# Bayesian model criticism and comparison: practical exercises

Anne Presanis

Full worked solutions are provided in section 7 at the end of this document, or, as an R Markdown document (with R code blocks that can be run conveniently from RStudio) in `modelcrit_practical.Rmd`. Feel free to consult these as you are going along, if you are stuck on any of the exercises, or do ask any questions.

As you will have understood by now, there are usually multiple ways of coding something in `R`, so don't worry if the way you manipulate the posterior samples and produce plots is not exactly the same as in the provided solutions. You will see that in the provided solutions, we have tended to use the `posterior R` package and the `tidyverse`, including `ggplot2` to produce the plots in this practical. But if you are more comfortable coding, for example, in base `R`, then that is fine. The concepts of implementing and using the tools introduced for model criticism and comparison, rather than the precise way of coding them in `R`, are what is important to take away from this practical.

In this practical, you will:

- understand the role of prediction and it's use in both in- and out-of-sample prediction for the different purposes of:
  - prediction;
  - model criticism/checking/assessment (goodness of fit and prior-data consistency);
  - model comparison;
- practice manipulating posterior samples after having obtained them to derive various quantities of interest, such as residuals, deviance summaries and prior- and posterior-predictive p-values, to aid in model assessment and comparison;
- practice plotting credible and prediction intervals and other posterior summaries to aid in visual model assessment and comparison;
- practice thinking about the different aspects of model fit, sensitivity analysis and model comparison that are the key ingredients in the model development and assessment cycle.

Note that throughout this practical, recognising that the two examples used (the HIV avidity index and the Beta-Binomial drug example) are straightforward enough that convergence is good, we have omitted some model fitting aspects, such as considering initial values (letting `JAGS/rjags` set them by default) and checking convergence with trace plots and some of the diagnostics seen in previous sessions. We are instead concentrating on model assessment here. However, in general, you should always consider these aspects of model fitting with any new model you are developing and fitting.

Outline of exercises in this practical:

1. Introducing the HIV avidity index example

   1.1 In- and out-of-sample prediction for purposes of prediction

2. Bayesian p-values: posterior-predictive checks

   2.1 Exercises on posterior-predictive checks

3. Bayesian residuals
4. Saturated deviance
5. Checking the prior: sensitivity analyses and prior-data conflict

6. Model comparison

# 1 Non-linear regression example: HIV avidity index

We will start by implementing the HIV avidity index example seen first in the regression session 3 then in the model criticism session 4. Recall that the "avidity index" (AI) is an immunological biomarker of recency of HIV infection, where a value of 0.8 acts as a threshold to define "recency". For a set $y_i, i = 1, \dots N$ of observations of the AI, at times $t_i, i = 1, \dots N$ since seroconversion (time $t = 0$, assumed to be the midpoint between last negative and last positive HIV test), we assume a non-linear growth model of the form:

$$y_i \sim \mathrm{N}(\mu_i, \sigma^2), \ i = 1, \dots N$$

where

$$\mu_i = \mu_\infty - \mu_{\text{diff}} \gamma_i^{t_i}.$$

Assuming a log-normal prior centred on 1 and with most prior mass $< 5$ for both the asymptote $\mu_\infty$ and the change in AI $\mu_{\text{diff}}$, a uniform prior for the proportion $\gamma$ of final size left to grow at time $t = 1$ year since seroconversion, and a relatively flat Gamma prior for the precision, gives us this implementation in `JAGS`:

```
hivai_mod <- "model {
for (i in 1:N) {
  y[i] ~ dnorm(mu[i], prec)
  mu[i] <- mu_inf - mu_diff*gamma^t[i]
 }
 mu_inf ~ dlnorm(log(1), 0.3)
 mu_diff ~ dlnorm(log(1), 0.3)
 gamma ~ dunif(0, 1)
 prec ~ dgamma(0.001, 0.001)
}"
```

The observations $y_i$ at times $t_i$ are saved in a tibble, that we turn into a list:

```
library(tidyverse)

hivai_dat <- read_csv(file = "hivai.csv")
(hivai_lst <- list(t = hivai_dat$time,
                   y = hivai_dat$AI,
                   N = nrow(hivai_dat)))
```

```
## $t
##  [1]  0.1  0.2  0.5  0.7  1.5  1.7  2.3  2.6  3.0  4.0  5.0  8.0 10.0
##
## $y
##  [1] 0.5424213 0.4887628 0.8294032 0.9268438 1.0614872 0.9936381 1.1669538
##  [8] 1.0060968 1.0486331 1.0916076 0.9871549 0.9379852 1.1524494
##
## $N
## [1] 13
```

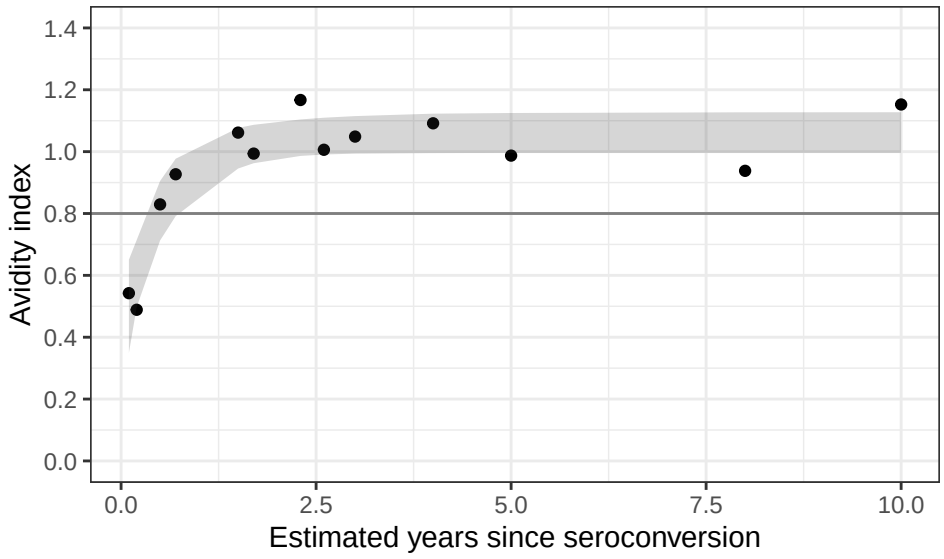
Let's run the model and plot the fitted mean $\mu_i$ against the observed data. `geom_point()` plots the observations as points, `geom_ribbon` plots the credible interval for $\mu_i$ as a filled band:

```
library(rjags)
library(posterior)

# number of chains and iterations to run, and parameters to monitor
nChains <- 2
```

```r
nIter <- 10000
hivai_pars <- c("mu","gamma","mu_inf","mu_diff","prec")

# fit the model
hivai_jag <- jags.model(textConnection(hivai_mod), data=hivai_lst,
                        quiet=TRUE, n.chains = nChains)
hivai_sam <- coda.samples(hivai_jag, hivai_pars, n.iter=nIter)

# obtain posterior summaries of the fitted mean mu
hivai_draws <- as_draws(hivai_sam)
(hivai_mu <- summary(hivai_draws, ~quantile(.x, probs = c(0.025,0.5,0.975))) %>%
  filter(str_detect(variable, "mu\\[")))
```

```
## # A tibble: 13 x 4
##    variable `2.5%` `50%` `97.5%`
##    <chr>     <dbl> <dbl>   <dbl>
##  1 mu[1]     0.349 0.496   0.651
##  2 mu[2]     0.497 0.600   0.714
##  3 mu[3]     0.713 0.806   0.905
##  4 mu[4]     0.792 0.888   0.977
##  5 mu[5]     0.945 1.02    1.08
##  6 mu[6]     0.963 1.03    1.09
##  7 mu[7]     0.986 1.05    1.10
##  8 mu[8]     0.991 1.05    1.11
##  9 mu[9]     0.994 1.05    1.11
## 10 mu[10]    0.995 1.06    1.12
## 11 mu[11]    0.996 1.06    1.12
## 12 mu[12]    0.996 1.06    1.13
## 13 mu[13]    0.996 1.06    1.13
```

```r
# plot the observed AI values by years since seroconversion, and overlay the
# fitted mean (combine observed and posterior in one tibble first)
hivai_dat %>%
  bind_cols(hivai_mu) %>%
  ggplot() +
  aes(x = time, y = AI) +
  geom_point() +
  geom_ribbon(aes(ymin=`2.5%`,ymax=`97.5%`), alpha = 0.2) +
  labs(x="Estimated years since seroconversion", y="Avidity index") +
  scale_y_continuous(breaks=seq(0, 1.4, by=0.2), limits=c(0,1.4)) +
  geom_hline(yintercept=0.8, col="gray50") +
  theme_bw() +
  theme(legend.position = "bottom")
```




### 1.1 Exercises: Prediction and out-of-sample prediction

a. Amend the `JAGS` model code to obtain the posterior-predictive distribution of the data and rerun the model.

b. Add the 95% posterior-predictive interval to the plot of the observed vs fitted $\mu_i$ above.

c. Alternatively, you can use the posterior samples you have already obtained to obtain the posterior-predictive distribution in `R` directly.

d. Referring to the lecture slides 8-9, amend the dataset so that you can predict the avidity index at times 1/12 and 11 years.

e. Reproduce the plot on slide 9 showing the credible interval for $\mu_i$ and the posterior-predictive interval, as well as the two predicted values of AI.

## 2 Bayesian p-values: posterior-predictive checks

To illustrate the use of posterior-predictive checks to assess model fit, we will artificially create an outlier in the HIV avidity index dataset and re-run the model to obtain the posterior of $\mu_i$ and posterior-predictive distribution of $y_i^{rep}$:

```
## Artificially create an outlier in the HIV AI example at time 8
hivai_out <- hivai_dat %>%
  mutate(
    AI = if_else(time == 8, 0.7, AI)
  )
hivai_out_lst <- list(t = hivai_out$time, y = hivai_out$AI, N = hivai_lst$N)

## Rerun model
hivai_out_jag <- jags.model(textConnection(hivai_mod), data=hivai_out_lst,
                            quiet=TRUE, n.chains = nChains)
hivai_out_sam <- coda.samples(hivai_out_jag, hivai_pars, n.iter=nIter)

# obtain posterior summaries of the fitted mean mu and posterior-predictive
hivai_out_draws <- as_draws(hivai_out_sam)
hivai_out_mu_pp <- summary(
    subset_draws(hivai_out_draws, c(paste0("mu[",1:hivai_lst$N,"]"),
```

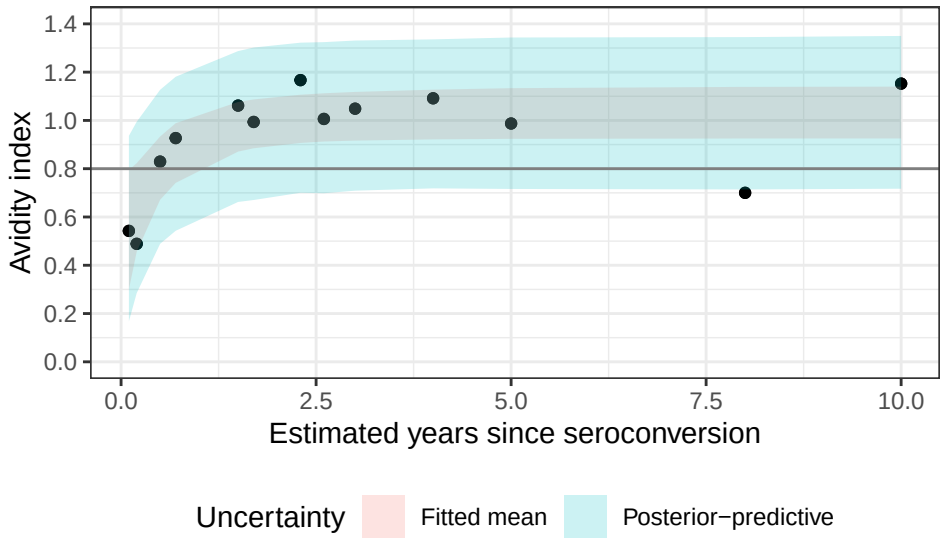
```
                                    paste0("yrep[",1:hivai_lst$N,"]"))),
    ~quantile(.x, probs = c(0.025,0.5,0.975))
  )
hivai_out_mu_pp %>% print(n = Inf)
```

```
## # A tibble: 26 x 4
##    variable `2.5%` `50%` `97.5%`
##    <chr>     <dbl> <dbl>   <dbl>
##  1 mu[1]     0.300 0.530   0.795
##  2 mu[2]     0.456 0.622   0.823
##  3 mu[3]     0.671 0.803   0.933
##  4 mu[4]     0.741 0.872   0.988
##  5 mu[5]     0.871 0.986   1.08
##  6 mu[6]     0.885 0.996   1.09
##  7 mu[7]     0.907 1.01    1.11
##  8 mu[8]     0.913 1.02    1.11
##  9 mu[9]     0.916 1.02    1.12
## 10 mu[10]    0.921 1.03    1.13
## 11 mu[11]    0.924 1.03    1.13
## 12 mu[12]    0.925 1.03    1.14
## 13 mu[13]    0.925 1.03    1.14
## 14 yrep[1]   0.167 0.528   0.936
## 15 yrep[2]   0.284 0.622   0.994
## 16 yrep[3]   0.489 0.801   1.13
## 17 yrep[4]   0.543 0.871   1.18
## 18 yrep[5]   0.662 0.986   1.29
## 19 yrep[6]   0.670 0.995   1.30
## 20 yrep[7]   0.700 1.01    1.32
## 21 yrep[8]   0.698 1.02    1.32
## 22 yrep[9]   0.709 1.02    1.33
## 23 yrep[10]  0.718 1.03    1.34
## 24 yrep[11]  0.716 1.03    1.34
## 25 yrep[12]  0.714 1.03    1.35
## 26 yrep[13]  0.717 1.03    1.35
```

We have a look at the usual plot and note that both the credible interval for $\mu_i$ and the posterior-predictive interval are wider than in our earlier plots, to accommodate (just) the outlier:

```
# combine observed and posterior in one tibble first
# then plot the observed AI values by years since seroconversion
# and overlay the fitted mean and posterior predictive
hivai_out_mu_pp %>% # posterior summaries
  mutate(       # create an observation number (index)
                # and type of uncertainty
    index = rep(1:hivai_out_lst$N, 2),
    Uncertainty = if_else(str_detect(variable, "mu"),
                          "Fitted mean",
                          "Posterior-predictive")
  ) %>%
  left_join(    # join with observed data by observation number (index)
    hivai_out %>%
      mutate(index = 1:hivai_lst$N),
    by = "index"
  ) %>%
  ggplot() +
```

```
  aes(x = time, y = AI) +
  geom_point() +
  geom_ribbon(aes(ymin=`2.5%`,ymax=`97.5%`,fill=Uncertainty), alpha = 0.2) +
  labs(x="Estimated years since seroconversion", y="Avidity index") +
  scale_y_continuous(breaks=seq(0, 1.4, by=0.2), limits=c(0,1.4)) +
  geom_hline(yintercept=0.8, col="gray50") +
  theme_bw() +
  theme(legend.position = "bottom")
```




### 2.1 Exercises: posterior predictive checks

a. Recreate the plot from slide 12 showing where the outlying observation at time 8 lies in its posterior-predictive distribution and the corresponding Bayesian p-value. Remember that given a sample of $M$ draws from a distribution, you can obtain the p-value corresponding to the tail area below an observation by comparing the draws to the observation and taking the mean. Note that you can use the `ggplot2` functions `geom_density()` to plot a density given a set of samples from the distribution; `geom_vline()` to plot a vertical line at a particular x-axis intercept; and `annotate()` to add text to the plot.

b. Obtain the posterior-predictive p-values for each of the avidity index observations and produce a histogram (see `geom_histogram()`) to summarise them. Since the sample size is small, the histogram may not give a good idea of whether the p-values are uniform or not, but the idea should give you some feeling for how posterior-predictive p-values may be conservative, but nevertheless useful for detecting outliers.

## 3 Bayesian residuals

We will use the same HIV example with the artificial outlier to explore the use of Bayesian versions of residual plots to assess goodness of fit.

a. Use your previously obtained posterior samples in Section 3 to calculate the posterior samples of the standardised Pearson residuals and the corresponding p-values. Recall that the p-values are obtained from the Normal(0,1) cumulative distribution function (`pnorm()`), at the quantiles given by residuals.

b. Recreate the residual density plot on slide 18 of the lecture. To plot densities of the posterior residuals, note that you can use the `ggridges R` package, using `geom_density_ridges()`. To order the densities

on the y-axis by increasing time since seroconversion from the top of the axis to the bottom (the reverse of the default), use the `scale_y_reverse()` function in `ggplot2`.

c. Plot the corresponding p-values, first obtaining the appropriate posterior quantiles, grouped by time/observation. To plot horizontal error bars, you can use `geom_errorbarh()`.

We will break here for the second half of the model criticism lecture.

# 4 Saturated deviance

We will now turn to the Beta-Binomial drug model, to illustrate assessing model fit with the posterior mean saturated deviance, using this dataset:

```
y <- c(11, 11, 11, 10, 11, 13, 13, 14, 8, 14)
N <- 20
m <- length(y)
```

a. Referring to slide 28, implement the model, including in the `JAGS` model code the calculation of the saturated binomial deviance. Recall that for $m$ i.i.d. observations $y_1, \ldots, y_m \sim \text{Binomial}(n, \pi)$, this deviance has the form:

$$D_S(\pi) = -2 \sum_{i=1}^{m} \left[ y_i \log \frac{\pi}{y_i/n} + (n - y_i) \log \frac{(1-\pi)}{(1-(y_i/n))} \right]$$

If you are equivalently calculating the deviance for a single one of these observations $y_i$ at a time, the corresponding expression is:

$$D_S(\pi) = -2 \left[ y_i \log \frac{\pi}{y_i/n} + (n - y_i) \log \frac{(1-\pi)}{(1-(y_i/n))} \right]$$

b. Alternatively, if you had not included an expression for the deviance in the model code, you could calculate it in `R` rather than `JAGS`.

# 5 Checking the prior

We will continue with the Beta-Binomial drug example to explore sensitivity analyses to the prior and prior-data conflict.

First, let's make a slight amendment to the model code and data file so that the parameters of the Beta prior for $\pi$ are not hard-coded into the model code, but instead are set in the data file:

```
# model
bb_mod <- "
model {
      # Prior
      pi ~ dbeta(a,b)

      for(i in 1:m) {
        # Likelihood
        y[i] ~ dbinom(pi, N)

        # Posterior predictive
        yrep[i] ~ dbinom(pi, N)
      }
    }
"
bb_pars <- c("pi","yrep")
```

a. Run the model for each of the three priors discussed in slides 31-32, i.e. the vague Beta(0.5,0.5) prior, the original prior Beta(9.2, 13.8) and the second informative prior Beta(9.2, 36.8).

b. Reproduce the plot on slide 32 comparing the posterior distributions of $\pi$ from each of the three models. Recall that you can use `geom_density()` to plot the density of a set of samples. Notice the differences in the posteriors - which prior is most informative?

c. Before carrying out the inferences, we could have run prior-predictive checks to assess the consistency of the most informative prior with the observed data. Following the `R` code in slide 34, obtain the prior-predictive p-values for each observed value of $y$ for the most informative prior. Is the prior consistent with any of the data points?

# 6 Model comparison

We return to the HIV avidity index example, the version of the data with the artificial outlier, to explore model comparison.

a. Following the lecture notes in slides 39-40, implement the HIV aviditiy index model using a $t$-distribution with 2 degrees of freedom, instead of a Normal distribution for the sampling distribution.

b. Compare the $t$ and Normal models informally by recreating the plot on slide 41 comparing the posterior distributions for $\mu_i$ and the posterior-predictive distributions for $y_i^{rep}$ for each sampling distribution.

c. Use the `JAGS/rjags` provided implementations of DIC and penalised expected deviance to recreate the table on slide 42. Have a look at the help file for `dic.samples()` to see how to use it to obtain the penalised deviance statistics. Remember that you will need to load the DIC module (`load.module("dic")`) of `rjags` first, and note that you will require having run at least 2 chains to use the DIC module. Note also that the values you obtain may not be exactly the same as the ones in the slides, but will be within Monte Carlo error. Which model fits the data better?

d. For the Normal model, calculate your own saturated deviance $D_S(\mu_i, \sigma^2)$ for each observation $y_i$; and therefore calculate the plug-in deviance at the posterior means of the parameters, $p_D$ and the $DIC$. Remember that (slide 24) for $m$ i.i.d. observations $y_1, \ldots, y_m \sim N(\mu, \sigma^2)$, the saturated deviance is

$$D_S(\mu, \sigma) = m \log\left(\frac{\sigma^2}{\hat{\sigma}^2}\right) + \frac{1}{\sigma^2}\sum_{i=1}^{m}(y_i - \mu)^2 - \frac{1}{\hat{\sigma}^2}\sum_{i=1}^{m}(y_i - \hat{\mu})^2$$

where $\hat{\mu}$ and $\hat{\sigma}$ are the MLEs of $\mu$ and $\sigma$. However, in this case, we have a different mean $\mu_i$ for each observation, $y_1, \ldots, y_m \sim N(\mu_i, \sigma^2)$, so we only have a single observation for each time point. In this case, the saturated deviance has the form

$$D_S(\mu_i, \sigma) = \frac{(y_i - \mu_i)^2}{\sigma^2}$$

# 7 Solutions

## 7.1 Prediction and out-of-sample prediction

a. Amend the JAGS model code to obtain the posterior-predictive distribution of the data and rerun the model. If amending the JAGS model code, the posterior-predictive distribution can be obtained with the addition of a replicate `yrep` drawn from a distribution with the same parameters as the sampling distribution:

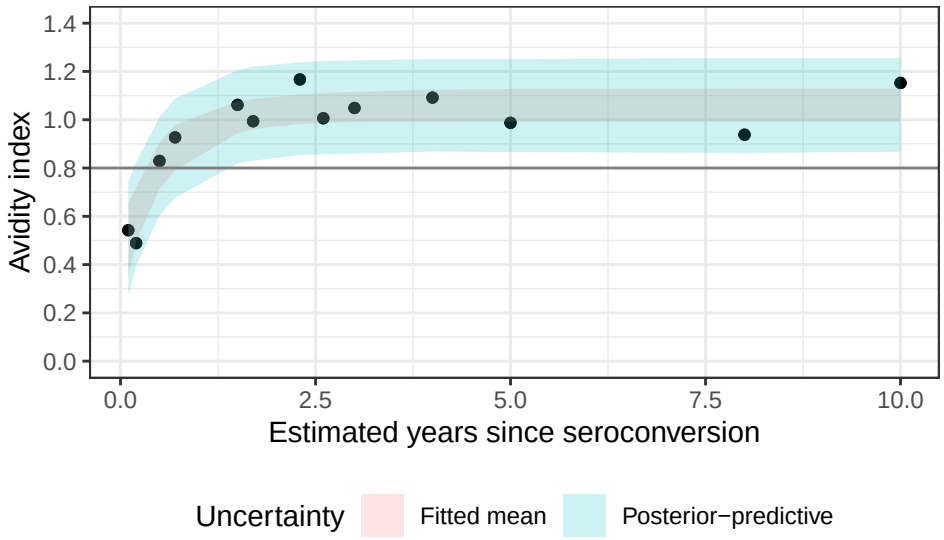
```
hivai_mod <- "model {
for (i in 1:N) {
  y[i] ~ dnorm(mu[i], prec)
  yrep[i] ~ dnorm(mu[i], prec)
  mu[i] <- mu_inf - mu_diff*gamma^t[i]
 }
 mu_inf ~ dlnorm(log(1), 0.3)
 mu_diff ~ dlnorm(log(1), 0.3)
 gamma ~ dunif(0, 1)
 prec ~ dgamma(0.001, 0.001)
}"

# Add yrep into the list of parameters to save
hivai_pars <- c("mu","gamma","mu_inf","mu_diff","prec","yrep")

# fit the model
hivai_jag <- jags.model(textConnection(hivai_mod), data=hivai_lst,
                    quiet=TRUE, n.chains = nChains)
hivai_sam <- coda.samples(hivai_jag, hivai_pars, n.iter=nIter)

# obtain posterior summaries of the fitted mean mu and posterior-predictive
hivai_draws <- as_draws(hivai_sam)
hivai_mu_pp <- summary(
  subset_draws(hivai_draws, c(paste0("mu[",1:hivai_lst$N,"]"),
                        paste0("yrep[",1:hivai_lst$N,"]"))),
  ~quantile(.x, probs = c(0.025,0.5,0.975))
)
hivai_mu_pp %>% print(n = Inf)
```

```
## # A tibble: 26 x 4
##    variable `2.5%` `50%` `97.5%`
##    <chr>     <dbl> <dbl>   <dbl>
##  1 mu[1]     0.350 0.499   0.655
##  2 mu[2]     0.496 0.602   0.718
##  3 mu[3]     0.713 0.806   0.906
##  4 mu[4]     0.791 0.887   0.978
##  5 mu[5]     0.944 1.02    1.08
##  6 mu[6]     0.961 1.03    1.09
##  7 mu[7]     0.983 1.05    1.10
##  8 mu[8]     0.987 1.05    1.11
##  9 mu[9]     0.990 1.05    1.12
## 10 mu[10]    0.992 1.06    1.12
## 11 mu[11]    0.993 1.06    1.13
## 12 mu[12]    0.993 1.06    1.13
## 13 mu[13]    0.993 1.06    1.13
## 14 yrep[1]   0.269 0.498   0.740
```

```
## 15 yrep[2]   0.395 0.602   0.825
## 16 yrep[3]   0.603 0.808   1.01
## 17 yrep[4]   0.677 0.888   1.09
## 18 yrep[5]   0.819 1.02    1.20
## 19 yrep[6]   0.830 1.03    1.22
## 20 yrep[7]   0.853 1.05    1.24
## 21 yrep[8]   0.857 1.05    1.24
## 22 yrep[9]   0.859 1.05    1.25
## 23 yrep[10]  0.868 1.06    1.25
## 24 yrep[11]  0.866 1.06    1.25
## 25 yrep[12]  0.862 1.06    1.26
## 26 yrep[13]  0.867 1.06    1.26
```

b. Add the 95% posterior-predictive interval to the plot of the observed vs fitted $\mu_i$ above. Starting with the posterior summaries created in 1a, we first create a variable to describe the type of uncertainty (either the fitted mean or the posterior-predictive interval). Then we combine with the observed data so that we can add another "ribbon" to the plot, by specifying that the "fill" colours for the ribbon are defined by the type of uncertainty:

```
# plot the observed AI values by years since seroconversion
# and overlay the fitted mean and posterior predictive intervals
hivai_mu_pp %>% # posterior summaries
  mutate(       # create an observation number (index)
           # and type of uncertainty
index = rep(1:hivai_lst$N, 2),
Uncertainty = if_else(str_detect(variable, "mu"),
                      "Fitted mean",
                      "Posterior-predictive")
  ) %>%
  left_join(    # join with observed data by observation number (index)
hivai_dat %>%
  mutate(index = 1:hivai_lst$N),
by = "index"
  ) %>%
  ggplot() +
  aes(x = time, y = AI) +
  geom_point() +
  geom_ribbon(aes(ymin=`2.5%`,ymax=`97.5%`,fill=Uncertainty), alpha = 0.2) +
  labs(x="Estimated years since seroconversion", y="Avidity index") +
  scale_y_continuous(breaks=seq(0, 1.4, by=0.2), limits=c(0,1.4)) +
  geom_hline(yintercept=0.8, col="gray50") +
  theme_bw() +
  theme(legend.position = "bottom")
```




c. Alternatively, you can use the posterior samples of you have already obtained to obtain the posterior-predictive distribution in `R` directly. Using the posterior samples of $\mu_i$ already obtained, we can sample from the posterior-predictive distribution as follows:

- turn the `draws` object into a two-dimensional matrix, with dimension `nIter*nChains` rows and number of columns given by the number of monitored variables

```
hivai_mat <- as_draws_matrix(hivai_draws)
```

- Create an array of the same dimension to store the posterior predictive samples

```
hivai_yrep <- array(dim = dim(hivai_mat[, paste0("mu[",1:hivai_lst$N,"]")]))
```

- For each time point/observation, draw `nIter*nChains` replicates from the posterior-predictive distribution, i.e. a Normal with mean given by $mu_t$ and standard deviation by $(1/\text{prec})^{0.5}$

```
for(t in 1:hivai_lst$N) {
  hivai_yrep[,t] <- rnorm(nIter*nChains,
                  mean = hivai_mat[, paste0("mu[",t,"]")],
                  sd = (1 / hivai_mat[, "prec"])^0.5)
}
```

- make sure the `yrep` array has appropriate column names

```
colnames(hivai_yrep) <- paste0("yrep[",1:hivai_lst$N,"]")
```

- summarise to obtain the posterior-predictive quantiles

```
(hivai_yrep_post <- summary(as_draws(hivai_yrep),
                        ~quantile(.x, probs = c(0.025,0.5,0.975))))
```

```
## # A tibble: 13 x 4
##    variable `2.5%` `50%` `97.5%`
##    <chr>     <dbl> <dbl>   <dbl>
##  1 yrep[1]   0.269 0.498   0.752
##  2 yrep[2]   0.393 0.603   0.820
##  3 yrep[3]   0.598 0.807   1.01
##  4 yrep[4]   0.683 0.888   1.09
##  5 yrep[5]   0.816 1.02    1.21
##  6 yrep[6]   0.832 1.03    1.22
##  7 yrep[7]   0.849 1.05    1.24
```

```
##  8 yrep[8]   0.859 1.05    1.24
##  9 yrep[9]   0.860 1.05    1.25
## 10 yrep[10]  0.864 1.06    1.25
## 11 yrep[11]  0.866 1.06    1.25
## 12 yrep[12]  0.869 1.06    1.25
## 13 yrep[13]  0.866 1.06    1.25
```

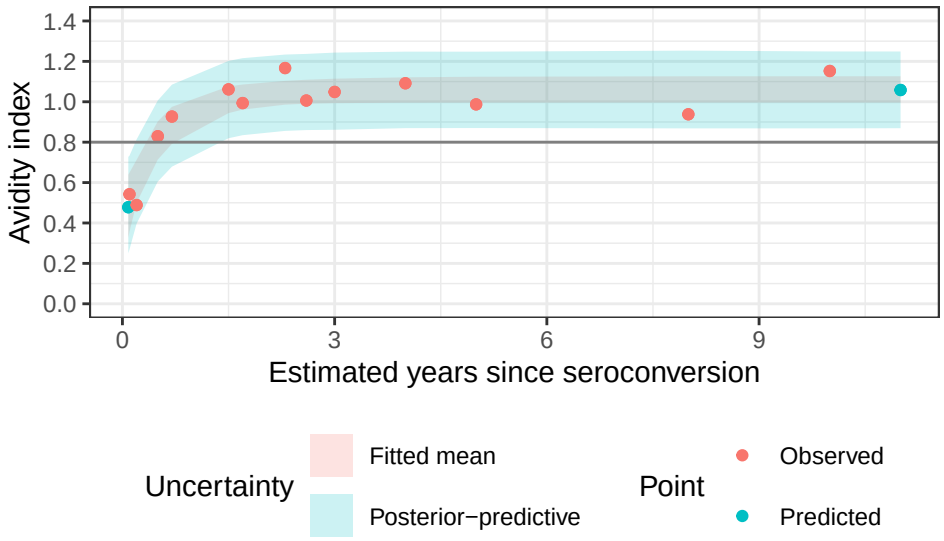
d. Referring to the lecture slides 8-9, amend the dataset so that you can predict the avidity index at times 1/12 and 11 years.

```
# Use missing values (NAs) in the data file
hivai_miss <- hivai_lst
hivai_miss$t <- c(1/12, hivai_lst$t, 11)
hivai_miss$y <- c(NA, hivai_lst$y, NA)
hivai_miss$N <- hivai_lst$N + 2

# rerun model
hivai_miss_jag <- jags.model(textConnection(hivai_mod), data=hivai_miss,
                         quiet=TRUE, n.chains = nChains)
hivai_miss_sam <- coda.samples(hivai_miss_jag, hivai_pars, n.iter=nIter)

# obtain posterior summaries of the fitted mean mu and posterior-predictive
hivai_miss_draws <- as_draws(hivai_miss_sam)
hivai_miss_mu_pp <- summary(
subset_draws(hivai_miss_draws,
             c(paste0("mu[",1:hivai_miss$N,"]"),
               paste0("yrep[",1:hivai_miss$N,"]"))
             ),
~quantile(.x, probs = c(0.025,0.5,0.975))
  )
```

e. Reproduce the plot on slide 9 showing the credible interval for $\mu_i$ and the posterior-predictive interval, as well as the two predicted values of AI. As in 1b., we first create an appropriate tibble with the credible interval for $\mu_i$, the posterior-predictive interval for $y_i^{rep}$, the observed data, and variables giving the labels for each that will be used in the plot. Then we use the `fill` aspect in `geom_ribbon` to colour the ribbon by type of uncertainty and the `col` aspect in `geom_point` to colour the points by whether they are observed or predicted.

```
hivai_miss_mu_pp %>% # posterior summaries
  mutate(
index = rep(1:hivai_miss$N,2),
Uncertainty = if_else(
    str_detect(variable,"mu"),
    "Fitted mean", "Posterior-predictive"
  )
  ) %>%
  left_join(    # join with observed data by observation number (index)
tibble(time = hivai_miss$t, AI = hivai_miss$y) %>%
  mutate(index = 1:hivai_miss$N),
by = "index"
  ) %>%
  mutate(       # rename to use as labels in plot
Point = if_else(is.na(AI), "Predicted", "Observed"),
AI = if_else(is.na(AI), `50%`, AI)
  ) %>%
  ggplot() +
```

```
  aes(x = time, y = AI) +
  geom_ribbon(aes(ymin=`2.5%`,ymax=`97.5%`,fill=Uncertainty), alpha = 0.2) +
  geom_point(aes(col = Point)) +
  labs(x="Estimated years since seroconversion", y="Avidity index") +
  scale_y_continuous(breaks=seq(0, 1.4, by=0.2), limits=c(0,1.4)) +
  geom_hline(yintercept=0.8, col="gray50") +
  theme_bw() +
  theme(legend.position = "bottom") +
  guides(col=guide_legend(nrow=2,byrow=TRUE), fill=guide_legend(nrow=2,byrow=TRUE))
```




## 7.2 Bayesian p-values: posterior-predictive checks

a. Recreate the plot from slide 12 showing where the outlying observation at time 8 lies in its posterior-predictive distribution and the corresponding Bayesian p-value.

- We first create an indicator for which observation is at time 8 (the 12th observation):

```
ind8 <- (1:nrow(hivai_out))[hivai_out$time == 8]
```

- Next we pull out the posterior predictive distribution for the outlying observation:

```
yrep8 <- hivai_out_draws %>%
  extract_variable(paste0("yrep[",ind8,"]")) %>%
  as_tibble()
```

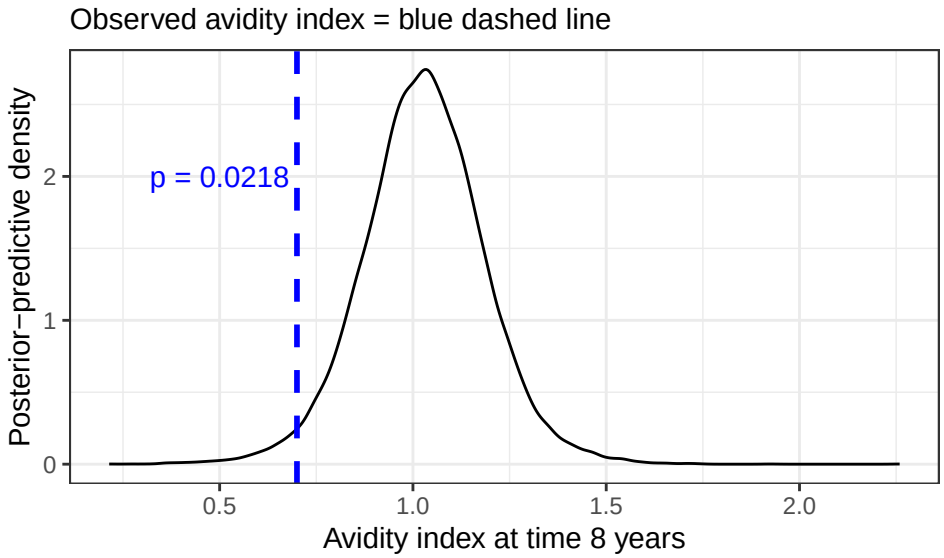
- And the outlying observation:

```
obs8 <- as.numeric(hivai_out$AI[ind8])
```

- And finally create the plot. We use `geom_density()` to plot the density of the posterior-predictive samples in `yrep8`; `geom_vline()` to plot a vertical line at the observed outlier `obs8`; and `annotate()` to add the posterior-predictive p-value to the plot, calculating it as the mean over the posterior-predictive samples of the comparison of `yrep8` to `obs8`.

```
yrep8 %>%
  ggplot() +
  aes(x = value) +
  geom_density() +
  geom_vline(aes(xintercept = obs8),
```

```
        color="blue", linetype="dashed", linewidth=1) +
  labs(x = "Avidity index at time 8 years",
   y = "Posterior-predictive density",
   subtitle = "Observed avidity index = blue dashed line") +
  annotate(geom="text", x=0.5, y=2, label=paste0("p = ", mean(yrep8$value < obs8)),
        color="blue") +
  theme_bw()
```




b. Obtain the posterior-predictive p-values for each of the avidity index observations and produce a histogram (see `geom_histogram()`) to summarise them.

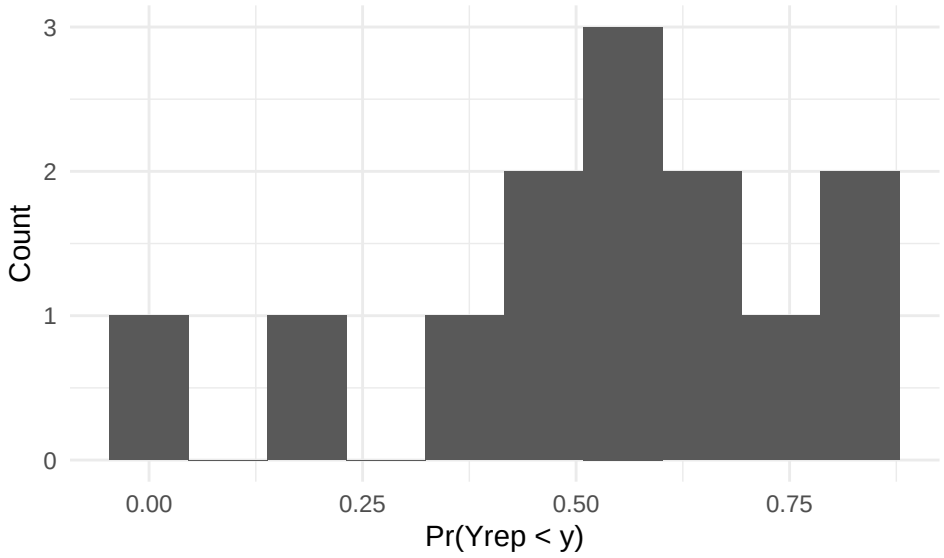
- We first combine the posterior-predictive samples and observed data in a single tibble, repeating the observations for each MCMC iteration:

```
hivai_out_ppobs <- as_draws_matrix(hivai_out_draws) %>%
  subset_draws(paste0("yrep[",1:hivai_out_lst$N,"]")) %>%
  as_tibble(rownames = "Iter") %>%  # posterior-predictive samples as a tibble
  pivot_longer(                     # reshape to long format to ease combining with data
cols = contains("yrep"),
names_to = "Index",
values_to = "yrep"
  ) %>%
  mutate(                           # make Index an indicator for observation number
Index = as.integer(str_remove(str_remove(Index, "yrep\\["), "\\]"))
  ) %>%
  left_join(                        # join observed data, repeating for each iteration
hivai_out %>%
  mutate(Index = 1:nrow(hivai_out)),
by = "Index"
  )
```

- Then we obtain the p-values by summarising over each observation (`Index`) and produce the histogram using `geom_histogram`, setting the number of bins for the p-values to be grouped in to 10:

```
hivai_out_ppobs %>%
  group_by(Index) %>%
  summarise(
```

```
pval = mean(yrep < AI),
.groups = "keep"
  ) %>%
  ungroup() %>%
  ggplot() +
  aes(x = pval) +
  geom_histogram(bins = 10) +
  labs(x = "Pr(Yrep < y)", y = "Count") +
  theme_minimal()
```




- Since the sample size is small, the histogram may not give a good idea of whether the p-values are uniform or not, but the idea should give you some feeling for how posterior-predictive p-values may be conservative, but nevertheless useful for detecting outliers. Note the clustering of p-values around 0.5.

## 7.3 Bayesian residuals

a. Use your previously obtained posterior samples in Section 3 to calculate the posterior samples of the standardised Pearson residuals and the corresponding p-values. Recall that the p-values are obtained from the Normal(0,1) cumulative distribution function (`pnorm()`), at the quantiles given by residuals.

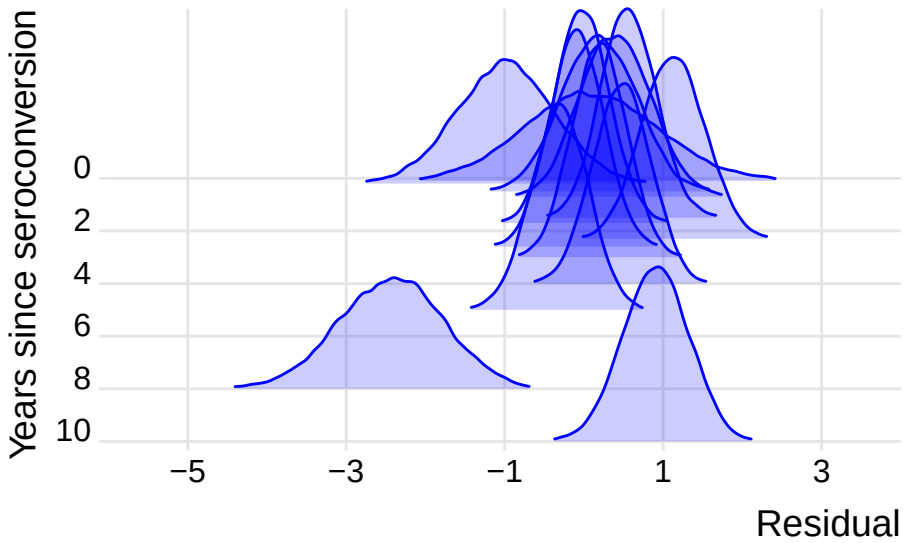
```
## Calculate residuals for example with artificial outlier
hivai_out_resids <- as_draws_matrix(hivai_out_draws) %>%
  subset_draws(variable = c(paste0("mu[",1:hivai_out_lst$N,"]"),"prec")) %>%
  as_tibble(rownames = "Iter") %>%
  pivot_longer(
cols = contains("mu"),
names_to = "Index",
values_to = "mu"
  ) %>%
  mutate(
Iter = as.integer(Iter),
Index = as.integer(str_remove_all(str_remove_all(Index, "mu\\["), "\\]")),
time = rep(hivai_out$time, nIter*nChains),    # add data into tibble, repeated for
y = rep(hivai_out$AI, nIter*nChains),         # each MCMC iteration
sigma = (1 / prec)^0.5,
```

```
res = (y - mu) / sigma,                    # standardised Pearson residual
p.res = pnorm(res)                         # p-value
  )
```

b. Recreate the residual density plot on slide 18 of the lecture. To plot densities of the posterior residuals, we use the `ggridges R` package with the `geom_density_ridges()` function. To order the densities on the y-axis by increasing time since seroconversion from the top of the axis to the bottom (the reverse of the default), we use the `scale_y_reverse()` function in `ggplot2`.

```
library(ggridges)

# plot posterior distribution of each residual
hivai_out_resids %>%
  ggplot() +
  aes(x = res, y = time, group = time) +
  geom_density_ridges(scale = 10, size = 0.5, alpha = 0.2, rel_min_height = 0.01,
                      col = "blue", fill = "blue") +
  theme_ridges() +
  scale_x_continuous(limits = c(-6, 4), breaks = seq(-5,3,2), expand = c(0, 0)) +
  scale_y_reverse(
breaks = seq(12, 0, by = -2),
expand = c(0, 0)
  ) +
  coord_cartesian(clip = "off") +
  labs(x = "Residual", y = "Years since seroconversion")
```

```
## Picking joint bandwidth of 0.0566
```




c. Plot the corresponding p-values, first obtaining the appropriate posterior quantiles, grouped by time/observation. We order the rows in order of increasing `Median` p-value, using the `tidyverse arrange()` function:

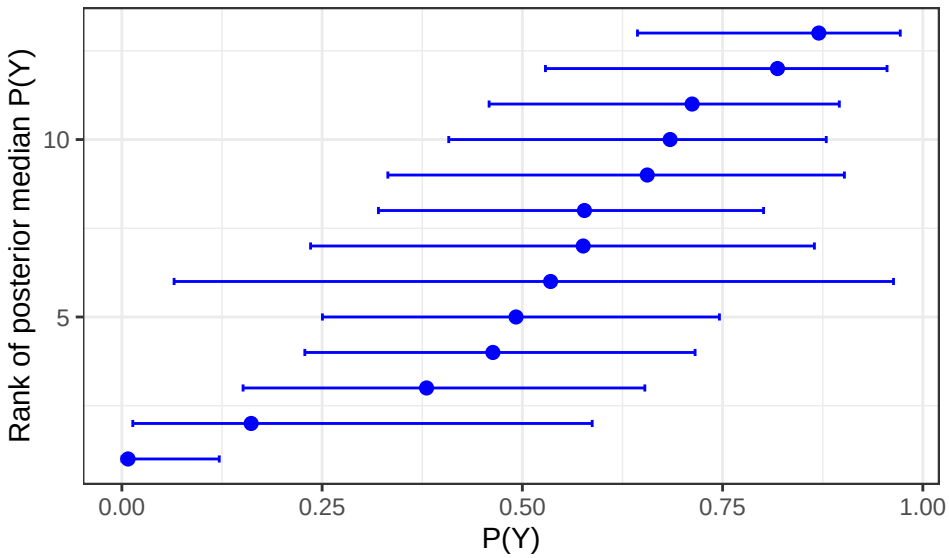
```
hivai_out_resids_pvals <- hivai_out_resids %>%
  select(time, p.res, Iter) %>%
  mutate(p.res = as.double(p.res)) %>% # Make sure the p-values are numeric (moving from
                                       # draws object to tibble leaves columns as draws).
  group_by(time) %>%                       # Group by time/observation
  summarise(Median = quantile(p.res, probs = 0.5),
```

```
        Lower = quantile(p.res, probs = 0.025),
        Upper = quantile(p.res, probs = 0.975), # Obtain quantiles
        .groups = "keep") %>%
  ungroup() %>%
  arrange(Median)
hivai_out_resids_pvals
```

```
## # A tibble: 13 x 4
##     time  Median     Lower Upper
##    <dbl>   <dbl>     <dbl> <dbl>
##  1   8   0.00768 0.0000826 0.122
##  2   0.2 0.161   0.0136    0.587
##  3   5   0.380   0.151     0.653
##  4   2.6 0.463   0.228     0.716
##  5   1.7 0.492   0.250     0.746
##  6   0.1 0.535   0.0652    0.963
##  7   0.5 0.576   0.236     0.865
##  8   3   0.577   0.320     0.801
##  9   0.7 0.656   0.332     0.902
## 10   4   0.684   0.408     0.879
## 11   1.5 0.712   0.458     0.896
## 12  10   0.819   0.529     0.955
## 13   2.3 0.870   0.644     0.972
```

- Next we plot the quantiles of the p-values, using `geom_point()` for the median and `geom_errorbarh()` for the credible interval:

```
hivai_out_resids_pvals %>%
  ggplot() +
  aes(x = Median, y = 1:nrow(hivai_out_resids_pvals)) +
  geom_point(size = 2, col = "blue") +
  geom_errorbarh(aes(xmin = Lower, xmax = Upper, height = 0.2),
                 col = "blue") +
  labs(x = "P(Y)", y = "Rank of posterior median P(Y)") +
  theme_bw()
```



## 7.4 Saturated deviance

a. Referring to slide 28, implement the model, including in the `JAGS` model code the calculation of the saturated binomial deviance. We add in a line to the code to calculate the deviance as `dev.pi[i] <- -2 * ((y[i] * (log(pi) - log(y[i]/N))) + ((N-y[i]) * (log(1-pi) - log(1 - (y[i]/N)))))`, run the model, and obtain the posterior quantiles and mean of the $\pi$, the deviance, and $\pi N$:

```
# Model
bb_mod <- "
model {
  # Prior
  pi ~ dbeta(9.2, 13.8)

  for(i in 1:m) {
# Likelihood
y[i] ~ dbinom(pi, N)

# Binomial saturated deviance
dev.pi[i] <- -2 * ((y[i] * (log(pi) - log(y[i]/N))) +
    ((N-y[i]) * (log(1-pi) - log(1 - (y[i]/N)))))
  }
}
"

# data and parameters to save
bb_dat <- list(y = y, m = m, N = N)
bb_pars <- c("pi","dev.pi")

# run model
bb_jag <- jags.model(textConnection(bb_mod), data=bb_dat,
                 quiet=TRUE, n.chains = nChains)
bb_sam <- coda.samples(bb_jag, bb_pars, n.iter=nIter)

# calculate pi*N at each MCMC iteration
bb_draws <- as_draws(bb_sam) %>%
  mutate_variables(yhat = pi * N)

# calculate posterior mean deviance contributions of each data point
bb_dev <- summary(bb_draws,
              ~quantile(.x, probs = c(0.025,0.5,0.975)),
              mean = ~mean(.x)) %>%
  bind_cols(y = c(y,NA,NA))
bb_dev
```

```
## # A tibble: 12 x 6
##    variable     `2.5%`   `50%` `97.5%`    mean     y
##    <chr>         <dbl>   <dbl>   <dbl>   <dbl> <dbl>
##  1 dev.pi[1]  0.000103  0.0447   0.495  0.0990    11
##  2 dev.pi[2]  0.000103  0.0447   0.495  0.0990    11
##  3 dev.pi[3]  0.000103  0.0447   0.495  0.0990    11
##  4 dev.pi[4]  0.00292   0.303    1.29   0.392     10
##  5 dev.pi[5]  0.000103  0.0447   0.495  0.0990    11
##  6 dev.pi[6]  0.0545    0.652    1.94   0.739     13
##  7 dev.pi[7]  0.0545    0.652    1.94   0.739     13
##  8 dev.pi[8]  0.497     1.62     3.43   1.71      14
```

```
##  9 dev.pi[9]  0.737     2.09     4.13   2.19       8
## 10 dev.pi[10] 0.497     1.62     3.43   1.71      14
## 11 pi         0.496     0.561    0.625  0.561     NA
## 12 yhat       9.91     11.2     12.5   11.2       NA
```

b. Alternatively, if you had not included an expression for the deviance in the model code, you could calculate it in `R` rather than `JAGS`:

- First, we define a function to calculate binomial saturated deviance:

```
dev.bin <- function(pi, y, n){
  -2 * ((y * (log(pi) - log(y/n))) +
      ((n-y) * (log(1-pi) - log(1 - (y/n)))))
}
```

- Next, we combine the observed data and posterior samples in one tibble, repeating the observations for each iteration, for ease of manipulation:

```
bb_dev <- as_tibble(subset_draws(as_draws_matrix(bb_draws),
                                 variable = c("pi","yhat")),
                    rownames = "Iter") %>%
  bind_cols(
tibble(y = y, index = paste0("y_",1:length(y))) %>%
  pivot_wider(names_from = index, values_from = y)
  ) %>%
  pivot_longer(              # long format
cols = contains("y_"),
names_to = "index",
values_to = "y",
names_prefix = "y_"
  )
```

- Finally, we calculate the deviances at each MCMC iteration, then summarise them, grouped by observation number, to obtain the posterior means:

```
bb_dev %>%
  mutate(
index = as.integer(index),
dev = dev.bin(pi = pi, y = y, n = N)
  ) %>%
  group_by(index) %>%
  summarise(
pi = mean(pi),
y = mean(y),
yhat = mean(yhat),
dev = mean(dev),
.groups = "keep"
  ) %>%
  ungroup()
```

```
## # A tibble: 10 x 5
##    index    pi     y  yhat    dev
##    <int> <dbl> <dbl> <dbl>  <dbl>
##  1     1 0.561    11  11.2 0.0990
##  2     2 0.561    11  11.2 0.0990
##  3     3 0.561    11  11.2 0.0990
##  4     4 0.561    10  11.2 0.392
```

```
##  5     5 0.561    11  11.2 0.0990
##  6     6 0.561    13  11.2 0.739
##  7     7 0.561    13  11.2 0.739
##  8     8 0.561    14  11.2 1.71
##  9     9 0.561     8  11.2 2.19
## 10    10 0.561    14  11.2 1.71
```

## 7.5 Checking the prior: sensitivity analyses and prior-data conflict

a. Run the model for each of the three priors discussed in slides 31-32, i.e. the vague Beta(0.5,0.5) prior, the original prior Beta(9.2, 13.8) and the second informative prior Beta(9.2, 36.8):

```
# fit model using each of the three priors
bb1_dat <- list(y = y, m = m, N = N, a = 0.5, b = 0.5)
bb1_jag <- jags.model(textConnection(bb_mod), data=bb1_dat, quiet=TRUE, n.chains = nChains)
bb1_sam <- as_draws(coda.samples(bb1_jag, bb_pars, n.iter=nIter))

bb2_dat <- list(y = y, m = m, N = N, a = 9.2, b = 13.8)
bb2_jag <- jags.model(textConnection(bb_mod), data=bb2_dat, quiet=TRUE, n.chains = nChains)
bb2_sam <- as_draws(coda.samples(bb2_jag, bb_pars, n.iter=nIter))

bb3_dat <- list(y = y, m = m, N = N, a = 9.2, b = 36.8)
bb3_jag <- jags.model(textConnection(bb_mod), data=bb3_dat, quiet=TRUE, n.chains = nChains)
bb3_sam <- as_draws(coda.samples(bb3_jag, bb_pars, n.iter=nIter))
```

b. Reproduce the plot on slide 32 comparing the posterior distributions of $\pi$ from each of the three models.

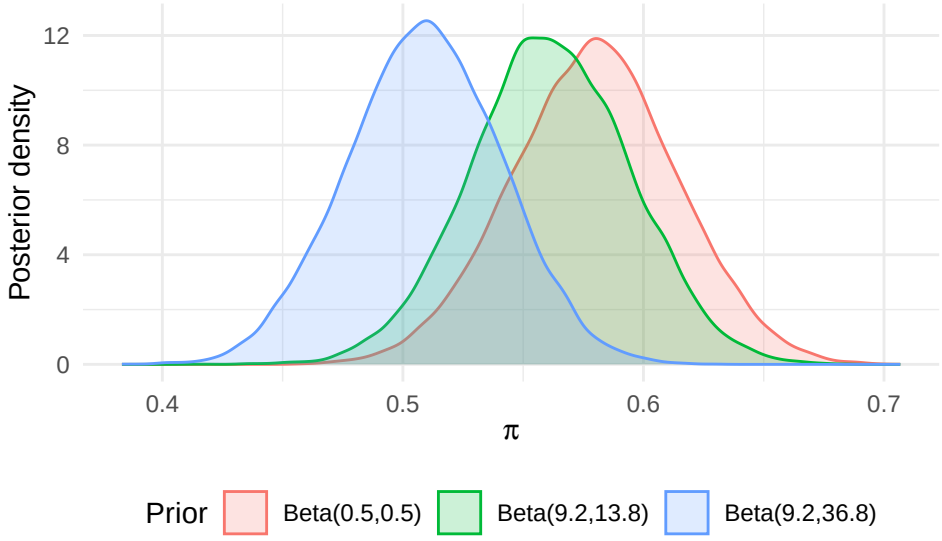
- First, we combine the three sets of posterior samples in a single tibble:

```
pi3priors <- tibble(
  pi = extract_variable(bb1_sam, "pi"),
  Prior = "Beta(0.5,0.5)",
  Iter = 1:(nIter*nChains)
) %>%
  bind_rows(
tibble(
  pi = extract_variable(bb2_sam, "pi"),
  Prior = "Beta(9.2,13.8)",
  Iter = 1:(nIter*nChains)
)
  ) %>%
  bind_rows(
tibble(
  pi = extract_variable(bb3_sam, "pi"),
  Prior = "Beta(9.2,36.8)",
  Iter = 1:(nIter*nChains)
)
  )
```

- Next we plot the posterior of $\pi$ under each prior, using `geom_density()` to plot the density of each set of posterior samples. We set `alpha=0.2` to control the transparency of the fill colours, so that we can easily see the overlaps between the three densities.

```
pi3priors %>%
  ggplot() +
  aes(x = pi, col = Prior, fill = Prior) +
  geom_density(alpha = 0.2) +
```

```
  labs(x = expression(pi), y = "Posterior density") +
  theme_minimal() +
  theme(legend.position = "bottom")
```




- Notice the differences in the posteriors: the vague Beta(0.5, 0.5) prior, as expected, results in a posterior that is close to the empirical distribution of the data (i.e. the posterior reflects mostly the likelihood); the original Beta(9.2, 13.8) prior suggest a distribution of $\pi$ that is slightly smaller than the data alone would suggest; and the second Beta(9.2, 36.8) prior is most informative, pulling the posterior even further away from the data.
- Although the most informative prior has most influence on the inference, we are not (at this stage) suggesting that it is somehow "wrong": it might be that two previous studies did truly find two different results, and that by carrying out these sensitivity analyses, we are understanding how different the posterior estimates could be under the two alternative priors.
- If we believed both prior studies were equally valid, we could have used a prior that was a compromise between the two.

c. Before carrying out the inferences, we could have run prior-predictive checks to assess the consistency of the most informative prior with the observed data. Following the `R` code in slide 34, obtain the prior-predictive p-values for each observed value of $y$ for the most informative prior.

```
# Prior
pi_pri <- rbeta(n = nChains*nIter, shape1 = 9.2, shape2 = 36.8)

# Prior-predictive samples
yrep_pri <- sapply(pi_pri, function(x) rbinom(n = 1, size = N, prob = x))

# Prior-predictive p-values
tibble(
  y = y,
  prior_p = sapply(y, function(x) { mean(yrep_pri < x) })
)
```

```
## # A tibble: 10 x 2
##        y prior_p
##    <dbl>   <dbl>
##  1    11   0.997
##  2    11   0.997
##  3    11   0.997
```

```
##  4    10   0.989
##  5    11   0.997
##  6    13   1.00
##  7    13   1.00
##  8    14   1.00
##  9     8   0.939
## 10    14   1.00
```

- Note that the prior-predictive probabilities are all greater than 0.93, suggesting that the prior is inconsistent with all of the data points (except perhaps marginally the 9th observation, which has the smallest value 8).

## 7.6 Model comparison

a. Following the lecture notes in slides 39-40, implement the HIV avidity index model using a $t$-distribution with 2 degrees of freedom, instead of a Normal distribution for the sampling distribution. We use `dt(mu[i], prec, 2)` to draw from the $t$-distribution with 2 degrees of freedom, and print out the resulting posterior quantiles:

```
# model
hivai_t_mod <- "model {
for (i in 1:N) {
  y[i] ~ dt(mu[i], prec, 2)
  yrep[i] ~ dt(mu[i], prec, 2)
  mu[i] <- mu_inf - mu_diff*gamma^t[i]
 }
 mu_inf ~ dlnorm(log(1), 0.3)
 mu_diff ~ dlnorm(log(1), 0.3)
 gamma ~ dunif(0, 1)
 prec ~ dgamma(0.001, 0.001)
}"

# Run model
hivai_t_jag <- jags.model(textConnection(hivai_t_mod), data=hivai_out_lst,
                      quiet=TRUE, n.chains = nChains)
hivai_t_draws <- as_draws(coda.samples(hivai_t_jag, hivai_pars, n.iter=nIter))

# posterior summaries
summary(hivai_t_draws,
    ~quantile(.x, probs = c(0.5,0.025,0.975))) %>%
  print(n = Inf)
```

```
## # A tibble: 30 x 4
##    variable   `50%`  `2.5%` `97.5%`
##    <chr>      <dbl>   <dbl>   <dbl>
##  1 gamma      0.136  0.0299   0.412
##  2 mu[1]      0.517  0.296    0.724
##  3 mu[2]      0.616  0.473    0.770
##  4 mu[3]      0.818  0.713    0.911
##  5 mu[4]      0.897  0.791    0.978
##  6 mu[5]      1.02   0.942    1.08
##  7 mu[6]      1.03   0.958    1.09
##  8 mu[7]      1.05   0.980    1.12
##  9 mu[8]      1.05   0.983    1.12
## 10 mu[9]      1.06   0.986    1.13
```

```
## 11 mu[10]     1.06   0.989    1.14
## 12 mu[11]     1.06   0.989    1.14
## 13 mu[12]     1.06   0.989    1.15
## 14 mu[13]     1.06   0.990    1.15
## 15 mu_diff    0.669  0.385    1.02
## 16 mu_inf     1.06   0.990    1.15
## 17 prec     190.    47.6    643.
## 18 yrep[1]    0.512  0.120    0.919
## 19 yrep[2]    0.615  0.254    0.997
## 20 yrep[3]    0.817  0.457    1.17
## 21 yrep[4]    0.896  0.546    1.24
## 22 yrep[5]    1.02   0.655    1.38
## 23 yrep[6]    1.03   0.677    1.37
## 24 yrep[7]    1.05   0.687    1.42
## 25 yrep[8]    1.05   0.695    1.40
## 26 yrep[9]    1.06   0.702    1.40
## 27 yrep[10]   1.06   0.697    1.43
## 28 yrep[11]   1.06   0.700    1.42
## 29 yrep[12]   1.06   0.708    1.42
## 30 yrep[13]   1.06   0.704    1.43
```

b. Compare the $t$ and Normal models informally by recreating the plot on slide 41 comparing the posterior distributions for $\mu_i$ and the posterior-predictive distributions for $y_i^{rep}$ for each sampling distribution.

- We first obtain the credible and predictive intervals from the $t$-distribution model and the equivalents from the Normal model:

```
# t-distribution
hivai_t_sum <- summary(
subset_draws(
  as_draws_matrix(hivai_t_draws), c("mu","yrep")
),
~quantile(.x, probs = c(0.025,0.975))
  )

# Normal distribution
hivai_n_sum <- summary(
  subset_draws(
as_draws_matrix(hivai_out_draws), c("mu","yrep")
  ),
  ~quantile(.x, probs = c(0.025,0.975))
)
```

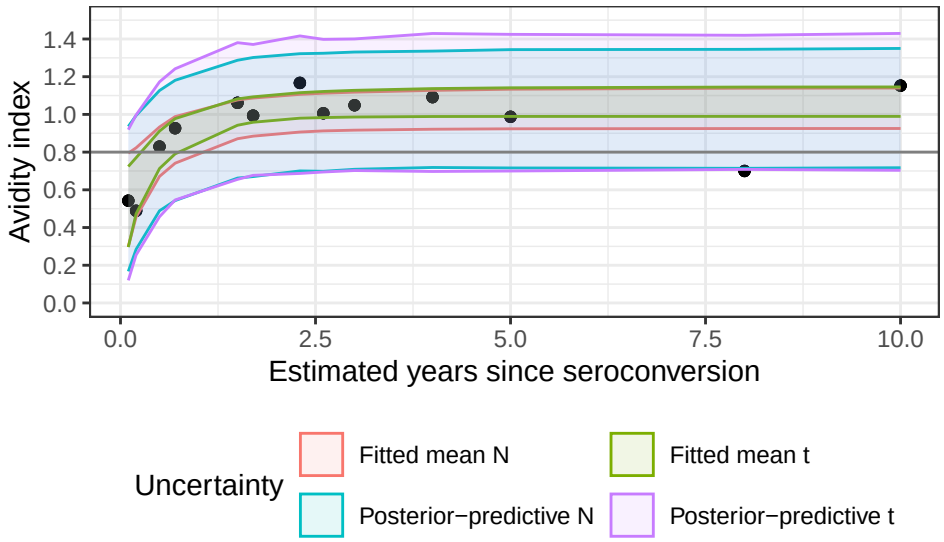
- Then we combine them in a single tibble for plotting, together with the observed data, creating a variable to label the model and type of uncertainty:

```
hivai_tVsN <- as_tibble(hivai_t_sum) %>%
  mutate(
Uncertainty = if_else(str_detect(variable,"mu"),
                      "Fitted mean t", "Posterior-predictive t")
  ) %>%
  bind_rows(
as_tibble(hivai_n_sum) %>%
  mutate(
    Uncertainty = if_else(str_detect(variable,"mu"),
                          "Fitted mean N", "Posterior-predictive N")
```

```
  )
  ) %>%
  mutate(
index = rep(1:hivai_out_lst$N, 4),
AI = rep(hivai_out_lst$y, 4),
time = rep(hivai_out_lst$t, 4)
  )
```

- And finally we plot them, using again the `fill` attribute to assign colours to the two models and two types of uncertainty:

```
hivai_tVsN %>%
  ggplot() +
  aes(x = time, y = AI) +
  geom_point() +
  geom_ribbon(aes(ymin=`2.5%`,ymax=`97.5%`,col=Uncertainty,fill=Uncertainty),
          alpha = 0.1) +
  labs(x="Estimated years since seroconversion", y="Avidity index") +
  scale_y_continuous(breaks=seq(0, 1.5, by=0.2), limits=c(0,1.5)) +
  geom_hline(yintercept=0.8, col="gray50") +
  theme_bw() +
  theme(legend.position = "bottom") +
  guides(col=guide_legend(nrow=2,byrow=TRUE), fill=guide_legend(nrow=2,byrow=TRUE))
```




c. Use the `JAGS/rjags` provided implementations of DIC and penalised expected deviance to recreate the table on slide 42. We use the `dic.samples()` function from `JAGS/rjags`, after first loading the DIC module:

```
load.module("dic")
```

```
## module dic loaded
```

```
# Obtain both types of DIC from JAGS using unstandardised deviance
(hivai_N_out_dic <- dic.samples(hivai_out_jag, n.iter = 1000, type = "pD"))
```

```
## Mean deviance:  -14.64
## penalty 4.343
## Penalized deviance: -10.3
```

```
(hivai_N_out_ped <- dic.samples(hivai_out_jag, n.iter = 1000, type = "popt"))
```

```
## Mean deviance:  -14.7
## penalty 11.27
## Penalized deviance: -3.432
```

```
(hivai_t_out_dic <- dic.samples(hivai_t_jag, n.iter = 1000, type = "pD"))
```

```
## Mean deviance:  -18.43
## penalty 3.517
## Penalized deviance: -14.91
```

```
(hivai_t_out_ped <- dic.samples(hivai_t_jag, n.iter = 1000, type = "popt"))
```

```
## Mean deviance:  -18.61
## penalty 11.41
## Penalized deviance: -7.198
```

- Notice that posterior mean deviances are similar between the DIC and PED; and that the PED penalises stronger than the DIC. The $t$-distribution appears to fit the data better than the Normal distribution, whichever penalisation we use.

d. For the Normal model, calculate your own saturated deviance $D_S(\mu_i, \sigma^2)$ for each observation $y_i$; and therefore calculate the plug-in deviance at the posterior means of the parameters, $p_D$ and the $DIC$.

- First we define a function to calculate the saturated deviance for a Normal distribution. The expression used depends on whether we are calculating for a single data point or for multiple i.i.d. observations, as in the formulae in the question:

```
# function to calculate Normal deviance
dev.normal <- function(y, mu, sigma){
  if(length(y) == 1 | length(y) == length(mu)) {
ret <- (y - mu)^2 / (sigma^2)
 }
 else if(length(y) > length(mu)) {
   m <- length(y)
   muhat <- mean(y)
   sigmahat <- (1/m) * sum((y - muhat)^2)
   ret <- (m * log((sigma^2) / (sigmahat^2)))
+ ((1 / (sigma^2))*sum((y-mu)^2)) - ((1 / (sigmahat^2))*sum((y-muhat)^2))
 }
 else
   ret <- NA

 ret
}
```

- Next we combine the posterior samples of $\mu_i$ and $y_i^{rep}$ in a single tibble:

```
hivai_N_post <- as_tibble(
subset_draws(
  as_draws_matrix(hivai_out_draws),
  c(paste0("mu[",1:hivai_out_lst$N,"]"),
    paste0("yrep[",1:hivai_out_lst$N,"]"))
),
rownames = "Iter"
  )
```

- We reshape to long format for easier manipulation and combine with the observed data (repeated for each iteration):

```
hivai_N_post <- hivai_N_post %>%
  pivot_longer(
cols = -Iter,
names_to = "variable",
values_to = "value"
  ) %>%
  mutate(
index = rep(rep(1:hivai_out_lst$N, 2), nChains*nIter),
y = rep(rep(hivai_out_lst$y, 2), nChains*nIter),
variable = if_else(str_detect(variable, "mu"), "mu", "yrep")
  )
```

- Next, we reformat back to wide form and combine with the posterior samples of the precision:

```
hivai_N_post <- hivai_N_post %>%
  pivot_wider(
id_cols = c(Iter, index, y),
names_from = variable,
values_from = value
  ) %>%
  left_join(
as_tibble(
  subset_draws(as_draws_matrix(hivai_out_draws),"prec"),
  rownames = "Iter"
),
by = "Iter"
  )
```

- We calculate the saturated deviance at each MCMC iteration:

```
hivai_N_post <- hivai_N_post %>%
  mutate(
mu = as.double(mu),
prec = as.double(prec),
yrep = as.double(yrep),
sigma = (1/prec)^0.5,
dev = dev.normal(y = y, mu = mu, sigma = sigma)
  )
```

- And finally, we calculate the plug-in deviance, $p_D$ and DIC:

```
hivai_N_dic <- hivai_N_post %>%
  group_by(index) %>%
  summarise(
y = mean(y),
mu = mean(mu),
sigma = mean(sigma),
Dbar = mean(dev),
.groups = "keep"
  ) %>%
  mutate(
Dplug = dev.normal(y = y, mu = mu, sigma = sigma),
pD = Dbar - Dplug,
DIC = Dbar + pD
```

```
  ) %>%
  ungroup() %>%
  select(-index)

hivai_N_dic
```

```
## # A tibble: 13 x 7
##        y    mu sigma  Dbar       Dplug    pD   DIC
##    <dbl> <dbl> <dbl> <dbl>       <dbl> <dbl> <dbl>
##  1 0.542 0.534 0.143 0.714 0.00321     0.711 1.42
##  2 0.489 0.626 0.143 1.36  0.922       0.439 1.80
##  3 0.829 0.803 0.143 0.257 0.0348      0.222 0.480
##  4 0.927 0.870 0.143 0.361 0.156       0.205 0.566
##  5 1.06  0.983 0.143 0.440 0.302       0.138 0.579
##  6 0.994 0.994 0.143 0.116 0.000000865 0.116 0.233
##  7 1.17  1.01  0.143 1.43  1.16        0.270 1.70
##  8 1.01  1.02  0.143 0.122 0.00615     0.116 0.239
##  9 1.05  1.02  0.143 0.151 0.0357      0.116 0.267
## 10 1.09  1.03  0.143 0.357 0.206       0.151 0.509
## 11 0.987 1.03  0.143 0.225 0.0834      0.142 0.368
## 12 0.7   1.03  0.143 6.39  5.31        1.08  7.48
## 13 1.15  1.03  0.143 0.991 0.726       0.264 1.25
```