(y|M_2)}{p(y|M_1)}, \quad (6)$$

favoring M_2 if $BF(M_2|y) > 1$, i.e., if $p(y|M_2) > p(y|M_1)$, and calling the **better model** M^*; (b) **computing the Bayes factor** in favor of M^* over M_3, calling the **better model** M^*; and so on up through M_m.

Notice that there's something else a bit **funny** about this: $p(y|M_j)$ is the **prior** **(not posterior) predictive distribution** for the data set y under model M_j, so the **Bayes factor rule** tells us to **choose the model that does a better job of predicting the data before any data arrives**.

Example. When you come upon a **new concept**, it's a good idea to **play with it** in a **simple setting** where you're pretty sure you know the **right answer**, to see if the new concept **gives you back known truth**; in that spirit, let's look at one of the **simplest possible inferential settings**: for $j = 1, 2$ and $i = 1, \ldots, n$

$$(y_i|\mu_j, M_j) \overset{\text{IID}}{\sim} N(\mu_j, \sigma^2) \quad \text{for known } \mu_j \text{ and } \sigma^2. \quad (7)$$

Bayes Factors (continued)

We can immediately **reduce by sufficiency** to the **equivalent models**

$$(\bar{y}|\mu_j, M_j) \sim N\left(\mu_j, \frac{\sigma^2}{n}\right) \quad \text{where } \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i; \tag{8}$$

in other words, the two models **agree** on the **Gaussian sampling story** and the **variance** but disagree on the **(underlying data-generating) mean**.

Intuition says that you should **favor** the model for which $\bar{y}$ and μ_j are **closest**; let's compute the **Bayes factor** and see if it agrees:

$$\begin{aligned} p(y|M_j) &= p(\bar{y}|M_j) = \frac{1}{\sqrt{2\pi\frac{\sigma^2}{n}}} \exp\left[-\frac{n}{2\sigma^2}(\bar{y}-\mu_j)^2\right], \text{ so} \\ BF(M_2|y) &= \frac{\frac{1}{\sqrt{2\pi\frac{\sigma^2}{n}}} \exp\left[-\frac{n}{2\sigma^2}(\bar{y}-\mu_2)^2\right]}{\frac{1}{\sqrt{2\pi\frac{\sigma^2}{n}}} \exp\left[-\frac{n}{2\sigma^2}(\bar{y}-\mu_1)^2\right]} \\ &= \exp\left\{-\frac{n}{2\sigma^2}\left[(\bar{y}-\mu_2)^2 - (\bar{y}-\mu_1)^2\right]\right\}, \end{aligned} \tag{9}$$

and this is **greater than 1** (i.e., we should **favor** M_2) iff $(\bar{y}-\mu_2)^2 < (\bar{y}-\mu_1)^2$, as **intuition suggested**.

OK so far, but now let's look at the **general problem of parametric model comparison**, in which model M_j has **its own parameter vector** θ_j (of length k_j) and is **specified** by

$$M_j\text{: } \left\{ \begin{array}{c} (\theta_j|M_j) \sim p(\theta_j|M_j) \\ (y|\theta_j, M_j) \sim p(y|\theta_j, M_j) \end{array} \right\}, \tag{10}$$

for $y = (y_1, \ldots, y_n)$.

Bayes Factors (continued)

Here the quantity $p(y|M_j)$ that **defines the Bayes factor** is

$$p(y|M_j) = \int p(y|\theta_j, M_j)\, p(\theta_j|M_j)\, d\theta_j; \tag{11}$$

this is called an **integrated likelihood** (or **marginal likelihood**) because it tells us to take a **weighted average** of the **sampling distribution/likelihood** $p(y|\theta_j, M_j)$, but **NB** **weighed by the prior** for θ_j in model M_j; as noted above, this may seem **surprising**, but it's **correct**, and it can lead to **trouble**, as follows.

The first trouble is **technical**: the **integral** in (11) can be **difficult to compute**, and may not even be much fun to **approximate** (more on this below).

The second thing to **notice** is that (11) can be **rewritten** as

$$p(y|M_j) = E_{(\theta_j|M_j)}\ p(y|\theta_j, M_j). \tag{12}$$

In other words the **integrated likelihood** is the **expectation** of the **sampling distribution** over the **prior** for θ_j in model M_j (evaluated at the **observed data** y); in other words, if **scientific context** suggests that $p(\theta_j|M_j)$ is **diffuse**, this expectation can be **unstable** with respect to **small details** in how the **diffuseness is specified**.

This can be **seen directly** by trying to **approximate** (12) via **Monte Carlo**: the **expectation** suggests that we (a) pick some large number N, (b) make N **IID** draws θ_{ij}^* from the **prior** $p(\theta_j|M_j)$, and (c) compute

$$p(y|M_j) \doteq \frac{1}{N} \sum_{i=1}^{N} p(y|\theta_{ij}^*, M_j). \tag{13}$$

Imagine trying to do this with (e.g.) a $\Gamma(\epsilon, \epsilon)$ prior on a parameter living on $(0, \infty)$, and think about how **unstable** the result would be.

Instability of Bayes Factors

Example: **Gaussian** sampling model with **known mean** μ and **unknown variance**:

$$M_j: \left\{ \begin{array}{c} (\sigma_j^2 | M_j) \sim \Gamma(\epsilon, \epsilon) \\ (y_i | \sigma_j^2, M_j) \overset{\text{IID}}{\sim} N(\mu, \sigma_j^2), \; i = 1, \ldots, n \end{array} \right\}. \tag{14}$$

In this **model**

$$\begin{aligned} p(y|\sigma_j^2, M_j) &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left[-\frac{1}{2\sigma_j^2}(y_i - \mu)^2 \right] \\ &= (2\pi)^{-\frac{n}{2}} \left(\sigma_j^2\right)^{-\frac{n}{2}} \exp\left[-\frac{1}{2\sigma_j^2} \sum_{i=1}^{n} (y_i - \mu)^2 \right]. \end{aligned} \tag{15}$$

Unfortunately, the $\Gamma(\epsilon, \epsilon)$ prior puts **so much of its mass near 0** that about **47%** of the draws from it are regarded by `R` as **exactly equal to 0**:

```
epsilon <- 0.001

M <- 100000

sigma.star2 <- rgamma( M, epsilon, epsilon )

sum( sigma.star2 == 0 ) / M
[1] 0.47424
```

So the **Monte Carlo approximation** to $p(y|M_j)$ in equation (13) will **fail** with the $\Gamma(\epsilon, \epsilon)$ prior unless it's **modified to be bounded away from 0**; why not try a Uniform(ϵ, A) prior **instead**?

```
mu <- 0

n <- 10

y <- rnorm( n, mu, 1 )

s <- sum( ( y - mu )^2 )
```

Instability of Bayes Factors

```
M <- 100000

sensitivity <- function( M, epsilon, A, n, s ) {

  sigma2.star <- runif( M, epsilon, A )

  sampling.distribution <- ( 2 * pi )^( - n / 2 ) *
    sigma2.star^( - n / 2 ) * exp( - s / ( 2 * sigma2.star ) )

  return( c( mean( sampling.distribution) ) )

}

sensitivity( M, 0.001, 5, n, s )
[1] 5.068015e-07

sensitivity( M, 0.01, 5, n, s )
[1] 5.08333e-07

sensitivity( M, 0.1, 5, n, s )
[1] 5.16355e-07
```

OK, that's **good**: with the Uniform(ϵ, A) prior the **Monte Carlo approximation** to $p(y|\sigma_j^2, M_j)$ is **not sensitive** to ϵ; what about A?

```
sensitivity( M, 0.001, 5, n, s )
[1] 5.053105e-07

sensitivity( M, 0.001, 10, n, s )
[1] 2.529732e-07

sensitivity( M, 0.001, 15, n, s )
[1] 1.695316e-07

sensitivity( M, 0.001, 50, n, s )
[1] 5.134128e-08

sensitivity( M, 0.001, 500, n, s )
[1] 4.911668e-09
```

Instability of Bayes Factors

With A the news is **not good**: as A **increases** the **apparent plausibility** of the **same data** under the **same model** goes down; in fact, if A is **increased** by a factor of **10**, $p(y|\sigma_j^2, M_j)$ **goes down by approximately the same factor of 10**; and remember: A is **not a number that comes from the science of the problem** (we haven't even seen any **data** yet).

Example: **Integer-valued** data $y = (y_1, \ldots, y_n)$;

$M_1 =$ **Geometric**(θ_1) likelihood with **Beta**(α_1, β_1) prior on θ_1;

$M_2 =$ **Poisson**(θ_2) likelihood with **Gamma**(α_2, β_2) prior on θ_2.

The **Bayes factor** in favor of M_1 over M_2 turns out to be

$$\frac{\Gamma(\alpha_1+\beta_1)\Gamma(n+\alpha_1)\Gamma(n\bar{y}+\beta_1)\Gamma(\alpha_2)(n+\beta_2)^{n\bar{y}+\alpha_2}\left(\prod_{i=1}^n y_i!\right)}{\Gamma(\alpha_1)\Gamma(\beta_1)\Gamma(n+n\bar{y}+\alpha_1+\beta_1)\Gamma(n\bar{y}+\alpha_2)\beta_2^{\alpha_2}.} \tag{16}$$

Diffuse priors: take $(\alpha_1, \beta_1) = (1, 1)$ and $(\alpha_2, \beta_2) = (\epsilon, \epsilon)$ for some $\epsilon > 0$.

Bayes factor reduces to

$$\frac{\Gamma(n+1)\Gamma(n\bar{y}+1)\Gamma(\epsilon)(n+\epsilon)^{n\bar{y}+\epsilon}\left(\prod_{i=1}^n y_i!\right)}{\Gamma(n+n\bar{y}+2)\Gamma(n\bar{y}+\epsilon)\epsilon^{\epsilon}}. \tag{17}$$

This goes to $+\infty$ as $\epsilon \downarrow 0$, i.e., you can make the evidence in **favor** of the **Geometric model** over the **Poisson** as **large** as you want, **no matter what the data says**, as a function of a quantity near 0 that **scientifically** you have **no basis** to specify.

If instead you **fix and bound** (α_2, β_2) away from 0 and let $(\alpha_1, \beta_1) \downarrow 0$, you can **completely reverse** this and make the evidence in **favor** of the **Poisson model** over the **Geometric** as **large** as you want; and a Uniform$(0, A)$ prior on θ_2 **doesn't help** either.

Instability of Bayes Factors

The **bottom line** is that, when **scientific context** suggests **diffuse priors** on the parameter vectors in the models being compared, the **integrated likelihood values** that are at the heart of **Bayes factors** can be **hideously sensitive** to **small arbitrary details** in how the diffuseness is specified.

This has been **well-known** for quite awhile now, and it's given rise to **an amazing amount of fumbling around**, as people who like Bayes factors have tried to find a way to **fix** the problem: at this point the list of attempts includes {**partial, intrinsic, fractional**} **Bayes factors, well-calibrated priors, conventional priors, intrinsic priors, expected posterior priors**, ... (e.g., Pericchi 2004), and all of them exhibit a level of **ad-hockery** that's **otherwise absent** from the **Bayesian paradigm**.

Approximating integrated likelihoods. We **want**

$$p(y|M_j) = \int p(y|\theta_j, M_j)\, p(\theta_j|M_j)\, d\theta_j; \tag{18}$$

maybe we can find an **analytic approximation** to this that will suggest how to **avoid trouble**.

Laplace (1785) already faced this problem **225 years ago**, and he offered a **solution** that's often useful, which we now call a **Laplace approximation** in his honor (it's also known in the **applied mathematics literature** as a **saddle-point approximation**).

Let* $P^*(\theta_j) = p(y|\theta_j, M_j)\, p(\theta_j|M_j)$; we want an **approximation** for $\int P^*(\theta_j)\, d\theta_j$, in which we notice that $P^*(\theta_j)$ is an **un-normalized probability density** (namely, in our case, the **posterior distribution** $p(\theta_j|y, M_j)$.

*I've drawn on something written by **David Mackay** in creating this explanation of the idea.

Laplace Approximation

Laplace said to himself: with **large** n this posterior distribution should be **close to Gaussian**, centered at the **posterior mode** $\widehat{\theta}_j$; this means that its **logarithm** should be close to **quadratic around that mode**, so let's take a **Taylor expansion** (Brook Taylor, 1685–1731, English mathematician, published **Taylor's theorem** in 1715) of $\log P^*(\theta_j)$ around $\widehat{\theta}_j$ and retain only the terms out to **second order**.

First let's look at this idea **univariately**, with θ_j a vector of length $k_j = 1$: you'll recall that if $g(x)$ is a **function** of a **single real variable** x, then for x near some point x_0,

$$g(x) \doteq g(x_0) + g'(x_0)(x - x_0) + \frac{1}{2} g''(x_0)(x - x_0)^2. \qquad (19)$$

Here $x = \theta_j, x_0 = \widehat{\theta}_j$, and $g(x) = \log P^*(\theta_j)$, from which $g'(x) = \frac{\partial}{\partial \theta_j} \log P^*(\theta_j) = \frac{(P^*)'(\theta_j)}{P^*(\theta_j)}$ and $g'(x_0) = \frac{(P^*)'(\widehat{\theta}_j)}{P^*(\widehat{\theta}_j)} = 0$ because $\widehat{\theta}_j$ is the **mode** of $P^*(\theta_j)$; thus the **approximation** becomes

$$\begin{aligned} \log P^*(\theta_j) &\doteq \log P^*(\widehat{\theta}_j) - \frac{f_j}{2}(\theta_j - \widehat{\theta}_j)^2, \quad \text{where} \\ f_j &= -\frac{\partial^2}{\partial \theta_j^2} \log P^*(\theta_j) \Big|_{\theta_j = \widehat{\theta}_j}. \end{aligned} \qquad (20)$$

Thus $P^*(\theta_j) \doteq P^*(\widehat{\theta}_j) \exp\left[-\frac{f_j}{2}(\theta_j - \widehat{\theta}_j)^2\right]$ and

$$\begin{aligned} p(y|M_j) &= \int p(y|\theta_j, M_j)\, p(\theta_j|M_j)\, d\theta_j = \int P^*(\theta_j)\, d\theta_j \qquad (21) \\ &\doteq p(y|\widehat{\theta}_j, M_j)\, p(\widehat{\theta}_j|M_j) \int_{-\infty}^{\infty} \exp\left[-\frac{f_j}{2}(\theta_j - \widehat{\theta}_j)^2\right] d\theta_j \\ &= p(y|\widehat{\theta}_j, M_j)\, p(\widehat{\theta}_j|M_j) \sqrt{\frac{2\pi}{f_j}}, \end{aligned}$$

Laplace Approximation (continued)

and this can be expressed on the **log scale** as

$$\log p(y|M_j) \doteq \log p(y|\widehat{\theta}_j, M_j) + \log p(\widehat{\theta}_j|M_j) + \frac{1}{2}\log 2\pi - \frac{1}{2}\log f_j. \quad (22)$$

With θ_j a vector of length $k_j > 1$, the details are **similar** except that we base the **approximation** on a **multivariate Gaussian**: you'll probably recall that if $g(x)$ is a **scalar-valued function** of a **vector** x, then for x near some point x_0,

$$g(x) \doteq g(x_0) + (x - x_0)' Dg(x_0) + \frac{1}{2}(x - x_0)' D^2 g(x_0)(x - x_0), \quad (23)$$

where $Dg(x_0)$ is the **gradient** (the **vector of first partial derivatives**) of g evaluated at x_0 and $D^2 g(x_0)$ is the **Hessian** (the **matrix of second partial derivatives**) of g evaluated at x_0.

Here (as before) $x = \theta_j, x_0 = \widehat{\theta}_j$, and $g(x) = \log P^*(\theta_j)$, from which the **linear term vanishes** (as before) and the **approximation** is

$$\log P^*(\theta_j) \doteq \log P^*(\widehat{\theta}_j) - \frac{1}{2}(\theta_j - \widehat{\theta}_j)' \widehat{H}_j (\theta_j - \widehat{\theta}_j), \quad (24)$$

where $\widehat{H}_j$ is **minus the Hessian** of $\log P^*(\theta_j)$ evaluated at $\widehat{\theta}_j$.

This means that $P^*(\theta_j) \doteq P^*(\widehat{\theta}_j) \exp\left[-\frac{1}{2}(\theta_j - \widehat{\theta}_j)' \widehat{H}_j (\theta_j - \widehat{\theta}_j)\right]$, which has a nice **(un-normalized) multivariate Gaussian form**, and therefore

$$\begin{aligned} p(y|M_j) &= \int p(y|\theta_j, M_j)\, p(\theta_j|M_j)\, d\theta_j = \int P^*(\theta_j)\, d\theta_j \\ &\doteq p(y|\widehat{\theta}_j, M_j)\, p(\widehat{\theta}_j|M_j) \cdot \quad (25) \\ &\qquad \int \exp\left[-\frac{1}{2}(\theta_j - \widehat{\theta}_j)' \widehat{H}_j (\theta_j - \widehat{\theta}_j)\right] d\theta_j \\ &= p(y|\widehat{\theta}_j, M_j)\, p(\widehat{\theta}_j|M_j) \left|2\pi \widehat{H}_j{}^{-1}\right|^{\frac{1}{2}}; \end{aligned}$$

Laplace Approximation (continued)

and since $\left|2\pi \hat{H}_j{}^{-1}\right|^{\frac{1}{2}} = (2\pi)^{\frac{k_j}{2}} \left|\hat{H}_j\right|^{-\frac{1}{2}}$ the result on the **log scale** is

$$\begin{aligned}\log p(y|M_j) &\doteq \log p(y|\hat{\theta}_j, M_j) + \log p(\hat{\theta}_j|M_j) \\ &\quad + \frac{k_j}{2}\log 2\pi - \frac{1}{2}\log|\hat{H}_j|, \end{aligned} \tag{26}$$

in which (as a reminder) $\hat{\theta}_j$ is the **posterior mode** of the **parameter vector** θ_j under model M_j and $\hat{H}_j$ is the **(observed) information matrix** (minus the Hessian) of $\log P^*(\theta_j) = \log p(y|\theta_j, M_j) + \log p(\theta_j|M_j)$ under model M_j, evaluated at the **posterior mode** $\hat{\theta}_j$.

This is not quite the **standard way Laplace approximations** are used in computing **Bayes factors**; by analysis of the **terms of third and higher order** in the Taylor expansion, it can be shown that the **error of this approximation** is of **order** $\frac{1}{n}$:

$$\begin{aligned}\log p(y|M_j) &= \log p(y|\hat{\theta}_j, M_j) + \log p(\hat{\theta}_j|M_j) \\ &\quad + \frac{k_j}{2}\log 2\pi - \frac{1}{2}\log|\hat{H}_j| + O\left(\frac{1}{n}\right); \end{aligned} \tag{27}$$

and it can be further shown that the **approximation still holds to order** $\frac{1}{n}$ if (a) you replace the **posterior mode** with the **MLE** (the **likelihood mode**) and (b) you take $\hat{H}_j$ to be $\hat{I}_j$, the **usual observed information matrix** (minus the Hessian of the **log likelihood**, evaluated at the **MLE**), so the **official Laplace approximation** we'll use is

$$\begin{aligned}\log p(y|M_j) &= \log p(y|\hat{\theta}_j, M_j) + \log p(\hat{\theta}_j|M_j) \\ &\quad + \frac{k_j}{2}\log 2\pi - \frac{1}{2}\log|\hat{I}_j| + O\left(\frac{1}{n}\right), \end{aligned} \tag{28}$$

in which $\hat{\theta}_j$ is the **MLE** of the **parameter vector** θ_j under model M_j and $\hat{I}_j$ is the **observed information matrix** for model M_j.

Laplace Approximation (continued)

Example: **Gaussian** sampling model with **known mean** μ and **unknown variance** (continued):

$$M_j\text{: } \left\{ \begin{array}{c} (\sigma_j^2|M_j) \sim \text{Uniform}(\epsilon, A) \\ (y_i|\sigma_j^2, M_j) \overset{\text{IID}}{\sim} N(\mu, \sigma_j^2),\ i = 1, \ldots, n \end{array} \right\}. \tag{29}$$

In this model $\theta_j = \sigma_j^2$ and $k_j = 1$; the **likelihood function** is an **arbitrary positive constant** c times the **sampling distribution** $p(y|\sigma_j^2, M_j) = p(y|\theta_j, M_j)$ noted earlier:

$$l(\theta_j|y, M_j) = c\,(2\pi)^{-\frac{n}{2}}\,(\theta_j)^{-\frac{n}{2}} \exp\left[-\frac{1}{2\theta_j}\sum_{i=1}^{n}(y_i - \mu)^2\right], \tag{30}$$

leading to the **log likelihood function**

$$ll(\theta_j|y, M_j) = c - \frac{n}{2}\log(2\pi) - \frac{n}{2}\log\theta_j - \frac{1}{2\theta_j}\sum_{i=1}^{n}(y_i - \mu)^2, \tag{31}$$

whose **first and second partial derivatives** are

$$\frac{\partial}{\partial\theta_j} ll(\theta_j|y, M_j) = -\frac{n}{2\theta_j} + \frac{1}{2\theta_j^2}\sum_{i=1}^{n}(y_i - \mu)^2 \quad \text{and} \tag{32}$$

$$\frac{\partial^2}{\partial\theta_j^2} ll(\theta_j|y, M_j) = \frac{n}{2\theta_j^2} - \frac{1}{\theta_j^3}\sum_{i=1}^{n}(y_i - \mu)^2;$$

the **MLE** $\widehat{\theta}_j$ solves $\frac{\partial}{\partial\theta_j} ll(\theta_j|y, M_j) = 0$ and is $\widehat{\theta}_j = \frac{s}{n}$, where $s = \sum_{i=1}^{n}(y_i - \mu)^2$ is a **(minimal) sufficient statistic** in this model, and the **observed information** is given by

$$\widehat{I}_j = -\left.\frac{\partial^2}{\partial\theta_j^2} ll(\theta_j|y, M_j)\right|_{\theta_j = \widehat{\theta}_j} = -\frac{n}{2\widehat{\theta}_j^2} + \frac{1}{\widehat{\theta}_j^3}\sum_{i=1}^{n}(y_i - \mu)^2 = \frac{n}{2\widehat{\theta}_j^2}. \tag{33}$$

Laplace Approximation (continued)

Thus the **Laplace approximation** to the **log integrated likelihood** is

$$\begin{aligned} \log p(y|M_j) &\doteq \log p(y|\widehat{\theta}_j, M_j) + \log p(\widehat{\theta}_j|M_j) \\ &\quad + \frac{k_j}{2}\log 2\pi - \frac{1}{2}\log|\widehat{I}_j|, \end{aligned} \tag{34}$$

in which $\log p(y|\widehat{\theta}_j, M_j) = -\frac{n}{2}\log 2\pi - \frac{n}{2}\log\widehat{\theta}_j - \frac{n}{2}$; in this **model** the **prior** is $p(\theta_j|M_j) = \frac{1}{A-\epsilon}I_{(\epsilon,A)}(\theta_j)$, so the **log prior**, evaluated at the MLE (which will be **between** ϵ and A **by choice** of both of those values), is $\log p(\widehat{\theta}_j|M_j) = -\log(A-\epsilon)$; and the **Laplace approximation** finally becomes

$$\begin{aligned} \log p(y|M_j) &\doteq -\frac{n}{2}\log 2\pi - \frac{n}{2}\log\widehat{\theta}_j - \frac{n}{2} - \log(A-\epsilon) \\ &\quad + \frac{1}{2}\log 2\pi - \frac{1}{2}\log\left(\frac{n}{2\widehat{\theta}_j^2}\right). \end{aligned} \tag{35}$$

Here's some `R` code to compare the earlier **Monte Carlo approximation**, to the **log integrated likelihood**, with this **Laplace approximation**:

```
log.integrated.likelihood <- function( n, mu,
  sigma.data.generating, M, epsilon, A ) {

  y <- rnorm( n, mu, sigma.data.generating )

  s <- sum( ( y - mu )^2 )

  sigma2.star <- runif( M, epsilon, A )

  sampling.distribution <- ( 2 * pi )^( - n / 2 ) *
    sigma2.star^( - n / 2 ) * exp( - s / ( 2 * sigma2.star ) )

  monte.carlo.approximation <- log( mean( sampling.distribution ) )

  theta.hat <- s / n
```

Laplace Approximation (continued)

```
  laplace.approximation <- - ( n / 2 ) * log( 2 * pi ) -
    ( n / 2 ) * log( theta.hat ) - n / 2 - log( A - epsilon ) +
    log( 2 * pi ) / 2 - log( n ) / 2 + log( 2 ) / 2 +
    log( theta.hat )

  return( c( monte.carlo.approximation, laplace.approximation ) )

}

mu <- 0

sigma.data.generating <- 1

M <- 100000

epsilon <- 0.001

A <- 5

log.integrated.likelihood( 10, mu,
  sigma.data.generating, M, epsilon, A )

[1] -14.72792 -14.95642

log.integrated.likelihood( 10, mu,
  sigma.data.generating, M, epsilon, A )

[1] -13.25275 -13.49012
```

With different **randomly-generated data sets** the **approximate log integrated likelihood** values bounce around quite a bit, but you can see that the **Laplace approximation** is already **decent** with $n = 10$ in this model, and with $n = 100$ it's **right on the money**:

```
log.integrated.likelihood( 100, mu,
  sigma.data.generating, M, epsilon, A )

[1] -144.1683 -144.1803

log.integrated.likelihood( 100, mu,
  sigma.data.generating, M, epsilon, A )

[1] -157.0197 -157.0448
```

BIC

As derived above, the standard **Laplace approximation** used in evaluating **Bayes factors** is

$$\begin{aligned}\log p(y|M_j) &= \log p(y|\widehat{\theta}_j, M_j) + \log p(\widehat{\theta}_j|M_j) \\ &\quad + \frac{k_j}{2}\log 2\pi - \frac{1}{2}\log|\widehat{I}_j| + O\left(\frac{1}{n}\right), \quad (36)\end{aligned}$$

in which $\widehat{\theta}_j$ and $\widehat{I}_j$ are the **MLE** of the **parameter vector** and the **observed information matrix** under model M_j, respectively.

Notice that the **prior** on θ_j in model M_j enters into this **approximation** through $\log p(\widehat{\theta}_j|M_j)$, and this is a term that **won't go away with more data**: as n increases this term is $O(1)$ (i.e., **bounded** but **not going to 0**).

Using a **less precise Taylor expansion**, Schwarz (1978, *Annals of Statistics*, **6**, 461–464) obtained a **different approximation** that's the basis of what has come to be known as the **Bayesian information criterion (BIC)**:

$$\log p(y|M_j) = \log p(y|\widehat{\theta}_j, M_j) - \frac{k_j}{2}\log n + O(1). \quad (37)$$

To see where it goes, let's work out what **implied prior BIC is using**, from the point of view of the **Laplace approximation**: if the **two approximations** are supposed to be **equal**, then

$$\log p(\widehat{\theta}_j|M_j) = -\frac{k_j}{2}\log(2\pi) - \frac{k_j}{2}\log n + \frac{1}{2}\log|\widehat{I}_j|; \quad (38)$$

what **prior** on θ_j would reduce to **expression** (38) when $\theta_j = \widehat{\theta}_j$?

Right away that makes me think of a **multivariate Gaussian distribution centered** at $\widehat{\theta}_j$, i.e., $N_{k_j}(\widehat{\theta}_j, \Sigma)$ for some Σ; on the **log scale** that prior is

BIC (continued)

$$\log p(\theta_j|M_j) = \log|2\pi\Sigma|^{-\frac{1}{2}} - \frac{1}{2}(\theta_j - \widehat{\theta}_j)'\Sigma^{-1}(\theta_j - \widehat{\theta}_j), \qquad (39)$$

and the **second term vanishes** for $\theta_j = \widehat{\theta}_j$.

So let's **set**

$$\log|2\pi\Sigma|^{-\frac{1}{2}} = -\frac{k_j}{2}\log(2\pi) - \frac{k_j}{2}\log n + \frac{1}{2}\log|\widehat{I}_j| \qquad (40)$$

and **solve** for Σ; the left-hand side **simplifies** to

$$\begin{aligned} \log|2\pi\Sigma|^{-\frac{1}{2}} &= \log\left[(2\pi)^{-\frac{k_j}{2}}|\Sigma|^{-\frac{1}{2}}\right] \\ &= -\frac{k_j}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma|, \qquad (41) \end{aligned}$$

and **equating** and **simplifying** yields

$$\log|\Sigma| = k_j\log n - \log|\widehat{I}_j| = \log|n\widehat{I}_j^{-1}|, \qquad (42)$$

from which we have the **answer**: a **prior** on θ_j in model M_j **implied** by the **BIC** $O(1)$ **approximation** to the **Laplace** $O\left(\frac{1}{n}\right)$ **approximation** is

$$(\theta_j|M_j) \sim N_{k_j}(\widehat{\theta}_j, n\widehat{I}_j^{-1}). \qquad (43)$$

In the **literature** this is called a **unit-information prior**, for the following reason.

With **large** n we know from our earlier **asymptotic** work that when estimating the **parameter vector** θ_j in **model** M_j, the **likelihood function** will be **approximately multivariate Gaussian** — $l(\theta_j|y, M_j) \sim N_{k_j}(\widehat{\theta}_j, \widehat{I}_j^{-1})$ — leading to the **Bayes's-Theorem updating procedure**

$$\begin{aligned} (\theta_j|M_j) &\sim N_{k_j}(\widehat{\theta}_j, n\widehat{I}_j^{-1}) \qquad (44) \\ l(\theta_j|y, M_j) &\sim N_{k_j}(\widehat{\theta}_j, \widehat{I}_j^{-1}). \end{aligned}$$

Unit-Information Prior

Now we know that a **Gaussian mixture of Gaussians is Gaussian**, with **posterior mean vector** in this case given by $\widehat{\theta}_j$ (since both the **prior** and **likelihood** mean vectors **coincide** at that value) and **posterior precision matrix** given by the **sum** of the **prior** and **likelihood precision matrices**, namely

$$\left[(n\widehat{I}_j^{-1})^{-1} + \widehat{I}_j\right] = \frac{n+1}{n}\widehat{I}_j; \tag{45}$$

thus for **large** n the updating rule for θ_j in **model** M_j will be approximately

$$\begin{aligned} (\theta_j|M_j) &\sim N_{k_j}(\widehat{\theta}_j, n\widehat{I}_j^{-1}) \\ l(\theta_j|y, M_j) &\sim N_{k_j}(\widehat{\theta}_j, \widehat{I}_j^{-1}) = N_{k_j}(\widehat{\theta}_j, \frac{n}{n}\widehat{I}_j^{-1}) \\ p(\theta_j|y, M_j) &\sim N_{k_j}(\widehat{\theta}_j, \frac{n}{n+1}\widehat{I}_j^{-1}) \end{aligned} \tag{46}$$

Since the **approximate likelihood distribution** is $N_{k_j}(\widehat{\theta}_j, \frac{n}{n}\widehat{I}_j^{-1})$, the **posterior information** is **identical** to the **likelihood information** except that the **uncertainty** about θ_j has been reduced from $\frac{n}{n}\widehat{I}_j^{-1}$ to $\frac{n}{n+1}\widehat{I}_j^{-1}$, i.e., it's as if the **prior** were **equivalent** to **1 more observation** that **behaves exactly like the data** (hence the name **unit-information prior**).

Example: What does this approach produce for a **unit-information prior** with the **sampling model**

$$(y_i|\mu) \overset{\text{IID}}{\sim} N(\mu, \sigma^2), \quad i = 1, \ldots, n, \tag{47}$$

with σ^2 **known**?

Since $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ is a **minimal sufficient statistic** in this model, we can immediately **reduce by sufficiency** to the **equivalent** and **simpler** model

$$(\bar{y}|\mu) \sim N\left(\mu, \frac{\sigma^2}{n}\right). \tag{48}$$

Unit-Information Prior

In this **model** $k_j = 1$ and $\theta_j = \mu$; the **likelihood function** is

$$l(\mu|\bar{y}) \sim N\left(\bar{y}, \frac{\sigma^2}{n}\right), \tag{49}$$

from which we can **see directly** that $\widehat{I}_j^{-1} = \frac{\sigma^2}{n}$, so a **unit-information prior** is

$$\mu \sim N(\bar{y}, \sigma^2). \tag{50}$$

This also makes **good intuitive sense**: with this **prior** you get the **same posterior** as if you had (a) an **improper, completely flat prior** on μ with **0 information content** and (b) a **data set** of $(n+1)$ observations with **mean** equal to the $\bar{y}$ observed in the **actual data set** of n observations.

Notice that, like **Jeffrey's prior** — $p(\theta_j|M_j) = c|\widehat{I}_j|^{\frac{1}{2}}$, chosen to achieve both **diffuseness** and **invariance under reparameterization** — this **unit-information prior depends on the data**, but in both cases the **dependence is rather gentle**: as we saw in the **example** above, a **unit-information prior** is **equivalent** to (a) a completely **flat** prior and (b) a data set of $(n+1)$ **observations** having the **same sufficient statistics** as those **observed** in the **actual data set**, and with **moderate** n this amounts to a **quite diffuse prior** indeed.

Why BIC? As **derived** above, using the **unit-information prior** (43) we get

$$\log p(y|M_j) = \log p(y|\widehat{\theta}_j, M_j) - \frac{k_j}{2}\log n + O(1). \tag{51}$$

This **approximation** has the **extremely desirable property** that it's **free** of the **hideous instability** of **integrated likelihoods** with respect to **tiny details**, in how **diffuse priors** are specified, that **do not arise directly from the science of the problem**; in my view, if you're going to use **Bayes factors** to choose among models, you're **well advised** to use a **method like BIC** that **protects you from yourself** in **mis-specifying those tiny details**.

Unit-Information Prior (continued)

The previous **unit-information prior** (UIP) could be called an **asymptotically-conjugate** UIP, because it depends on the **likelihood function** being **approximately multivariate normal** with large n; as pointed out by someone in class, this has the potentially **undesirable** property in small samples that it **may not respect the range of possible parameter values** (e.g., placing a **normal** prior on a parameter that lives on $(0, 1)$).

It turns out that in any given problem **there may be more than one UIP**; let's try for a **small-sample-conjugate UIP** that remedies the **range problem**.

Example: **Poisson** sampling model, **conjugate** prior: for $i = 1, \ldots, n$,

$$\begin{array}{rcl} \lambda & \sim & \Gamma(\alpha, \beta) \\ (y_i|\lambda) & \overset{\text{IID}}{\sim} & \text{Poisson}(\lambda); \end{array} \tag{52}$$

with this **model** $(\lambda|y) \sim \Gamma(\alpha + s, \beta + n)$, where $s = \sum_{i=1}^{n} y_i$ (a **sufficient statistic**).

Q: What **choices** of α and β achieve the **goals** of the **UIP** (**prior** worth **one observation**, **posterior** with this prior **equivalent** to {**flat prior** + **data set** of $(n + 1)$ **observations** having the **same sufficient statistics** as those **observed** in the **actual data set**})?

A: In this model the **prior sample size** is β, so set that to **1**; the **posterior mean** with the $\Gamma(\alpha, \beta)$ prior is $\frac{\alpha+s}{\beta+n}$; a **completely flat** prior would be **equivalent** to $\Gamma(0, 0)$, leading to a **posterior mean** of $\bar{y} = \frac{s}{n}$; so **set** $\left(\beta = 1, \frac{\alpha+s}{\beta+n} = \bar{y}\right)$ and **solve** for α, obtaining $\alpha = \bar{y}$; thus a **small-sample-conjugate UIP** for this **sampling model** is $\lambda \sim \Gamma(\bar{y}, 1)$.

Diffuse Priors

Even faster: the **data mean** is $\bar{y}$; the **posterior mean** is a **weighted average** of the **prior** and **data means**; to get a **posterior mean** of $\bar{y}$, set the **prior mean** $\frac{\alpha}{\beta} = \bar{y}$; set the **prior sample size** $\beta = 1$; $\rightarrow \alpha = \bar{y}$.

Example: **Bernoulli** sampling model, **conjugate** prior: for $i = 1, \ldots, n$,

$$\begin{array}{rcl} \theta & \sim & \text{Beta}(\alpha, \beta) \\ (y_i|\theta) & \overset{\text{IID}}{\sim} & \text{Bernoulli}(\theta); \end{array} \tag{53}$$

you can **show** (take-home test 2) that the **small-sample-conjugate UIP** is Beta$(\bar{y}, 1 - \bar{y})$.

Further refinement: ϵ**-information conjugate prior** for any $\epsilon > 0$ — follow this line of reasoning with ϵ in place of a **prior sample size** of 1; in the **Gaussian**, **Poisson** and **Bernoulli sampling models** you get $\mu \sim N\left(\bar{y}, \frac{\sigma^2}{\epsilon}\right)$, $\lambda \sim \Gamma(\epsilon\bar{y}, \epsilon)$, and $\theta \sim \text{Beta}(\epsilon\bar{y}, \epsilon(1 - \bar{y}))$, respectively; is this **worth pursuing** in practice? (take-home test 2).

To **clarify** the **role** of, and **potential difficulties** with, **diffuse** priors:

Bayesian task 1: given your **data set** $y = (y_1, \ldots, y_n)$ and a **parametric model** M_j: $(y|\theta_j, M_j) \sim p(y|\theta_j, M_j)$ (with **parameter vector** θ_j of length k_j) on which you're willing to **condition**, **learn** about θ_j from the **data**, assuming **little relevant information** about θ_j **external** to y.

In this task, as long as n is **decently large** (**relative** to k_j), the **manner** in which you **specify** a **diffuse prior** $p(\theta_j|M_j)$ **won't matter much**; by our earlier **asymptotic results**, **two such priors** will lead to **essentially the same posterior**.

Diffuse Priors (continued)

Bayesian task 2: given y, **compare two or more models** M_j to see if one is **better** than the other.

In this task, if you use **Bayes factors** based on either the **Monte-Carlo approximation**

$$p(y|M_j) \doteq \frac{1}{N}\sum_{i=1}^{N} p(y|\theta_{ij}^*, M_j) \tag{54}$$

(where N is **large** and the θ_{ij}^* are **IID draws** from the prior $p(\theta_j|M_j)$) or the **full Laplace approximation**

$$\begin{aligned}\log p(y|M_j) \;\doteq\; & \log p(y|\widehat{\theta}_j, M_j) + \log p(\widehat{\theta}_j|M_j) \\ & + \frac{k_j}{2}\log 2\pi - \frac{1}{2}\log|\widehat{I}_j|\end{aligned} \tag{55}$$

(in which $\widehat{\theta}_j$ is the **MLE** of the **parameter vector** θ_j under model M_j and $\widehat{I}_j$ is the **observed information matrix** for model M_j), the **manner** in which you **specify** a **diffuse prior** $p(\theta_j|M_j)$ can **matter a great deal**, even with **large** n.

I'm aware of **two remedies**: (a) use **Laplace** with the **asymptotically-conjugate UIP** (which is equivalent to **BIC**), or **Laplace** with some **other UIP** (the **combination** should be **stable** with respect to the **diffuseness**), or (b) do your **model comparison** with **something other than Bayes factors** (more on this below).

Why is **BIC** called the Bayesian **information** criterion?

$$\log p(y|M_j) \doteq \log p(y|\widehat{\theta}_j, M_j) - \frac{k_j}{2}\log n. \tag{56}$$

People often work with a **multiple** of this for **model comparison** (although **Schwarz** himself proposed (56)):

$$BIC(M_j|y) = -2\log p(y|\widehat{\theta}_j, M_j) + k_j \log n \tag{57}$$

(the -2 **multiplier** comes from **deviance** considerations (see below)).

Information Criteria

With the -2 **multiplier**, **good models** have **small values** of BIC.

$$BIC(M_j|y) = -2 \log p(y|\widehat{\theta}_j, M_j) + k_j \log n \qquad (58)$$

The **first** term on the **right** side **rewards** the **quality** of the **model fit**; the **better** the model **fits** the **data**, the **bigger** $\log p(y|\widehat{\theta}_j, M_j)$ will be.

But you can always **make the fit better** just by **including more parameters** (think of the **IHGA** case study, with the **model** $(y_i|\lambda_i) \overset{\text{indep}}{\sim} \text{Poisson}(\lambda_i)$, with n **observations** and n **parameters**; by taking $\widehat{\lambda}_i \; y_i$ the model **fits the data perfectly**), so we need a **penalty** for making the model **overly complicated**.

The **second** term on the **right** side, $k_j \log n$, **penalizes model complexity**, and **BIC** does this at a $\log n$ **rate** for each **new parameter** added; the **model** with the **lowest BIC** will achieve the **best trade-off** between **model fit** and **model complexity** (at least as **BIC measures these quantities**; as we'll see, there are **other ways to measure them**).

By now there are **many proposed information criteria (IC)** for **model selection**, and this is the way most of them **operate**, by creating a **tug of war** between **model fit** and **model complexity**.

In **1974**, reasoning along **different** (and, in my view, **ad hoc**) lines, Hirotugu **Akaike** (now an **emeritus statistician** at the **Institute of Statistical Mathematics** in **Tokyo**) proposed what he called **AIC**:

$$AIC(M_j|y) = -2 \log p(y|\widehat{\theta}_j, M_j) + 2k_j. \qquad (59)$$

He (coyly) proposed the name "**an information criterion**" (AIC) for it; note (importantly) that it **penalizes model complexity** only at an $O(1)$ **rate**.

DIC

By choosing the $k_j \log n$ **penalty for complexity**, BIC has the **arguably important property** of **consistency in model selection**: if the **actual data-generating model** M^* is in the set $\mathcal{M} = \{M_1, M_2, \ldots\}$ over which you're **assessing your model uncertainty**, then as n **increases** with $\mathcal{M}$ fixed the **probability that BIC selects** M^* **goes to 1**.

With the $2k_j$ **penalty for complexity**, AIC **fails to achieve model-selection consistency**; with even **moderate** n you can see that $2k_j < k_j \log n$, so **AIC tends to select overly-complicated models** (compared with the **BIC consistency standard**).

There's another **IC-based Bayesian model-selection criterion** it's good to know about: the **deviance information criterion (DIC)**.

The **deviance** for a model M_j measures the **fit** of M_j (on the **log likelihood scale**) when compared to the **best possible fit** you could achieve; it's defined to be

$$D(M_j|y) = -2\left[\log p(y|\widehat{\theta}_j, M_j) - \log p(y|\widehat{\theta}_S, M_S)\right], \qquad (60)$$

in which $\widehat{\theta}_j$ is the **MLE** under model M_j and $\widehat{\theta}_S$ is the vector of **MLEs** from a **saturated model** M_S with a **parameter for every observation** (so that the **model fit** is **as good as it can be**).

Example: **Gaussian** with **known variance** σ^2 and **unknown mean** μ — M_j: $(y_i|\mu) \overset{\text{IID}}{\sim} N\left(\mu, \sigma^2\right), i = 1, \ldots, n.$

We already know that **in this model** (with $\theta_j = \mu$)

$$\log p(y|\theta_j, M_j) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mu)^2, \qquad (61)$$

DIC (continued)

and the **MLE** for μ is $\bar{y}$, so

$$\log p(y|\widehat{\theta}_j^2, M_j) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \bar{y})^2. \quad (62)$$

Here the **saturated model** corresponding to M_j (one **parameter** for each **data point**) is

$$M_S\colon (y_i|\mu_i) \overset{\text{indep}}{\sim} N\left(\mu_i, \sigma^2\right), i = 1, \ldots, n;$$

thus $\theta_S = (\mu_1, \ldots, \mu_n)$ and

$$\log p(y|\theta_S, M_S) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mu_i)^2; \quad (63)$$

the **MLE** of θ_S is $\widehat{\theta}_S = (y_1, \ldots, y_n)$ and

$$\log p(y|\widehat{\theta}_S, M_S) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log\sigma^2; \quad (64)$$

so the **deviance** is

$$\begin{aligned} D(M_j|y) &= -2\left[\log p(y|\widehat{\theta}_j, M_j) - \log p(y|\widehat{\theta}_S, M_S)\right] \\ &= \frac{1}{\sigma^2}\sum_{i=1}^{n}(y_i - \bar{y})^2. \end{aligned} \quad (65)$$

The reason for the -2 in the definition of **BIC**, **AIC** and the **deviance** is that with the -2 **in front of the log likelihood**, the **asymptotic repeated-sampling behavior** of quantities **related to the deviance** is χ^2; for example, you know from AMS 205 that in the **IID Gaussian model with unknown mean and variance**, $\frac{(n-1)\widehat{\sigma}^2}{\sigma^2} \sim_{RS} \chi^2_{n-1}$ where $\widehat{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2$, but in this case $\frac{(n-1)\widehat{\sigma}^2}{\sigma^2} = D(M_j|y)$ so the **deviance** even has a **small-sample** χ^2 story in this model.

It's often a **pain** to work with the $\log p(y|\widehat{\theta}_S, M_S)$ term in the **deviance** (e.g., sometimes it's not so clear what the **saturated** model should be); fortunately, if you're **comparing two models** M_1 and M_2 on the **same data set**,

DIC (continued)

(a) it's obviously **natural** to work with quantities like $[D(M_2|y) - D(M_1|y)]$ and (b) the **saturated model** is the **same** for both M_1 and M_2, so you **don't have to compute** $\log p(y|\widehat{\theta}_S, M_S)$ because it **cancels in the subtraction**:

$$D(M_2|y) - D(M_1|y) = -2\left[\log p(y|\widehat{\theta}_2, M_2) - \log p(y|\widehat{\theta}_1, M_1)\right], \tag{66}$$

the **difference** of **maximum log likelihood values** under the two models; for this reason the **deviance** is often simply **defined** as

$$D(M_j|y) = -2\log p(y|\widehat{\theta}_j, M_j) + c. \tag{67}$$

whenever it's to be used **solely for model comparison on the same data set**.

OK, now that we know what **deviance** is, what's **DIC**?

Given a **parametric model** M_j: $(y|\theta_j, M_j) \sim p(y|\theta_j, M_j)$, the `WinBUGS` people (Spiegelhalter et al., 2002) define the **deviance information criterion** (DIC) (by analogy with other **information criteria**) to be an estimate $D(\bar{\theta}_j)$ of the model (lack of) **fit** (as measured by the **deviance**: $D(\theta_j) = -2\log p(y|\theta_j, M_j)$) plus a **penalty for complexity** equal to twice the **effective number of parameters** p_{Dj} of the model:

$$DIC(M_j|y) = D(\bar{\theta}_j) + 2\,\widehat{p}_{Dj}, \tag{68}$$

where $\bar{\theta}_j$ is the posterior mean of θ_j; they suggest that models with **low DIC** values are to be **preferred** over those with higher values.

When p_{Dj} is **difficult to read directly from the model** (e.g., in **complex hierarchical models**, especially those with **random effects**), they motivate the following **estimate**, which is easy to compute from **standard MCMC output**:

DIC (continued)

$$\widehat{p}_{Dj} = \overline{D(\theta_j)} - D(\bar{\theta}_j), \tag{69}$$

i.e., the difference between the **posterior mean of the deviance** and the **deviance evaluated at the posterior mean** of the parameter vector (`WinBUGS` has a button to **estimate** these quantities, but it's also easy to **write your own MCMC code** to **estimate DIC**: $-2 \log p(y|\theta_j, M_j)$ is just a **scalar-valued function of the parameter vector** that can be **monitored** as **another column** in the **MCMC data set**).

With **this particular way** of estimating p_{Dj}, DIC becomes

$$DIC(M_j|y) = 2\overline{D(\theta_j)} - D(\bar{\theta}_j). \tag{70}$$

So far this just looks like a **Bayesian version of AIC**, and indeed **DIC** (like **AIC**) is **not asymptotically consistent** (this **doesn't bother** Spiegelhalter et al.: "We are not greatly concerned about this: we **neither believe in a true model** nor would we **expect** the **list of models being considered** to **remain static** as the **sample size increases**"); it does have **three advantages** over AIC:

- it's **(small-sample) Bayesian**, whereas **AIC** is **(large-sample) likelihood-based** (i.e., with **small** n and **non-Gaussian likelihoods**, DIC's emphasis on **posterior means** rather than **likelihood modes** may yield **better calibration performance** in **choosing a good model**);

- it's **readily computed** from **MCMC output**; and

- it provides a way to **estimate the effective number of parameters** in a model in settings (e.g., **hierarchical models with random effects**) in which it's **not obvious** how to **count the number of parameters**.

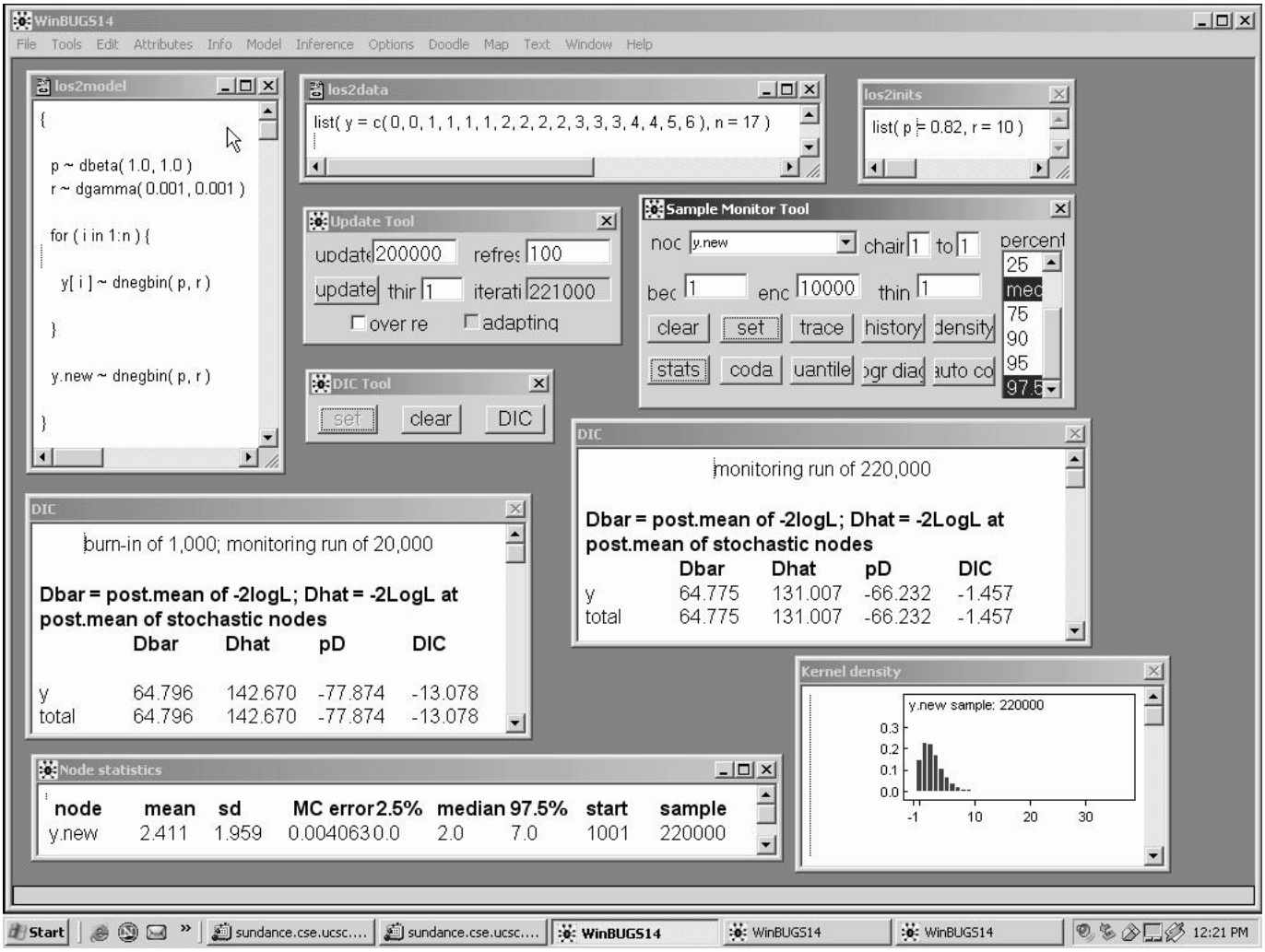
But **DIC** also has a **big disadvantage**: it turns out that, because of the **line of reasoning** leading to $\widehat{p}_{Dj}$,

DIC (continued)

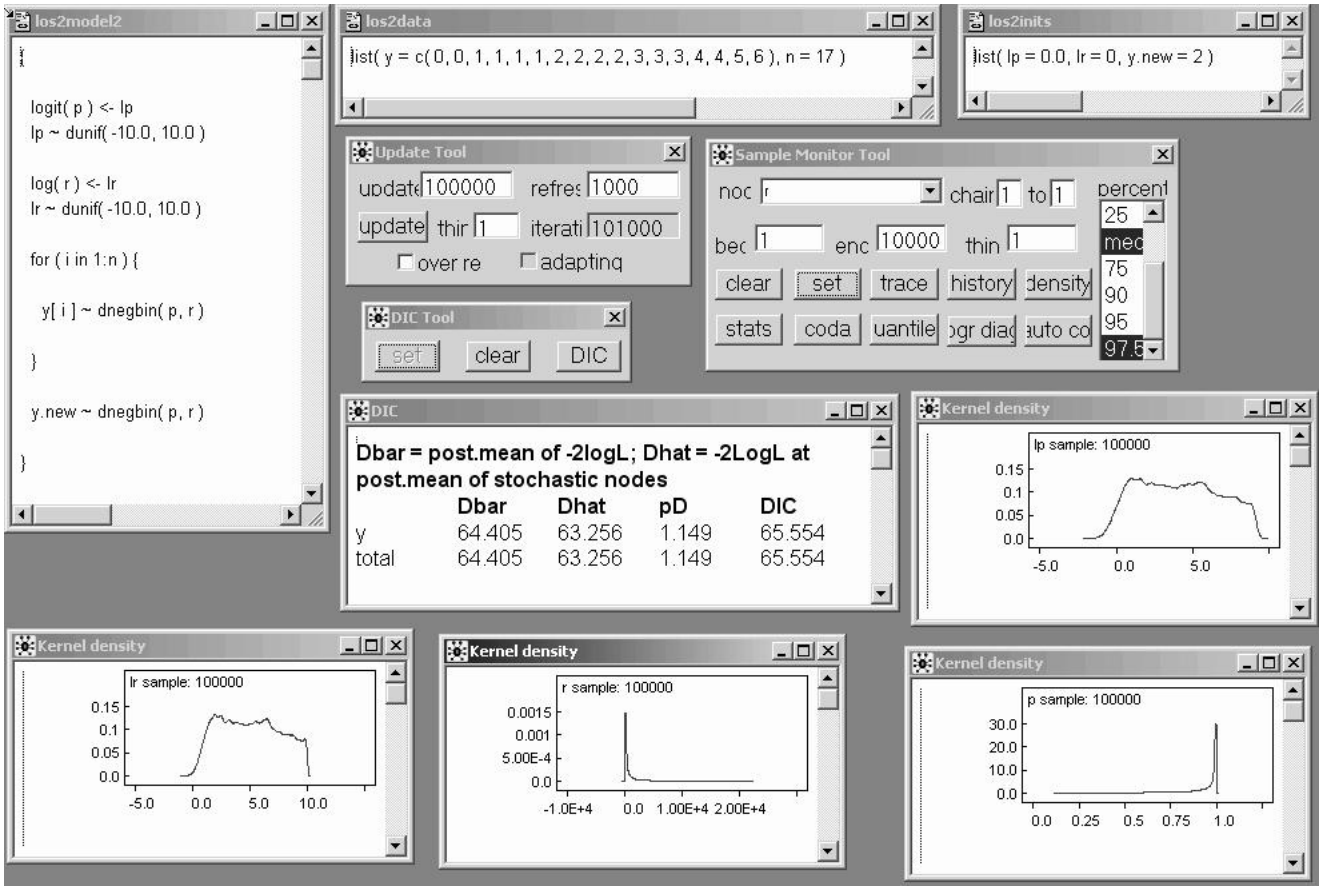
the **approximation** $\overline{D(\theta_j)} - D(\bar{\theta}_j)$ **only works well if you can find a parameterization in which the posterior distribution** $p(\theta_j|y, M_j)$ **is close to (multivariate) normal**.

Example: $y = (0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 4, 4, 5, 6)$ is a **data set** generated from the **negative binomial** distribution with parameters $(p, r) = (0.82, 10.8)$ (in `WinBUGS` notation); y has `mean 2.35` and `VTMR 1.22`.

Using **standard diffuse priors** for p and r as in the `BUGS` examples manuals, the **effective number of parameters** p_D for the negative binomial model (which **fits the data quite well**) is estimated at **−66.2**, leading to a **DIC value** of **−1.5**:



DIC (continued)



The **problem**, as mentioned earlier, is that in the **original parameterization**, $0 < p < 1$, $r > 0$ and the **marginal posteriors** for both of these quantities were **violently non-normal**; forcing `WinBUGS` to **parameterize** instead in terms of $lp = \log\left(\frac{p}{1-p}\right)$ and $lr = \log(r)$ (by placing **priors** on these **transformed parameters** instead of on p and r) **improves** the **DIC approximation dramatically**, but the **estimate** of p_{Dj} is **still too low by 43%** (the **correct number of parameters** is **2**, and DIC **estimates** it as **1.15**).

The problem is **even worse in mixture models** — for example, working with a **sampling distribution** that's a **mixture of two Gaussians**, as in $p(y_i|\gamma, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2, M_j) = \gamma N(y; \mu_1, \sigma_1^2) + (1-\gamma)N(y; \mu_2, \sigma_2^2)$; in these models **DIC performs so badly that the** `WinBUGS` **people had to disable the DIC button**.

A Predictive Criterion

I said back on page 6 of this set of notes that **it would be good** to have a rather **general-purpose utility-based** way to decide if M_2 is **better** than M_1, and I promised **two ideas** along these lines; **Bayes factors** were the **first idea**; here's the **second**.

A hallmark of **good scientific work** is that **good models make good predictions and bad models make bad predictions**; this suggests developing a **utility structure** based on **predictive accuracy**.

Consider first a setting in which the y_i are **real-valued** and the **models** to be compared are (as before)

$$M_j: \left\{ \begin{array}{c} (\theta_j|M_j) \sim p(\theta_j|M_j) \\ (y|\theta_j, M_j) \sim p(y|\theta_j, M_j) \end{array} \right\}, \quad (71)$$

for $y = (y_1, \ldots, y_n)$.

I can readily construct a **posterior predictive distribution** for a new data value y^* under **each model**:

$$\begin{aligned} p(y^*|y, M_j) &= \int p(y^*|\theta_j, y, M_j)\, p(\theta_j|y, M_j)\, d\theta_j \\ &= \int p(y^*|\theta_j, M_j)\, p(\theta_j|y, M_j)\, d\theta_j. \end{aligned} \quad (72)$$

This is **similar to an integrated likelihood** except that we take the **weighted average** of the **sampling distribution**, weighted by the **posterior** for θ_j in model M_j, which is reason for **excitement**: all of that **nonsense** about **diffuse priors** in **Bayes factors** will **disappear** in this approach.

How should I **assess** the **quality** of model M_j's **predictions**?

If I had **new data values** y^*, I could **compare them** (in some way) with their **predictive distributions** $p(y^*|y, M_j)$ under **all of the models**, and the model with the **best predictions** would be **favored**; how should this be **quantified**?

Log Scoring

Evidently I need **two things**: **new data values** y^*, and a way to **compare** a **number** y^* with a **predictive distribution** $p(y^*|y, M_j)$ for it.

One **natural approach** to obtaining **new data values** is **cross-validation**: **partition** the data into **two (non-overlapping and exhaustive) subsets** y_M (for **modeling**) and y_V (for **validation**), and fit **predictive distributions** $p(y_V|y_M, M_j)$ for the **validation values** given the **modeling values**.

A **simple** way to do this is with a **jack-knife (leave-one-out)** form of **cross-validation**: let y_{-i} stand for the **data set** y with observation i **omitted** and **compare** y_i with $p(y_i|y_{-i}, M_j)$ for all $i = 1, \ldots, n$.

OK, so how do I **compare a data point** y_i with its **predictive distribution** $p(y_i|y_{-i}, M_j)$?

This is called the problem of **scoring a predictor**, and it's been given a lot of thought in fields like **meteorology** (where people **predict aspects of the weather** every day).

Good scoring rules (according to **definitions** of the following terms that I'll omit for brevity; see, e.g., O'Hagan and Forster, 2004) are **impartial, symmetric** and **proper**; **math fact:** all **impartial, symmetric** and **proper scoring rules** are **linear functions** of the **logarithm** of the **height** of the **predictive density** $p(y_i|y_{-i}, M_j)$ at the **actual observed value** y_i (i.e., log scores).

Once we have these **log scores** $\log p(y_i|y_{-i}, M_j)$, one for each **data value**, a natural way to **combine them** is to **average them**.

This suggest a **model-selection criterion** I'll call the **cross-validation log score** LS_{CV}:

$$LS_{CV}(M_j|y) = \frac{1}{n}\sum_{i=1}^{n} \log p(y_i|y_{-i}, M_j). \qquad (73)$$

Log Scoring (continued)

This can be given a direct **decision-theoretic justification**: with a **utility function** for model j given by

$$U(M_j|y) = \log p(y^*|y, M_j), \tag{74}$$

where y^* is a **future data value**, the **expectation** in the process of **maximizing expected utility** (MEU) is over our **uncertainty about** y^*; this expectation can be **estimated** (assuming **exchangeability**) by $LS_{CV}(M_j|y)$.

The **naive** approach to calculating LS_{CV}, when **MCMC** is needed to compute the **predictive distributions**, requires n MCMC runs, **one for each omitted observation**; it would be nice to have a **version of log scoring** that could be evaluated with a **single MCMC run** for each model.

This motivates what **Draper and Krnjajić** (2010) call the **full-sample log score**: in the **one-sample situation**, for instance, compute a **single predictive distribution** $p(y^*|y, M_j)$ for a **future data value** y^* with each model M_j under consideration, based on the **entire data set** y (without omitting any observations), and define (cf. Laud and Ibrahim, 1995)

$$LS_{FS}(M_j|y) = \frac{1}{n} \sum_{i=1}^{n} \log p(y_i|y, M_j). \tag{75}$$

This **uses the data twice**, but does so in a way that **matters less and less** as n increases (and already **matters little** for even **moderate** n).

Remarkably, Draper and Krnjajić (2010) have shown that not only is LS_{FS} **faster to compute** than naive implementations of LS_{CV}, it can actually do a **better job of model discrimination in small samples** than either LS_{CV} or **DIC** (see my 4 Feb 2010 talk posted on the course web page).

• The **log score approach** works equally well with **parametric** and **nonparametric** Bayesian models; DIC is **only defined for parametric models**.

Log Scoring (continued)

• When **parametric** model M_j with parameter vector θ_j is fit via **MCMC**, the **predictive ordinate** $p(y^*|y, M_j)$ in LS_{FS} is **easy to approximate**: with m **identically distributed (not necessarily independent)** MCMC **monitoring** draws $(\theta_j)_k^*$ from $p(\theta_j|y, M_j)$,

$$\begin{aligned} p(y^*|y, M_j) &= \int p(y^*|\theta_j, M_j)\, p(\theta_j|y, M_j) d\theta_i \\ &= E_{(\theta_j|y,M_j)}\left[p(y^*|\theta_j, M_j)\right] \\ &\doteq \frac{1}{m}\sum_{k=1}^{m} p(y^*|(\theta_j)_k^*, M_j), \quad \text{and} \\ LS_{FS}(M_j|y) &\doteq \frac{1}{n}\sum_{i=1}^{n} \log\left[\frac{1}{m}\sum_{k=1}^{m} p(y_i|(\theta_j)_k^*, M_j)\right]. \end{aligned} \tag{76}$$

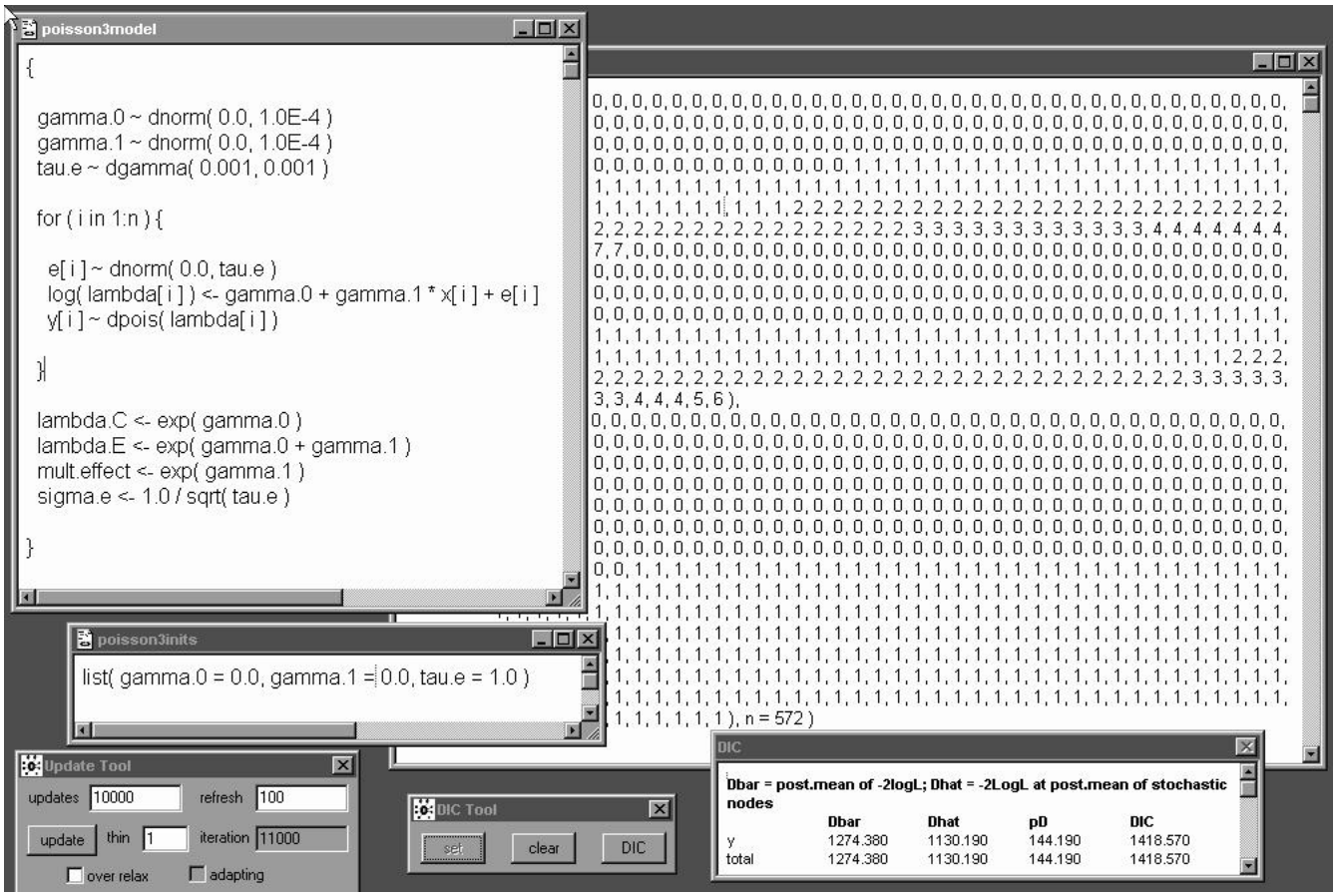
Example: Revisiting the IHGA case study, four **possible models** for the data (not all of them good):

- **Two-independent-sample Gaussian** (diffuse priors);
- **One-sample Poisson** (diffuse prior), pretending treatment and control λs are equal;
- **Two-independent-sample Poisson** (diffuse priors), which is equivalent to **fixed-effects Poisson regression** (FEPR); and
- **Random-effects Poisson regression** (REPR), because C and T **variance-to-mean ratios** (VTMRs) are 1.63 and 1.32, respectively:

$$\begin{aligned} (y_i\,|\lambda_i) &\overset{\text{indep}}{\sim} \text{Poisson}(\lambda_i) \\ \log(\lambda_i) &= \beta_0 + \beta_1 x_i + e_i \\ e_i &\overset{\text{IID}}{\sim} N\left(0, \sigma_e^2\right) \\ \left(\beta_0, \beta_1, \sigma_e^2\right) &\sim \text{diffuse}\ , \end{aligned} \tag{77}$$

where $x_i = 1$ is a **binary indicator** for T/C status.

IHGA example



To use the **DIC feature** in `WinBUGS` to produce the screen shot above, I **fit** the REPR model as usual, did a **burn-in** of 1,000, **selected** `DIC` as a pull-down option from the `Inference` menu, **clicked** the `set` button in the `DIC Tool` window that popped up, **changed** the 1,000 to 10,000 in the `updates` window of the `Update Tool`, **clicked** `update`, and then **clicked** `DIC` in the `DIC Tool` when the monitoring run of 10,000 was finished—the `DIC` **results window** appears, with the `Dbar` ($\overline{D(\theta)}$), `Dhat` ($D(\bar{\theta})$), `pD` ($\widehat{p}_D$), and `DIC` ($DIC(y)$) values.

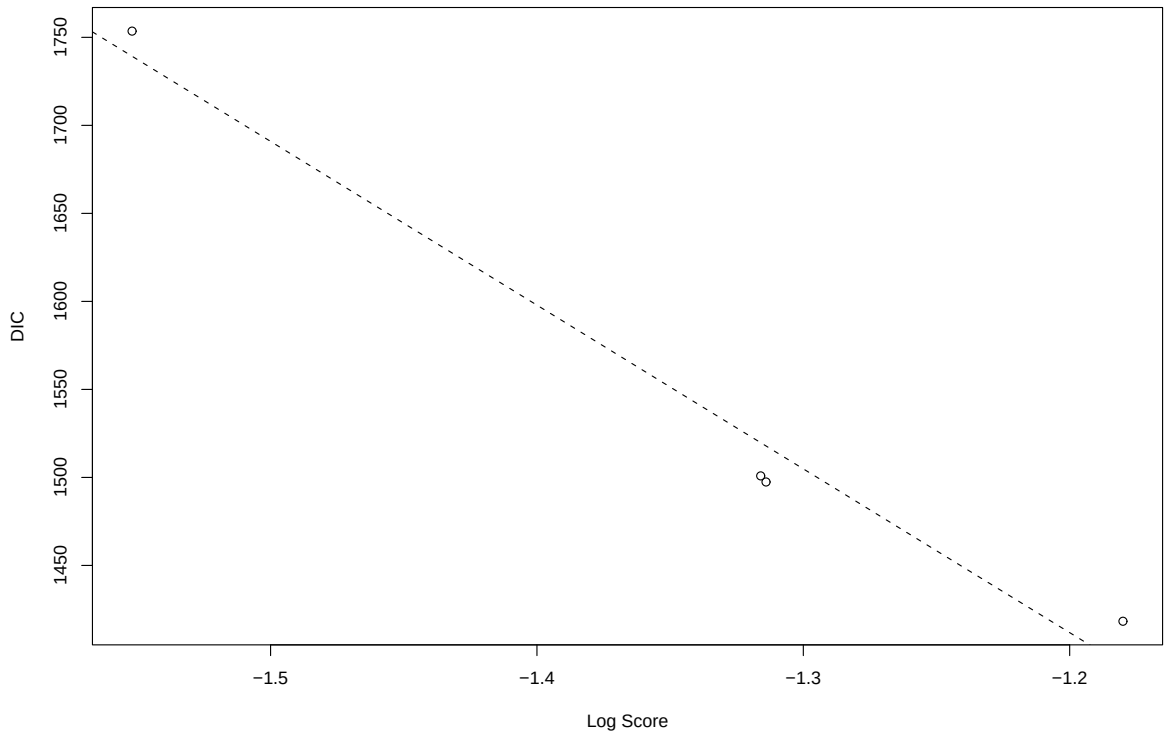
IHGA example (continued)

DIC and **LS** results on these four models:

Model	$\overline{D(\theta)}$	$D(\bar{\theta})$	$\widehat{p}_D$	$DIC(y)$	$LS(y)$
1 (Gaussian)	1749.6	1745.6	3.99	1753.5	-1.552
2 (Poisson, common λ)	1499.9	1498.8	1.02	1500.9	-1.316
3 (FEPR, different λs)	1495.4	1493.4	1.98	1497.4	-1.314
4 (REPR)	1275.7 1274.7 1274.4	1132.0 1131.3 1130.2	143.2 143.5 144.2	1418.3 1418.2 1418.6	-1.180

(3 REPR rows were based on **different monitoring runs**, all of length 10,000, to give idea of Monte Carlo noise level.)

As $\sigma_e \to 0$ in the **REPR** model, you get the **FEPR** model, with $p_D = 2$ parameters; as $\sigma_e \to \infty$, in effect **all subjects in study have their own** λ and p_D would be 572; in between at $\sigma_e \doteq 0.675$ (the posterior mean), `WinBUGS` estimates that there are about **143 effective parameters in REPR model**, but its deviance $D(\bar{\theta}_j)$ is so much lower that it **wins the DIC contest** hands down.



The **correlation** between LS and DIC across these four models is **−0.98**.

Example of LS_{FS} Calculations

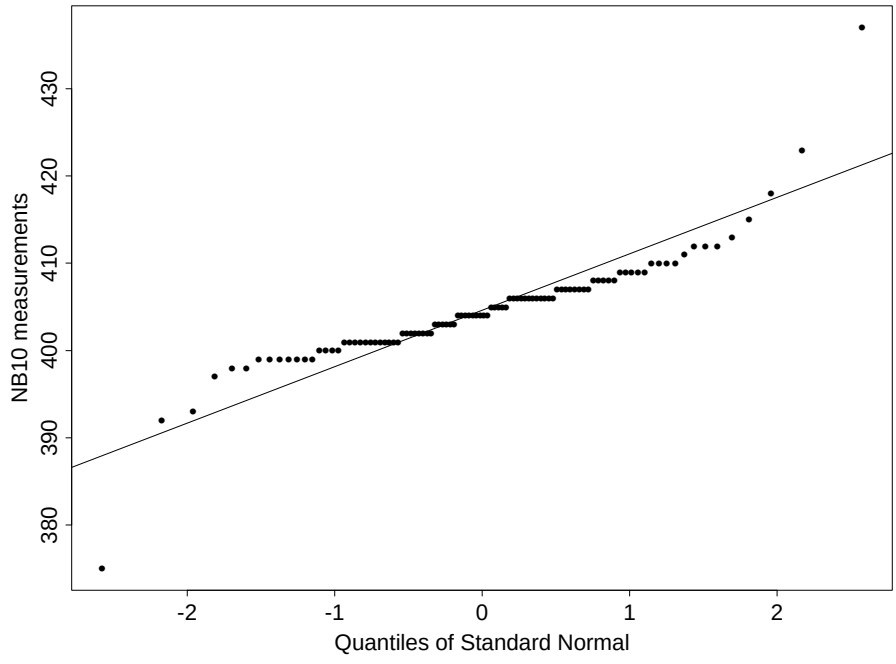
Case Study: **Measurement of physical constants**. What used to be called the National Bureau of Standards (NBS) in Washington, DC, conducts extremely high precision measurement of physical constants, such as the actual weight of so-called **check-weights** that are supposed to serve as reference standards (like the official kg).

In 1962–63, for example, $n = 100$ weighings (listed below) of a block of metal called **NB10**, which was supposed to weigh exactly 10g, were made under conditions **as close to IID as possible** (Freedman et al., 1998); measurements are given as **micrograms below 10g**.

Value	375	392	393	397	398	399	400	401
Frequency	1	1	1	1	2	7	4	12
Value	402	403	404	405	406	407	408	409
Frequency	8	6	9	5	12	8	5	5
Value	410	411	412	413	415	418	423	437
Frequency	4	1	3	1	1	1	1	1

Q: How much does NB10 **really weigh**?

The graph below is a **normal qqplot** of the 100 measurements $y = (y_1, \ldots, y_n)$, which have a mean of $\bar{y} = 404.6$ (the units are **micrograms below 10g**) and an SD of $s = 6.5$.



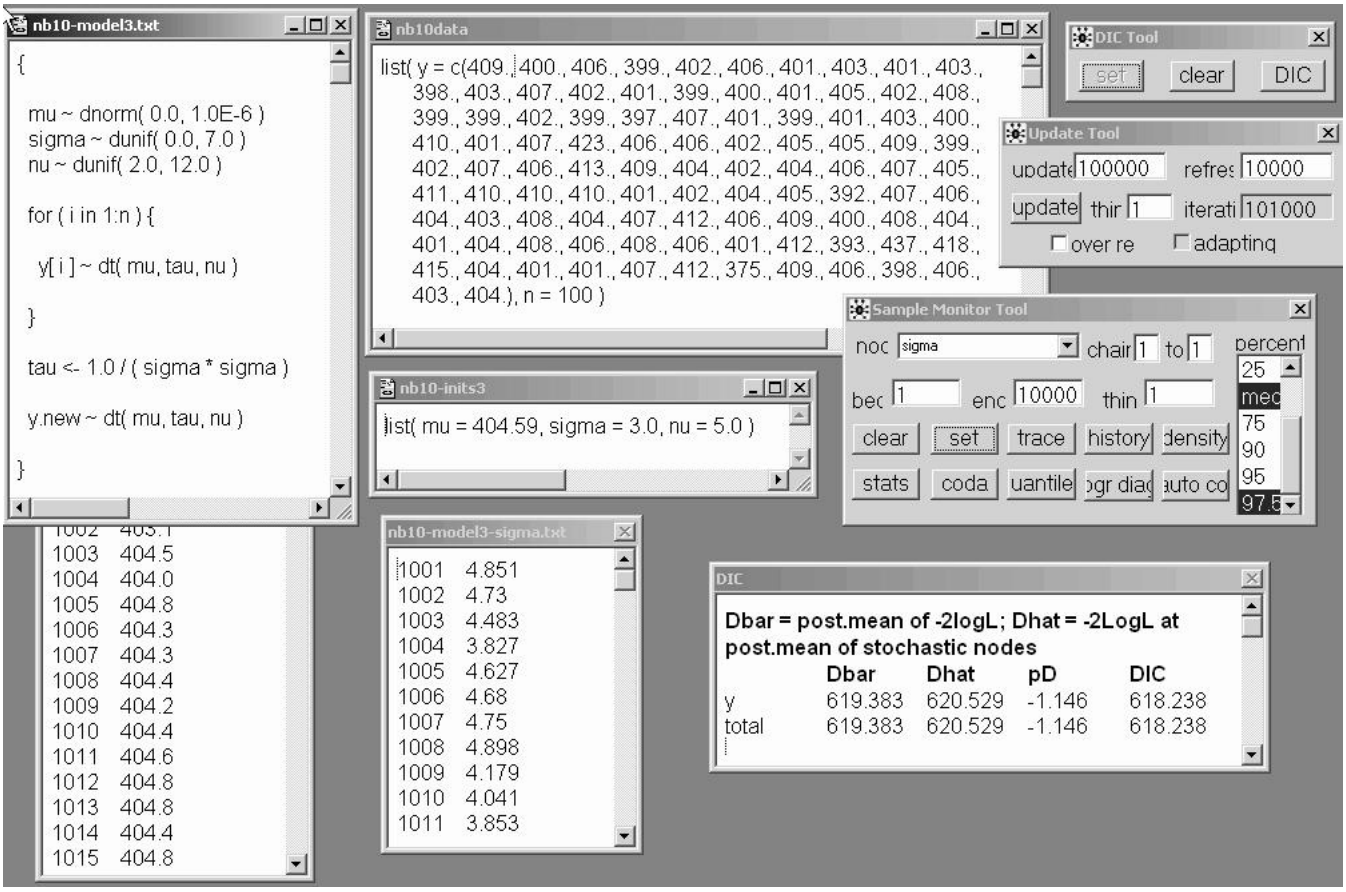
LS_{FS} Calculations (continued)

You can see that the data are **symmetric but heavy-tailed**; two models to compare would be M_1: **Gaussian with unknown mean and variance** (probably **doesn't fit well**) and M_2: t with **unknown mean, variance and shape** (should **fit better**).

The **files** `ams207-nb10-model2.txt`, `ams207-nb10-data.txt`, and `ams207-nb10-inits2.txt` on the course web page contain the `WinBUGS` **implementation** of

$$M_2\text{: } \mu \sim N(0, \text{precision} = \text{1.0E-6}), \sigma \sim U(0, 7.0),$$
$$\nu \sim U(2.0, 12.0), (y_i|\mu, \sigma, \nu) \overset{\text{IID}}{\sim} t_\nu(\mu, \sigma^2)$$

I've **stored** the μ, σ and ν columns of the **MCMC data set** from this model in **files** called `nb10-model2-mu.txt`, `nb10-model2-sigma.txt` and `nb10-model2-nu.txt`, respectively.



The DIC value for this model is **618.2** (note that DIC has **misbehaved** again: p_{D2} is estimated to be **−1.1**; I tried

LS_{FS} Calculations (continued)

diffuse priors on $\log \sigma$ and $\log \nu$ to improve $\widehat{p}_{D2}$, but that made it **worse (−4.4)**).

I go through a **similar process** with the files `nb10-model1.txt`, `nb10-data.txt`, and `nb10-inits1.txt` to fit

$$M_1\text{: } \mu \sim N(0, \text{precision} = \text{1.0E-6}), \sigma \sim U(0, 9.0),$$
$$(y_i | \mu, \sigma) \overset{\text{IID}}{\sim} N(\mu, \sigma^2)$$

and store the μ and σ columns of the MCMC data set in files called `nb10-model1-mu.txt` and `nb10-model1-sigma.txt`, respectively; this time the DIC value is **660.1** and DIC is **better-behaved** (p_D is estimated to be **1.9**, which is **about right**).

On the basis of DIC I would conclude that M_2 (**618.2** with 3 parameters) is (substantially) better than M_1 (**660.1** with 2).

Here's some `R` **code** (also available on the web page) to compute the **log score** values for **both models**.

```
> y <- dget( "nb10-data.txt" )

> y <- sort( y$y )

> mu.t <- matrix( scan( "nb10-model2-mu.txt" ),
    100000, 2, byrow = T )[ , 2 ]

> sigma.t <- matrix( scan( "nb10-model2-sigma.txt" ),
    100000, 2, byrow = T )[ , 2 ]

> nu.t <- matrix( scan( "nb10-model2-nu.txt" ),
    100000, 2, byrow = T )[ , 2 ]

> mu.G <- matrix( scan( "nb10-model1-mu.txt" ),
    100000, 2, byrow = T )[ , 2 ]

> sigma.G <- matrix( scan( "nb10-model1-sigma.txt" ),
    100000, 2, byrow = T )[ , 2 ]
```

LS_{FS} Calculations (continued)

```
> dt.s <- function( y, mu, sigma, nu ) {

>   exp( lgamma( ( nu + 1 ) / 2 ) - ( ( nu + 1 ) / 2 ) *
>     log( 1 + ( y - mu )^2 / ( nu * sigma^2 ) ) -
>     lgamma( nu / 2 ) - log( nu * pi ) / 2 - log( sigma ) )

> }

> LS.contributions <- matrix( 0, 100, 2 )

> for ( j in 1:100 ) {

>   LS.contributions[ j, 1 ] <- log( mean( dt.s( y[ j ],
>     mu.t, sigma.t, nu.t ) ) )

>   LS.contributions[ j, 2 ] <- log( mean( dnorm( y[ j ],
>     mu.G, sigma.G ) ) )

> }

> cbind( y, LS.contributions,
>   0 + LS.contributions[ , 1 ] > LS.contributions[ , 2 ] )

                                t
                                better
                                than
              t       Gaussian  G

  [1,] 375 -8.586208 -12.204954 1
  [2,] 392 -5.349809  -4.639139 0
  [3,] 393 -5.077313  -4.362693 0
  [4,] 397 -3.903555  -3.475233 0
  [5,] 398 -3.602015  -3.309458 0
  [6,] 398 -3.602015  -3.309458 0
  [7,] 399 -3.307381  -3.166624 0
  [8,] 399 -3.307381  -3.166624 0
```

LS_{FS} Calculations (continued)

```
 [9,] 399 -3.307381  -3.166624 0
[10,] 399 -3.307381  -3.166624 0
[11,] 399 -3.307381  -3.166624 0
[12,] 399 -3.307381  -3.166624 0
[13,] 399 -3.307381  -3.166624 0
[14,] 400 -3.028685  -3.046933 1
[15,] 400 -3.028685  -3.046933 1
[16,] 400 -3.028685  -3.046933 1
[17,] 400 -3.028685  -3.046933 1
[18,] 401 -2.778176  -2.950552 1
[19,] 401 -2.778176  -2.950552 1
[20,] 401 -2.778176  -2.950552 1
[21,] 401 -2.778176  -2.950552 1
[22,] 401 -2.778176  -2.950552 1
[23,] 401 -2.778176  -2.950552 1
[24,] 401 -2.778176  -2.950552 1
[25,] 401 -2.778176  -2.950552 1
[26,] 401 -2.778176  -2.950552 1
[27,] 401 -2.778176  -2.950552 1
[28,] 401 -2.778176  -2.950552 1
[29,] 401 -2.778176  -2.950552 1
[30,] 402 -2.571441  -2.877618 1
[31,] 402 -2.571441  -2.877618 1
[32,] 402 -2.571441  -2.877618 1
[33,] 402 -2.571441  -2.877618 1
[34,] 402 -2.571441  -2.877618 1
[35,] 402 -2.571441  -2.877618 1
[36,] 402 -2.571441  -2.877618 1
[37,] 402 -2.571441  -2.877618 1
[38,] 403 -2.426129  -2.828236 1
[39,] 403 -2.426129  -2.828236 1
[40,] 403 -2.426129  -2.828236 1
[41,] 403 -2.426129  -2.828236 1
[42,] 403 -2.426129  -2.828236 1
[43,] 403 -2.426129  -2.828236 1
[44,] 404 -2.358212  -2.802475 1
```

LS_{FS} Calculations (continued)

```
[45,] 404 -2.358212  -2.802475 1
[46,] 404 -2.358212  -2.802475 1
[47,] 404 -2.358212  -2.802475 1
[48,] 404 -2.358212  -2.802475 1
[49,] 404 -2.358212  -2.802475 1
[50,] 404 -2.358212  -2.802475 1
[51,] 404 -2.358212  -2.802475 1
[52,] 404 -2.358212  -2.802475 1
[53,] 405 -2.376305  -2.800373 1
[54,] 405 -2.376305  -2.800373 1
[55,] 405 -2.376305  -2.800373 1
[56,] 405 -2.376305  -2.800373 1
[57,] 405 -2.376305  -2.800373 1
[58,] 406 -2.477698  -2.821932 1
[59,] 406 -2.477698  -2.821932 1
[60,] 406 -2.477698  -2.821932 1
[61,] 406 -2.477698  -2.821932 1
[62,] 406 -2.477698  -2.821932 1
[63,] 406 -2.477698  -2.821932 1
[64,] 406 -2.477698  -2.821932 1
[65,] 406 -2.477698  -2.821932 1
[66,] 406 -2.477698  -2.821932 1
[67,] 406 -2.477698  -2.821932 1
[68,] 406 -2.477698  -2.821932 1
[69,] 406 -2.477698  -2.821932 1
[70,] 407 -2.649778  -2.867123 1
[71,] 407 -2.649778  -2.867123 1
[72,] 407 -2.649778  -2.867123 1
[73,] 407 -2.649778  -2.867123 1
[74,] 407 -2.649778  -2.867123 1
[75,] 407 -2.649778  -2.867123 1
[76,] 407 -2.649778  -2.867123 1
[77,] 407 -2.649778  -2.867123 1
[78,] 408 -2.875393  -2.935880 1
[79,] 408 -2.875393  -2.935880 1
[80,] 408 -2.875393  -2.935880 1
```

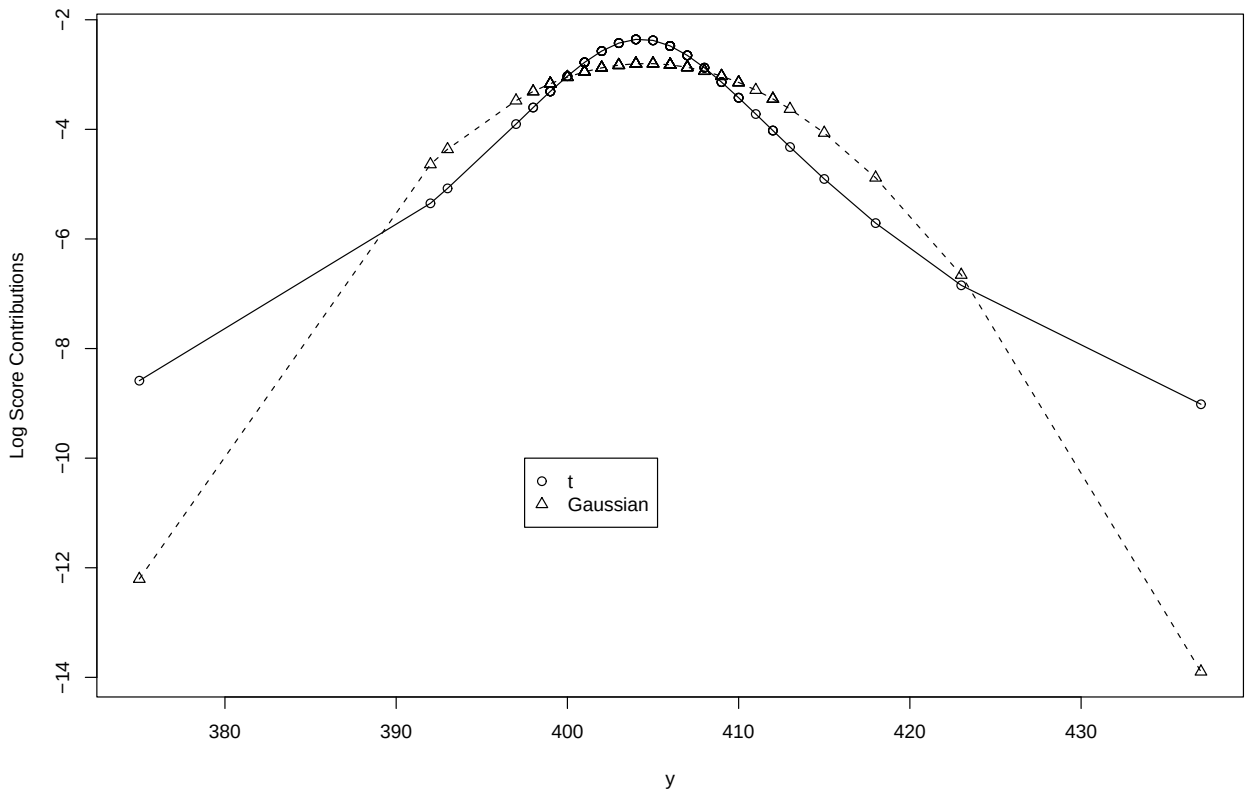
LS_{FS} Calculations (continued)

```
 [81,] 408 -2.875393  -2.935880 1
 [82,] 408 -2.875393  -2.935880 1
 [83,] 409 -3.137771  -3.028107 0
 [84,] 409 -3.137771  -3.028107 0
 [85,] 409 -3.137771  -3.028107 0
 [86,] 409 -3.137771  -3.028107 0
 [87,] 409 -3.137771  -3.028107 0
 [88,] 410 -3.422943  -3.143672 0
 [89,] 410 -3.422943  -3.143672 0
 [90,] 410 -3.422943  -3.143672 0
 [91,] 410 -3.422943  -3.143672 0
 [92,] 411 -3.720225  -3.282413 0
 [93,] 412 -4.021816  -3.444136 0
 [94,] 412 -4.021816  -3.444136 0
 [95,] 412 -4.021816  -3.444136 0
 [96,] 413 -4.322196  -3.628616 0
 [97,] 415 -4.905384  -4.064801 0
 [98,] 418 -5.710652  -4.882504 0
 [99,] 423 -6.845648  -6.656119 0
[100,] 437 -9.016222 -13.896384 1

> sum( LS.contributions[ , 1 ] > LS.contributions[ , 2 ] ) /
>   length( y )

[1] 0.71

# Thus t model is predictively better than Gaussian for
# 71% of the data points.

LS.t <- mean( LS.contributions[ , 1 ] )

LS.G <- mean( LS.contributions[ , 2 ] )

c( LS.t, LS.G )

[1] -3.082331 -3.262142
```

LS_{FS} Calculations (continued)

Although it's not immediately **obvious**, the **log score** for the t model (-3.08) is **substantially higher** than that for the Gaussian model (-3.26), so LS and DIC have reached the **same conclusion** here.

```
> plot( y, LS.contributions[ , 1 ],
>   ylim = c( min( LS.contributions ),
>   max( LS.contributions ) ),
>   ylab = 'Log Score Contributions' )

> lines( y, LS.contributions[ , 1 ], lty = 1 )

> points( y, LS.contributions[ , 2 ], pch = 2 )

> lines( y, LS.contributions[ , 2 ], lty = 2 )

> legend( 397.5, -10, c( "t", "Gaussian" ), pch = c( 1, 2 ) )
```



The t model **fits better** both **in the tails** (where the **most influential observations** are from the Gaussian point of view) and in the **center** (where **most** of the data values are).

Is M_1 good enough?

On page 5 of this set of notes I argued that the **two basic questions** in **model search** are **"Is M_1 better than M_2?** and **"Is M_1 good enough?"**; we've talked quite a bit about the **former question**; what about the **latter**?

As I argued on page 5, answering the question **"Is M_1 good enough?"** requires first answering **another question — "Good enough for what purpose?"** — and this makes answering both of the two basic questions a **decision problem**, requiring the specification of a **utility function** that quantifies your **value judgments** among **good and bad possibilities**.

So this question — **"Is M_1 good enough?"** — can't be answered in a **general** way; but we can make progress on a **general answer** to a **related** question: **"Could the data have arisen from M_1?"**.

This is a **model-checking** question, and there are **many ways** to try to answer it: all sorts of **model-specific diagnostics** will occur to you (**graphical** [e.g., **plot residuals**] and **numerical**), and you should **use** them (e.g., the **variance-to-mean-ratios** in the E and C groups in the **IHGA case study** were **substantially greater than 1**, so the IGHA data **could not have come from a Poisson model**).

But here's another **general-purpose tool** along these lines: M_1 gives a particular value of $LS_{FS}(M_1|y)$ with the actual data set y; how **unusual** is this value if M_1 really were the **data-generating mechanism**?

To answer this question we can **simulate** from M_1 many times, **developing** a distribution of LS_{FS} values, and **see how unusual** the actual data set's **log score** is in this distribution (Draper and Krnjajić, 2010).

Posterior Predictive Model-Checking

This is related to the **posterior predictive model-checking** method of Gelman, Meng and Stern (1996; see GCSB chapter 6); however, this sort of thing cannot be done **naively**, or the result will be **poor calibration** — indeed, Robins et al. (2000) demonstrated that the Gelman et al. procedure may be (sharply) **conservative** (and yet GCSB have **not changed** their **posterior predictive text** to **reflect this** in the **current** (second) **edition** of their book, published in **2004**).

Using a **modification** of an idea in Robins et al., Milovan and I have developed a method for **accurately calibrating the log score scale**.

Inputs to our procedure: (1) A **data set** (e.g., with regression structure); (2) A **model** (can be parametric, non-parametric, or semi-parametric).

Simple example: data set $y = (1, 2, 2, 3, 3, 3, 4, 6, 7, 11)$, $n = 10$.

Given **model** $(*)$

$$\begin{aligned} (\lambda) &\sim \text{Gamma}(0.001, 0.001) \\ (y_i|\lambda) &\overset{\text{IID}}{\sim} \text{Poisson}(\lambda) \end{aligned} \tag{78}$$

Step 1:

Calculate LS_{FS} for this data set; say you get $LS_{FS} = -1.1$; call this the **actual log score** (ALS).

Obtain the posterior for λ given y based on this data set; call this the **actual posterior**.

Calibrating LS_{FS} Scale

Step 2:

```
for ( i in 1:m1 ) {

  make a lambda draw from the actual posterior;
    call it lambda[ i ]

  generate a data set of size n from the second
    line of model (*) above, using
    lambda = lambda[ i ]

  compute the log score for this generated
    data set; call it LS[ i ]

}
```

The **output of this loop** is a **vector of log scores**; call this **V.LS**.

Locate the ALS in this distribution of LS_{FS} values by computing the percentage of LS_{FS} values in **V.LS** that are $\leq$ ALS; call this percentage the **unadjusted actual tail area** (say this is 0.22).

So far this is just Gelman et al. with LS_{FS} as the **discrepancy function**.

Milovan and I know from our own simulations and the literature (Robins et al. 2000) that this tail area (a p-value for a **composite null hypothesis**, e.g., Poisson(λ) with λ unspecified) is **conservative**, i.e., with the 0.22 example above an **adjusted version** of it that's well calibrated would be **smaller**.

In other words, if **Gelman** gives you a value of **0.01** you know that there's a **problem** with your model (because the **correct (calibrated) value** would be **even smaller**), but if he gives you a value of **0.40** the **correct (calibrated) value** will be **smaller** and might be as small as (say) **0.04**, which might well lead you to a **different conclusion**.

Calibrating $LSFS$ Scale (continued)

We've **modified** and implemented one of the ways suggested by **Robins** et al., and we've shown that it **does indeed work** even in **rather small-sample situations**, although our approach to implementing the basic idea can be **computationally intensive**.

Step 3:

```
for ( j in 1:m2 ){

  make a lambda draw from the actual posterior;
    call it lambda*.

  generate a data set of size n from the second line
    of model (*) above, using lambda = lambda*;
    call this the simulated data set

  repeat steps 1, 2 above on this
    simulated data set

}
```

The result will be a vector of unadjusted tail areas; call this **V.P**.

Compute the percentage of tail areas in V.P that are $\leq$ the unadjusted actual tail area; this is the **adjusted actual tail area**.

Calibrating LS_{FS} Scale (continued)

The claim is that the 3-step procedure above is **well-calibrated**, i.e., if the sampling part of model $(*)$ really did generate the observed data, the distribution of adjusted actual tail areas obtained in this way would be **uniform**, apart from simulation noise.

Step 3 in this procedure **solves the calibration problem** by applying the old idea that if $X \sim F_X$ then $F_X(X) \sim U(0,1)$.

This claim can be verified by building a **big loop** around steps 1–3 as follows:

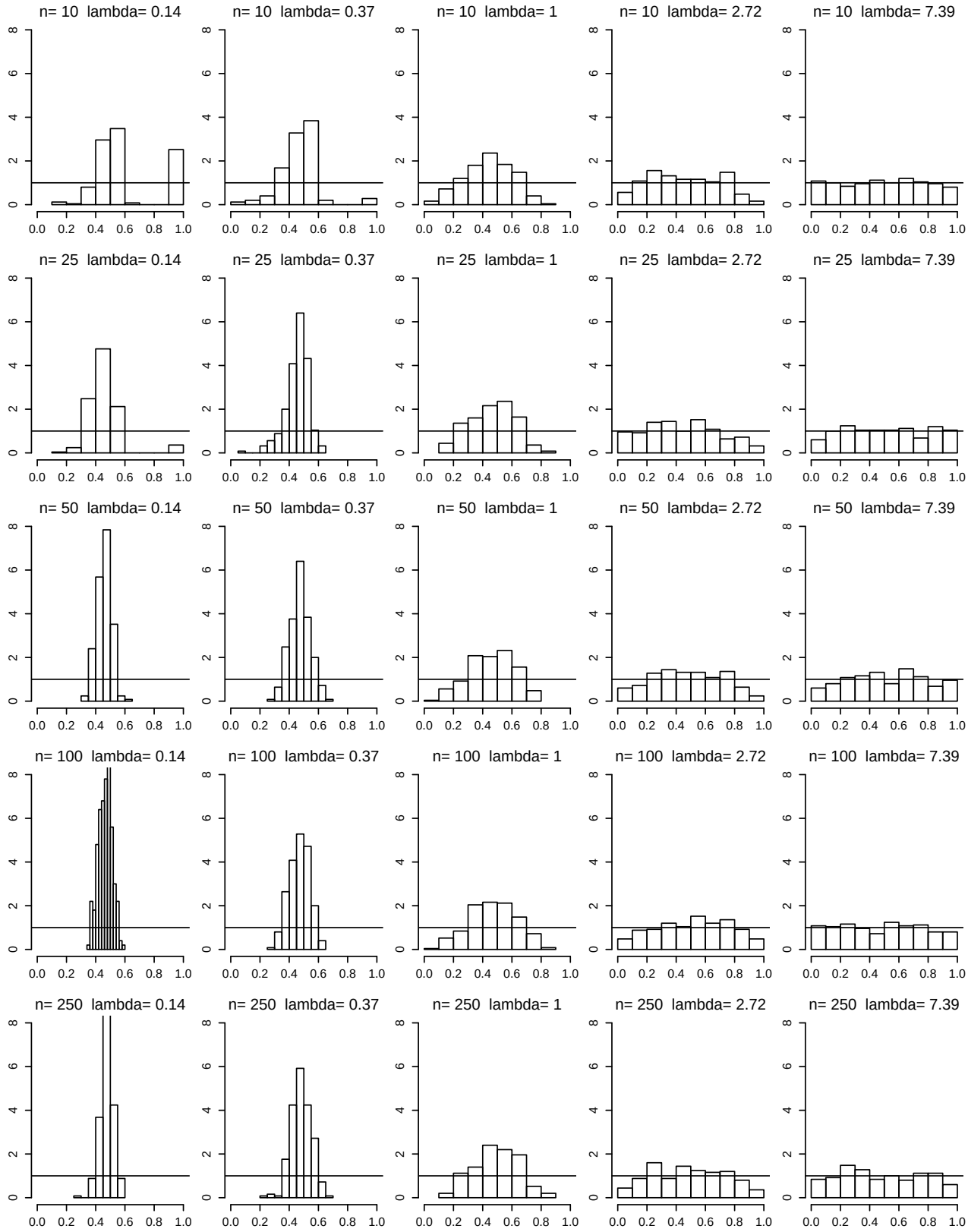
```
Choose a lambda value of interest; call it lambda.sim

for ( k in 1:m3 ) {

  generate a data set of size n from the
    second line of model (*) above, using
    lambda = lambda.sim; call this the
    validation data set

  repeat steps 1-3 on the validation data set

}
```

The result will be a vector of **adjusted P-values**; call this **V.Pa**.

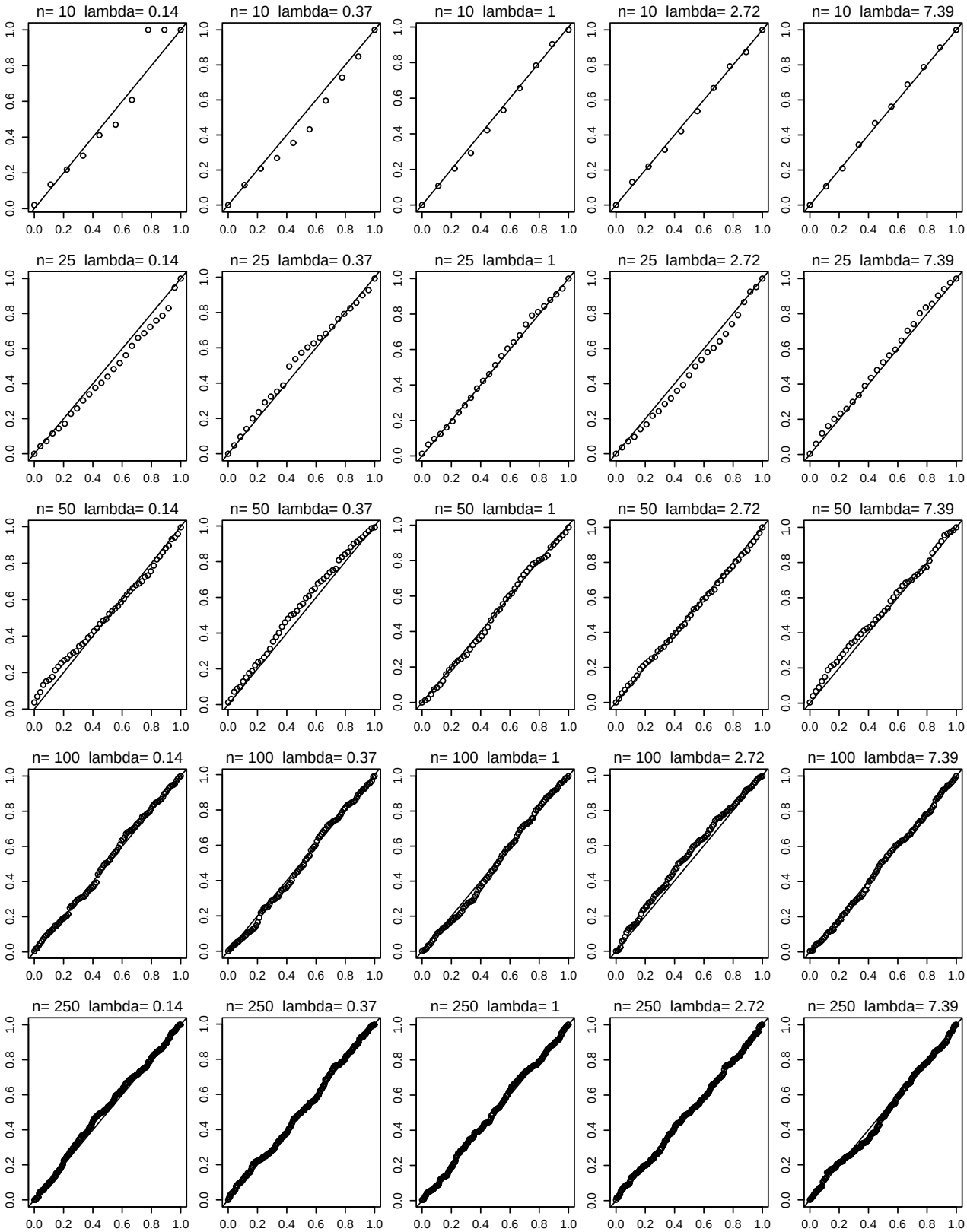
We have **verified** (via simulation) in several simple (and some less simple) situations that the values in V.Pa are close to $U(0,1)$ in distribution.

Two **examples**—Poisson(λ) and Gaussian(μ, σ^2):

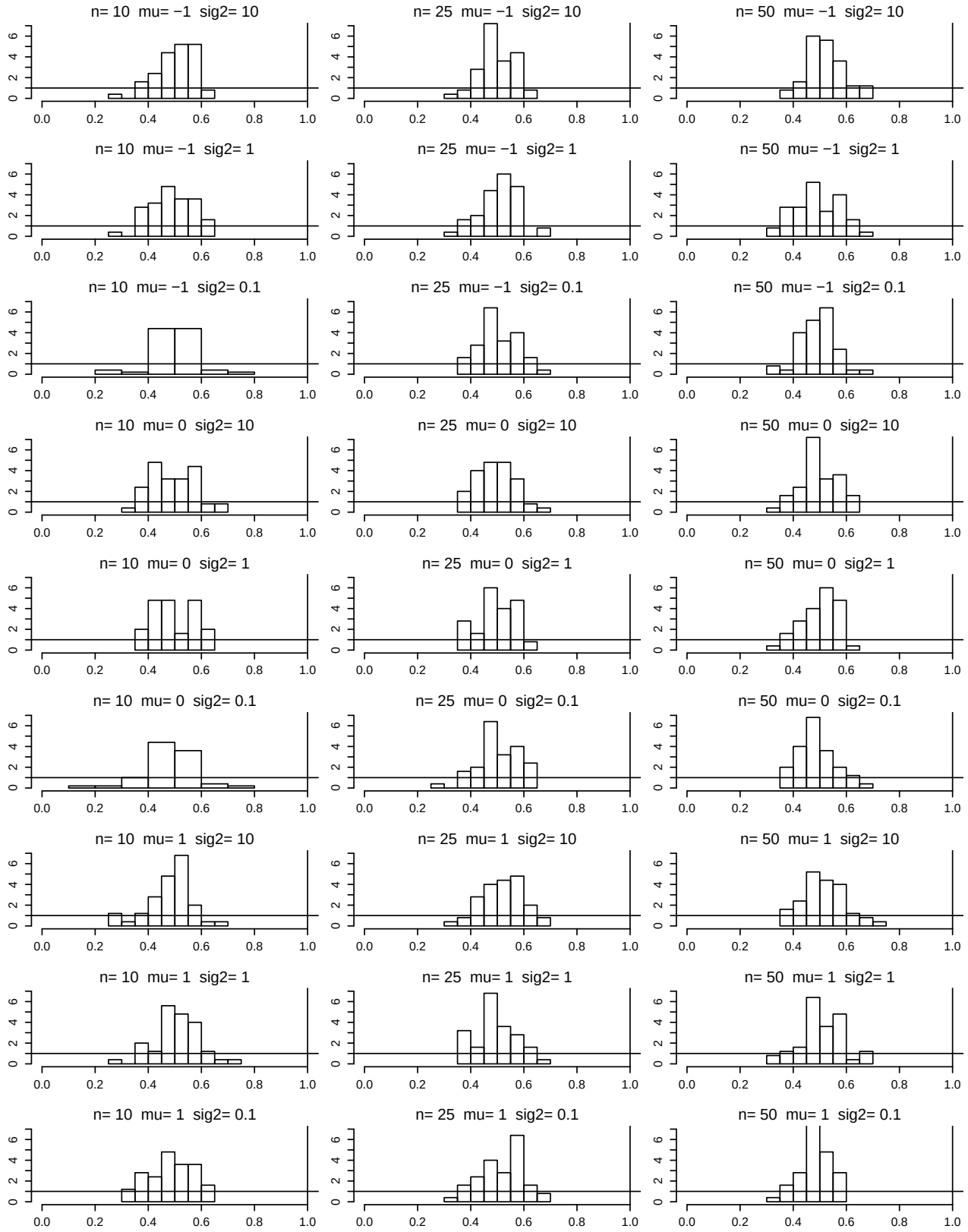
Uncalibrated p-values



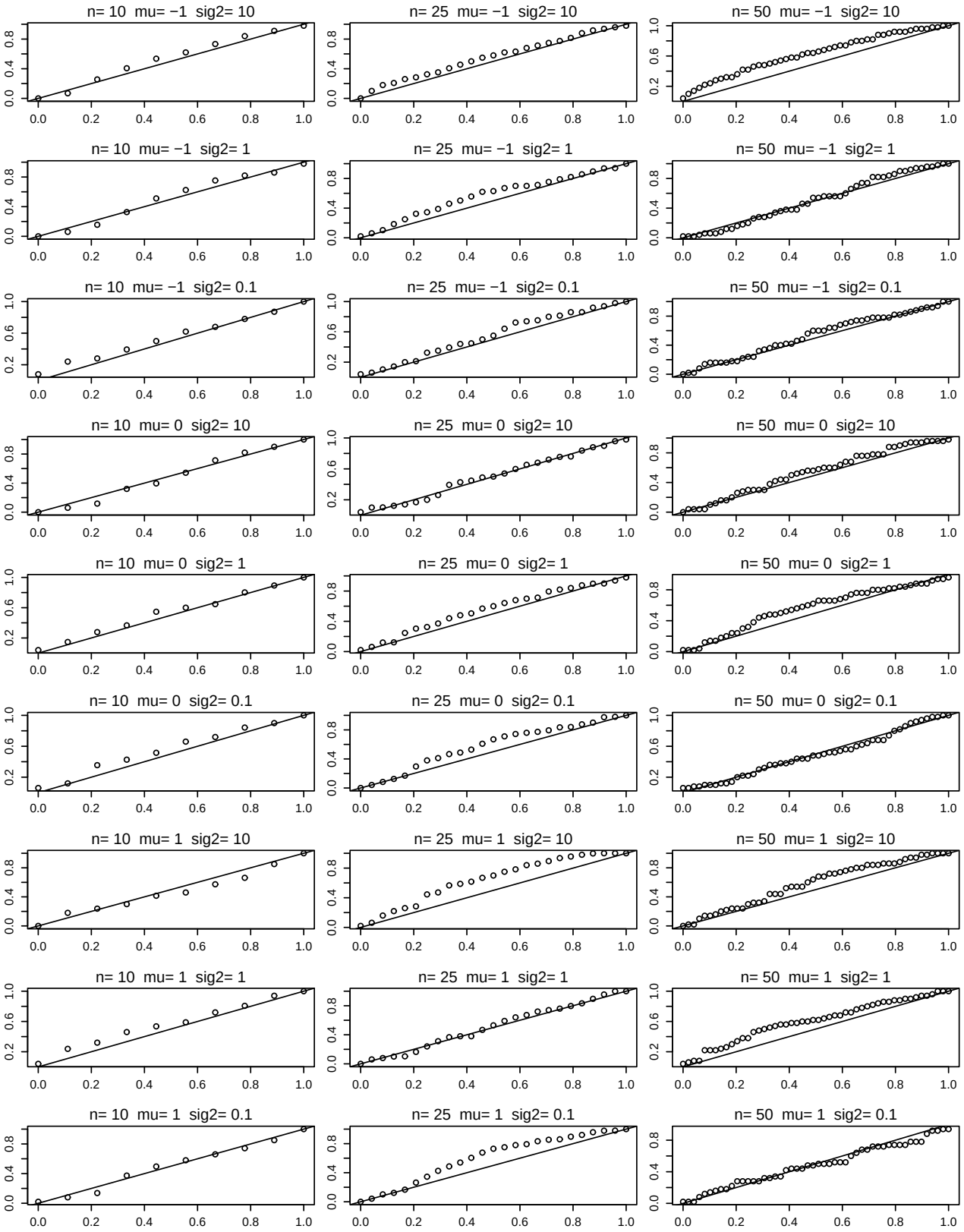
Calibrated p-values



Uncalibrated p-values



Calibrated p-values



R Implementation

Here's some R code (available at the course web site) to **implement** our method for **calibrating** the **log score** scale in a **one-sample Poisson** setting, applied first to a simple data set on **length of stay (LoS)** in the **hospital** for **mothers** admitted to **give birth** and then to a **simulated data set** that was **not generated by the Poisson model**.

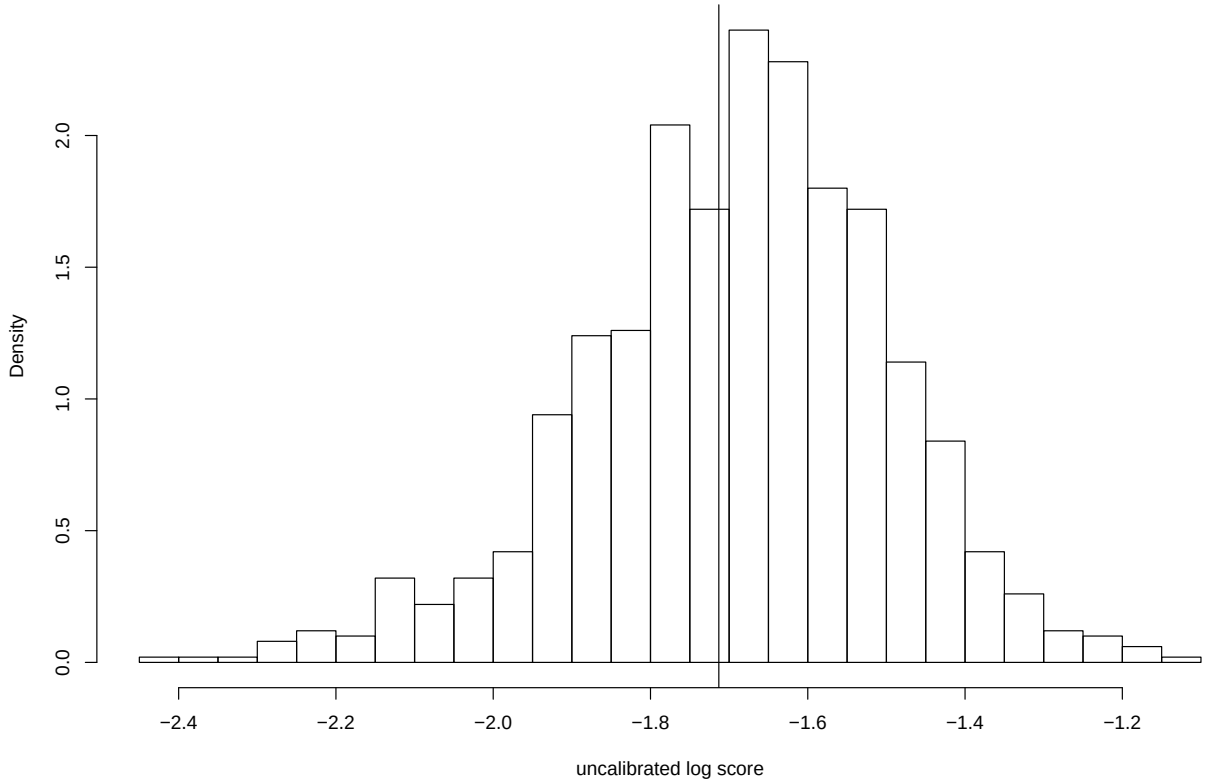
```
> print( y <- c( 0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 4, 6 ) )
 [1] 0 1 1 1 1 1 2 2 2 2 3 3 4 6

> print( epsilon <- 0.001 )
[1] 0.001

> ln.poisson.gamma <- function( y, alpha, beta ) {
+
+   lgamma( alpha + y ) + alpha * log( beta /
+     ( beta + 1 ) ) + y * log( 1 / ( beta + 1 ) ) -
+     lgamma( alpha ) - lgamma( y + 1 )
+
+ }

> step1 <- function( y, epsilon ) {
+
+   n <- length( y )
+
+   s <- sum( y )
+
+   als <- mean( ln.poisson.gamma( y, epsilon + s,
+     epsilon + n ) )
+
+   return( c( n, s, als ) )
+
+ }

> print( step1.result <- step1( y, epsilon ) )
[1] 14.00000 29.00000 -1.71309
```

So the **actual log score** for the LoS data set is -1.71, but is this **unusually small if the data really were Poisson**?

R Implementation (continued)

```
> step2 <- function( n, s, epsilon, als, m1 ) {
+
+   lambda <- rgamma( m1, epsilon + s, epsilon + n )
+
+   ls <- rep( 0, m1 )
+
+   for ( i in 1:m1 ) {
+
+     y.star <- rpois( n, lambda[ i ] )
+
+     s.star <- sum( y.star )
+
+     ls[ i ] <- mean( ln.poisson.gamma( y.star,
+       epsilon + s.star, epsilon + n ) )
+
+   }
+
+   uata <- sum( ls <= als ) / m1
+
+   write( ls, "ls.out" )
+
+   return( uata )
+
+ }

> m1 <- 1000
>
> print( step2.result <- step2( step1.result[ 1 ],
+   step1.result[ 2 ], epsilon, step1.result[ 3 ], m1 ) )
[1] 0.418

> v.ls <- scan( "ls.out" )
Read 1000 items
>
> hist( v.ls, nclass = 20, probability = T,
+   main = '', xlab = 'uncalibrated log score' )
>
> abline( v = step1.result[ 3 ] )
```

R Implementation (continued)



The **actual log score** doesn't look at all **unusual** in this plot, but recall from the discussion above that it **may not yet be properly calibrated**.

```
> step3 <- function( y, epsilon, m1, m2 ) {
+
+   step1.result <- step1( y, epsilon )
+
+   n <- step1.result[ 1 ]
+
+   s.actual <- step1.result[ 2 ]
+
+   uata <- step2( step1.result[ 1 ], step1.result[ 2 ],
+    epsilon, step1.result[ 3 ], m1 )
+
+   v.p <- rep( 0, m2 )
```

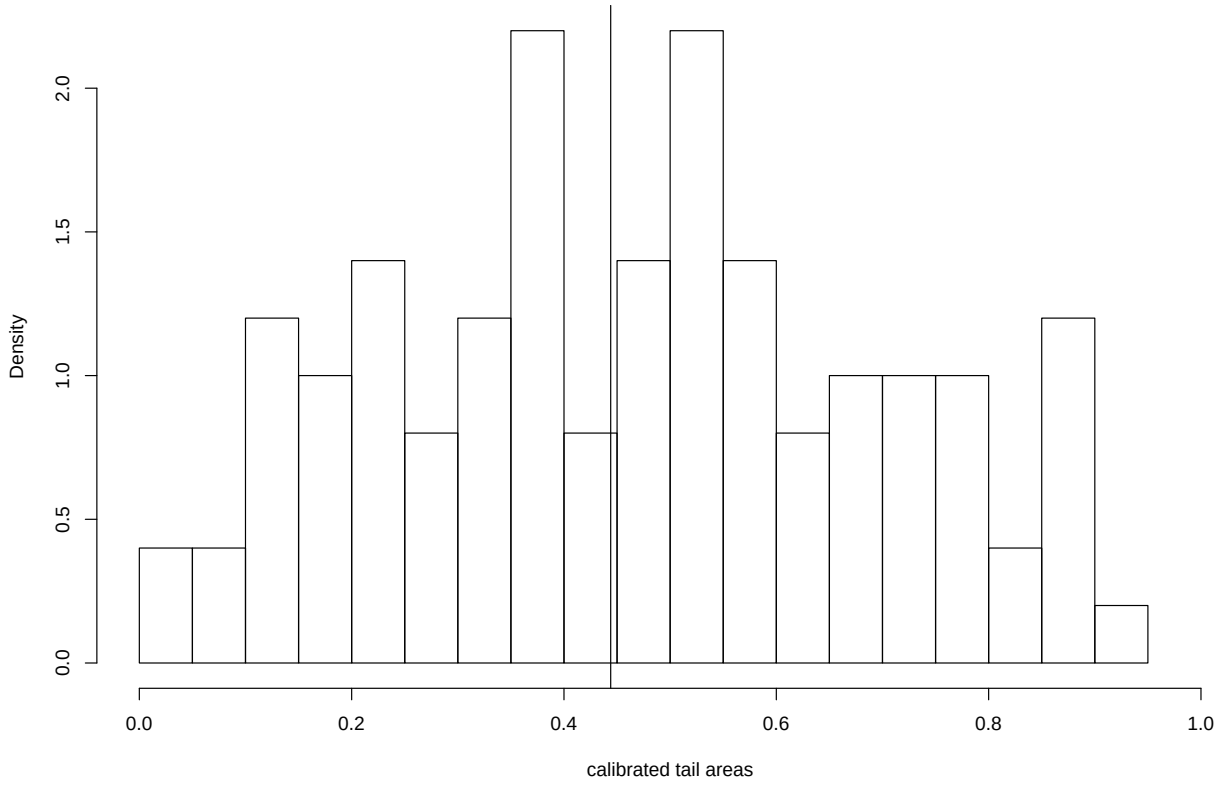
R Implementation (continued)

```
+   for ( j in 1:m2 ) {
+
+     lambda.star <- rgamma( 1, epsilon + s.actual,
+      epsilon + n )
+
+     y.sim <- rpois( n, lambda.star )
+
+     step1.result <- step1( y.sim, epsilon )
+
+     v.p[ j ] <- step2( step1.result[ 1 ],
+       step1.result[ 2 ], epsilon, step1.result[ 3 ], m1 )
+
+   }
+
+   aata <- sum( v.p <= uata ) / m2
+
+   write( v.p, "v.p.out" )
+
+   return( aata )
+
+ }

> m2 <- 100
>
> print( step3.result <- step3( y, epsilon, m1, m2 ) )
[1] 0.4
```

Here the **recalibration** has **not had much effect**, but (as the plots above showed) **this will not always be the case**.

R Implementation (continued)

```
> v.p <- scan( "v.p.out" )
Read 100 items
>
> hist( v.p, nclass = 20, probability = T, xlim = c( 0, 1 ),
+   main = '', xlab = 'calibrated tail areas' )
>
> abline( v = step2.result )
```



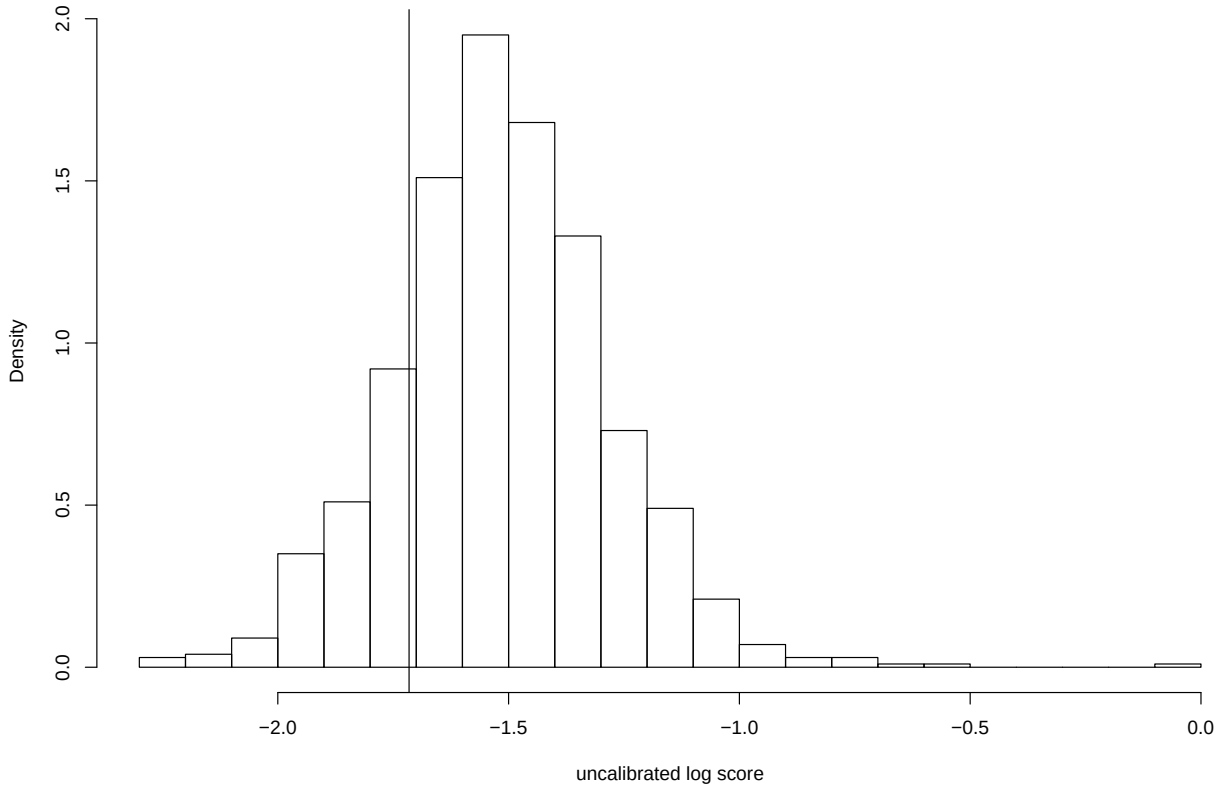
For a **second example** let's look at a **data set** generated as a **lognormal mixture of Poissons** with a **substantial VTMR**.

```
> n <- 10
>
> e <- rnorm( n, 0.0, 0.5 )
>
> mu <- 0
>
> lambda <- rep( 0, n )
```

R Implementation (continued)

```
> y <- rep( 0, n )

> for ( i in 1:n ) {
+
+   lambda[ i ] <- exp( mu + e[ i ] )
+
+   y[ i ] <- rpois( 1, lambda[ i ] )
+
+ }

> print( y <- sort( y ) )

[1] 0 0 0 1 1 1 2 3 4 4

> var( y ) / mean( y )

[1] 1.555556

> print( step1.result <- step1( y, epsilon ) )

[1] 10.000000 16.000000 -1.715601

> print( step2.result <- step2( step1.result[ 1 ],
+   step1.result[ 2 ], epsilon, step1.result[ 3 ], m1 ) )

[1] 0.178

> v.ls <- scan( "ls.out" )

> hist( v.ls, nclass = 20, probability = T,
+   main = '', xlab = 'uncalibrated log score' )

> abline( v = step1.result[ 3 ] )
```

R Implementation (continued)

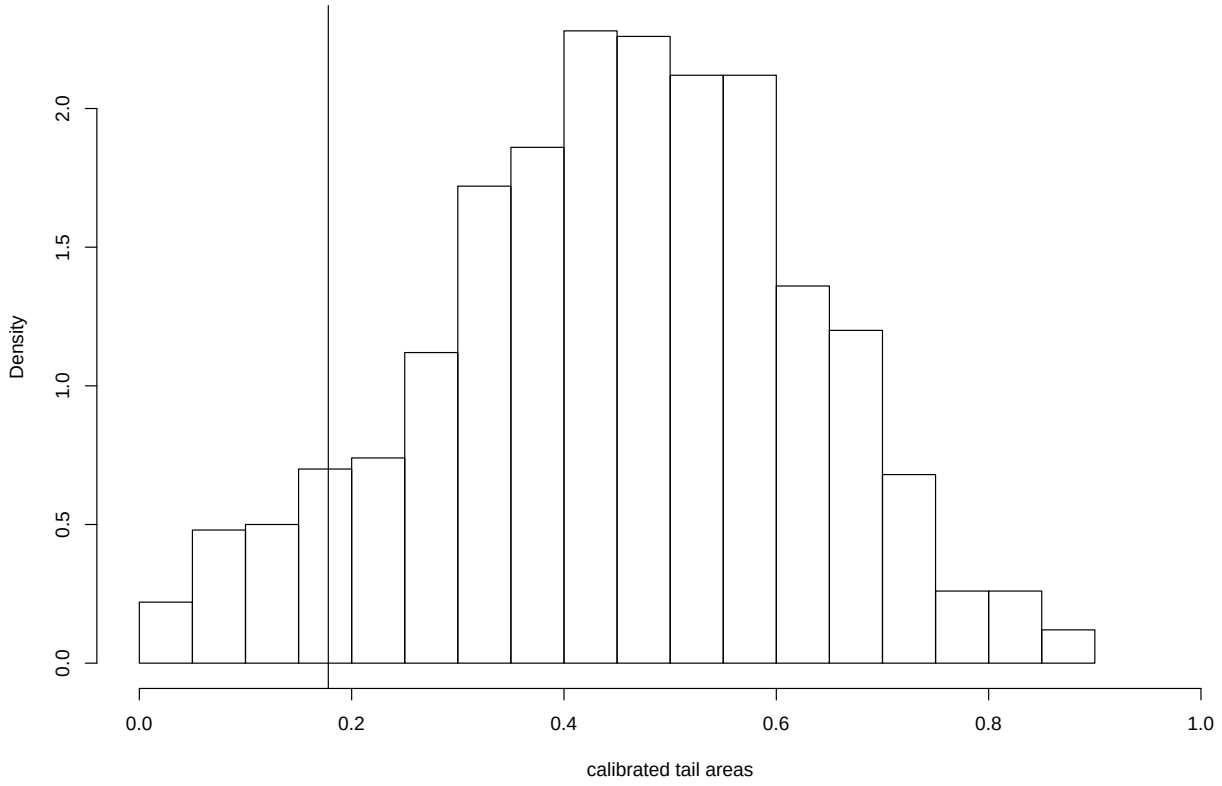



```
> m2 <- 1000

> print( step3.result <- step3( y, epsilon, m1, m2 ) )

[1] 0.099
```

So here's an example where the **uncalibrated tail area** is **about twice as big as it should be**.

```
> v.p <- scan( "v.p.out" )

> hist( v.p, nclass = 20, probability = T, xlim = c( 0, 1 ),
+   main = '', xlab = 'calibrated tail areas' )

> abline( v = step2.result )
```

R Implementation (continued)



The **true calibrated tail-area distribution** is **far from uniform**, so 0.178 is actually **substantially farther out in the true tail than it seems**.

LS_{CV}, LS_{FS} and DIC Model Discrimination

Here are **three behavioral rules**: **maximize** LS_{CV}, **maximize** LS_{FS}, **minimize** DIC; with (e.g.) two models to choose between, how **accurately** do these behavioral rules **discriminate** between M_1 and M_2?

Example: Consider **comparing** the following two models, with **diffuse priors** and $i = 1, \ldots, n$:

$$M_1: \left\{ \begin{array}{ccc} \lambda & \sim & p(\lambda) \\ (y_i|\lambda) & \overset{\text{IID}}{\sim} & \text{Poisson}(\lambda) \end{array} \right\} \quad \text{versus} \tag{79}$$

$$M_2: \left\{ \begin{array}{ccc} (\beta_0, \sigma^2) & \sim & p(\beta_0, \sigma^2) \\ (y_i|\lambda_i) & \overset{\text{indep}}{\sim} & \text{Poisson}(\lambda_i) \\ \log(\lambda_i) & = & \beta_0 + e_i \\ e_i & \overset{\text{IID}}{\sim} & N(0, \sigma^2) \end{array} \right\} \tag{80}$$

Milovan and I **generated data from** M_2 and computed LS_{CV}, LS_{FS}, and DIC for models M_1 and M_2 in **full-factorial grid** $\{n = 32, 42, 56, 100\}$, $\{\beta_0 = 0.0, 1.0\}$, $\sigma^2 = 0.1, 0.25, 0.5, 1.0, 1.5, 2.0\}$, with **100** simulation replications in each cell, and monitored **percentages of correct model choice** (here M_2 is always correct).

Examples of **results** for (e.g.) LS_{CV}:

$n = 32$

% Correct Decision

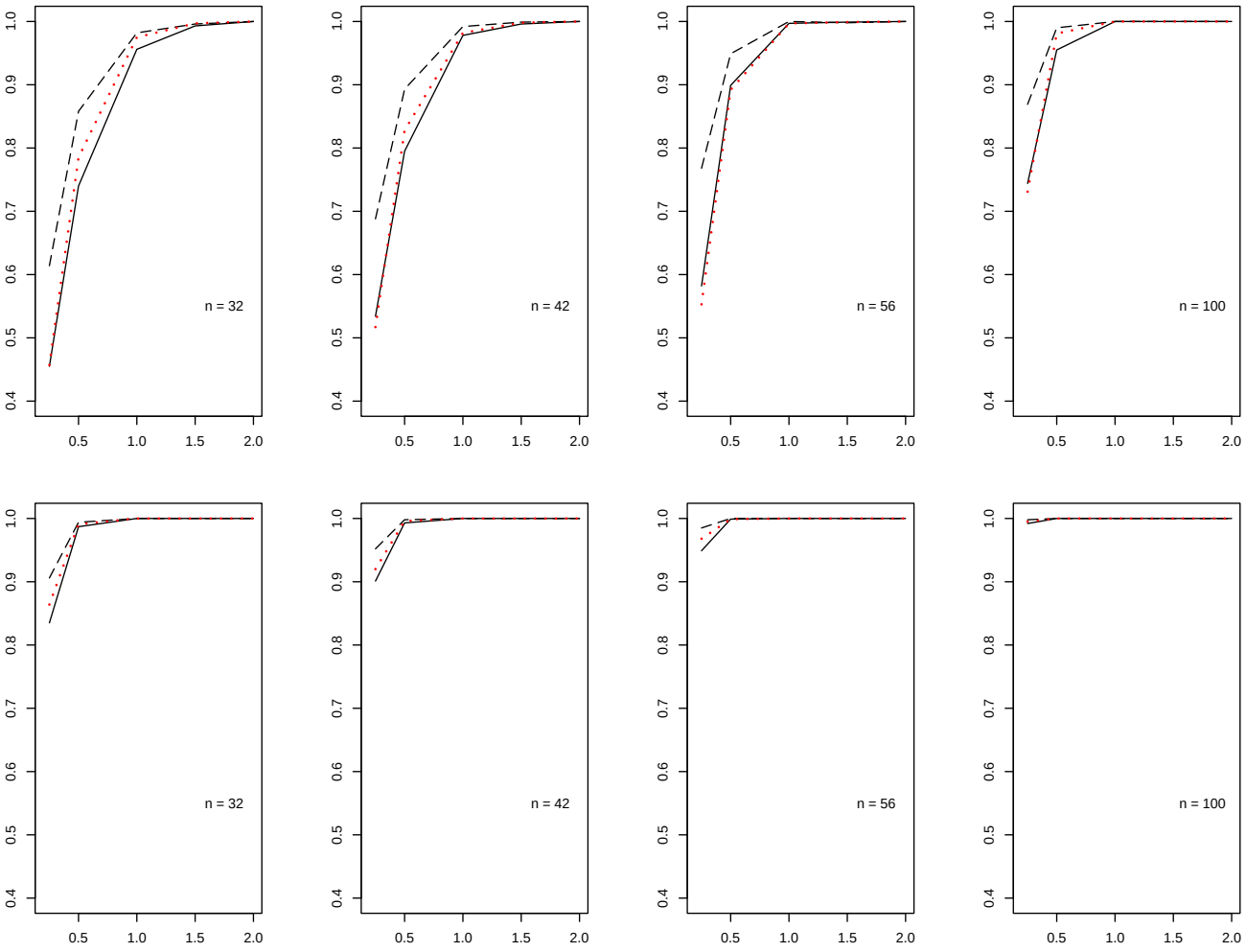
	β_0	
σ^2	0	1
0.10	31	47
0.25	49	85
0.50	76	95
1.00	97	100
1.50	98	100
2.00	100	100

Mean Absolute Difference in LS_{CV}

	β_0	
σ^2	0	1
0.10	0.001	0.002
0.25	0.002	0.013
0.50	0.017	0.221
1.00	0.237	4.07
1.50	1.44	17.4
2.00	12.8	63.9

Even with n only **32**, LS_{CV} makes the right model choice **more than 90% of the time** when $\sigma^2 > 0.5$ for $\beta_0 = 1$ and when $\sigma^2 > 1.0$ for $\beta_0 = 0$.

Model Discrimination (continued)



The plots above compare **Bayesian decision-theoretic power curves** for LS_{CV} (**solid** lines), LS_{FS} (**long dotted** lines), and DIC (**short dotted** lines) (row 1: $\beta_0 = 0$; row 2: $\beta_0 = 1$).

Remarkably, not only is LS_{FS} **much quicker computationally** than LS_{CV}, it's also **more accurate** at identifying the correct model than LS_{CV} or DIC.

To summarize, in **computational efficiency**

$$\text{naive } LS_{CV} < DIC \doteq LS_{FS} \tag{81}$$

and in **fixed-** and **random-effects Poisson modeling** the results in **model discrimination power** are

$$LS_{CV} \doteq DIC < LS_{FS} \tag{82}$$

What LS_{FS} Is Not

Consider the **likelihood** part of a (parametric) model M_j: $(y_i|\theta_j, M_j) \overset{\text{IID}}{\sim} p(y_i|\theta_j, M_j)(j = 1, 2)$, with **prior** $p(\theta_j|M_j)$ for model M_j.

The **Bayes factor** involves comparing quantities of the form

$$\begin{aligned} p(y|M_j) &= \int \left[\prod_{i=1}^{n} p(y_i|\theta_j, M_j)\right] p(\theta_j|M_j)\, d\theta_j, \\ &= E_{(\theta_j|M_j)} L(\theta_j|y, M_j), \end{aligned} \tag{83}$$

i.e., the Bayes factor involves comparing **expectations** of **likelihoods** with respect to the **priors** in the models under comparison (this is **why ordinary Bayes factors behave so badly with diffuse priors**).

Aitkin (1991) proposed instead **posterior Bayes factors**): compute the expectations with respect to the **posteriors**, i.e., **PBF:** favor model M_1 if $\log \bar{L}_1^A > \log \bar{L}_2^A$, where

$$\log \bar{L}_j^A = \log \int \left[\prod_{i=1}^{n} p(y_i|\theta_j, M_j)\right] p(\theta_j|y, M_j)\, d\theta_j. \tag{84}$$

This **solves** the problem of sensitivity to a diffuse prior but **creates new problems of its own**, e.g., it's **incoherent**.

It may **seem** at first glance (e.g., O'Hagan and Forster (2004) think so) that **PBF is the same thing as** LS_{FS}: favor model M_1 if

$$n\, LS_{FS}(M_1|y) > n\, LS_{FS}(M_2|y). \tag{85}$$

But **not so**:

$$nLS_{FS}(M_j|y) = \log \prod_{i=1}^{n} \left[\int p(y_i|\theta_j, M_j)\, p(\theta_j|y, M_j)\, d\theta_j\right], \tag{86}$$

and this is **not the same** because the **integral** and **product** operators **do not commute**.

What LS_{FS} Is Not (continued)

Also, some people (e.g., Geweke (2005)) like to compare models based on the **posterior expectation of the log likelihood** (this is **one of the ingredients** in DIC), and this is **not the same** as LS_{FS} either: by **Jensen's inequality**

$$
\begin{aligned}
nLS_{FS}(M_j|y) &= \sum_{i=1}^{n} \log p(y_i|y, M_j) \\
&= \sum_{i=1}^{n} \log \int p(y_i|\theta_j, M_j)\, p(\theta_j|y, M_j)\, d\theta_j \\
&= \sum_{i=1}^{n} \log E_{(\theta_j|y,M_j)} L(\theta_j|y_i, M_j) \\
&> \sum_{i=1}^{n} E_{(\theta_j|y,M_j)} \log L(\theta_j|y_i, M_j) \qquad (87) \\
&= E_{(\theta_j|y,M_j)} \sum_{i=1}^{n} \log L(\theta_j|y_i, M_j) \\
&= E_{(\theta_j|y,M_j)} \log \prod_{i=1}^{n} L(\theta_j|y_i, M_j) \\
&= E_{(\theta_j|y,M_j)} \log L(\theta_j|y, M_j).
\end{aligned}
$$