Chapter 6

Height model again and again...

- model:

$$
\begin{aligned}
h_i &\sim N(\mu_i, \sigma) \\
\mu_i &= \alpha + w_i \beta \\
\alpha_0 &\sim N(\mu_\alpha, \sigma_\alpha) \\
\beta &\sim N(\mu_\beta, \sigma_\beta) \\
\sigma &\sim U(a_\sigma, b_\sigma)
\end{aligned}
$$

Height model again and again...

- model:

$$
\begin{aligned}
h_i &\sim N(\mu_i, \sigma) \\
\mu_i &= \alpha + w_i \beta \\
\alpha_0 &\sim N(\mu_\alpha, \sigma_\alpha) \\
\beta &\sim N(\mu_\beta, \sigma_\beta) \\
\sigma &\sim U(a_\sigma, b_\sigma)
\end{aligned}
$$

- If h_i is in cm, and w_i is in kilo gram, what scale is the parameters in?

parameter scale

Model	scale
μ	cm
σ	cm
α	cm
β	$\frac{cm}{kg}$
μ_β	$\frac{cm}{kg}$
σ_β	$\frac{cm}{kg}$

- If you don't have any information about coefficients, choose vague prior. Think don't over think!
- .

```
library(NHANES)
data("NHANES")
NHANES = NHANES[duplicated(NHANES$ID)==F,]
NHANES = NHANES[NHANES$Age >20, ]
NHANES = NHANES[,c("Height","Weight","DirectChol","TotChol")]
NHANES = NHANES[complete.cases(NHANES),]
NHANES = data.frame(NHANES)
```

- **map** uses optimization to find the MAP value.
- If using very vague priors, set start values in **map**, otherwise it will take a sample from the prior as a starting guess.
- .

```
model <- map( flist = alist(...),
              data = .,
              start = list(nameOfParameter = initalValue))
```

Overfitting

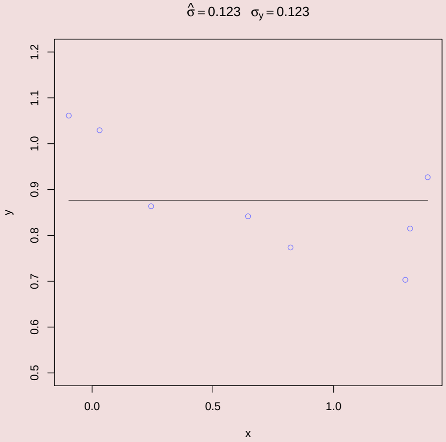


Figure : MAP estimate of $\mu = \beta_0$

Overfitting

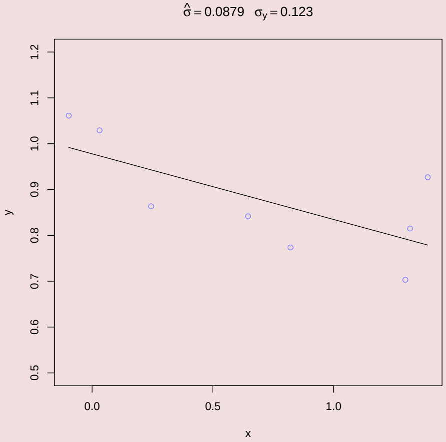


Figure : MAP estimate of $\mu = \beta_0 + x\beta_1$

Overfitting

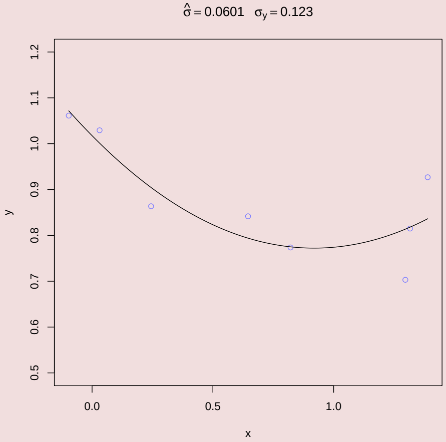


Figure : MAP estimate of $\mu = \beta_0 + x\beta_1 + x^2\beta_2$

Overfitting

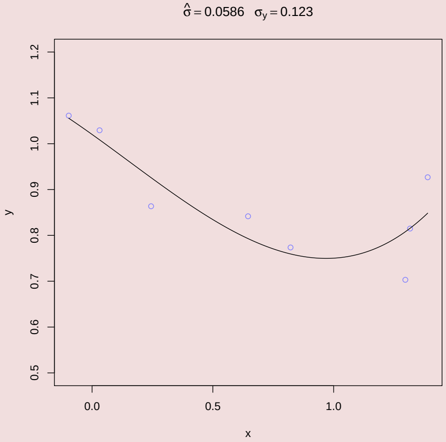


Figure : MAP estimate of $\mu = \beta_0 + x\beta_1 + x^2\beta_2 + x^3\beta_3$

Overfitting

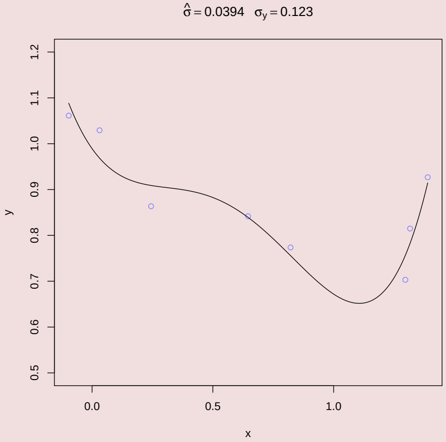


Figure : MAP estimate of $\mu = \beta_0 + x\beta_1 + x^2\beta_2 + x^3\beta_3 + x^4\beta_4$

Overfitting

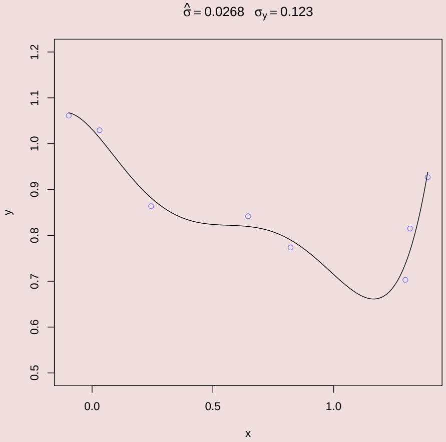


Figure : MAP estimate of $\mu = \beta_0 + x\beta_1 + x^2\beta_2 + x^3\beta_3 + x^4\beta_4 + x^5\beta_5$

Overfitting

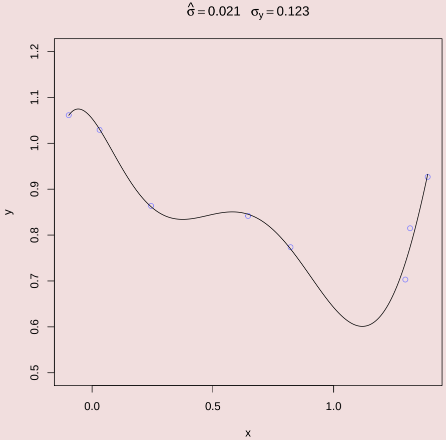


Figure : MAP estimate of
$\mu = \beta_0 + x\beta_1 + x^2\beta_2 + x^3\beta_3 + x^4\beta_4 + x^5\beta_5 + x^6\beta_6$

Removing one observations

- Loop over all data $j = 1, \ldots, n$:
- Remove one observation, y_j.
- Estimate β using $y_1, \ldots, y_{j-1}, y_{j+1}, \ldots, y_n$.
- Estimate σ using $\sqrt{\frac{1}{n} \sum (y_j - \hat{y}_j)^2}$.

Leave one out

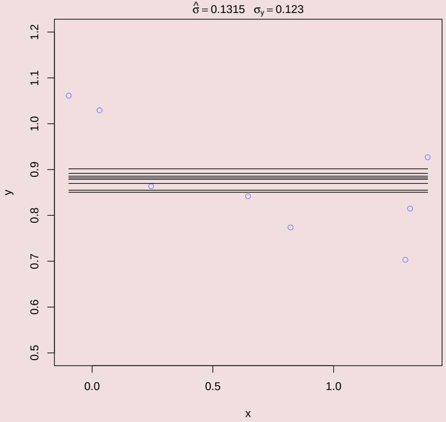


Figure : Leave one out estimate of $\mu = \beta_0$

Leave one out

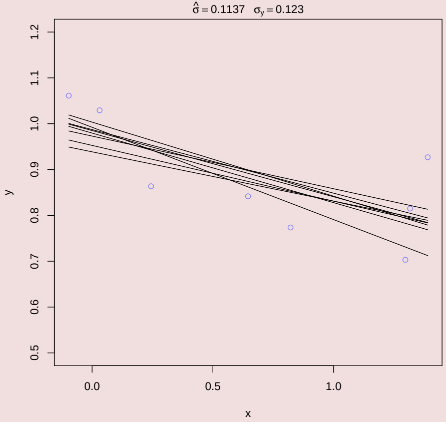


Figure : Leave one out of $\mu = \beta_0 + x\beta_1$

Leave one out

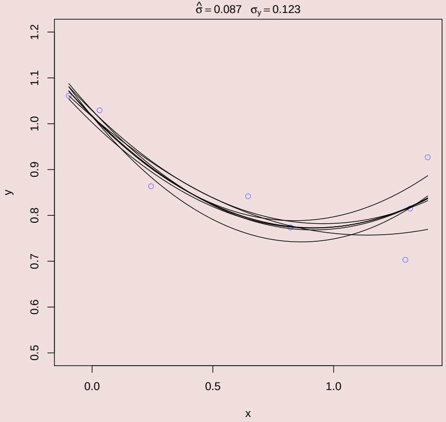


Figure : Leave one out of $\mu = \beta_0 + x\beta_1 + x^2\beta_2$

Leave one out

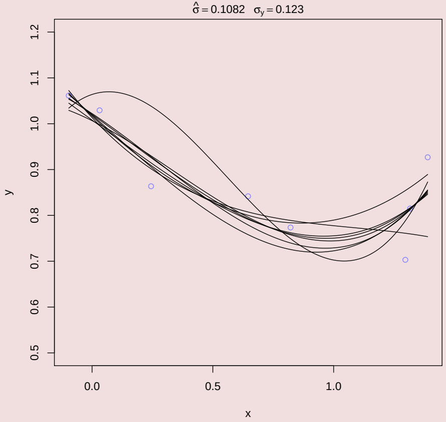


Figure : Leave one out of $\mu = \beta_0 + x\beta_1 + x^2\beta_2 + x^3\beta_3$

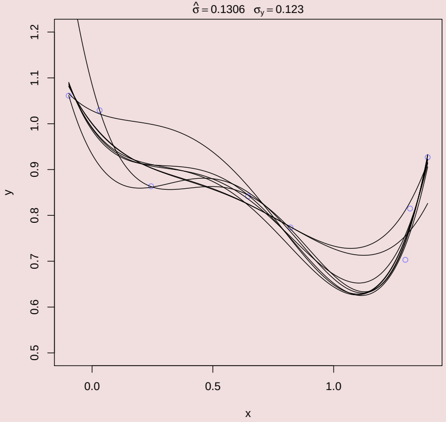


Figure : Leave one out of $\mu = \beta_0 + x\beta_1 + x^2\beta_2 + x^3\beta_3 + x^4\beta_4$

Leave one out

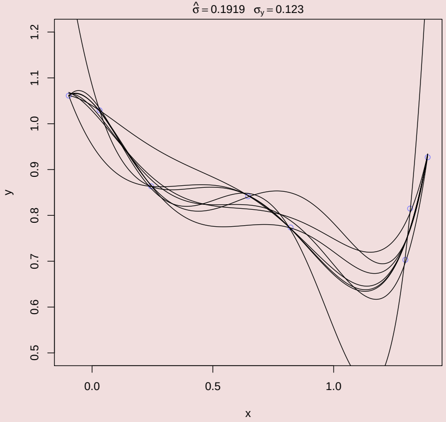


Figure : Leave one out of $\mu = \beta_0 + x\beta_1 + x^2\beta_2 + x^3\beta_3 + x^4\beta_4 + x^5\beta_5$

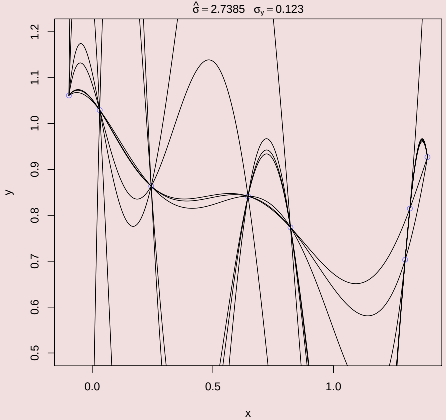


Figure : Leave one out of
$\mu = \beta_0 + x\beta_1 + x^2\beta_2 + x^3\beta_3 + x^4\beta_4 + x^5\beta_5 + x^6\beta_6$

- For linear, there exists many different measure R^2_{adj}.

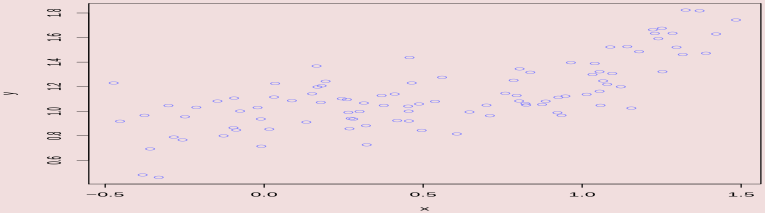
Beyond linear model

- For linear, there exists many different measure R^2_{adj}.
- For a general density/probability function what indicates a good fit?

Beyond linear model

- For linear, there exists many different measure R^2_{adj}.
- For a general density/probability function what indicates a good fit?
- Answer: $p(y_i)$, or equivalently $\log(p(y_i))$

- $AIC = -2\sum_{i=1}^{n} \log(p(y_i)) + 2d$,
  d the number of parameters in the model.
- The smaller the better.



- True model:

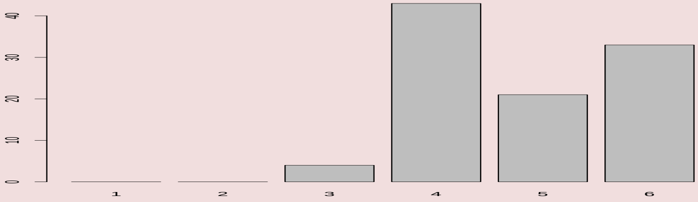
$$y_i = 1 + x0.1 - x^2 0.3 + 0.4x^3 + \epsilon_i,$$

  $\epsilon_i \sim N(0, 0.2)$
- Models with different number of parameters (d)

$$y_i = \alpha_0 + \sum_{i=1}^{d-1} x^i \beta_i + \epsilon_i.$$

- Choosing model by best (smallest) AIC.

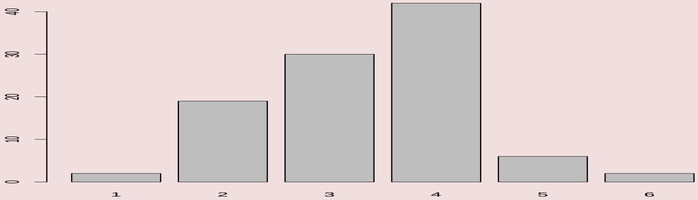
repeated experiment, in sample



- Simulate 100 observations, repeat 101 times.
- Record the best model by AIC.

Cross-validation

- Split data into two or three sets.
- Training data - fit the parameters.
- Test data - evaluate performance.
- Evaluation data (not used here).



- Simulate 100 observations, repeat 101 times.
- Split the data into two part, 60% training, 40% testing.
- Fit the parameters on the training data.
- Choose model on the testing data.

- The AIC is not well suited for the Bayesian modeling.
- Priors can affect over-fitting vs under-fitting.
- Possible to choose prior so they learn less from the data.
- Leaving, model selection for a slide to examine prior effect on the posterior distribution.

Model:

$$
\begin{aligned}
y_i &\sim N(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta x_i \\
\alpha &\sim N(0, 100) \\
\beta &\sim N(0, \sigma_\beta) \\
\sigma &\sim (0, 10)
\end{aligned}
$$

- What effect does σ_β on the posterior distribution.

Model:

$$
\begin{aligned}
y_i &\sim N(\mu_i, \sigma) \\
\mu_i &= \alpha + \sum_{j=1}^{d} x_{ij}\beta_j \\
\alpha &\sim N(0, \sigma_\beta) \\
\beta_j &\sim N(0, \sigma_\beta),\ j = 1, \dots, d \\
\sigma &\sim (0, 10)
\end{aligned}
$$

- σ_β is a hyperparameter.
- In Machine learning: choose the optimal σ_β gives best prediction.

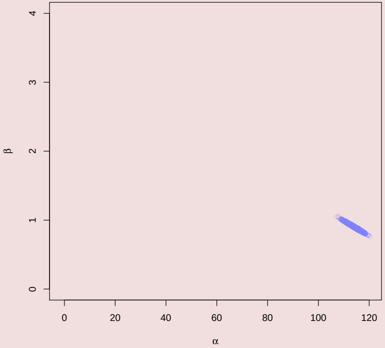


Figure : Posterior samples for $\sigma_\beta = 100$

Regularization

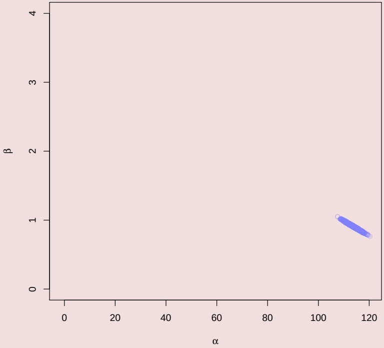


Figure : Posterior samples for $\sigma_\beta = 10$

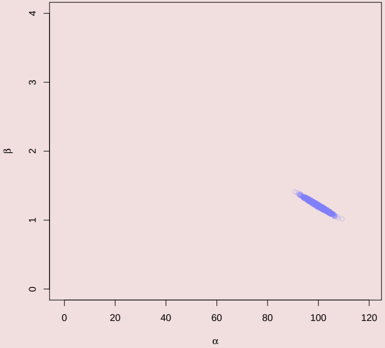


Figure : Posterior samples for $\sigma_\beta = 1$

Regularization

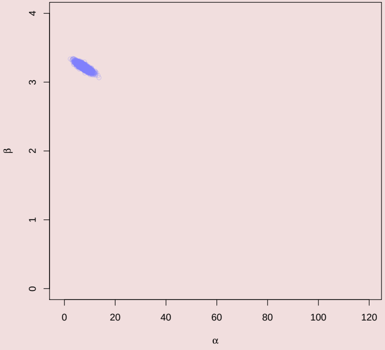


Figure : Posterior samples for $\sigma_\beta = 0.1$

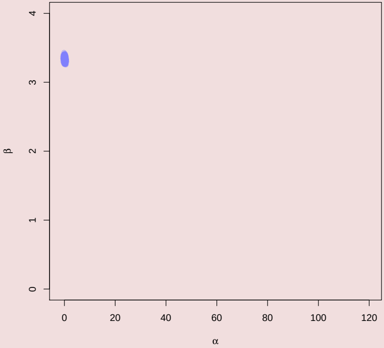


Figure : Posterior samples for $\sigma_\beta = 0.01$

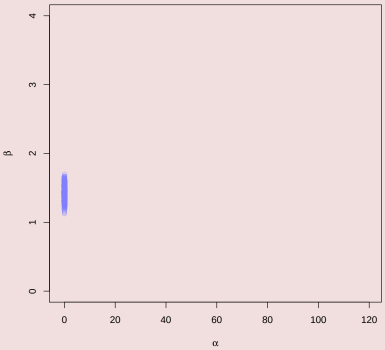


Figure : Posterior samples for $\sigma_\beta = 0.001$

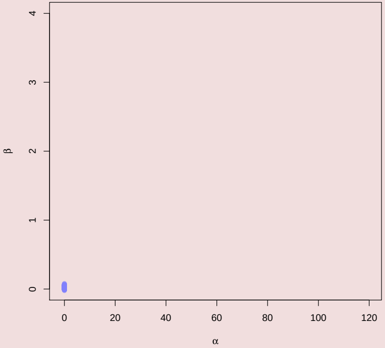


Figure : Posterior samples for $\sigma_\beta = 0.0001$

Regularization

- Prior makes the posterior conservative.
- How to compare models?
- For $\sigma_\beta = 0.0001$ and $\sigma_\beta = 100$ same number of parameters.

WAIC

WAIC balances how well a model predicts, with flexibility of the model

- How well does a model predict the data?

$$\sum_{j=1}^{n} \log(p(y_j|y_1,\ldots,y_n)) = \sum_{j=1}^{n} \log(\mathbb{E}[p(y_j|\alpha,\beta,\sigma,y_1,\ldots,y_n)])$$

$$= \sum_{j=1}^{n} \log\left(\int p(y_j|\tilde{\alpha},\tilde{\beta},\tilde{\sigma})\cdot \right.$$

$$\left. \cdot p(\tilde{\alpha},\tilde{\beta},\tilde{\sigma}|y_1,\ldots,y_n)\, d\tilde{\alpha}\, d\tilde{\beta}\, d\tilde{\sigma}\right)$$

- How flexible is the model?

$$p_{WAIC} = \sum_{j=1}^{n} \mathbb{V}[\log(p(y_j|\alpha,\beta,\sigma,y_1,\ldots,y_n))]$$

Building the model:

```
data(milk)
d            <- milk[ complete.cases(milk), ]
d$neocortex <- d$neocortex.perc / 100
model <- map(
              alist(kcal.per.g ~ dnorm(mu, sigma),
              mu <- alpha + bn * neocortex,
              alpha ~ dnorm(0,10),
              bn    ~ dnorm(0,10)),
              data = d
            )
n.sim <- 1000
post <- extract.samples(model, n = n.sim)
```

WAIC code

Computing $\log(p(y_j|\alpha^{(i)}, \beta^{(i)}, \sigma^{(i)}))$ for data $j = 1, \ldots, n$ and samples $i = 1, \ldots, n_{sim}$:

```
ll <- sapply( 1:n.sim ,
              function(j){
                     mu <- post$alpha[j] + d$neocortex * post$bn[j]
                     dnorm(d$kcal.per.g, mu, post$sigma[j], log=T )} )
```

WAIC code, deviation

Approximating:

$$\sum_{j=1}^{n} \log(\mathbb{E}[p(y_j|\alpha,\beta,\sigma,y_1,\ldots,y_n)]) = \sum_{j=1}^{n} \log\left(\int p(y_j|\tilde{\alpha},\tilde{\beta},\tilde{\sigma})p(\tilde{\alpha},\tilde{\beta},\tilde{\sigma})|y_1,\ldots,y_n)\, d\tilde{\alpha}\, d\tilde{\beta}\, d\tilde{\sigma}\right)$$

with

$$lppd = \sum_{j=1}^{n} \log\left(\frac{1}{n_{sim}}\sum_{i=1}^{n_{sim}} p(y_j|\alpha^{(i)},\beta^{(i)},\sigma^{(i)})\right)$$

```
n <- nrow(d)
lppd <- sum( sapply(1:n, function(j) log_sum_exp(ll[j,]) - log(n.sim)))
```

Approximating:

$$\sum_{j=1}^{n} \mathbb{V}[\log(p(y_j|\alpha, \beta, \sigma, y_1, \ldots, y_n))]$$

with the variance of the function $\log(p(y_j|\alpha^{(i)}, \beta^{(i)}, \sigma^{(i)})$

```
pWAIC <- sum(sapply(1:n, function(j) var(ll[j,])))
```

WAIC code, p_{WAIC}

Model1

$$\begin{aligned} y_i &\sim N(\mu_i, \sigma) \\ \mu_i &= \alpha + nero\beta_n \\ \alpha &\sim N(0, \sigma_\beta = 10) \\ \beta_n &\sim N(0, \sigma_\beta = 10) \end{aligned}$$

Results:

$-2lppd \approx -12$

$2p_{WAIC} \approx 6$

$WAIC = -2llpd + 2p_{WAIC} = -6$

Model2

$$\begin{aligned} y_i &\sim N(\mu_i, \sigma) \\ \mu_i &= \alpha + nero\beta_n \\ \alpha &\sim N(0, \sigma_\beta = 0.1) \\ \beta_n &\sim N(0, \sigma_\beta = 0.1) \end{aligned}$$

Results:

$-2lppd \approx 36$

$2p_{WAIC} \approx 1$

$WAIC = -2llpd + 2p_{WAIC} = 37$

Comparing models

```
model1 <- map(
          alist(kcal.per.g ~ dnorm(mu, sigma),
          mu <- alpha,
          alpha ~ dnorm(0,10)),
          data = d
          )
model2 <- map(
          alist(kcal.per.g ~ dnorm(mu, sigma),
          mu   <- alpha + bn * neocortex,
          alpha ~ dnorm(0,10),
          bn    ~ dnorm(0,10)),
          data = d
          )
model3 <- map(
          alist(kcal.per.g ~ dnorm(mu, sigma),
          mu   <- alpha + bn * neocortex + bm * log(mass),
          alpha ~ dnorm(0,10),
          bn    ~ dnorm(0,10),
          bm    ~ dnorm(0,10)),
          data = d
          )
```

WAIC model comparision

Comparing models

```
WAIC(model1)
```

```
[1] -7.410979
attr(,"lppd")
[1] 5.938131
attr(,"pWAIC")
[1] 2.232642
attr(,"se")
[1] 4.902928
```

WAIC model comparision

Comparing models

```
compare(model1, model2, model3)
```

	WAIC	pWAIC	dWAIC	weight	SE	dSE
model3	−13.6	5.5	0.0	0.93	8.10	NA
model1	−8.0	2.0	5.6	0.06	4.91	7.66
model2	−5.8	3.1	7.8	0.02	4.59	7.91

WAIC model comparision

- $WAIC_{min}$ is the smallest $WAIC$ of all compared models.
- $dWAIC_i = WAIC_i - WAIC_{min}$, gives the weight of a model

$$w_i = \frac{\exp(-\frac{1}{2} dWAIC_i)}{\sum_{j=1}^{m} \exp(-\frac{1}{2} dWAIC_j)}$$