Computational Cognitive Science 2024-2025

Tutorial 4: Bayesian Model Comparison

Nisbett and Wilson: Are people biased?

We previously used count data from a Nisbett and Wilson study to look at different approaches to parameter estimation. We will now use the same data to look at model comparison.

Options A-D are identical consumer products arranged from left to right in a display. Observations are the number of participants who chose each item in the array.

Choice	A	B	C	D
Observations	6	9	16	21

```
choice_data <- c(6,9,16,21)

choice_data/sum(choice_data)
```

```
## [1] 0.1153846 0.1730769 0.3076923 0.4038462
```

Let's compare two models:

- M1: Uniform categorical distribution. This captures the hypothesis that people are unbiased in their selections.
- M2: Dirichlet with $\alpha = 2$ priors. This captures the hypothesis that there is bias in participants' choices. We aren't using Jeffreys priors because they attach substantial weight to extreme biases, which are unlikely.

We will now try to see which model is supported by the data. We'll ignore the fact that the answer is fairly obvious, and a null hypothesis significance test would probably be adequate here.

Question 1: Are these descriptive or process models?

Solution:

Descriptive; there is no psychological content or account of why choices might look the way they do in either model.

Question 2: We have chosen these models for simplicity. If our goal is to detect left/right bias, can you think of changes that make one or both models more reasonable, without adding much complexity?

Solution:

There are many possible answers, but one issue is that there are many reasons we could prefer M2 over M1 aside from left/right bias, which means that data favoring M2 doesn't imply there's a left/right bias overall, e.g., c(100,0,0,100) would strongly favor M2 but provides evidence against a left/right bias. If detecting left/right bias is our goal, we might collapse A and B into one group and C and D into another and use a beta-binimonal distribution. We might also decide there's a range of biases that are so subtle that they don't matter, leading us to permit some variability in binomial parameters in M1. We might also have intuitions

about plausible biases that favor different priors, which are important: If we aren't prepared to defend our choice of priors, we might not trust our marginal likelihoods.

Question 3: What is the marginal log likelihood for M1?

Solution: *We don't need to integrate over a range of thetas because M1 commits to* $theta_i = .25$. A multinomial distribution describes the probability of independent choice counts. We can use R's built-in `dmultinom` function, which produces log probabilities directly.

```
dmultinom(choice_data,prob=c(.25,.25,.25,.25),log=TRUE)
```

```
## [1] -11.15955
```

Question 4: Finish implementing the importance sampler below.

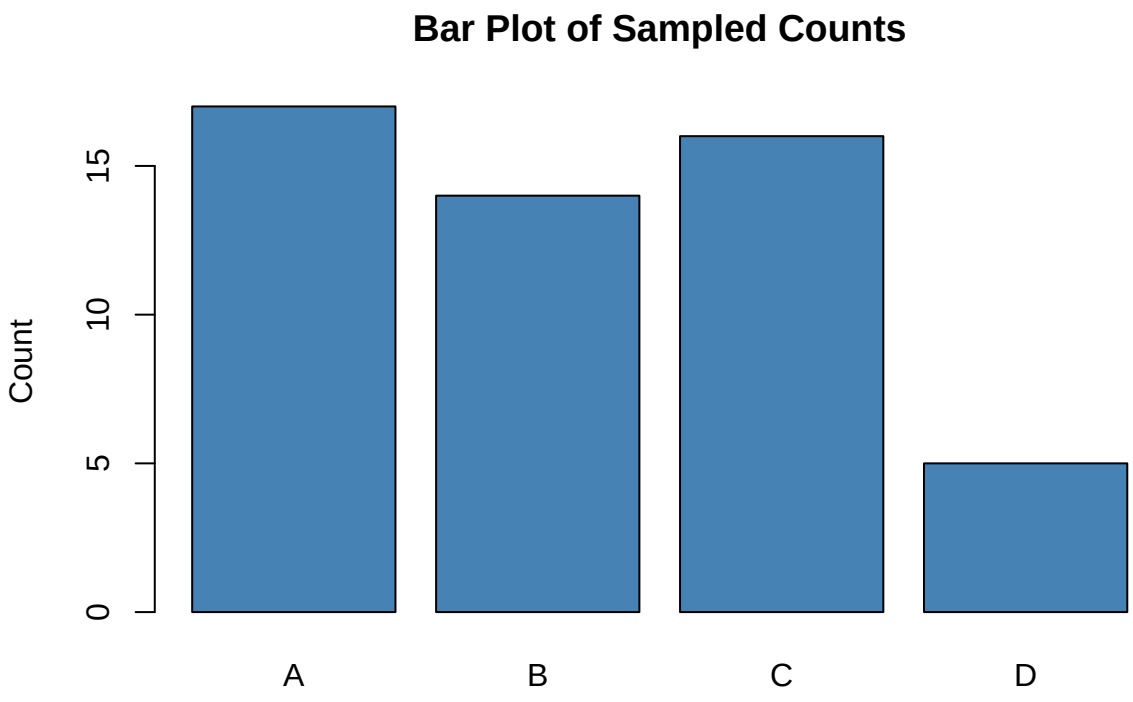
A quick visualization of the generative process for M2 may help clarify things:

```
# --- Parameters ---
# Alpha parameters for the Dirichlet prior
alpha_params <- c(2, 2, 2, 2)
# Total number of trials (from the original data)
total_trials <- sum(choice_data)
# Labels for the choices
choices <- c("A", "B", "C", "D")


# --- Sampling ---
# 1. Draw ONE sample of probabilities from the Dirichlet distribution
# This represents one plausible set of choice probabilities according to the prior
prob_sample <- rdirichlet(1, alpha_params)

# 2. Use those probabilities to draw ONE sample of counts from the multinomial distribution
# This simulates one possible experimental outcome given the probabilities from Step 1
count_sample <- rmultinom(1, size = total_trials, prob = prob_sample)

# --- Visualization ---
barplot(
  height = count_sample[,1],
  names.arg = choices,
  main = "Bar Plot of Sampled Counts",
  xlab = "Choice",
  ylab = "Count",
  col = "steelblue"
)
```




```
impSamp <- function(logTargD, # log target density
                    propR,    # a function to sample proposals
                    logPropD, # the log density for our proposal distribution
                    # Somewhat arbitarily, we are computing
                    # the expectation of the 4th param
                    ef=function(x) {x[,4]}) {
  nSamps <- 100000 # The more the better
  proposals <- propR(nSamps)
  pDens <- logPropD(proposals)
  unnP <- logTargD(proposals)
  w <- exp(unnP-pDens)
  print(paste("Expected value of target function:",
    sprintf("%2.3f",sum(w*ef(proposals))/sum(w))))
  print(paste("Log of average importance weight:",
    sprintf("%2.6f",log(sum(w)/nSamps))))
}
alpha <- 2

# Define logTargD, propR, logPropD. You can use ef to sanity-check --
#   it returns the posterior expectation 4th alpha entry, which you can
#   compute exactly: alpha_k/(sum_i alpha_i), combining real counts and
#   pseudocounts.
# Then call impSamp on these arguments.
```

Solution:

```
prop_alpha <- c(alpha,alpha,alpha,alpha)

propR <- function(n) {rdirichlet(n,prop_alpha)}
logPropD <- function(p) {ddirichlet(p,prop_alpha,log=TRUE)}
```

```r
# We want to compute the likelihood for each of the proposals, all in one go.
# dmultinom doesn't inherently support that, so we're using the apply function
# to apply dmultinom to each row/proposal.
likeli <- function(p) {apply(p,1,function(ps) {out <- dmultinom(choice_data,prob=ps,log=TRUE);out})}
m2_prior <- function(p) {out <- ddirichlet(p,alpha=c(alpha,alpha,alpha,alpha),log=TRUE);out}
# Note that the prior shows up both in the proposal density and the target;
# this is the special case of importance sampling we discussed in lecture.
# We could remove those terms from both density functions since they cancel out.
logTargD <- function(p) {
  lk <- likeli(p);
  pr <- m2_prior(p);
  lk+pr}

impSamp(logTargD,propR,logPropD,function(x) {x[,4]})
```

```
## [1] "Expected value of target function: 0.384"
## [1] "Log of average importance weight: -9.476419"
```

```r
impSamp2 <- function(logTargetDensity, # log of the unnormalized target density (e.g., likelihood * pri
                     proposalSampler,    # a function to sample proposals
                     logProposalDensity, # the log density for our proposal distribution
                     # An optional function to compute the expectation of some value
                     expectationFunction = function(x) {x[,4]}) {

  numSamples <- 100000 # The more samples, the more accurate the approximation

  # 1. Draw a large number of samples from the proposal distribution.
  proposals <- proposalSampler(numSamples)

  # 2. Calculate the log probability density of each proposal under the proposal distribution.
  logProposalDensities <- logProposalDensity(proposals)

  # 3. Calculate the log probability density of each proposal under the unnormalized target distributio
  logUnnormalizedTargetDensities <- logTargetDensity(proposals)

  # 4. Calculate the importance weights.
  # The weight is the ratio of the target density to the proposal density.
  # In log space, this is a subtraction. We use exp() to return to the original scale.
  importanceWeights <- exp(logUnnormalizedTargetDensities - logProposalDensities)

  # 5. Estimate the expectation of the target function by taking a weighted average.
  # The formula is: sum(weights * function(proposals)) / sum(weights)
  estimatedExpectation <- sum(importanceWeights * expectationFunction(proposals)) / sum(importanceWeigh
  print(paste("Expectation of the 4th parameter:",
    sprintf("%2.3f", estimatedExpectation)))

  # 6. Estimate the marginal likelihood (the "evidence").
  # This is approximated by the average of the importance weights.
  # We take the log for numerical stability and for comparison with other log-likelihoods.
  logMarginalLikelihood <- log(sum(importanceWeights) / numSamples)
  print(paste("Log Marginal Likelihood (Evidence) for the model:",
    sprintf("%2.6f", logMarginalLikelihood)))
}
```

```r
# Define the alpha parameter for the Dirichlet distribution
alpha_param <- 2

# The prior distribution for M2 is a Dirichlet with alpha=2 for each of the 4 choices.
# We will use this prior as our proposal distribution.
proposal_alphas <- c(alpha_param, alpha_param, alpha_param, alpha_param)

# A function that draws n samples from our proposal (prior) distribution.
proposalSampler <- function(n) {
  rdirichlet(n, proposal_alphas)
}

# A function that calculates the log density of a given set of parameters 'p'
# under our proposal (prior) distribution.
logProposalDensity <- function(p) {
  ddirichlet(p, alpha=proposal_alphas, log = TRUE)
}

# A function to calculate the log-likelihood of the data for a given set of choice probabilities 'p'.
# The `apply` function is used to calculate the multinomial likelihood for each row (each proposed para
calculateLogLikelihood <- function(p) {
  apply(p, 1, function(single_parameter_set) {
    dmultinom(choice_data, prob = single_parameter_set, log = TRUE)
  })
}

# A function to calculate the log-prior probability for a given set of parameters 'p'.
calculateLogPrior <- function(p) {
    ddirichlet(p, alpha=proposal_alphas, log=TRUE)
}

# The target density is proportional to likelihood * prior.
# In log space, this means log-likelihood + log-prior.
# Note: Since our proposal distribution is the prior, the prior term will cancel out
# when calculating the importance weights. We could have defined logTargetDensity
# as just calculateLogLikelihood, but this way is more explicit.
logTargetDensity <- function(p) {
  log_likelihoods <- calculateLogLikelihood(p)
  log_priors <- calculateLogPrior(p)
  log_likelihoods + log_priors
}

impSamp2(logTargetDensity, proposalSampler, logProposalDensity, function(x) {x[,4]})
```

```
## [1] "Expectation of the 4th parameter: 0.384"
## [1] "Log Marginal Likelihood (Evidence) for the model: -9.472205"
```

Because we are dealing with a conjugate prior, we can also compute the marginal likelihood directly, though deriving that is outside the scope of the course:

```r
ml <- function(pd,d) {lgamma(sum(pd))+lgamma(sum(d)+1) -
    lgamma(sum(pd)+sum(d))+sum(lgamma(pd+d))-sum(lgamma(pd)+lgamma(d+1))}
ml(c(alpha,alpha,alpha,alpha),choice_data)
```

```
## [1] -9.47508
```

Question 5: What is the Bayes factor favoring the more complex model? How should we interpret this?

Solution:

The BF is equal to M2 log likelihood minus the M1 log likelihood, exponentiated to yield a regular probability. The numbers below may vary from yours because the importance sampler has stochastic output.

```
exp(-9.47 - (-11.16))
```

```
## [1] 5.419481
```

It favors the model that suggests people have a bias. We would have to believe (a priori) that the more simpler model is about 5.4 times as probable to prefer it in light of our data.

Question 6: Which model is favored by the BIC? Does the BIC make sense to use, here?

Solution:

Fitting α to maximize likelihood will converge to the the same likelihood as setting multinomial parameters directly. We could also argue that the effective number of parameters is the same as in the multinomial case: 3 rather than 4 as the final parameter is determined by the sum-to-one requirement.

```
mle_nll <- -2*dmultinom(choice_data,prob=choice_data/sum(choice_data),log=TRUE)
penalty <- 4*log(sum(choice_data)) # or 3; see above. We get the same qualitative result.

print(mle_nll+penalty)
```

```
## [1] 27.23487
```

```
print(-2*dmultinom(choice_data,prob=c(.25,.25,.25,.25),log=TRUE))
```

```
## [1] 22.3191
```

No. If we can compute marginal likelihoods, there's little point in using the BIC, and it's a bit peculiar to fit a prior in this way, as conceptually it becomes somewhat superfluous; we could have just fit the multinomial parameters directly.