CGS698C, Lectures 15-18: Model comparison and hypothesis testing

Himanshu Yadav

2024-04-04

Contents

1 *How to judge the performance of a model?*

1. Does the model capture important properties of the observed data?
2. With what accuracy does the model predict future data?
   - Models differ in their predictive accuracy and complexity
   - We prefer models with fewer assumptions and higher predictive accuracy

1.1 *Posterior predictive checks*

- An approach to assess the fit of a model to data
- Suppose you are given observed data, y containing N data points (i.e., sample size is N)
- Our goal is to generate n samples, each containing N data points, from the posterior predictive distribution
- **Steps:**
  1. Fit the model to observed data, y
  2. Obtain the posterior distribution, $\hat{p}(\theta|y)$
  3. Generate posterior predictive data.
     Let i be the index of data samples to be generated from the posterior predictive distribution, $i \in 1, 2, ..n$. In order to generate i^{th} sample from the posterior predictive distribution:
     (a) Sample a set of parameter value(s) from the posterior distribution, $\theta_i \sim \hat{p}(\theta|y)$
     (b) Draw a sample of N data points from the sampling distribution conditional on θ_i, such that $y_{sim_i} \sim f(y|\theta_i)$. The sample y_{sim_i} contains N data points. We will call y_{sim_i} a *simulated data sample*
     (c) Generate n samples of simulated data, $y_{sim_1}, y_{sim_2}, .., y_{sim_n}$
  4. **Graphical posterior predictive checks:** Display observed data alongside simulated data samples
     – You can also choose to display a particular summary statistic of the samples, e.g., sample means
     – Do the simulated samples differ from observed data in terms of some key features e.g., mean, variance etc.?
  5. **Numerical posterior predictive checks:** Numerically evaluate the discrepancy between observed and simulated data in terms of certain properties, e.g., maximum, mean, median, range, etc.
  6. Choose a test statistic of samples, $T()$. T is a feature that you want to compare between simulated samples and the observed data sample. It could be, for example, mean, range, etc.
  7. Calculate test statistic for each simulate data sample, $T(y_{sim_i})$ and the actual data sample, $T(y)$
  8. How many $T(y_{sim_i})$ are larger than $T(y)$?

  - How many times a certain feature of simulated data exceeds that of actual data? **Does the model generate 'enough' and 'not too' extreme observations?**

9. Compute the so-called Bayesian p-values, $p = Pr(T(y_{sim}) > T(y)|y)$, i.e., the proportion of simulated samples that have $T(y_{sim_i})$ larger than $T(y)$
10. The model is a bad fit if $p < 0.2$ or $p > 0.8$
    - $p < 0.2$ means that the model does not generate 'enough extreme' observations
    - $p > 0.8$ means that the model generates 'too extreme' observations, that is, the model is overly cautious with uncertainty

Example. Assessing the fit of a binomial model

Suppose you are given 100 independent and identically distributed data points that are assumed to come from a Binomial distribution with sample size 10 and probability of success θ. Let y_i be the i^{th} data point,

$y_i \sim Binomial(n = 10, \theta)$
$\theta \sim Beta(a = 1, b = 1)$

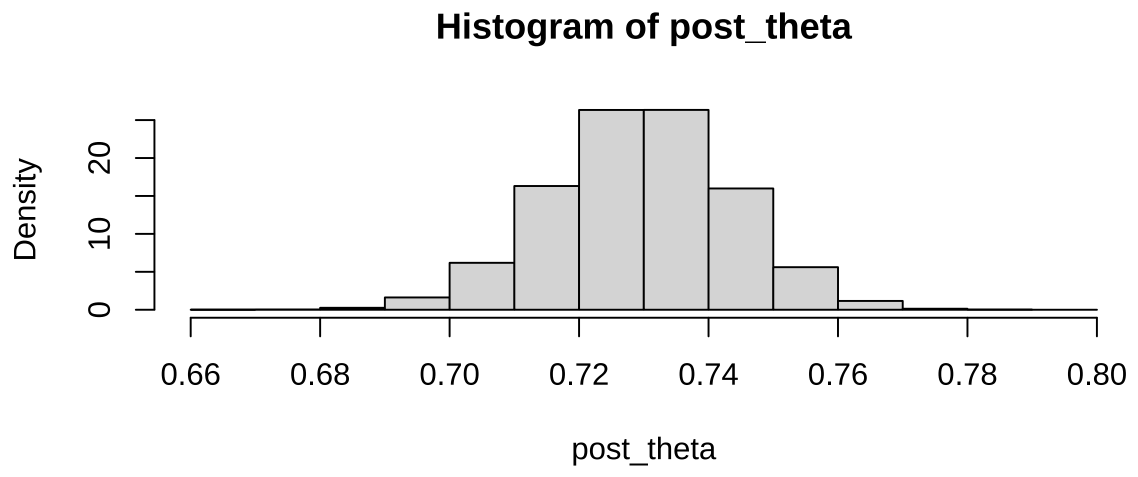
```
#Observed data
N_obs=100 # Number of data points
y <- rbinom(N_obs,size=10,prob=0.7)
```

Posterior distribution of θ

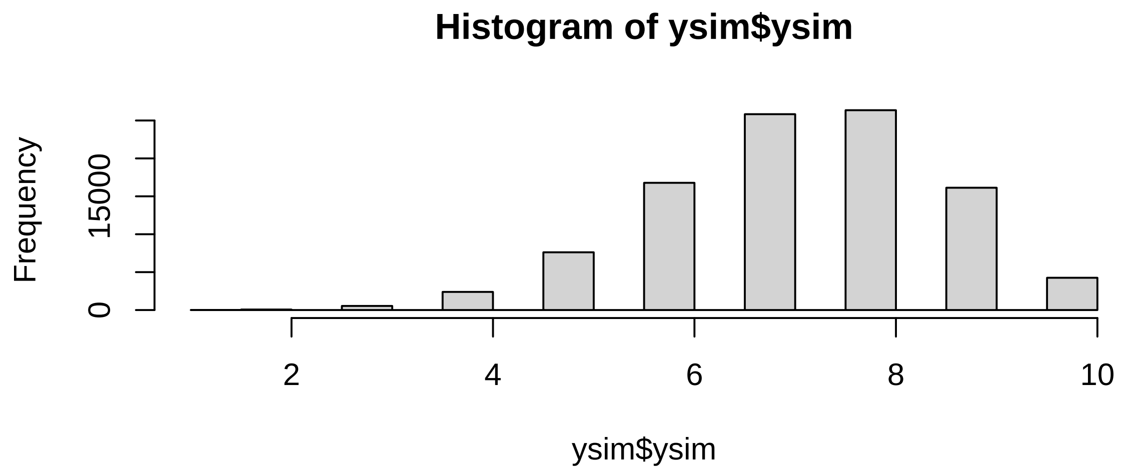
$\theta|y \sim Beta(a + \sum_{i=1}^{N_{obs}} y_i, b + N_{obs} * n - \sum_{i=1}^{N_{obs}} y_i)$

Where, N_{obs} is the number of observed data points, y_i is i^{th} data point, a and b are shape parameters of the beta prior on θ.

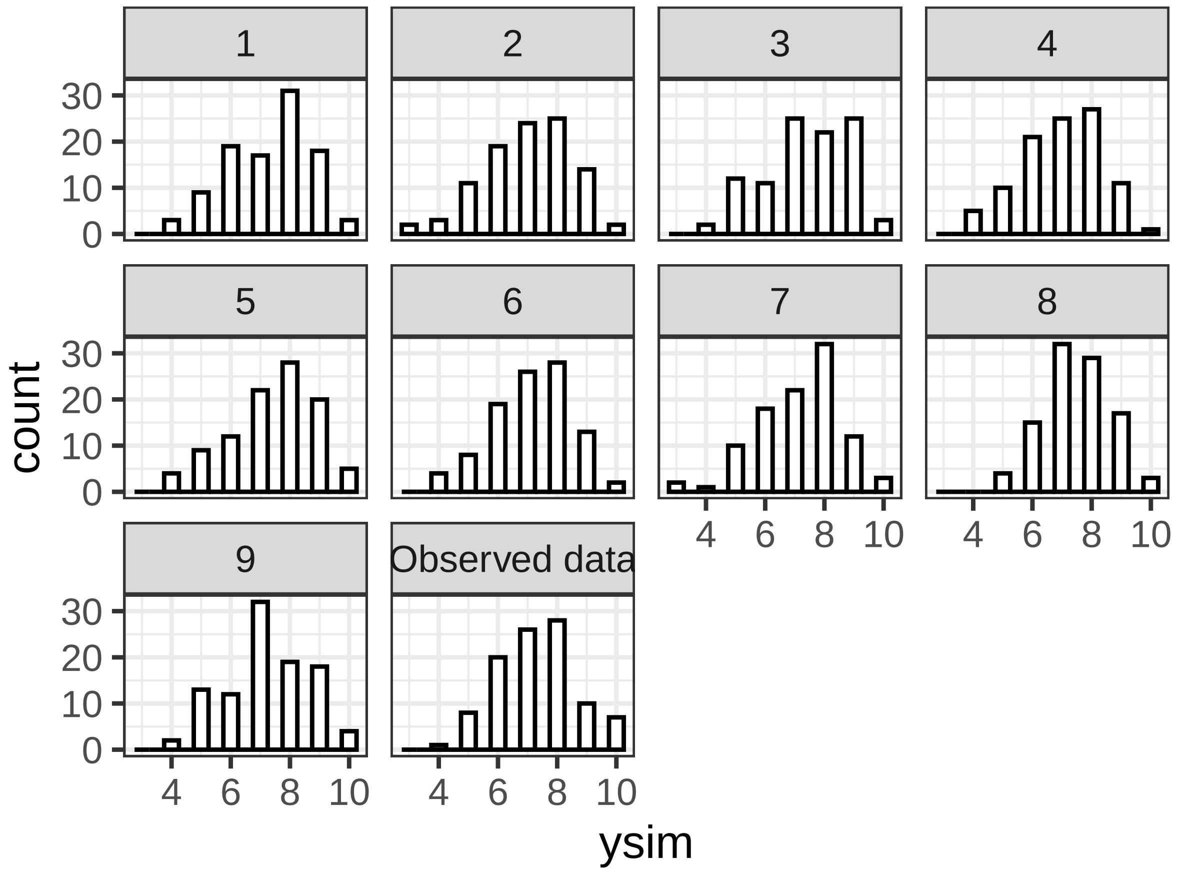
```
a <- 1
b <- 1
n <- 10
post_theta <- rbeta(100000,a+sum(y),b+(N_obs*n)-sum(y))
hist(post_theta,freq = FALSE)
```



```
# Collect samples from posterior predictive distribution
ysim <- data.frame(matrix(ncol = 2,nrow = 0))
colnames(ysim) <- c("iter","ysim")
for(i in 1:1000){
  sample_theta <- rbeta(1,1+sum(y),1+N_obs*n-sum(y))
  ypp <- rbinom(N_obs,10,sample_theta)
  ysim <- rbind(ysim,data.frame(iter=rep(i,N_obs),ysim=ypp))
}
hist(ysim$ysim)
```



```
ysim_plot <- rbind(subset(ysim,iter<10),
                   data.frame(iter=rep("Observed data",N_obs),ysim=y))

p <- ggplot(data=subset(ysim_plot),aes(x=ysim))
p <- p+geom_histogram(fill="white",color="black",binwidth=.5)
p+theme_bw()+facet_wrap(~iter)
```



```
# Choose test statistics to compare certain features
# of observed data with the simulated data samples
ysim_summary <- ysim %>%
  group_by(iter) %>%
  summarize(ybar=mean(ysim),yrange=range(ysim)[2]-range(ysim)[1])
# I have choosen mean and range as test statistics
# and calculated these statistics for each simulated data sample
head(ysim_summary)
```

```
## # A tibble: 6 x 3
##    iter  ybar yrange
##   <int> <dbl>  <int>
## 1     1  7.3       6
## 2     2  7.01      7
## 3     3  7.4       6
## 4     4  6.96      6
## 5     5  7.41      6
## 6     6  7.13      6
```

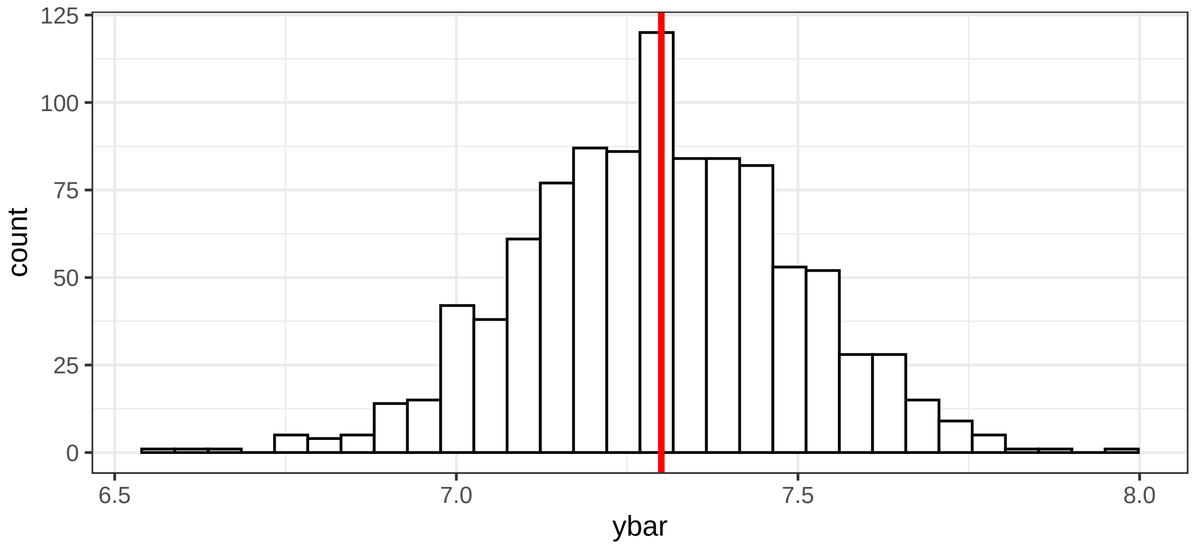
```
# Calculate Bayesian p-value
# p = Pr(T(ysim)>T(y)|y)
# When the test statistic is "mean"
exceeding_bar <- which(ysim_summary$ybar>mean(y))
p_bar <- length(exceeding_bar)/length(ysim_summary$iter)
p_bar
```

```
## [1] 0.464
```

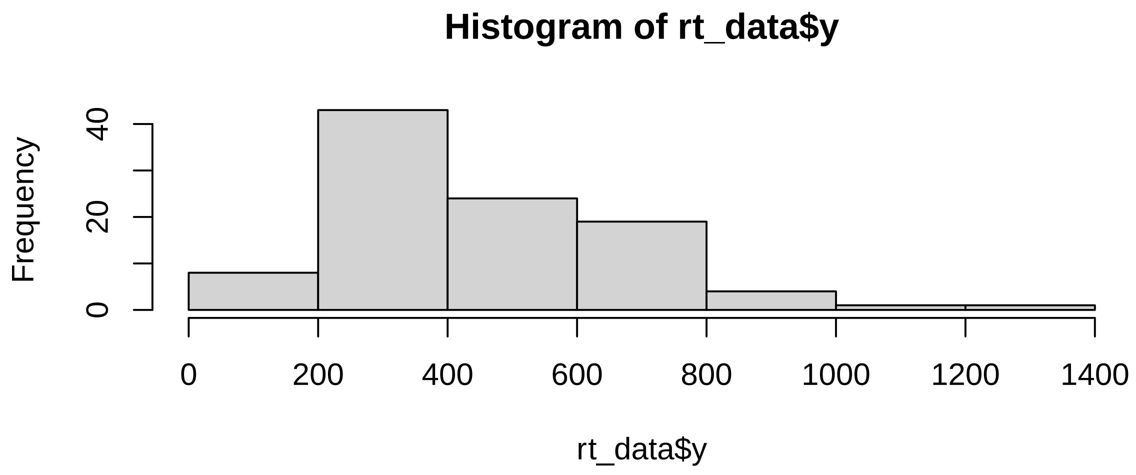
```
p <- ggplot(ysim_summary,aes(x=ybar))+geom_histogram(fill="white",color="black")
p+geom_vline(xintercept = mean(y),colour="red",size=1.2)+theme_bw()
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```



Example. Checking the fit of a normal model for reading time data
You are given reading time data consisting of 100 observations,



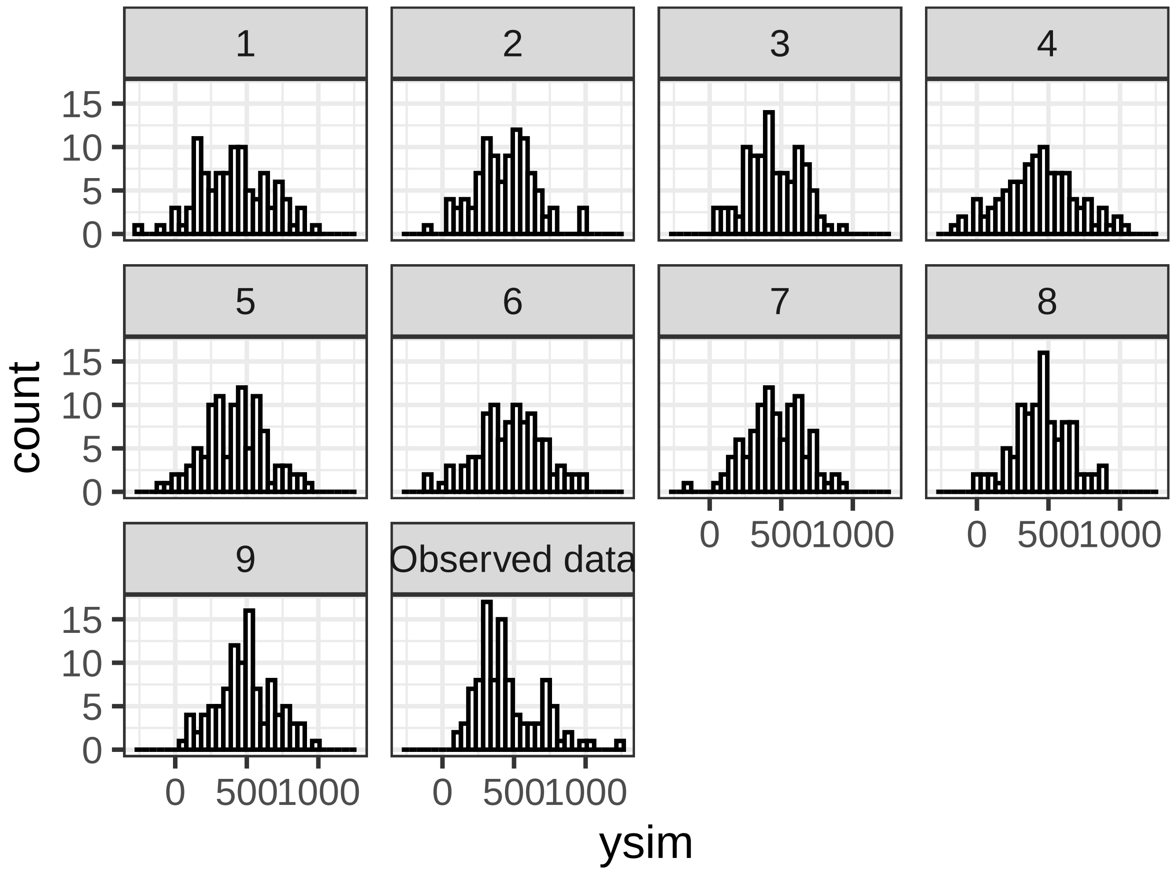
You assume that the data come from a normal distribution with mean μ and variance σ^2

```
#Fit the normal model
m1 <- brm(y~1,family = gaussian(),
          prior = c(prior(normal(200, 100), class = Intercept),
                    prior(normal(0, 1000), class = sigma)),
          data=rt_data,iter=500,cores = 4)
save(m1,file="FittedModels/ModelEvaluation_m1.Rda")
```

```
# Samples from the posterior distribution are
mu_samples <- posterior_samples(m1)$b_Intercept
sigma_samples <- posterior_samples(m1)$sigma

# Generate simulated data samples from the posterior predictive
ysim <- data.frame(matrix(ncol = 2,nrow = 0))
colnames(ysim) <- c("iter","ysim")
for(i in 1:length(mu_samples)){
  sample_mu <- mu_samples[i]
  sample_sigma <- sigma_samples[i]
  ypp <- rnorm(N_obs,sample_mu,sample_sigma)
  ysim <- rbind(ysim,data.frame(iter=rep(i,N_obs),ysim=ypp))
}
```

```
# Graphical posterior predictive checks
ysim_plot <- rbind(subset(ysim,iter<10),
                   data.frame(iter=rep("Observed data",N_obs),ysim=y))
p <- ggplot(data=subset(ysim_plot),aes(x=ysim))
p <- p+geom_histogram(fill="white",color="black")
p+theme_bw()+facet_wrap(~iter)
```

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```

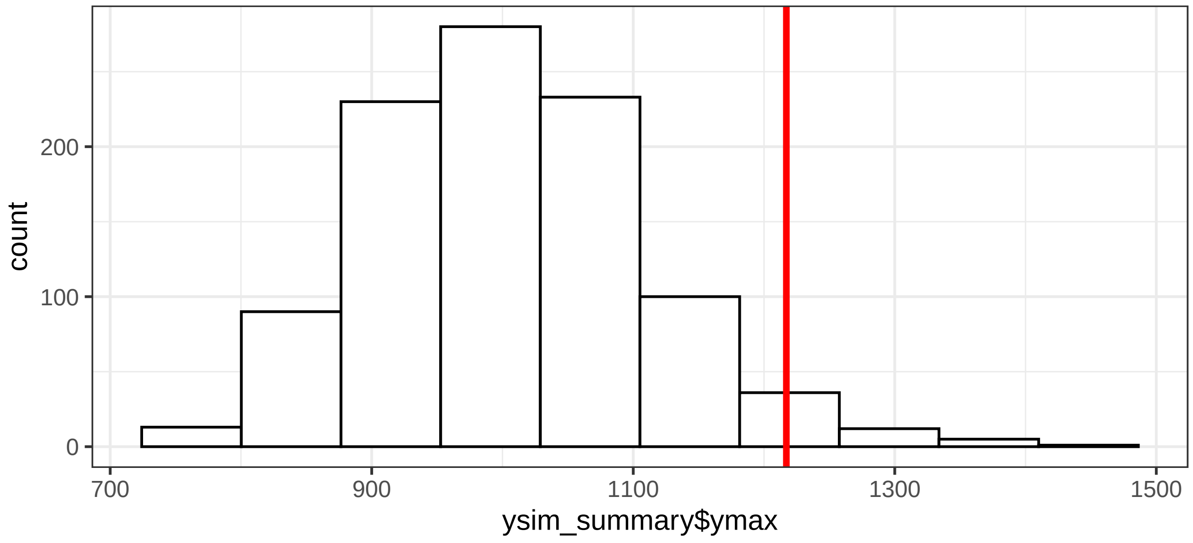


```
# Compute test statistics
ysim_summary <- subset(ysim) %>%
  group_by(iter) %>%
  summarize(ymin=min(ysim),
            ymax=max(ysim),
            ybar=mean(ysim))

exceeding_bar <- which(ysim_summary$ybar>mean(y))
p_bar <- length(exceeding_bar)/length(ysim_summary$iter)
p_bar

## [1] 0.367

exceeding_max <- which(ysim_summary$ymax>max(y))
p_max <- length(exceeding_max)/length(ysim_summary$iter)
p_max

## [1] 0.031

exceeding_min <- which(ysim_summary$ymin>min(y))
```

```
p_min <- length(exceeding_min)/length(ysim_summary$iter)
p_min
```

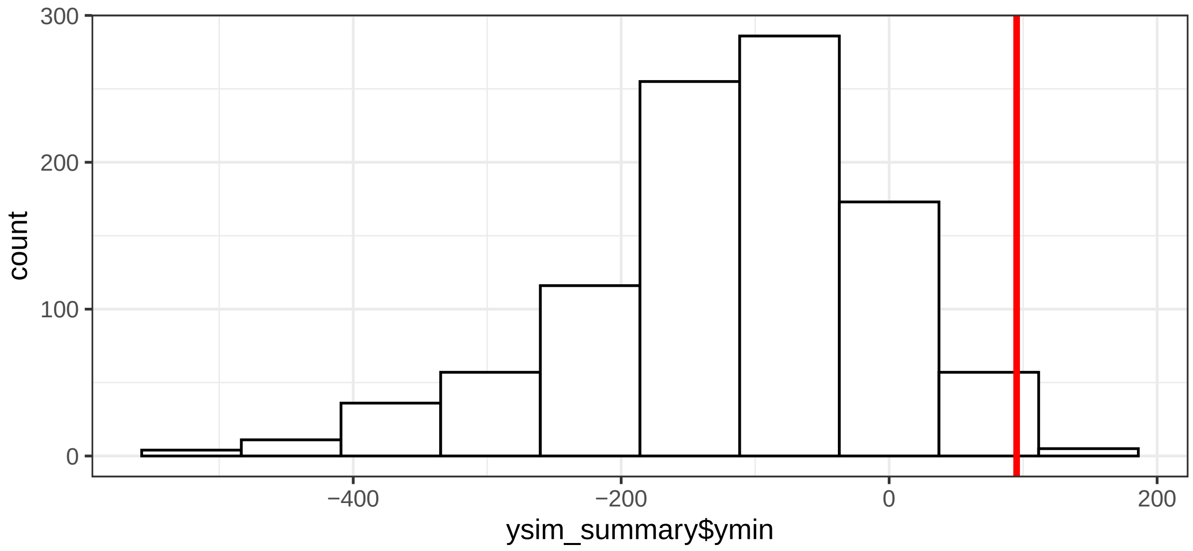
```
## [1] 0.007
```

```
p <- ggplot()+geom_histogram(aes(x=ysim_summary$ymax)
                      ,bins=10,fill="white",colour="black")
p+theme_bw()+geom_vline(xintercept = max(y),color="red",size=1.2)
```



```
# The observed data appear to be skewed towards the right
```

```
p <- ggplot()+geom_histogram(aes(x=ysim_summary$ymin)
                      ,bins=10,fill="white",colour="black")
p+theme_bw()+geom_vline(xintercept = min(y),color="red",size=1.2)
```

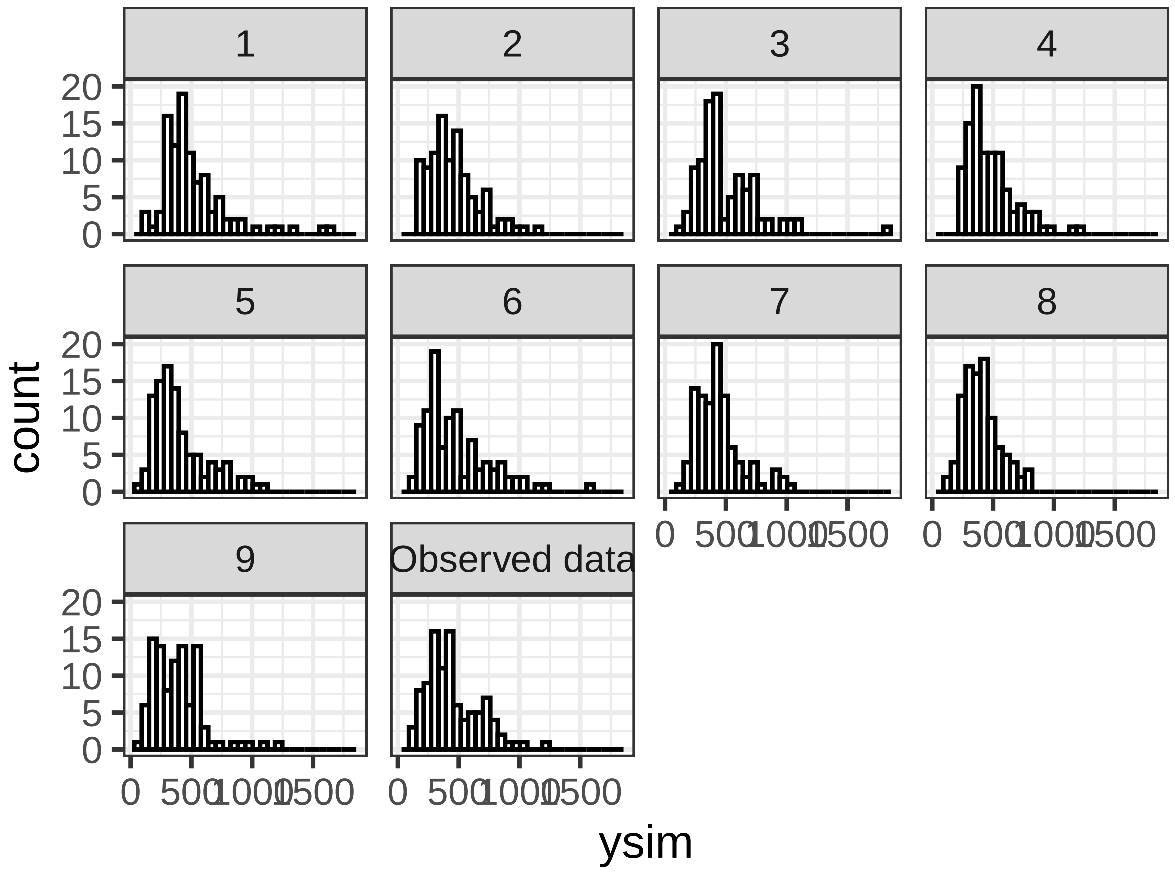


```
# The model generates negative reading times
```

You realize you are doing something wrong. The posterior predictive checks tell you that a normal model does not fit the observed reading times. Perheps reading times are not normally distributed.

You build another model. This time you assume that reading times come from a lognormal distribution with mean μ and variance σ^2.

```
#Fit the lognormal model
m2 <- brm(y~1,family = lognormal(),
          prior = c(prior(normal(6, 2), class = Intercept),
                    prior(normal(0, 2), class = sigma)),
          data=rt_data,iter=500,cores=4)
save(m2,file="FittedModels/ModelEvaluation_m2.Rda")

# Samples from the posterior distribution are
mu_samples <- posterior_samples(m2)$b_Intercept
sigma_samples <- posterior_samples(m2)$sigma

# Generate simulated data samples from the posterior predictive
ysim <- data.frame(matrix(ncol = 2,nrow = 0))
colnames(ysim) <- c("iter","ysim")
for(i in 1:length(mu_samples)){
  sample_mu <- mu_samples[i]
  sample_sigma <- sigma_samples[i]
  ypp <- rlnorm(N_obs,sample_mu,sample_sigma)
  ysim <- rbind(ysim,data.frame(iter=rep(i,N_obs),ysim=ypp))
}

# Graphical posterior predictive checks
ysim_plot <- rbind(subset(ysim,iter<10),
                   data.frame(iter=rep("Observed data",N_obs),ysim=y))
p <- ggplot(data=subset(ysim_plot),aes(x=ysim))
p <- p+geom_histogram(fill="white",color="black")
p+theme_bw()+facet_wrap(~iter)

## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```

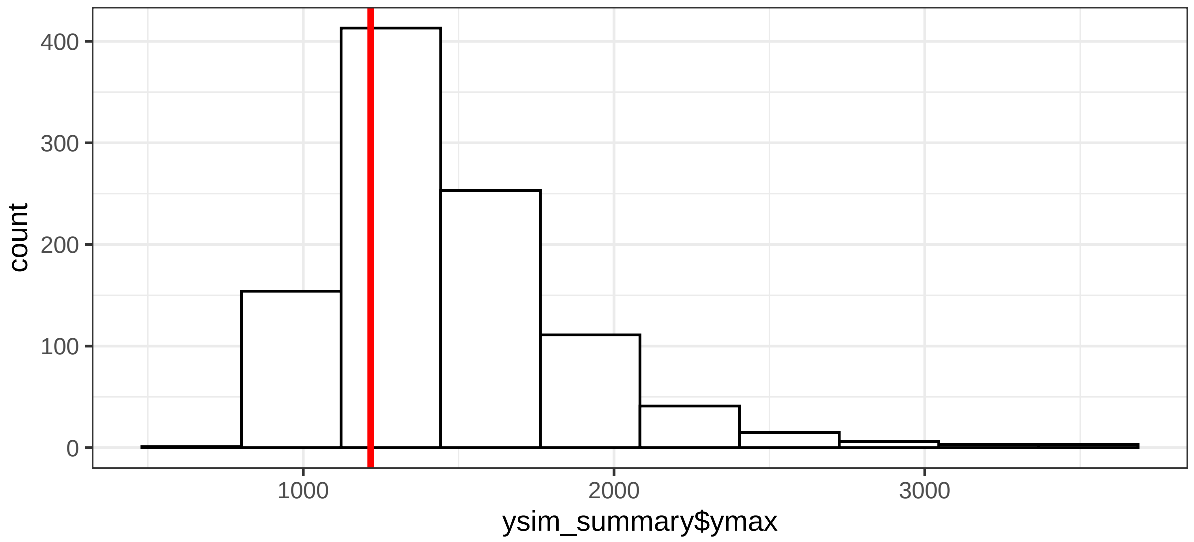


```
# Compute test statistics
ysim_summary <- subset(ysim) %>%
  group_by(iter) %>%
  summarize(ymin=min(ysim),
            ymax=max(ysim),
            ybar=mean(ysim))

exceeding_bar <- which(ysim_summary$ybar>mean(y))
p_bar <- length(exceeding_bar)/length(ysim_summary$iter)
p_bar

## [1] 0.569

exceeding_max <- which(ysim_summary$ymax>max(y))
p_max <- length(exceeding_max)/length(ysim_summary$iter)
p_max

## [1] 0.724

exceeding_min <- which(ysim_summary$ymin>min(y))
```

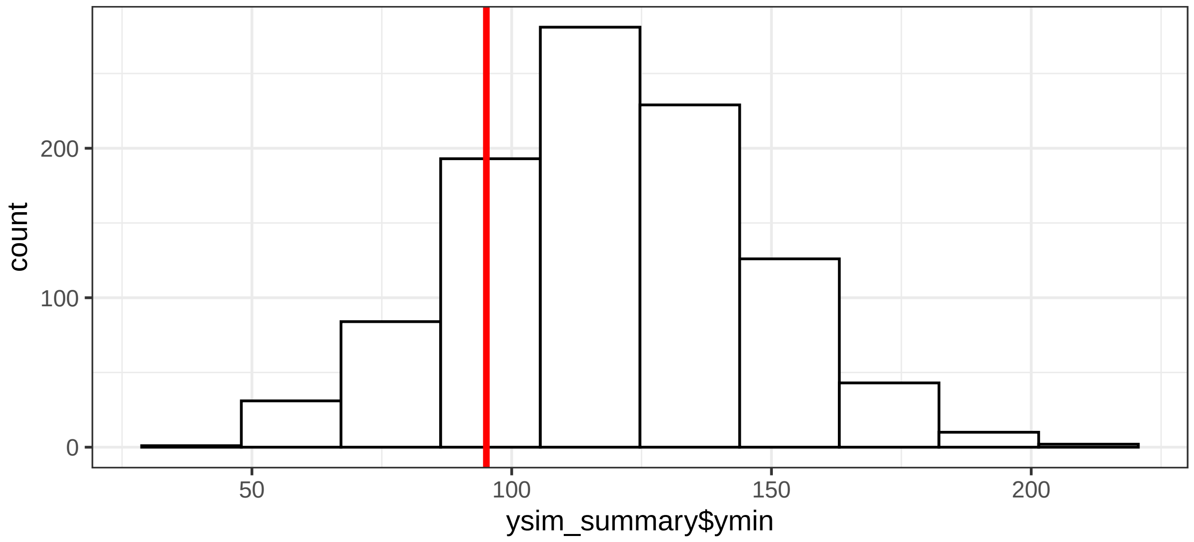
```
p_min <- length(exceeding_min)/length(ysim_summary$iter)
p_min
```

```
## [1] 0.805
```

```
p <- ggplot()+geom_histogram(aes(x=ysim_summary$ymax)
                      ,bins=10,fill="white",colour="black")
p+theme_bw()+geom_vline(xintercept = max(y),color="red",size=1.2)
```



```
p <- ggplot()+geom_histogram(aes(x=ysim_summary$ymin)
                      ,bins=10,fill="white",colour="black")
p+theme_bw()+geom_vline(xintercept = min(y),color="red",size=1.2)
```



1.1.1 What is wrong with the 'posterior predictive checks'?

1. Uses the same data twice; once to train the model and then to evaluate the model
   - Risk of overfitting
     - If you make your model more and more complex to fit the current data, the model would learn irregular features of data
     - There are both regular and irregular features in a data sample. If the model learns irregular features, it would fail to predict future data
2. Does not really quantify the performance of a model
   - Cannot be used for model comparison
   - Cannot be used for hypothesis testing

Posterior predictive checking is a 'useful' step in model building. It can answer a crucial question: Does the model ignore some key properties of the data? Does the model generate "valid" predictions about reality?

1.2 *Information-theoretic measures of predictive accuracy*

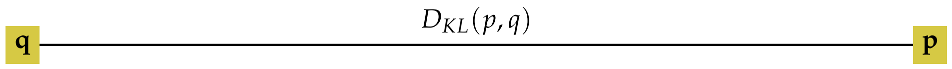
We are interested in the **predictive accuracy of a model**: with what accuracy does the model predict unseen data?

How should we measure the predictive accuracy of a model?

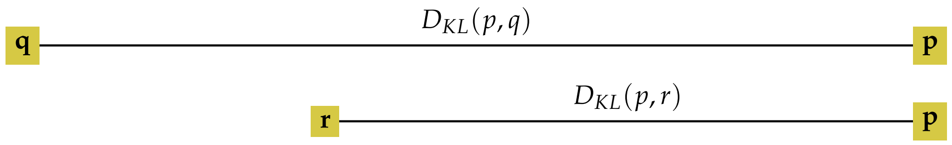
A perspective: predictive accuracy is *reduction in uncertainty*

- Predictive accuracy is related to "how much uncertainty is reduced by learning an outcome?"
  - If the reduction in uncertainty is larger, the model has poor predictive accuracy
- How to measure uncertainty?

  Consider, p_i represents the probability of occurrence of an event i in the sample space S; p represents the distribution of probabilities over n events in the sample space
  - A measure of uncertainty should be continuous, monotonically decreasing, non-negative, additive and it should increase with an increase in the number of events
  - **Information entroy:** $H(p) = -\sum_{i=1}^{n} p_i \log p_i$
  - In simple words, uncertainty is 'negative average log probability of an event'
  - Using this, we can measure uncertainty contained in a distribution
  - How to measure 'reduction in uncertainty'?
  - **Kullback-Leibler divergence (K-L divergence):**
    * Uncertainty induced by using probabilities from one distribution, say q, to predict another distribution, p
    * $D_{KL}(p, q) = \sum p_i \log p_i - \sum p_i \log q_i$
    * K-L divergence will tell us the distance between the model, q and 'true' distribution, p



- But we do not have access to the true distribution, p. No!



- However, we still can measure relative distance for two models say, q and r, as: $D_{KL}(p,q) - D_{KL}(p,r) = (\sum p_i \log p_i - \sum p_i \log q_i) - (\sum p_i \log p_i - \sum p_i \log r_i) = \sum p_i \log r_i - \sum p_i \log q_i$
- If p, q, and r are continuous variables:
  $D_{KL}(p,q) - D_{KL}(p,r) = \int p(y) \log r(y|\theta) dy - \int p(y) \log q(y|\phi) dy$
- If the models q and r were trained on data y^{obs} and we estimate their predictive accuracy on new data y^{new}:
  $D_{KL}(p,q) = \int p(y^{new}) \log p(y^{new}) dy^{new} - \int p(y^{new}) \log q(y^{new}|y^{obs}) dy^{new}$
  $D_{KL}(p,r) = \int p(y^{new}) \log p(y^{new}) dy^{new} - \int p(y^{new}) \log r(y^{new}|y^{obs}) dy^{new}$
  where $D_{KL}(p,q)$ and $D_{KL}(p,r)$ represent the distance between models q and r and the true density p. In the above equations, we do not have any control over the term $\int p(y^{new}) \log p(y^{new}) dy^{new}$ and it remains constant for both the models. To maximize the predictive accuracy of the model, the KL divergence should be minimized that means the term $-\int p(y^{new}) \log q(y^{new}|y^{obs}) dy^{new}$ should be minimized, or $\int p(y^{new}) \log q(y^{new}|y^{obs}) dy^{new}$ should be maximized. A model that has larger expected log predictive density of a new data $q(y^{new}|y^{obs})$ given a true distribution of the data $p(y^{new})$ will have higher predictive accuracy.
  Assuming y_{new} comes from true distribution $p(y)$, we are only concerned about the term $\sum \log q(y^{new}|y^{obs})$ where
  $\log q(y^{new}|y^{obs}) = \log \int q(y^{new}|\theta) p(\theta|y^{obs}) d\theta$
  The term $\sum \log q(y^{new}|y^{obs})$ is called the expected log pointwise predictive density ($\widehat{elppd}$) of the model.
- Above derivations show that we can use average log probabilities to approximate the relative predictive accuracy of a model
- In sum, **'reduction in uncertainty' can be measured using expected log predictive density of a model, i.e.,** $\sum \log q(y^{new}|y^{obs})$
  $\log q(y^{new}|y^{obs}) = \log \int q(y^{new}|\theta) p(\theta|y^{obs}) d\theta$
- We can also get a measure of relative model fit, called "Deviance"
- Deviance, $D(q) = -2 \sum_i log(q_i)$

- Different measures, such as AIC, WAIC, LOO, etc., use log predictive density in different ways to compute the predictive accuracy of a model
  - Risk of overfitting is still there!
  - The model is trained and evaluated on same data

- **Solution 1:** Penalize a more complex model. This would give a good approximation of the predictive accuracy of the model on future data.
  - $elpd_{AIC} = \log p(y|\hat{\theta}) - k$, where $\hat{\theta}$ maximum likelihood point estimate of parameter θ, K is number of parameters in the model
  - $elpd_{DIC} = \log p(y|\hat{\theta}) - k_{DIC}$ where $\hat{\theta}$ posterior mean estmate of parameter θ, k_{DIC} is effective number of parameters in the model. $k_{DIC} = 2var_{posterior} \log(p(y|\theta))$
  - $elppd_{WAIC} = \sum_{i=1}^{N} \log(E_{posterior} p(y_i|\theta)) - k_{WAIC}$, where $k_{WAIC} = \sum_{i=1}^{N} var_{posterior} \log(p(y_i|\theta))$
  1. Compute log predictive density (lpd) for each data point y_i, i.e., $lpd_i = \log \frac{1}{K} \sum_{j=1}^{K} p(y_i|\theta_j^*)$, where $\theta_j^* \sim p(\theta|y)$
  2. Compute log pointwise predictive density as sum of lpd across observations, $lppd = \sum_{i=1}^{N} lpd_i$
  3. Compute expected log pointwise predictive density by subtracting penalty term from the lppd, $elppd = lppd - k_{WAIC}$

Example. Predictive accuracies of two Binomial models having different Beta priors

Suppose you are given 10 data points that are assumed come from Binomial distribution with sample size 20 and probability of success θ

Let y_i be i^{th} data point.

Model 1:

$y_i \sim Binomial(20, \theta)$

$\theta \sim Beta(1, 1)$

Model 2:

$y_i \sim Binomial(20, \theta)$

$\theta \sim Beta(200, 600)$

```
#Observed data
y <- rbinom(10,20,0.7)
N_obs<-10
y
```

```
##  [1] 11 18 11 13 14 12 11 15 14 11
```

```
# Model1: Binom(20,theta), theta ~ beta(1,1)
# Model2: Binom(20,theta), theta ~ beta(200,600)

# Compute log pointwise predictive density (lppd) for the models
lppd_m1 <- 0
for(i in 1:N_obs){
  sample_theta <- rbeta(1000,1+sum(y),1+N_obs*20-sum(y))
  lpd_i <- log(mean(dbinom(y[i],20,sample_theta)))
  lppd_m1 <- lppd_m1 + lpd_i
}

lppd_m2 <- 0
for(i in 1:N_obs){
  sample_theta <- rbeta(1000,200+sum(y),600+N_obs*20-sum(y))
  lpd_i <- log(mean(dbinom(y[i],20,sample_theta)))
  lppd_m2 <- lppd_m2 + lpd_i
}

# Compute deviance, -2*lppd
# Less deviance means better predictive accuracy
-2*lppd_m1 # less deviance -- better predictive accuracy
```

```
## [1] 44.70003
```

```
-2*lppd_m2 # more deviance
```

```
## [1] 128.3949
```

- Comparison of log pointwise predictive density shows that Model 1 has better predictive accuracy than Model 2 for given data

1.2.1 Problems

- All these measure compute *in-sample deviance* of the model. We used the same data to fit the model and compute the predictive accuracy. This is overfitting!
- We do not know which model performs better in predicting new data
- Our goal is to build a model that performs better in predicting unseen data

```
ynew <- rbinom(5,20,0.4)
ynew
```

```
## [1]  8  8 10  9 11
```

```
lppd_m1 <- 0
for(i in 1:5){
  sample_theta <- rbeta(1000,1+sum(y),1+N_obs*20-sum(y))
  lpd_i <- log(mean(dbinom(ynew[i],20,sample_theta)))
  lppd_m1 <- lppd_m1 + lpd_i
}

lppd_m2 <- 0
for(i in 1:5){
  sample_theta <- rbeta(1000,200+sum(y),600+N_obs*20-sum(y))
  lpd_i <- log(mean(dbinom(ynew[i],20,sample_theta)))
  lppd_m2 <- lppd_m2 + lpd_i
}

-2*lppd_m1 # more deviance
```

```
## [1] 32.56102
```

```r
-2*lppd_m2 # less deviance -- better predictive accuracy
```

```
## [1] 25.82328
```

What if we do not have new data?

- Can we compare two models without overfitting?

2 Model evaluation using out-of-sample predictive density measures

2.1 Cross-validation

Approach: Split the data into training and test sets; fit the model on training data and evaluate the model on test data.

Versions: Leave-one-out cross validation, k-fold cross validation, stratified k-fold cross validation, cross-study-validation (Yadav et al., 2022)

Steps for stratified k-fold cross validation:

1. Divide the observed data, y into k representative subsets, $y_1, y_2, ..., y_k$
2. Create k sets of training and test data by leaving out 1 subset as test data

   We would end up with $(y_{train_1}, y_{test_1}), (y_{train_2}, y_{test_2}), \cdots, (y_{train_k}, y_{test_k})$
3. In each iteration i, use the training data y_{train_i} to fit the model and the corresponding test data y_{test_i} to evaluate the predictive accuracy of the model in terms of log pointwise predictive density, lpd_i

   (a) Sample a set of parameter value(s) from the posterior distribution that was estimated on training data, $\theta_j \sim \hat{p}(\theta|y_{train_i})$

   (b) Compute the probability density (i.e., likelihood) of generating the test data, y_{test_i} conditional on j^{th} set of parameter values from the posterior, such that $p(y_{test_i}|\theta_j)$.

   (c) Compute the log pointwise predictive density of the model in the i^{th} iteration as the log of likelihoods averaged over the posterior distribution estimated from the training data

   $$lpd_i = \log \frac{1}{N} \sum_{j=1}^{N} p(y_{test_i}|\theta_j) \text{ where } \theta_j \sim \hat{p}(\theta|y_{train_i})$$

   (d) A model's overall predictive accuracy, the expected log predictive density ($\widehat{elpd}$), is computed as the sum of its log predictive densities over k iterations

   $$\widehat{elpd} = \sum_{i=1}^{k} lpd_i$$

   - You can further calculate the standard error of $\widehat{elpd}$ using $SE_{\widehat{elpd}} = \sqrt{nV_{i=1}^{k}(lpd_i)}$ where $V_{i=1}^{k}(lpd_i)$ represents the variance of k log predictive density values

Implementation of k-fold cross validation

Suppose the test data y_{test_i} constains Ni datapoints indexed by ni. The log predictive density for datapoint $y_{test_{ni}}$ would be

$$lpd_{ni} = \log \frac{1}{N} \sum_{j=1}^{N} p(y_{test_{ni}}|\theta_j) \text{ where } \theta_j \sim \hat{p}(\theta|y_{train_i})$$

We can now compute log pointwise predictive density for the test data y_{test_i} using

$$lppd_i = \sum_{ni=1}^{Ni} lpd_{ni}$$

And, overall predictive accuracy of the model as,

$$\widehat{elpd} = \sum_{i=1}^{K} lppd_i$$

```
# k-fold cross validaton (using pointwise predictive density)

dat <- data.frame(y)
dat$obs <- 1:nrow(dat)
#Create k folds of test and training data
k <- 5
train <- list()
test <- list()
indices <- 1:nrow(dat)
for(i in 1:k){
  test_indices <- sample(indices,size=2,replace = FALSE)
  test[[i]] <- dat[test_indices,]
  train[[i]] <- dat[-test_indices,]
  indices <- setdiff(indices,test_indices)
}

# Fit models on training data and test their predictive accuracy on training data
lpd_m1 <- rep(NA,k)
for(i in 1:k){
  ytrain <- train[[i]]$y
  ytest <- test[[i]]$y
  sample_theta <- rbeta(1000,1+sum(ytrain),1+(length(ytrain)*20)-sum(ytrain))
  lppd_i <- 0
  for(j in 1:length(ytest)){
    lpd_ij <- log(mean(dbinom(ytest[j],20,sample_theta)))
    lppd_i <- lppd_i + lpd_ij
  }
  lpd_m1[i] <- lppd_i
}
```

```
lpd_m2 <- rep(NA,k)
for(i in 1:k){
  ytrain <- train[[i]]$y
  ytest <- test[[i]]$y
  sample_theta <- rbeta(1000,200+sum(ytrain),600+(length(ytrain)*20)-sum(ytrain))
  lppd_i <- 0
  for(j in 1:length(ytest)){
    lpd_ij <- log(mean(dbinom(ytest[j],20,sample_theta)))
    lppd_i <- lppd_i + lpd_ij
  }
  lpd_m2[i] <- lppd_i
}
##

elpd_m1 <- sum(lpd_m1)
elpd_m1

## [1] -23.94281

elpd_m1_SE <- sqrt(k*var(lpd_m1))
elpd_m1_SE

## [1] 3.083959

elpd_m2 <- sum(lpd_m2)
elpd_m2

## [1] -68.36895

elpd_m2_SE <- sqrt(k*var(lpd_m2))
elpd_m2_SE

## [1] 14.23218

# Leave-one-out cross validaton (LOO-CV)
elpd_m1 <- 0
for(i in 1:N_obs){
  ytrain <- dat$y[-i]
  ytest <- dat$y[i]
  sample_theta <- rbeta(1000,1+sum(ytrain),1+(N_obs-1)*20-sum(ytrain))
  lpd_i <- log(mean(dbinom(ytest,20,sample_theta)))
  elpd_m1 <- lppd_m1 + lpd_i
}

elpd_m2 <- 0
```

```
for(i in 1:N_obs){
  ytrain <- dat$y[-i]
  ytest <- dat$y[i]
  sample_theta <- rbeta(1000,200+sum(ytrain),600+(N_obs-1)*20-sum(ytrain))
  lpd_i <- log(mean(dbinom(ytest,20,sample_theta)))
  elpd_m2 <- lppd_m2 + lpd_i
}
```

```
elpd_m1
```

```
## [1] -18.53096
```

```
elpd_m2
```

```
## [1] -16.74185
```

2.2 *Marginal likelihood and Bayes factor*

Suppose we consider two competing models M_1 and M_2. We are interested in quantifying the relative plausibility of model M_1 compared to M_2 given the prior model probability and the evidence from the data, y. This relative plausibility is given by the ratio of the posterior probabilities of the models, i.e., $\frac{P(M_1|y)}{P(M_2|y)}$.

According to Bayes' rule, the posterior probability of model M_1 is
$P(M_1|y) = \frac{P(y|M_1)P(M_1)}{P(y)}$, where $P(y) = P(y|M_1)P(M_1) + P(y|M_2)P(M_2)$
The ratio of posterior probabilities of model M_1 and M_2 will be
$\frac{P(M_1|y)}{P(M_2|y)} = \frac{P(y|M_1)}{P(y|M_2)} \frac{P(M_1)}{P(M_2)}$
$P(y|M_1)$ and $P(y|M_2)$ are marginal likelihoods of model M_1 and model M_2
$P(y|M_1) = \int P(y|\theta)P(\theta)d\theta$

- Marginal likelihood is basically the prior predictive density of the model. You can also compute log prior predictive density if we you want.
- The ratio of marginal likelihoods of two models, i.e., $\frac{P(y|M_1)}{P(y|M_2)}$ is referred to as **Bayes factor**. Bayes factor is often used for hypothesis testing.
- However, the marginal likelihood cannot be derived analytically for most of the models
  - We need algorithms to approximate the marginal likelihood
  - For example, Monte Carlo integrator, importance sampling, bridge-sampling etc.

Analytically derived Bayes factor

```
k <- 2
n <- 10
# Model 1: theta ~ Beta(1,1)
# Model 2: theta ~ Beta(20,60)
# Analytically
# For a Binomial(n,theta) model with theta ~ Beta(a,b)
# Marginal likelihood = n!/k!(n-k)! * (k+a-1)!*(n-k+b-1)!/(n+a+b-1)!

# ML for model 1
a=1
b=1
k=2
n=10
ML_m1 <- 1/(n+1)
#ML for model 2
a=20
b=60
k=2
n=10
ML_m2 <- (factorial(n)/(factorial(k)*factorial(n-k)))*(
  factorial(k+a-1)*factorial(n-k+b-1)/factorial(n+a+b-1))

ML_m2

## [1] 5.079397e-21

ML_m1

## [1] 0.09090909

Bayes_factor <- ML_m1/ML_m2
Bayes_factor

## [1] 1.789762e+19
```

Bayes factor computed using Markov chain Monte Carlo (Metropolis-Hastings)

```
# Computationally (Using Metropolis-Hastings algorithm)
# ML for model 1
a=1
b=1
k=2
n=10

theta_p <- c(0.4)  # initialization of markov chain
i <- 2
ML <- 0
while(i<50000){
  sample_theta <- rnorm(1,theta_p[i-1],.08)
  if(sample_theta>0&sample_theta<1){
    post_new <- dbinom(k,n,sample_theta)*dbeta(sample_theta,a,b)
    post_prev <- dbinom(k,n,theta_p[i-1])*dbeta(theta_p[i-1],a,b)
    proposal_density <- (post_new*dnorm(theta_p[i-1],sample_theta,.08))/
      (post_prev*dnorm(sample_theta,theta_p[i-1],.08))
    p_str <- min(proposal_density,1)
    if(p_str>runif(1,0,1)){
      theta_p[i] <- sample_theta
      lkl <- dbinom(k,n,sample_theta)*dbeta(sample_theta,a,b)/
        dnorm(sample_theta,theta_p[i-1],.08)
      ML <- ML+lkl
      i <- i+1
    }
  }
}
ML/50000
hist(theta_p)
```