Easy

E1

1. The measure needs to be continuous so that a small change in probabilities doesn't lead to an arbitrary change in uncertainty.
2. Uncertainty should scale with the possible outcomes. As the number of outcomes increases, the amount of uncertianty about the outcome should increase.
3. The uncertainty about combinations of events should be the sum of the uncertainty of the individual events.

E2

```r
H <- function(p){
  -sum(p*log(p))
}

p <- c(.3, .7)

H(p)
```

```
[1] 0.6108643
```

E3

```r
dice_probs <- c(.2, .25, .25, .3)

H(dice_probs)
```

```
[1] 1.376227
```

E4

```r
dice_probs <- c(1/3, 1/3, 1/3)

H(dice_probs)
```

```
[1] 1.098612
```

Medium

M1

AIC is the deviance plus 2 times the number of parameters. It assumes that the priors are flat, the posterior is multivariate gaussian, and that N is much greater than K.

DIC is the average deviance minus the difference between the average deviance and the deviance at the posterior mean. The difference between the average deviance and the deviance at the posterior mean is a proxy for the number of parameters, since a big difference would indicate more parameters (more variability).

WAIC is looks at both the lppd which is the sum of the log probability of each observation. This log probability is the average over the whole posterior of the parameters. You then subtract the sum of the variances of the probability of seeing y over the entire distribution.

The WAIC is probably the most general as it doesn't make any distributional assumptions and averages over the entire posterior distribution.

M2

With model selection, we take the best fitting model and throw out the rest. With model averaging, we give each model a weight based on how well it should predict out of sample data and then average across all these models. When we use model selection, we lose uncertainty about the structure of the data generating process. This is analogous to choosing a point estimate of a parameter and loosing uncertainty about the parameter.

When we use model averaging we may let unreasonable models water better models.

M3

Because we don't actually know the reference distribution, we are just comparing multiple distributions to that same reference. If we use different data for different models, we wouldn't be comparing to the same reference and we can't use this to compare models.

M4

By concentrating the prior, this would lead to there being a lower variance in the posterior and a lower variance in the likelihood of seeing the data, which would decrease the effective number of parameters.

Let's test this with stan!

```r
library(cmdstanr)
set.seed(12042014)
# generate some data
n <- 10
a <- .5
b <- .5
sigma <- 2
x <- rnorm(n)
y <- a + x*b + rnorm(n, 0, sigma)


# make separate data lists (only difference is the priors) for the informative
and
# noninformative model.
```

```r
data_informative <- list(
  N = n,
  x = x,
  y = y,
  b_scale = .5,
  a_scale = .5,
  sigma_prior = 2
)

data_not_informative <- list(
  N = n,
  x = x,
  y = y,
  b_scale = 1000,
  a_scale = 1000,
  sigma_prior = .1
)

# compile the model
mod <- cmdstan_model("stan_model/stan_mod.stan")

# fit the model with informative priors
fit_inf <- mod$sample(
  data = data_informative,
  chains = 4,
  parallel_chains = 4,
  show_messages = F
)

# fit the model with noninformative priors
fit_non_inf <- mod$sample(
  data = data_not_informative,
  chains = 4,
  parallel_chains = 4,
  show_messages = F
)

# extract the dataframe of log likelihoods of the data for each iteration from
# the informative and not informative model
liks_inf <- fit_inf$draws("log_lik", format = "df")[,1:n]

liks_n_inf <- fit_non_inf$draws("log_lik", format = "df")[,1:n]

# calculate pwaic for the informative
pwaic_inf <- sum(apply(liks_inf, 2, var))

# calculate pwaic for the non informative
pwaic_n_inf <- sum(apply(liks_n_inf, 2, var))
```

```r
# print the results
print(paste0("Informative p_waic = ", round(pwaic_inf, 2),
             " Noninformative p_waic = ", round(pwaic_n_inf, 2)))
```

```
[1] "Informative p_waic = 1.3 Noninformative p_waic = 1.96"
```

M5

Informative priors reduce overfitting by tempering likelihoods that have learned too much from the data.

M6

If the prior is too informative, the prior will overwhelm the likelihood and not let the model learn anything from the data.

Hard

```r
library(rethinking)
data(Howell1)
d <- Howell1
d$age <- (d$age - mean(d$age))/sd(d$age)
set.seed(1000)
i <- sample(1:nrow(d), size = nrow(d)/2)
d1 <- d[i,]
d2 <- d[-i,]
```

I think a pretty regularizing prior on alpha would be N(0,1), and on the betas would be N(0,20) in this case

```r
mod <- cmdstan_model("stan_model/hard_mods.stan")

n <- nrow(d1)
x_tilde <- seq(min(d$age), max(d$age), l = 200)

alpha_scale <- 20
beta_scale <- 20

dat_1 <- list(
  N = n,
  K = 1,
  J = 200,
  y = d1$height,
  X = matrix(d1$age),
  y2 = d2$height,
  X2 = matrix(d2$age),
  X_tilde = matrix(x_tilde),
  alpha_scale = alpha_scale,
  beta_scale = beta_scale
)

fit1 <- mod$sample(
  data = dat_1,
  chains = 4,
  parallel_chains = 4,
  show_messages = F
)

fit1$diagnostic_summary()
```

```
$num_divergent
[1] 0 0 0 0

$num_max_treedepth
[1] 0 0 0 0

$ebfmi
[1] 1.143728 1.186664 1.052444 1.113562
```

```
dat_2 <- list(
  N = n,
  K = 2,
  J = 200,
  y = d1$height,
  X = matrix(c(d1$age, d1$age^2), ncol = 2),
  y2 = d2$height,
  X2 = matrix(c(d2$age, d2$age^2), ncol = 2),
  X_tilde = matrix(c(x_tilde, x_tilde^2), ncol = 2),
  alpha_scale = alpha_scale,
  beta_scale = beta_scale
)

fit2 <- mod$sample(
  data = dat_2,
  chains = 4,
  parallel_chains = 4,
  show_messages = F
)

fit2$diagnostic_summary()
```

```
$num_divergent
[1] 0 0 0 0

$num_max_treedepth
[1] 0 0 0 0

$ebfmi
[1] 1.045437 1.071895 0.998991 1.243517
```

```r
dat_3 <- list(
  N = n,
  K = 3,
  J = 200,
  y = d1$height,
  X = matrix(c(d1$age, d1$age^2, d1$age^3), ncol = 3),
  y2 = d2$height,
  X2 = matrix(c(d2$age, d2$age^2, d2$age^3), ncol = 3),
  X_tilde = matrix(c(x_tilde, x_tilde^2, x_tilde^3), ncol = 3),
  alpha_scale = alpha_scale,
  beta_scale = beta_scale
)

fit3<- mod$sample(
  data = dat_3,
  chains = 4,
  parallel_chains = 4,
  show_messages = F
)

fit3$diagnostic_summary()
```

```
$num_divergent
[1] 0 0 0 0

$num_max_treedepth
[1] 0 0 0 0

$ebfmi
[1] 1.072613 1.085938 1.099025 1.026645
```

```r
dat_4 <- list(
  N = n,
  K = 4,
  J = 200,
  y = d1$height,
  X = matrix(c(d1$age, d1$age^2, d1$age^3, d1$age^4), ncol = 4),
  y2 = d2$height,
  X2 = matrix(c(d2$age, d2$age^2, d2$age^3, d2$age^4), ncol = 4),
  X_tilde = matrix(c(x_tilde, x_tilde^2, x_tilde^3, x_tilde^4), ncol = 4),
  alpha_scale = alpha_scale,
  beta_scale = beta_scale
)

fit4 <- mod$sample(
  data = dat_4,
  chains = 4,
  parallel_chains = 4,
  show_messages = F
)

fit4$diagnostic_summary()
```

```
$num_divergent
[1] 0 0 0 0

$num_max_treedepth
[1] 0 0 0 0

$ebfmi
[1] 0.8999595 1.1210804 1.0069562 0.9730276
```

```r
dat_5 <- list(
  N = n,
  K = 5,
  J = 200,
  y = d1$height,
  X = matrix(c(d1$age, d1$age^2, d1$age^3, d1$age^4, d1$age^5), ncol = 5),
  y2 = d2$height,
  X2 = matrix(c(d2$age, d2$age^2, d2$age^3, d2$age^4, d2$age^5), ncol = 5),
  X_tilde = matrix(c(x_tilde, x_tilde^2, x_tilde^3, x_tilde^4, x_tilde^5), ncol
= 5),
  alpha_scale = alpha_scale,
  beta_scale = beta_scale
)

fit5 <- mod$sample(
  data = dat_5,
  chains = 4,
  parallel_chains = 4,
  show_messages = F
)
```

```r
fit5$diagnostic_summary()
```

```
$num_divergent
[1] 0 0 0 0

$num_max_treedepth
[1] 0 0 0 0

$ebfmi
[1] 0.9880485 0.9683732 0.9386670 0.9447720
```

```r
dat_6 <- list(
  N = n,
  K = 6,
  J = 200,
  y = d1$height,
  X = matrix(c(d1$age, d1$age^2, d1$age^3, d1$age^4, d1$age^5, d1$age^6), ncol
= 6),
  y2 = d2$height,
  X2 = matrix(c(d2$age, d2$age^2, d2$age^3, d2$age^4, d2$age^5, d2$age^6), ncol
= 6),
   X_tilde  =  matrix(c(x_tilde,  x_tilde^2,  x_tilde^3,  x_tilde^4,  x_tilde^5,
x_tilde^6), ncol = 6),
  alpha_scale = alpha_scale,
  beta_scale = beta_scale
)

fit6 <- mod$sample(
  data = dat_6,
  chains = 4,
  parallel_chains = 4,
  show_messages = F
)

fit6$diagnostic_summary()
```

```
$num_divergent
[1] 0 0 0 0

$num_max_treedepth
[1] 0 0 0 0

$ebfmi
[1] 1.044830 1.047019 1.083258 1.043966
```

H1

```
# use loo instead of waic, it should be very similar
library(loo)
loo1 <- loo(fit1$draws("log_lik"))
loo2 <- loo(fit2$draws("log_lik"))
loo3 <- loo(fit3$draws("log_lik"))
loo4 <- loo(fit4$draws("log_lik"))
loo5 <- loo(fit5$draws("log_lik"))
loo6 <- loo(fit6$draws("log_lik"))
loo_list <- list(loo1, loo2, loo3, loo4, loo5, loo6)
loo_compare(loo_list)
```

```
       elpd_diff se_diff
model6    0.0       0.0
model4   -0.6       1.9
model5   -1.3       1.8
model3  -14.0       5.9
model2 -107.9      14.3
model1 -243.2      15.5
```

```
loo_model_weights(loo_list)
```

```
Method: stacking
------
       weight
model1 0.000
model2 0.022
model3 0.000
model4 0.398
model5 0.007
model6 0.572
```

```
comp <- loo_compare(loo_list)
```

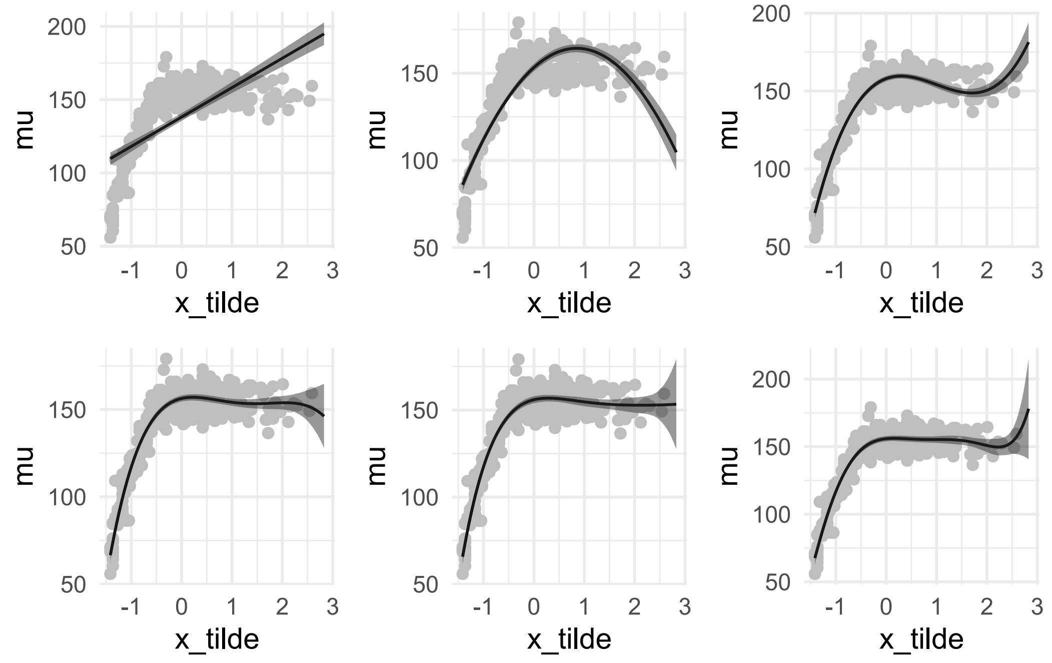
H2

```
library(tidyverse)
library(cowplot)

plot_mu <- function(fit, x_tilde, x = d1$age, y = d1$height){
  mu_post <- fit$draws("mu_tilde", format = "df")[,1:200]
  mu <- apply(mu_post, 2, mean)
  upr <- apply(mu_post, 2, quantile, .985)
  lwr <- apply(mu_post, 2, quantile, .015)
  dat <- data.frame(x, y)

  data.frame(x_tilde, mu, upr, lwr) %>%
    ggplot(aes(x = x_tilde, y = mu)) +
    geom_point(data = dat, aes(x = x, y = y), color = "grey") +
    geom_line() +
    geom_ribbon(aes(x = x_tilde, ymax = upr, ymin = lwr),
                alpha = .5) +
    theme_minimal()
}

plot_grid(
  plot_mu(fit1, x_tilde, d1$age, d1$height),
  plot_mu(fit2, x_tilde, d1$age, d1$height),
  plot_mu(fit3, x_tilde, d1$age, d1$height),
  plot_mu(fit4, x_tilde, d1$age, d1$height),
  plot_mu(fit5, x_tilde, d1$age, d1$height),
  plot_mu(fit6, x_tilde, d1$age, d1$height),
  ncol = 3
)
```



H3

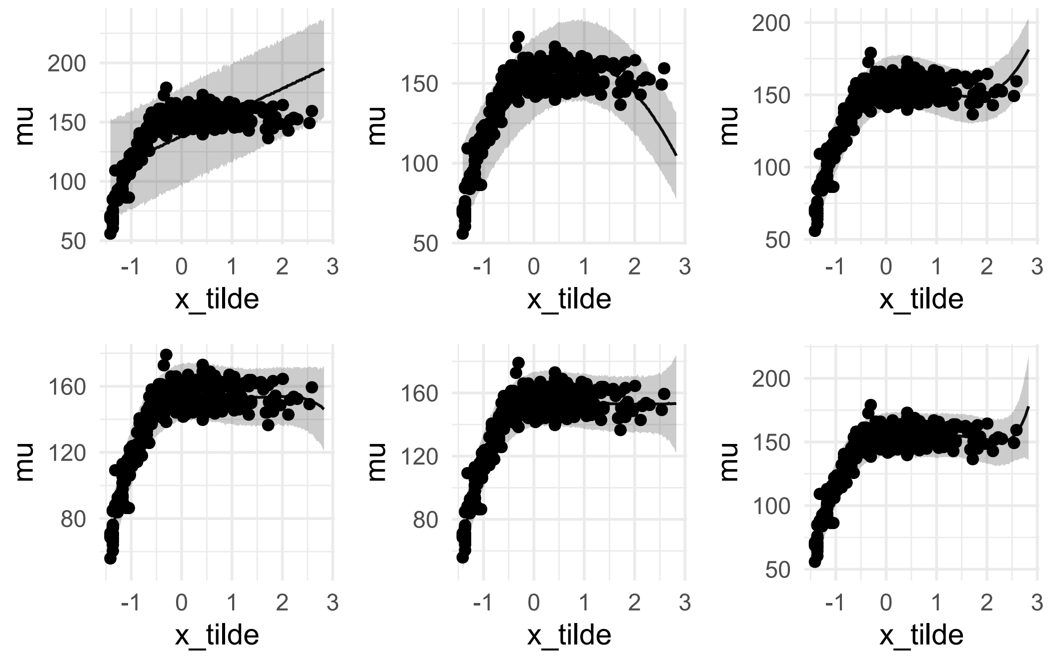
```
plot_pred <- function(fit, x_tilde, x = d1$age, y = d1$height){
  post <- fit$draws("y_tilde", format = "df")[,1:200]
```

```r
  mu <- apply(post, 2, mean)
  upr <- apply(post, 2, quantile, .985)
  lwr <- apply(post, 2, quantile, .015)

  dat <- data.frame(x, y)

  data.frame(x_tilde, mu, upr, lwr) %>%
    ggplot(aes(x = x_tilde, y = mu)) +
    geom_line() +
    geom_ribbon(aes(x = x_tilde, ymax = upr, ymin = lwr),
                alpha = .25) +
    geom_point(data = dat, aes(x = x, y = y)) +
    theme_minimal()
}

plot_grid(
  plot_pred(fit1, x_tilde, d1$age, d1$height),
  plot_pred(fit2, x_tilde, d1$age, d1$height),
  plot_pred(fit3, x_tilde, d1$age, d1$height),
  plot_pred(fit4, x_tilde, d1$age, d1$height),
  plot_pred(fit5, x_tilde, d1$age, d1$height),
  plot_pred(fit6, x_tilde, d1$age, d1$height),
  ncol = 3
)
```



H4

deviance −2sum(log(q))

```r
dev_fun <- function(fit, y = d2$height, X){
  # a_post <- fit$draws("alpha", format = "df")$alpha
  # b_post <- fit$draws("beta", format = "df")[1:ncol(X)]
  sig_post <- fit$draws("sigma", format = "df")$sigma
  # a <- mean(a_post)
  # b <- apply(b_post, 2, mean)
  post <-fit$draws("y2_pred", format = "df")[,1:length(y)]
  mu <- apply(post, 2, mean)
  sigma <- mean(sig_post)
  #log_lik <- sum(dnorm(y, a + X %*% b, sigma, log = T))
  log_lik <- sum(dnorm(y, mu, sigma, log = T))
  dev <- -2*log_lik
  return(dev)
}

devs <- c(
  dev_fun(fit1, X = dat_1$X2),
  dev_fun(fit2, X = dat_2$X2),
  dev_fun(fit3, X = dat_3$X2),
  dev_fun(fit4, X = dat_4$X2),
  dev_fun(fit5, X = dat_5$X2),
  dev_fun(fit6, X = dat_6$X2)
)

for(i in 1:6){
  print(paste0("Model ", i, " deviance = ", round(devs[i]), 2))
}
```

```
[1] "Model 1 deviance = 24302"
[1] "Model 2 deviance = 21742"
[1] "Model 3 deviance = 19542"
[1] "Model 4 deviance = 18942"
[1] "Model 5 deviance = 18952"
[1] "Model 6 deviance = 19102"
```

H5

```r
print(comp)
```

```
       elpd_diff se_diff
model6    0.0       0.0
model4   -0.6       1.9
model5   -1.3       1.8
model3  -14.0       5.9
model2 -107.9      14.3
model1 -243.2      15.5
```

```
devs - min(devs)
```

```
[1] 536.043868 280.072008  60.824885   0.000000   1.256125  16.372171
```

H6

```
n <- nrow(d1)
x_tilde <- seq(min(d$age), max(d$age), l = 200)

alpha_scale <- 20
beta_scale <- 5


dat_6 <- list(
  N = n,
  K = 6,
  J = 200,
  y = d1$height,
  X = matrix(c(d1$age, d1$age^2, d1$age^3, d1$age^4, d1$age^5, d1$age^6), ncol
= 6),
  y2 = d2$height,
  X2 = matrix(c(d2$age, d2$age^2, d2$age^3, d2$age^4, d2$age^5, d2$age^6), ncol
= 6),
   X_tilde  =  matrix(c(x_tilde,  x_tilde^2,  x_tilde^3,  x_tilde^4,  x_tilde^5,
x_tilde^6), ncol = 6),
  alpha_scale = alpha_scale,
  beta_scale = beta_scale
)

fit6_new <- mod$sample(
  data = dat_6,
  chains = 4,
  parallel_chains = 4,
  show_messages = F
)

fit6$diagnostic_summary()
```

```
$num_divergent
[1] 0 0 0 0

$num_max_treedepth
[1] 0 0 0 0

$ebfmi
[1] 1.044830 1.047019 1.083258 1.043966
```

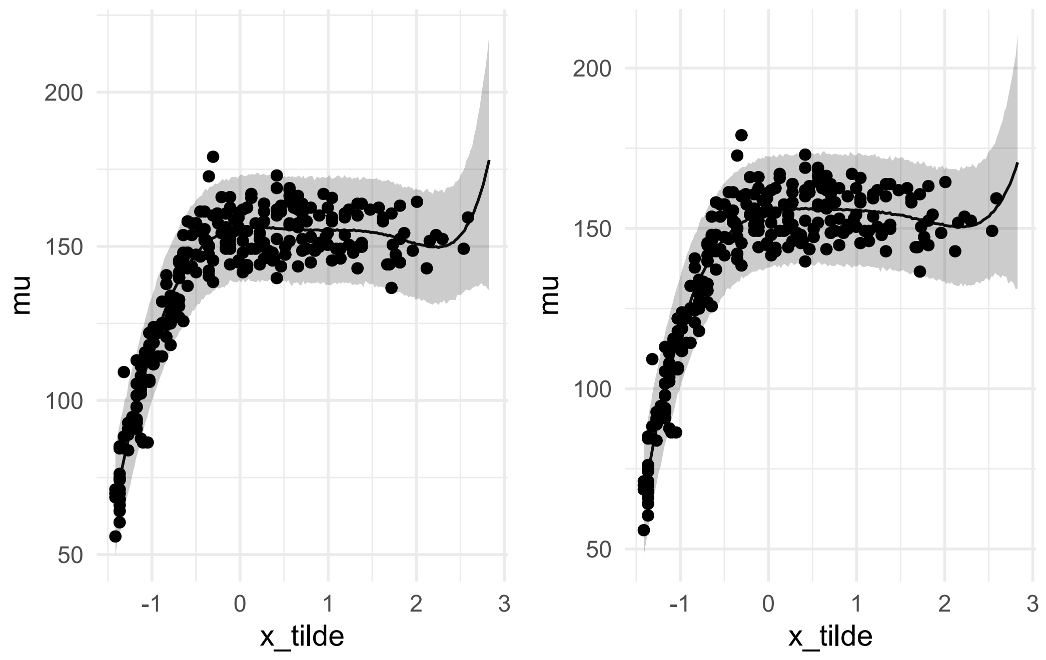
```
loo6_new <- loo(fit6_new$draws("log_lik"))
loo_list <- list(loo1, loo2, loo3, loo4, loo5, loo6, loo6_new)
loo_compare(loo_list)
```

```
       elpd_diff se_diff
model6    0.0       0.0 
model4   -0.6       1.9 
model5   -1.3       1.8 
model7   -1.3       1.7 
model3  -14.0       5.9 
model2 -107.9      14.3 
model1 -243.2      15.5 
```

```
loo_model_weights(loo_list)
```

```
Method: stacking
------
       weight
model1 0.000 
model2 0.024 
model3 0.000 
model4 0.326 
model5 0.000 
model6 0.649 
model7 0.001 
```

```
plot_grid(
  plot_pred(fit6, x_tilde, d1$age, d1$height),
  plot_pred(fit6_new, x_tilde, d1$age, d1$height)
)
```



```
dev_fun(fit6, X = dat_6$X2)
```

```
[1] 1909.979
```

```
dev_fun(fit6_new, X = dat_6$X2)
```

```
[1] 1905.413
```