Chapter 5: Process in Statistical Modelling

Noryanti Muhammad
Centre for Mathematical Sciences
College of Computing and Applied Sciences
Universiti Malaysia Pahang

Centre of Excellence (CoE) for Data Science & Artificial Intelligence
Research & Innovation Department
Universiti Malaysia Pahang

Expected Outcomes:

By the end of this chapter, students should be able:

- ✓ To understand the process in statistical modelling.
- ✓ To identify the appropriate model in statistical modelling.
- ✓ To determine the parameter estimation of the model including method of moments, MLE and Bayesian Analysis.
- ✓To understand and use the Bayesian analysis in the statistical modelling.

Content:

5.1 Model Identification

- Statistical modeling refers to the data science process of applying statistical analysis to datasets.
- A statistical model is a mathematical relationship between one or more random variables and other non-random variables.
- The application of statistical modeling to raw data helps data scientists approach data analysis in a strategic manner, providing intuitive visualizations that aid in identifying relationships between variables and making predictions.
- Common data sets for statistical analysis include Internet of Things (IoT) sensors, census data, public health data, social media data, imagery data, and other public sector data that benefit from real-world predictions.

5.1 Model Identification

Supervised learning techniques include regression models and classification models:

- **Regression model**: a type of predictive statistical model that analyzes the relationship between a dependent and an independent variable. Common regression models include logistic, polynomial, and linear regression models. Use cases include forecasting, time series modeling, and discovering the causal effect relationship between variables.

- **Classification model**: a type of machine learning in which an algorithm analyzes an existing, large and complex set of known data points as a means of understanding and then appropriately classifying the data; common models include models include decision trees, Naive Bayes, nearest neighbor, random forests, and neural networking models, which are typically used in Artificial Intelligence.

5.1 Model Identification

Unsupervised learning techniques include clustering algorithms and association rules:

- K-means clustering: aggregates a specified number of data points into a specific number of groupings based on certain similarities.
- Reinforcement learning: an area of deep learning that concerns models iterating over many attempts, rewarding moves that produce favorable outcomes and penalizing steps that produce undesired outcomes, therefore training the algorithm to learn the optimal process.

5.1 Model Identification

There are three main types of statistical models: parametric, nonparametric, and semiparametric:

- **Parametric:** a family of probability distributions that has a finite number of parameters.
- **Nonparametric:** models in which the number and nature of the parameters are flexible and not fixed in advance.
- **Semiparametric:** the parameter has both a finite-dimensional component (parametric) and an infinite-dimensional component (nonparametric).

5.1 Model Identification

Build Statistical Models

- The first step in building a statistical model is knowing how to choose a statistical model.
- Choosing the best statistical model is dependent upon several different variables.
- Is the purpose of the analysis to answer a very specific question, or solely to make predictions from a set of variables?
- How many explanatory and dependent variables are there?
- What is the shape of the relationships between dependent and explanatory variables?
- How many parameters will be included in the model?
- Once these questions are answered, the appropriate model can be selected.

5.1 Model Identification

Best practices

Once a statistical model is selected, it must be built. Best practices for how to make a statistical model include:

- Start with univariate descriptive and graphs. Visualizing the data helps with identifying errors, understanding the variables you're working with, how they look, how they are behaving and why.
- Build predictors in theoretically distinct sets first in order to observe how related variables work together, and then the outcome once the sets are combined.
- Next, run bivariate descriptive with graphs in order to visualize and understand how each potential predictor relates individually to every other predictor and to the outcome.
- Frequently record, compare and interpret results from models run with and without control variables.
- Eliminate non-significant interactions first; any variable involved in a significant interaction must be included in the model by itself.
- While identifying the many existing relationships between variables, and categorizing and testing every possible predictor, be sure not to lose sight of the research question.

Machine Learning vs Statistical Modeling

- **Machine learning** is a subfield of computer science and artificial intelligence that involves building systems that can learn from data rather than explicitly programmed instructions. Machine learning models seek out patterns hidden in data independent of all assumptions, therefore predictive power is typically very strong. Machine learning requires little human input and does well with large numbers of attributes and observations.

- **Statistical modeling** is a subfield of mathematics that seeks out relationships between variables in order to predict an outcome. Statistical models are based on coefficient estimation, are typically applied to smaller sets of data with fewer attributes and **require the human designer** to understand the relationships between variables before inputting.

5.2 Parameter estimation

5.2.1 Method of moments

5.2 Parameter estimation

5.2.2 Maximum Likelihood Estimation (MLE)

5.2 Parameter estimation

5.2.3 Bayesian Analysis

- The majority of experimental linguistic research has been analyzed using frequentist statistics - that is, we draw conclusions from our sample data based on the frequency or proportion of groups within the data, and then we attempt to extrapolate to the larger community based on this sample.
- In these cases, we are often comparing our data to a null hypothesis - is our data compatible with this “no difference” hypothesis?
- We obtain a p-value, which measures the (in)compatibility of our data with this hypothesis.
- These methods rely heavily on point values, such as means and medians or variance.

5.2 Parameter estimation

5.2.3 Bayesian Analysis

- Bayesian inference is an entirely different ballgame.
- Instead of relying on single points such as means or medians, it is a **probability-based system**.
- In this system there is a relationship between previously known information and your current dataset.
- The output of the analysis includes credible intervals - that is, based on previous information plus your current model, what is the most probable range of values for your variable of interest?
- Informally, Bayes' theorem is: **Posterior ∝ Prior × Likelihood** **OR** $P(\theta|data) \propto P(data|\theta) \times P(\theta)$

5.2 Parameter estimation

5.2.3 Bayesian Analysis

Prior- a probability distribution that represents what the model knows before seeing the data.

Posterior – a probability distribution that represents what the model knows after having seen the data.

Take a **prior probability distribution** use observations from the data to update a **posterior probability distribution**.

5.2 Parameter estimation

5.2.3 Bayesian Analysis

Bayesian analysis, **a method of statistical inference** (named for English mathematician Thomas Bayes) that allows one to combine prior information about a population parameter with evidence from information contained in a sample to guide the statistical inference process. The probability distribution which called posterior.

$$P(\theta|data) = \frac{P(data|\theta) \times P(\theta)}{\int P(data|\theta) \times P(\theta) d\theta}$$

OR

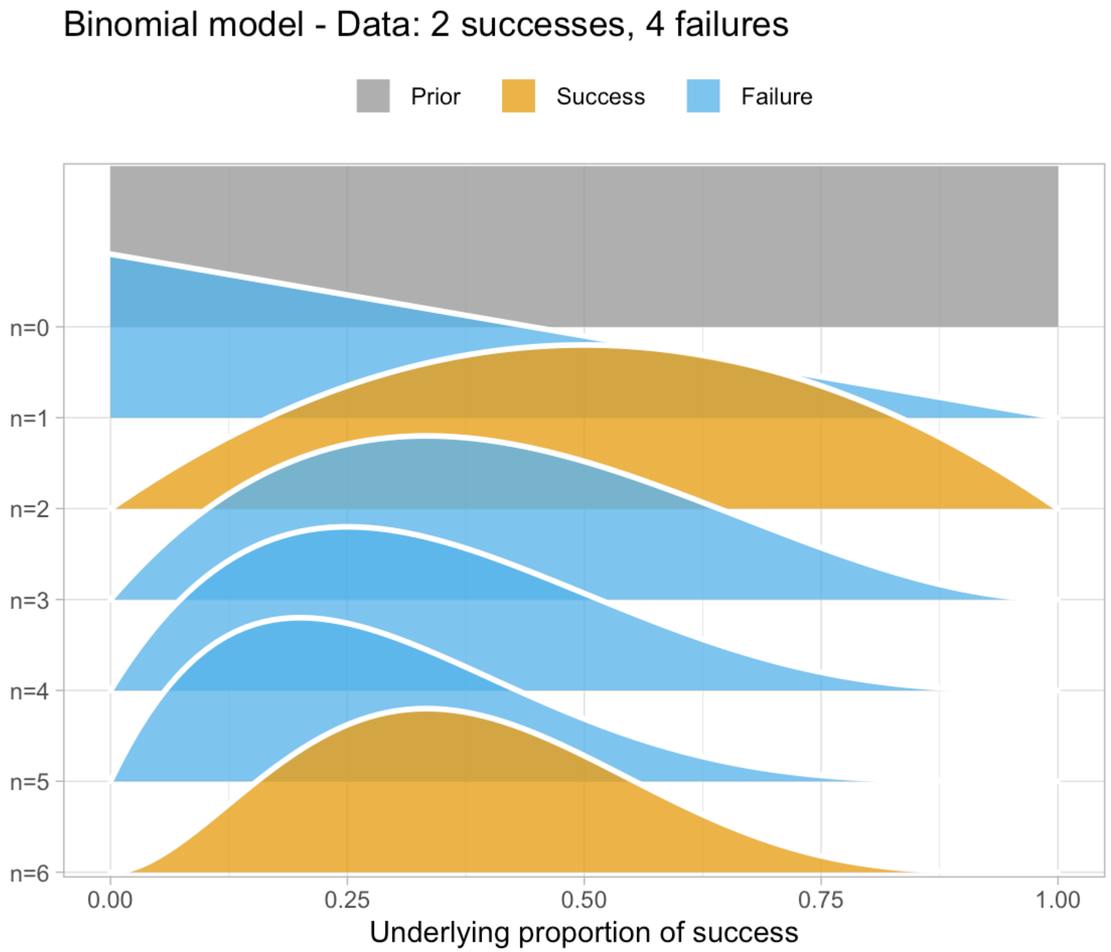
$$Posterior = \frac{Lik \times Prior}{AverageLik}$$

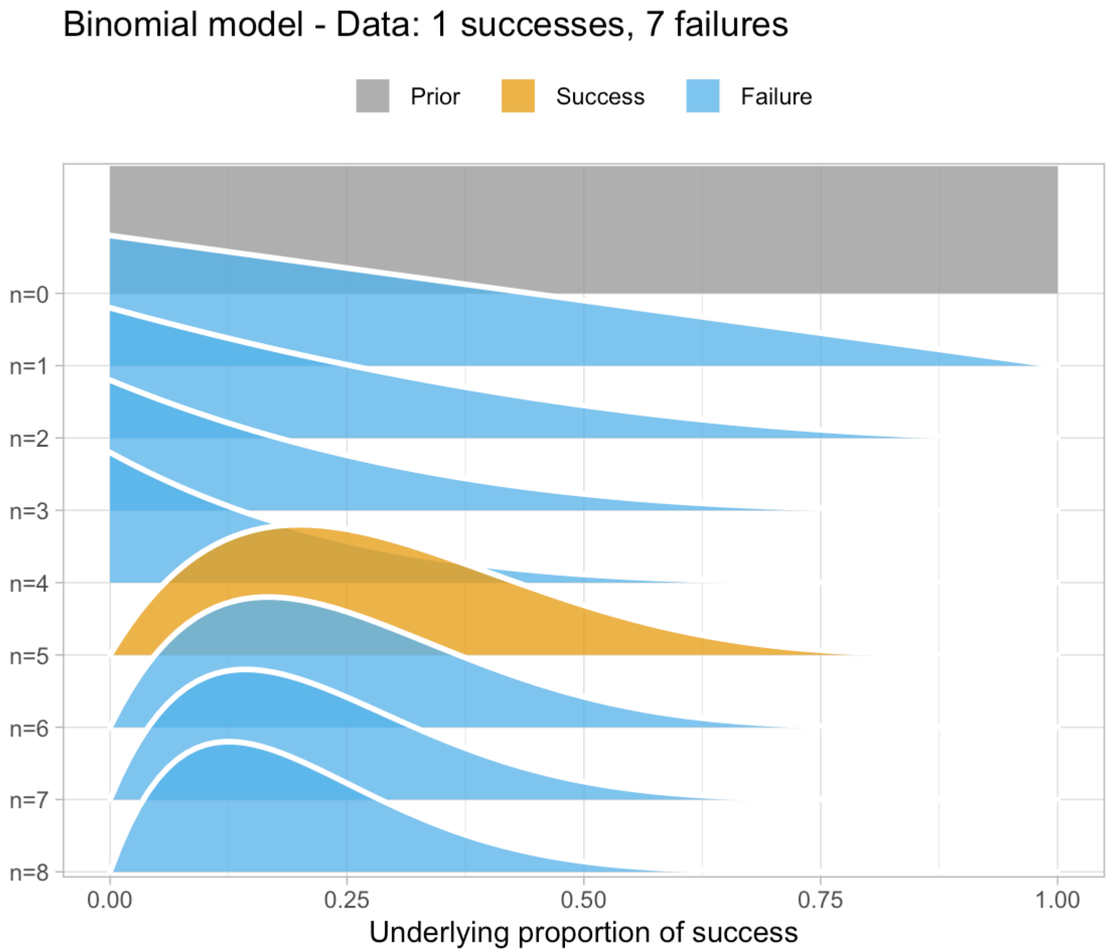
The posterior can be computed from three key ingredients:

- A likelihood distribution, P(data | \theta) ;
- A prior distribution, P(\theta) ;
- The ‘average likelihood’, \int P(data | \theta) \times P(\theta) d\theta = P(data).

5.2 Parameter estimation

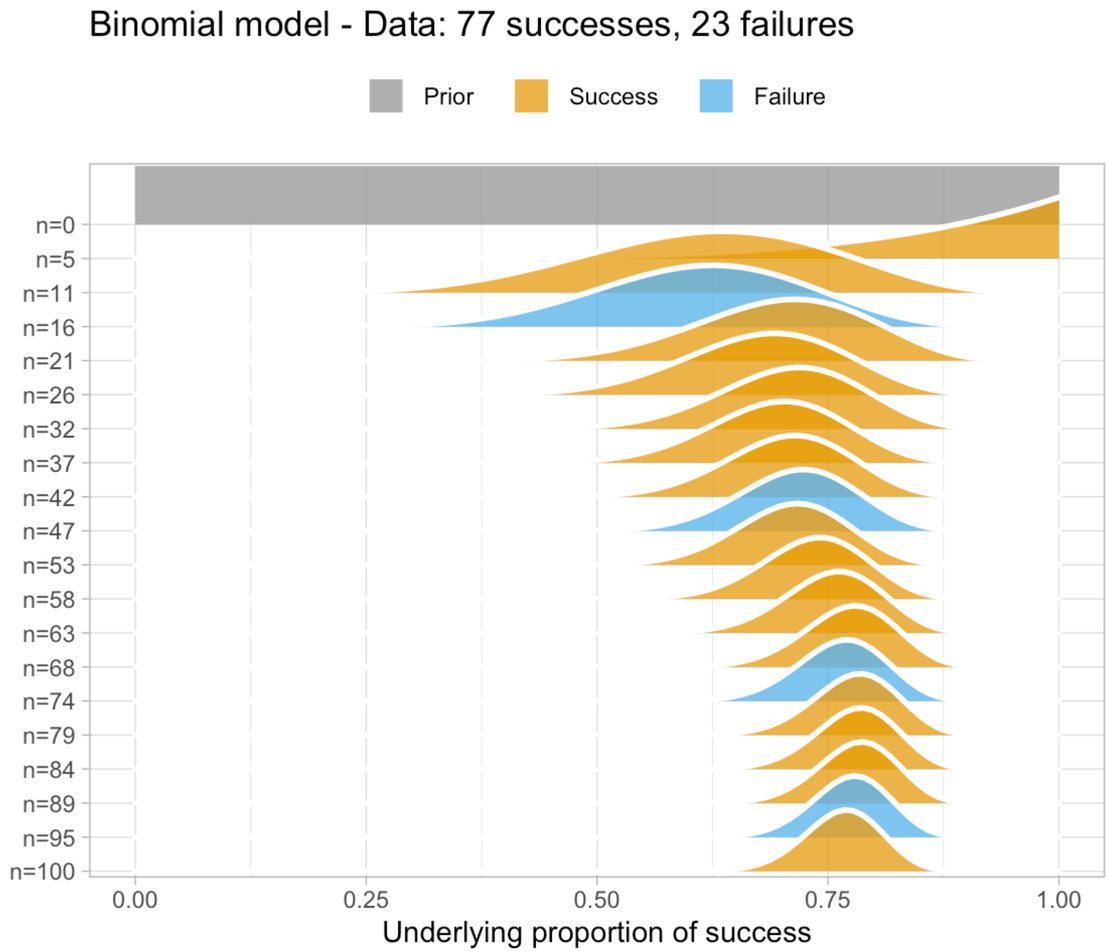
A simple Bayesian analysis

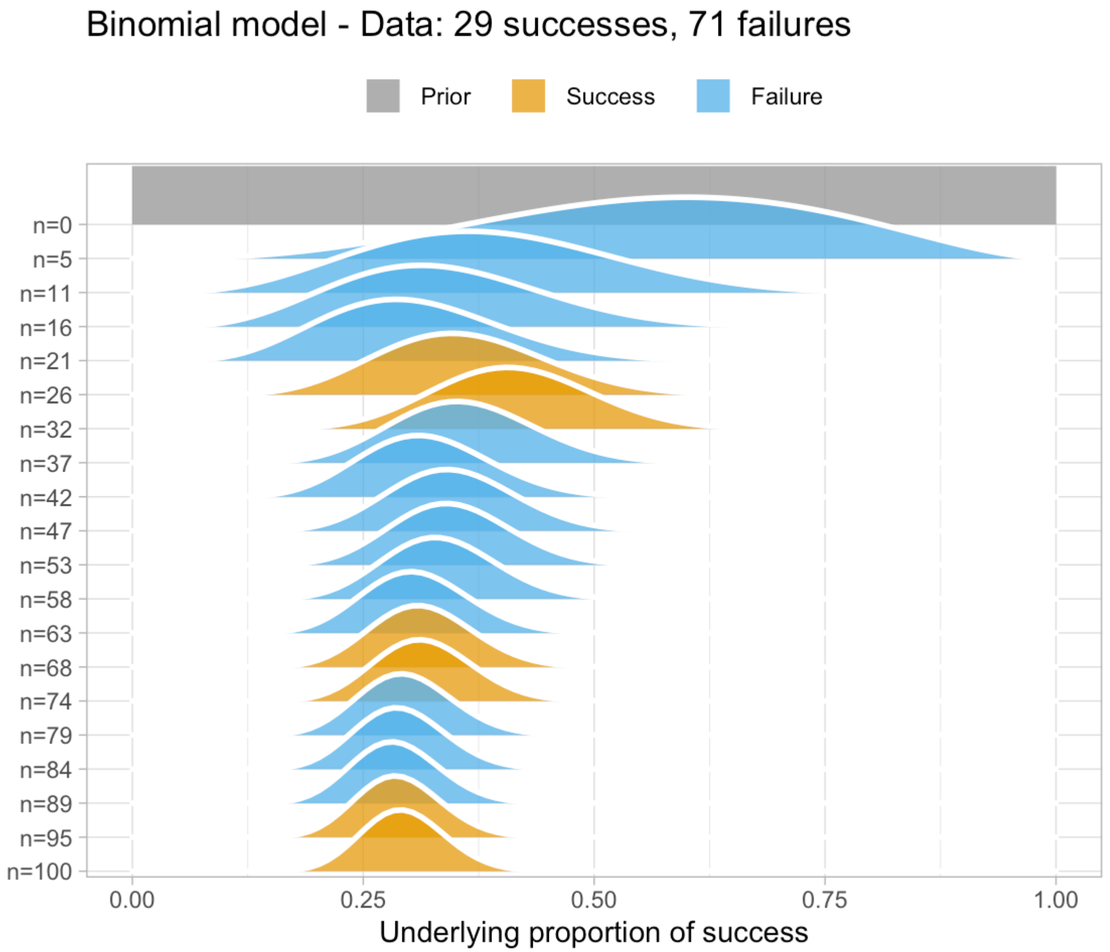




5.2 Parameter estimation

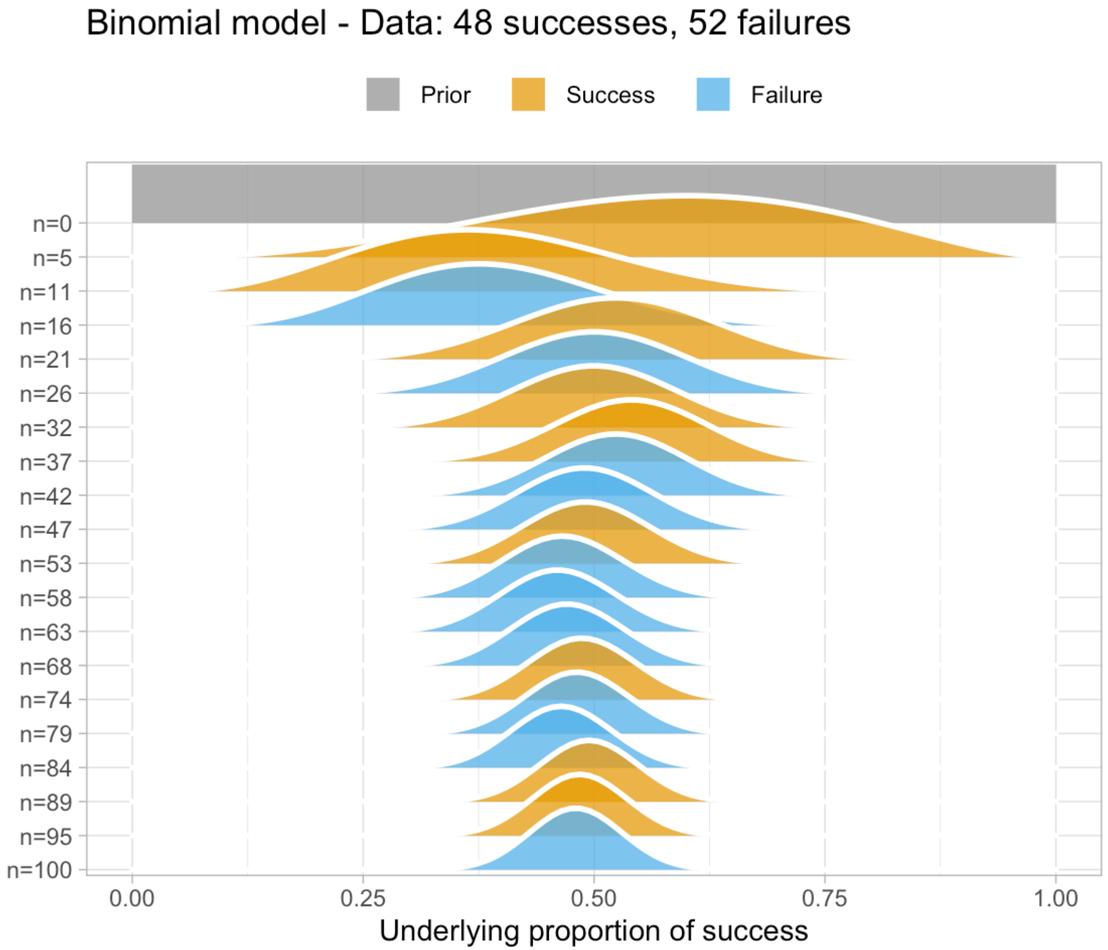
A simple Bayesian analysis





5.2 Parameter estimation

A simple Bayesian analysis



5.2 Parameter estimation

Why use Bayesian analysis?

There are many reasons to use Bayesian analysis instead of frequentist analytics. Bayesian analysis is flexible in that:

- You can include information sources in addition to the data. this includes background information given in textbooks or previous studies, common knowledge, etc.
- You can make any comparisons between groups or data sets.
- This is especially important for linguistic research.
- Bayesian methods allow us to directly the question we are interested in: How plausible is our hypothesis given the data?
- This allows us to quantify uncertainty about the data and avoid terms such as “prove”.
- Models are more easily defined and are more flexible, and not susceptible to things such as separation.

5.2 Parameter estimation

How to run a Bayesian analysis in R

- There are a bunch of different packages available for doing Bayesian analysis in R.
- These include RJAGS and rstanarm, among others.
- The development of the programming language Stan has made doing Bayesian analysis easier for social sciences.
- We will use the package brms, which is written to communicate with Stan, and allows us to use syntax analogous to the lme4 package.
- Note that some codding written for linguistic research use the rstan and rstanarm packages (such as Sorensen, Hohenstein and Vasishth, 2016 and Nicenbolm and Vasishth, 2016).
- A more recent codding (Vasishth et al., 2018) utilizes the brms package.

5.2 Parameter estimation

How to run a Bayesian analysis in R

Vasishth et al. (2018) identify five steps in carrying out an analysis in a Bayesian framework. They are:

- Explore the data using graphical tools; visualize the relationships between variables of interest.
- Define model(s) and priors.
- Fit model(s) to data.
- Check for convergence.
- Carry out inference by:
  - summarizing and displaying posterior distributions
  - computing Bayes factors with several different priors for theparameter being tested
  - evaluating predictive performance of competing models using k-fold cross-validation or approximations of leave-one-out cross-validation.

EXAMPLE

A Bayesian model for the proportion of success:

What is the proportion of successful treatments with a new drug?

- prop_model(data)
- data is a vector of failures represented by 1s and 0s
- There is an unknown underlying proportion of success
- If the data point is a success is only affected by the proportion of success
- Prior to seeing any data, any underlying proportion of success is equally likely
- The result is a probability distribution that represents what the model knows about the underlying proportion of success

EXAMPLE

A Bayesian model for the proportion of success:

What is the proportion of successful treatments with a new drug?

- The **output of prop_model** is a plot showing what the model learns about the underlying proportion of success from each data point in the order you entered them.
- At n=0 there is no data, and all the model knows is that it's equally probable that the proportion of success is anything from 0% to 100%.
- At n=4 all data has been added, and the model knows a little bit more.

EXAMPLE

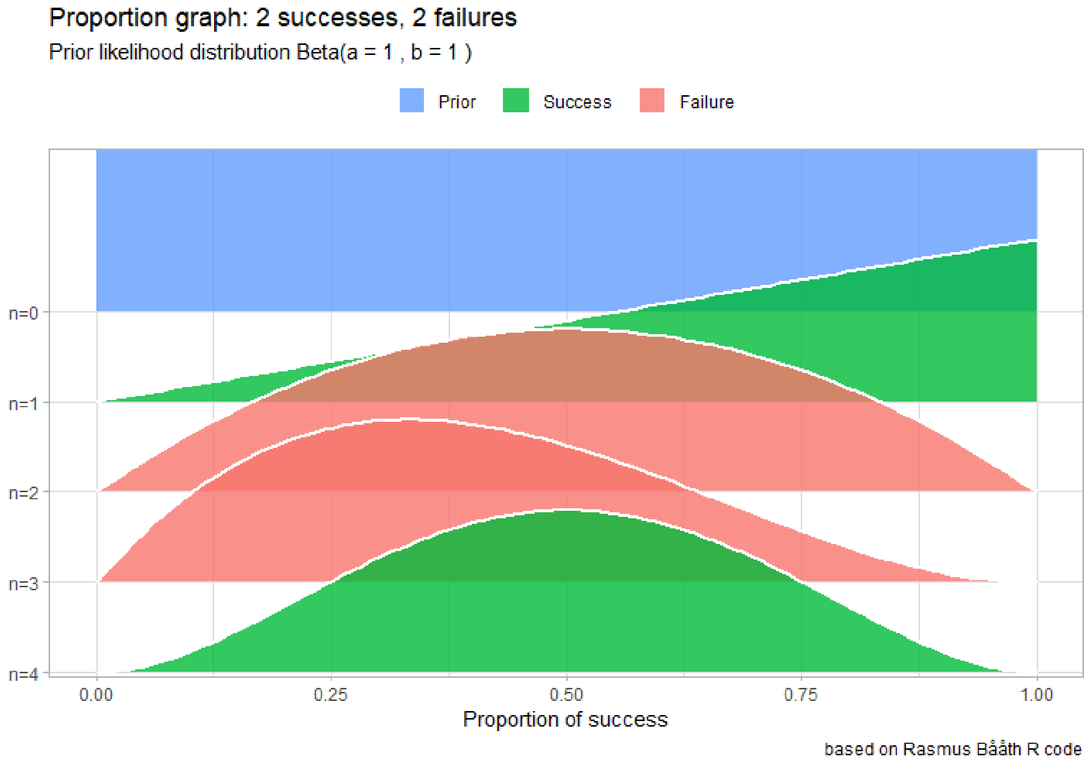
R Command:
Make sure the packages `tidyverse` and `ggridges` are installed, otherwise run:

- The function takes a number of successes and failures coded as a TRUE/FALSE or 0/1 vector. This should be given as the data argument.
- The result is a visualization of the how a Beta-Binomial model gradualy learns the underlying proportion of successes using this data.
- The function also returns a sample from the posterior distribution that can be further manipulated and inspected.
- The default prior is a Beta(1,1) distribution, but this can be set using the prior_prop argument.

EXAMPLE

There is a new drug to treat zombieism. It has never been tested before, but the results from this pilot test have two recoveries out of 4 subjects. Here, `prop_model` plots the successive posterior probabilities that the zombie antidote will be successful:

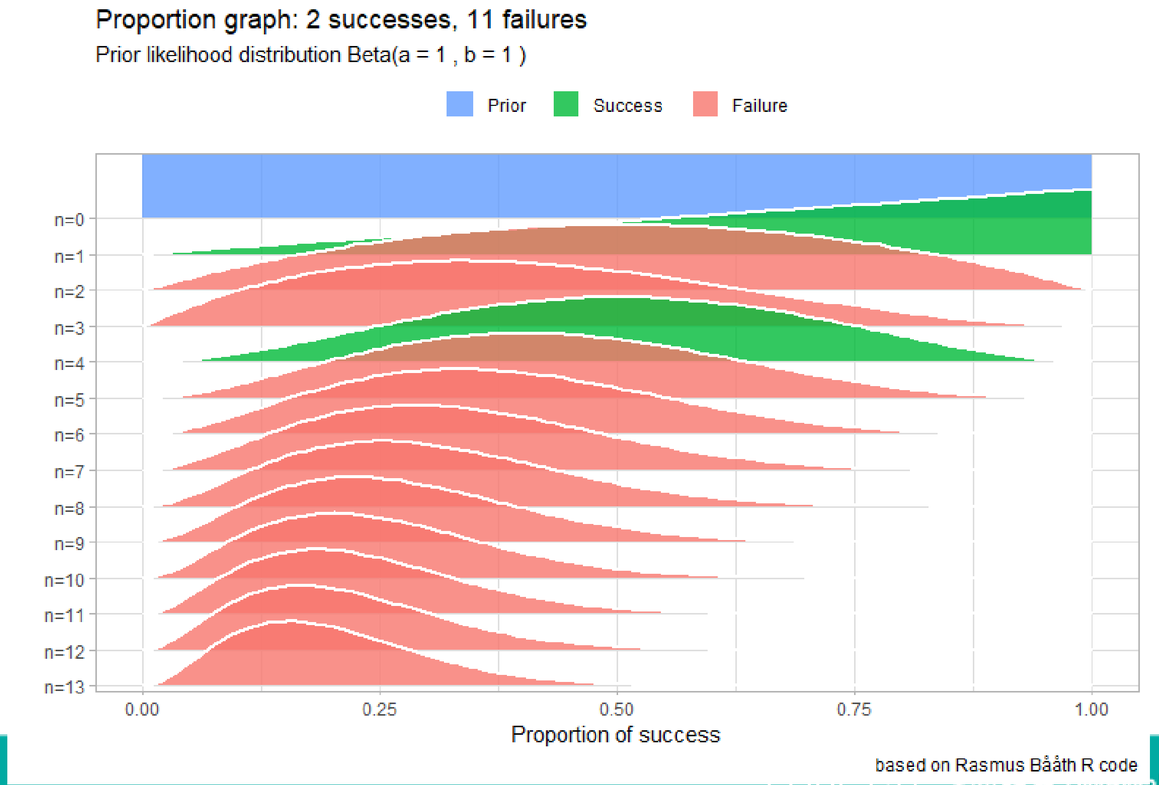
```
data <- c( 1, 0, 0, 1 )
prop_model( data )
```



EXAMPLE

Consider several more subjects, however, they are all failures, leaving just 2 successes out of 13 subjects. Here is how the posterior probability changes with the new case information:

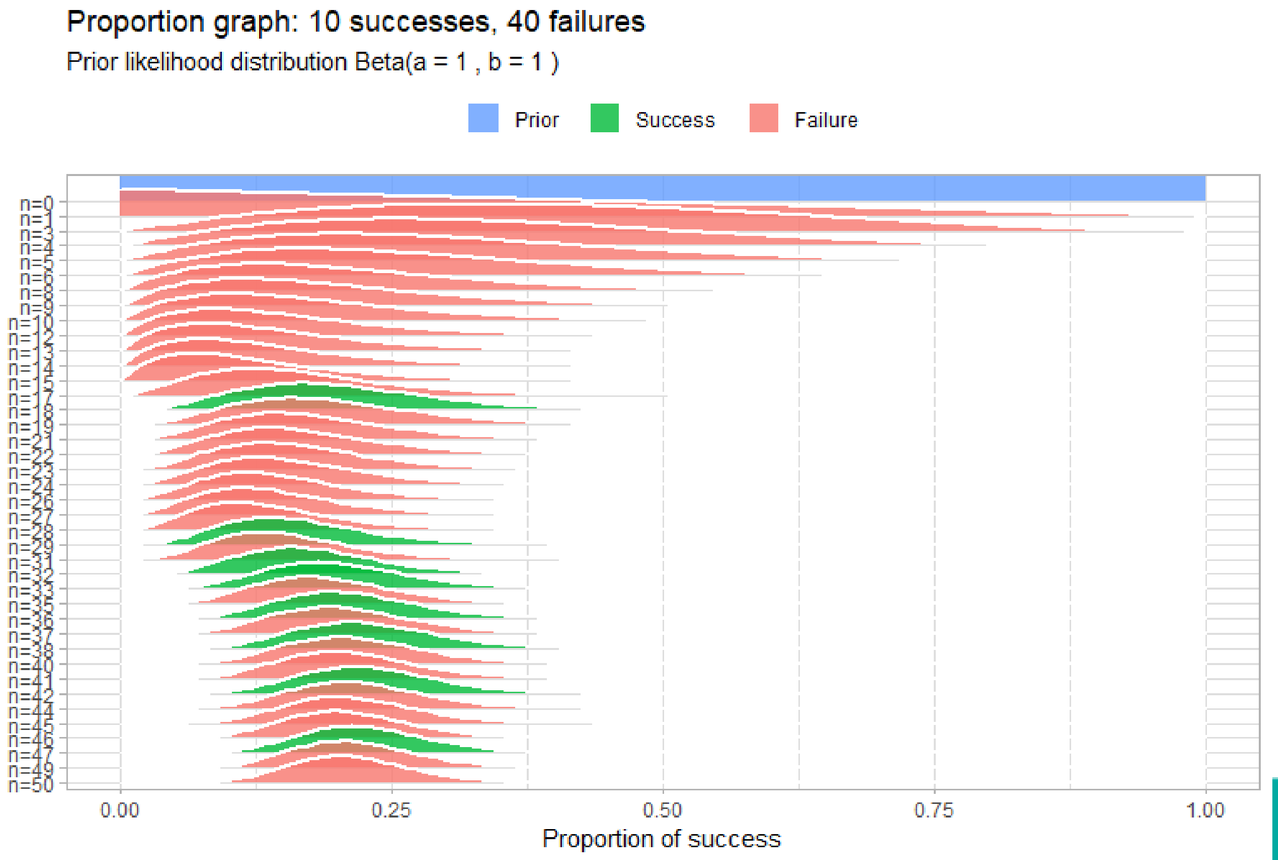
```
data <- c( 1, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0 )
prop_model( data )
```



EXAMPLE

Consider more subjects by using `rbinom()`. Here is how the posterior probability changes with the new case information:

```
data <- rbinom( 50, 1, 0.2 )
prop_model( data )
```

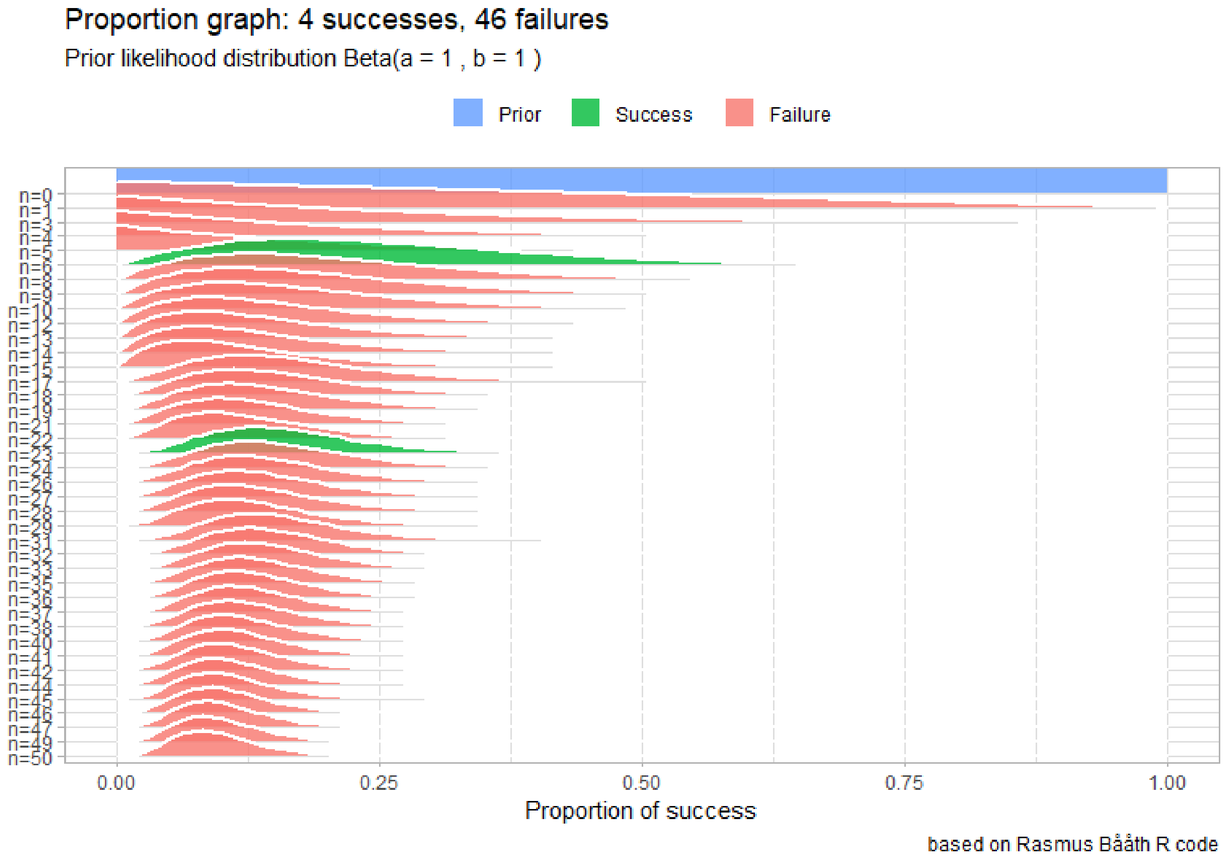


EXAMPLE

Consider we want to compare the experimental zombie drug's success with that of the currently used drug which has a 7% success rate

```
data2 <- rbinom( 50, 1, 0.07 )
posterior2 <- prop_model( data2 )
head( posterior2 )
```

```
> head( posterior2 )
[1] 0.08491134 0.05751564 0.04344934 0.11970881
0.06002682 0.03463283
```



EXAMPLE

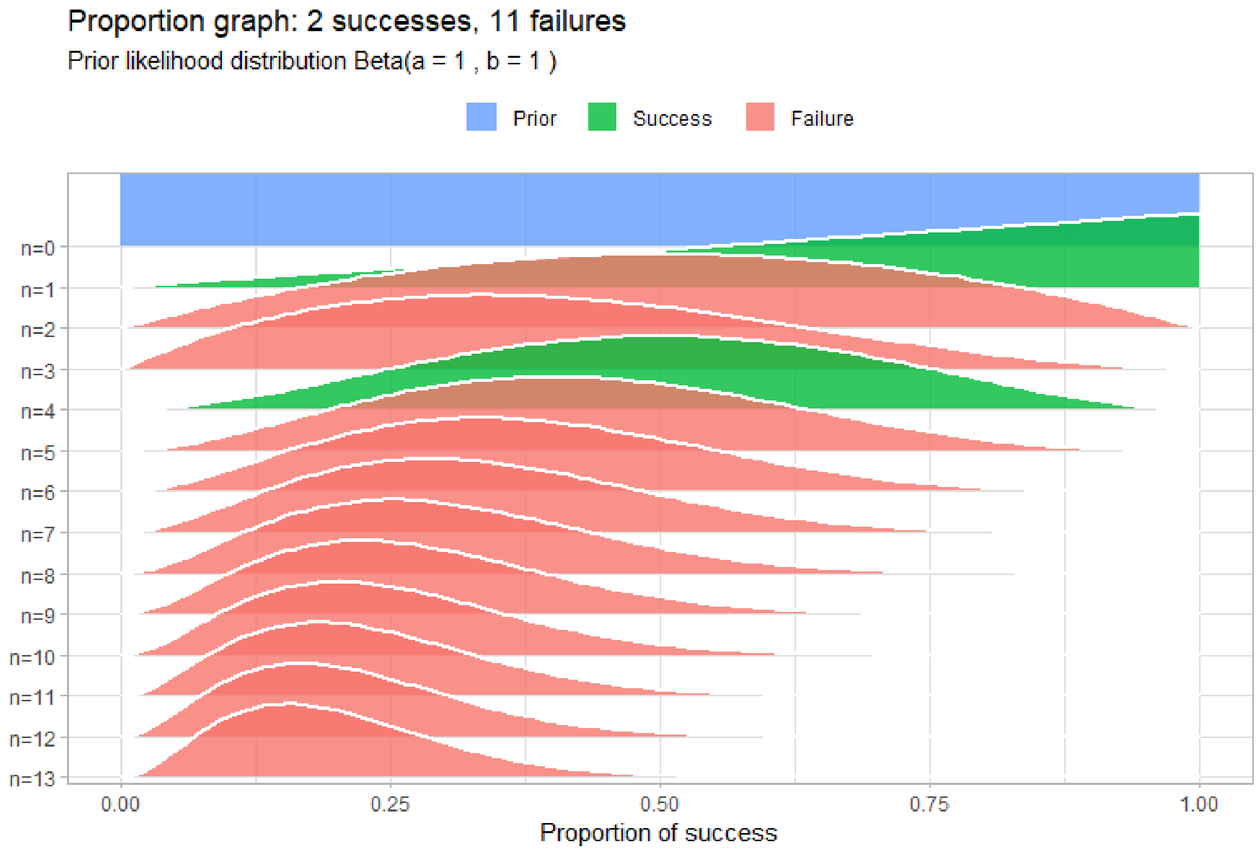
- Subjective observation time: the more information collected about a variable, the closer the observed prior distribution approaches the ‘true’ probability of a success.
- Take a prior probability distribution use observations from the data to update a posterior probability distribution
- **prior** - a probability distribution that represents what the model knows before seeing the data
- **posterior** - a probability distribution that represents what the model knows after having seen the data.

EXAMPLE

Consider the previous data (2 successes, 11 failures), extract the posterior.

```
data = c(1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0)
# Extract and explore the posterior
posterior <- prop_model( data )
```

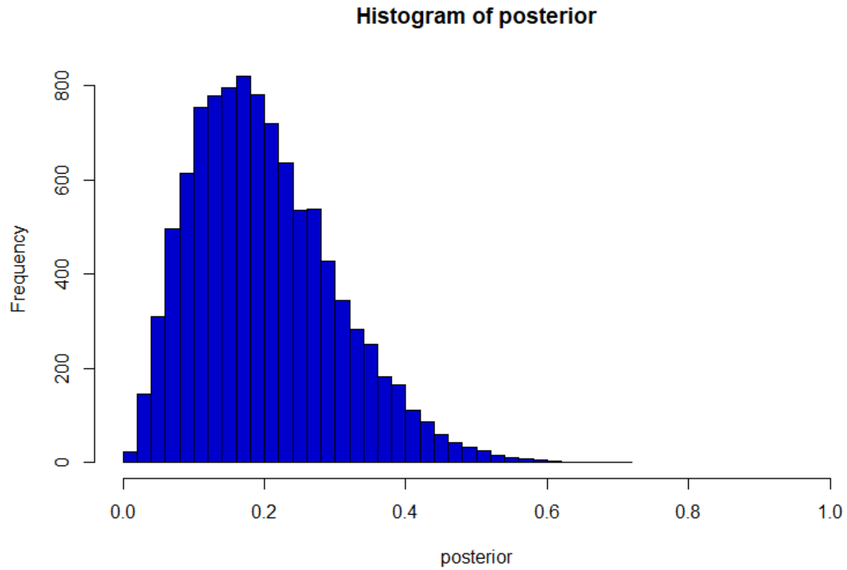
```
> head(posterior)
[1] 0.03039135 0.07150169 0.11515787 0.11851651
0.16231068 0.10970068
```



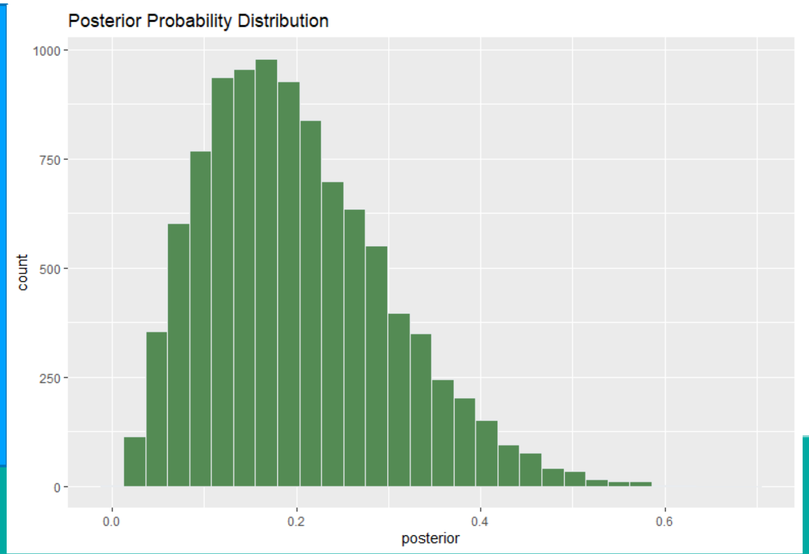
EXAMPLE

Plot the posterior distribution and Calculate a few descriptive statistics for the posterior distribution:

```
hist( posterior, breaks = 30, xlim = c( 0,1
), col = 'palegreen4' )
# Calculate the median
median(posterior)
# Calculate the credible 90% interval
quantile(posterior, c(0.05, 0.95))
# Calculate the probability that the success
rate is greater than 0.7%
sum( posterior > 0.07 )/length(posterior)
```




```
to generate a similar plot with ggplot
post_df <- data.frame( 'posterior' = posterior )
ggplot( post_df, aes( x = posterior ) ) +
  geom_histogram( fill = 'palegreen4', color = "#e9ecef" ) +
  ggtitle( 'Posterior Probability Distribution' )
summary( post_df )
```



EXAMPLE

How to interpret these summary statistics?

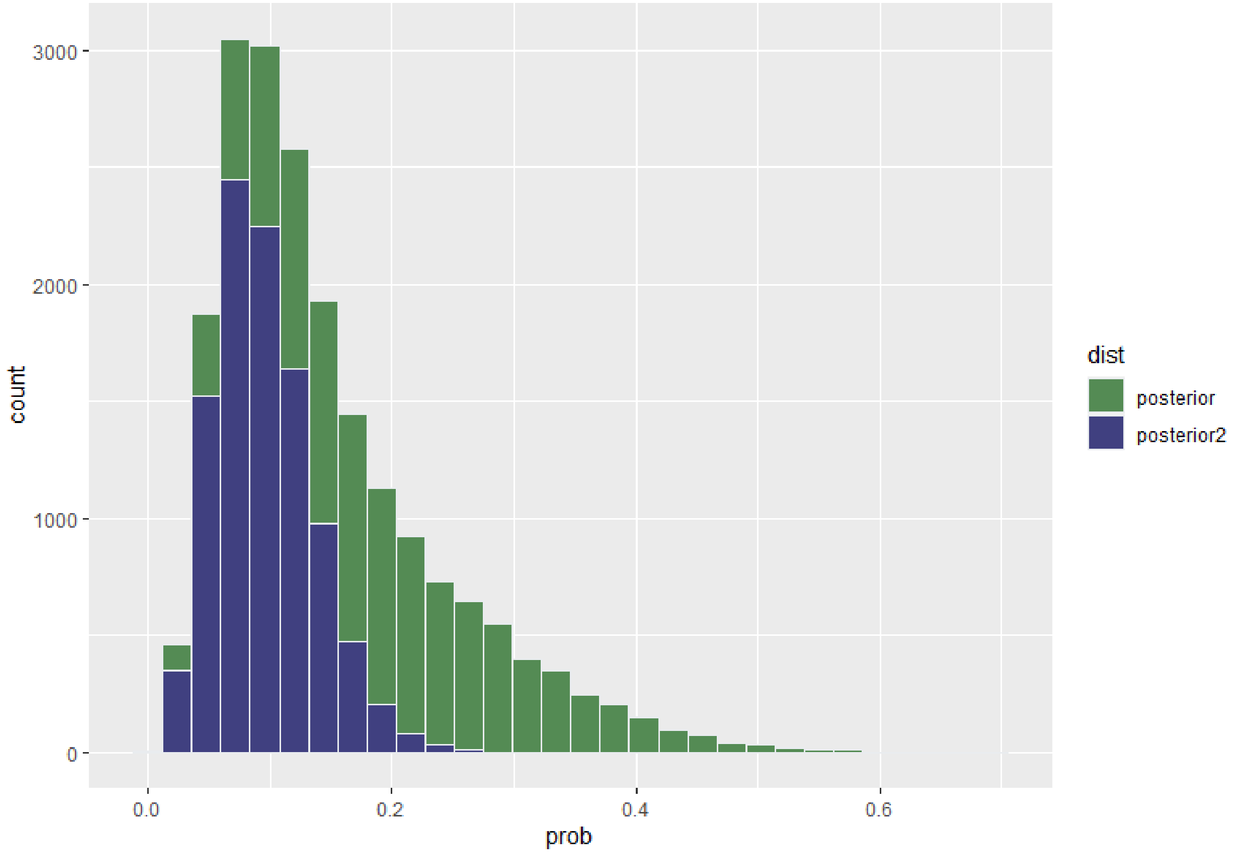
“Given the data of two cured and 11 relapsed zombies, there is a 90% probability that the drug cures between 6% and 39% of treated zombies. Further, there is a 93% probability that this new drug is more effective that the current zombie drug.”

```
> median(posterior)
[1] 0.186583
> # Calculate the credible 90% interval
> quantile(posterior, c(0.05, 0.95))
       5%        95%
0.06089382 0.38627503
> # Calculate the probability that the success rate is
greater than 0.7%
> sum( posterior > 0.07 )/length(posterior)
[1] 0.9291
```

EXAMPLE

Compare the 2 posterior probability distributions in an overlapped histogram:

```
post_df <- data.frame( 'posterior'
= posterior, 'posterior2' =
posterior2 )
post_long <- post_df %>%
  pivot_longer( cols =
everything(), names_to = 'dist',
values_to = 'prob' ) %>%
  ggplot( aes( x = prob, fill =
dist ) ) +
  geom_histogram( color =
"#e9ecef" ) +

scale_fill_manual(values=c("palegr
een4", "#404080"))
post_long
```



EXAMPLE

How does Bayesian Inference Work?

The Parts Needed for Bayesian Inference

Bayesian Inference:

Bayesian Inference =**Data** + a **Generative Model** + **Priors**

Generative Model: can be a computer program, mathematical expression or a set of rules that can be fed parameters values to be used to simulate data.

EXAMPLE

Simulation

```
#Paramters
prop_success <- 0.15
n_zombies <- 13
#simulate the data with rbinom()
data <- rbinom( n_zombies, 1, prop_success )
Data

#simulate from uniform
data <- c()
for( zombie in 1 : n_zombies ) {
  data[ zombie ] <- runif( 1, min = 0, max = 1
) < prop_success
}
data <- as.numeric( data )
Data
prop_success <- 0.42
n_zombies <- 100
#simulate the data with rbinom()
data <- rbinom( 1, n_zombies, prop_success )
data
```

The function gives an idea of the logic for the prevous code. However, using `rbinom()` is more efficient.

Then, run the generative model where the success is estimated at 42% and the test is done with 100 zombies. (37 zombies got cured in this run of the generative model)

How about if it is run 200 times?

EXAMPLE

Simulation: 200 simulation

```
prop_success <- 0.42
n_zombies <- 100
#simulate the data with rbinom()
data <- rbinom( 200, n_zombies, prop_success )
data

summary( data )
mean( data )
```

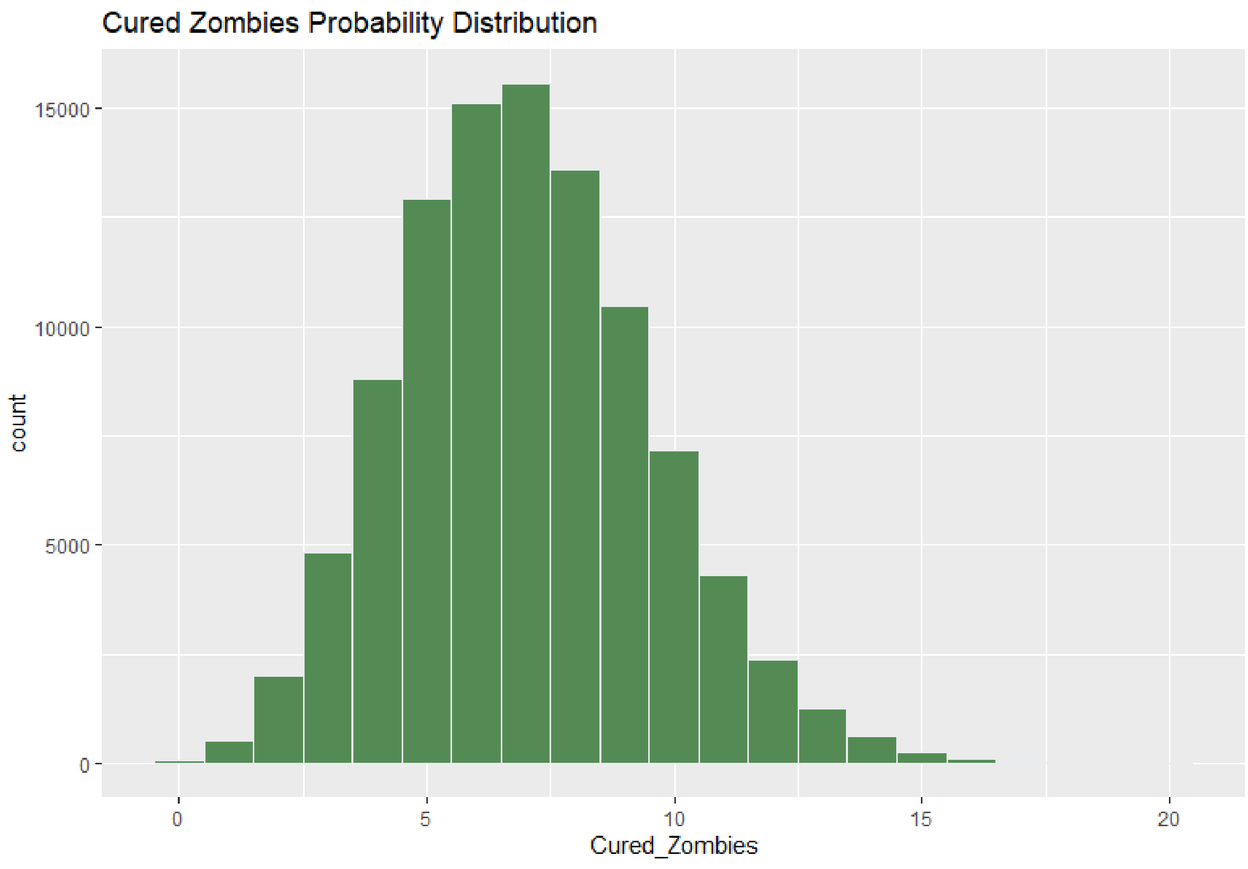
```
> data
  [1] 47 42 46 40 43 34 46 39 38 31 42 44 39 51 43 30 43 45 40 52
45 43 42 39 41 57 37 42 34 38 37 52 32 40 32 46 48 37 43 47 46 37
 [43] 39 43 40 43 34 37 34 48 44 45 46 50 35 46 47 40 39 38 53 46
42 52 42 34 39 38 51 47 42 40 38 40 39 36 46 43 36 35 35 49 38 45
 [85] 42 45 43 41 37 45 39 41 29 44 36 43 42 44 44 52 53 42 36 42
37 45 42 41 47 48 40 44 39 38 38 48 43 38 38 41 45 40 37 39 47 43
[127] 45 41 42 39 41 45 49 54 43 39 52 45 46 42 44 36 43 44 46 41
44 49 37 49 40 47 41 40 36 44 35 45 34 41 38 42 47 44 37 43 46 40
[169] 46 38 45 45 39 39 38 48 41 46 45 42 39 48 46 27 34 37 37 55
40 53 39 41 43 29 38 34 50 37 41 42
> summary( data )
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   27.0    38.0    42.0    41.9    45.0    57.0
> mean( data )
[1] 41.9
```

EXAMPLE

Using a Generative Model

Apply the generative model to a relatively large number of simulated trials. The results will for the probability distribution for the process:

```
cured_zombies <- rbinom( n = 100000, size = 100,
prob = 0.07 )
post_df <- data.frame( 'Cured_Zombies' =
cured_zombies )

ggplot( post_df, aes( x = Cured_Zombies ) ) +
  geom_histogram( binwidth = 1, fill =
'palegreen4', color = "#e9ecef" ) +
  ggtitle( 'Cured Zombies Probability
Distribution' )
```



5.2 Parameter estimation

Always it is more often the case that we have data, and do not need a generative model to create it.

However, we do not know the parameters (populations) that drive the process.

Bayesian Inference is used to invert the probability to work backwards and learn about the underlying process given the data at hand.

EXAMPLE

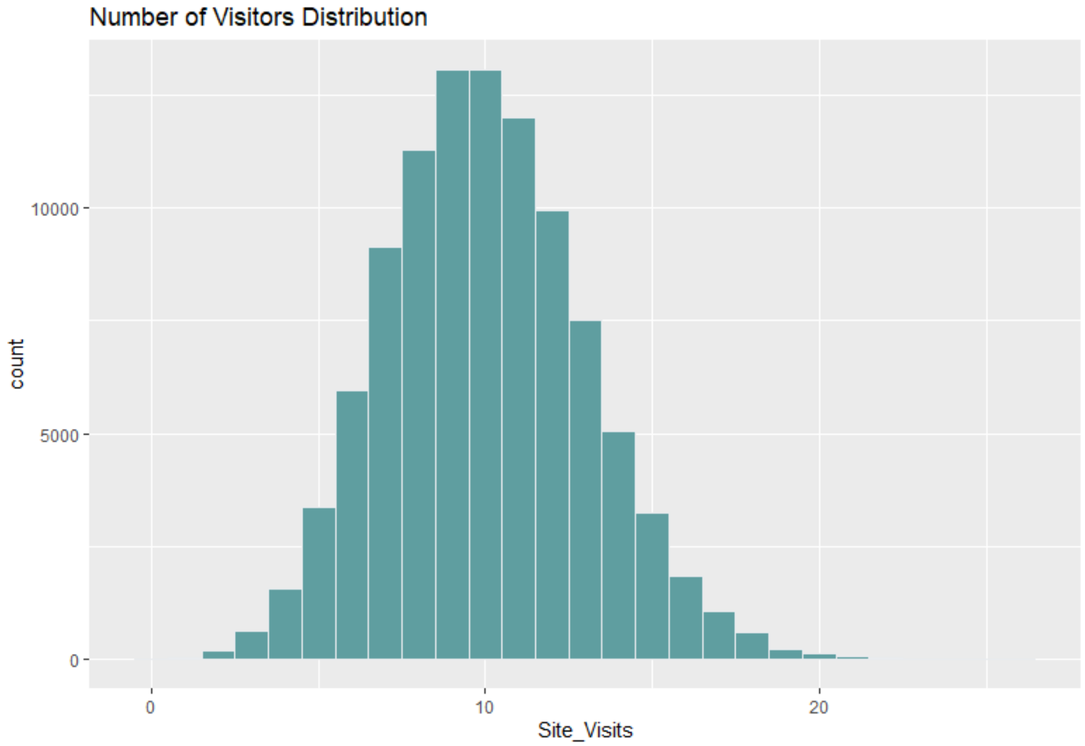
How many visitors to a site?

To get more visitors to your website you are considering paying for an advertisement (ads) to be shown 100 times on a popular social media site. According to the social media site, their ads get clicked on 10% of the time.

EXAMPLE

How many visitors to a site?

```
# Fill in the parameters
n_samples <- 100000
n_ads_shown <- 100
proportion_clicks <- 0.1
n_visitors <- rbinom(n_samples, size = n_ads_shown,
                     prob = proportion_clicks)

post_df <- data.frame( 'Site_Visits' = n_visitors )

ggplot( post_df, aes( x = Site_Visits ) ) +
  geom_histogram( binwidth = 1, fill = 'cadetblue', color =
"#e9ecef" ) +
  ggtitle( 'Number of Visitors Distribution' )
```



EXAMPLE

How many visitors to a site? :Prepresenting Uncertainty with Priors

To address the Priors.

- **Prior Probability Distribution**: represents how uncertain the model is about the parameters before seeing any data. Ideally, this uncertainty should reflect your uncertainty. The **reason** it is called a prior is because it represents the uncertainty before having included information about the data.

- You’re not so sure that your ad will get clicked on exactly 10% of the time. Instead of assigning `proportion_clicks` a single value you are now going to assign it a large number of values drawn from a probability distribution.

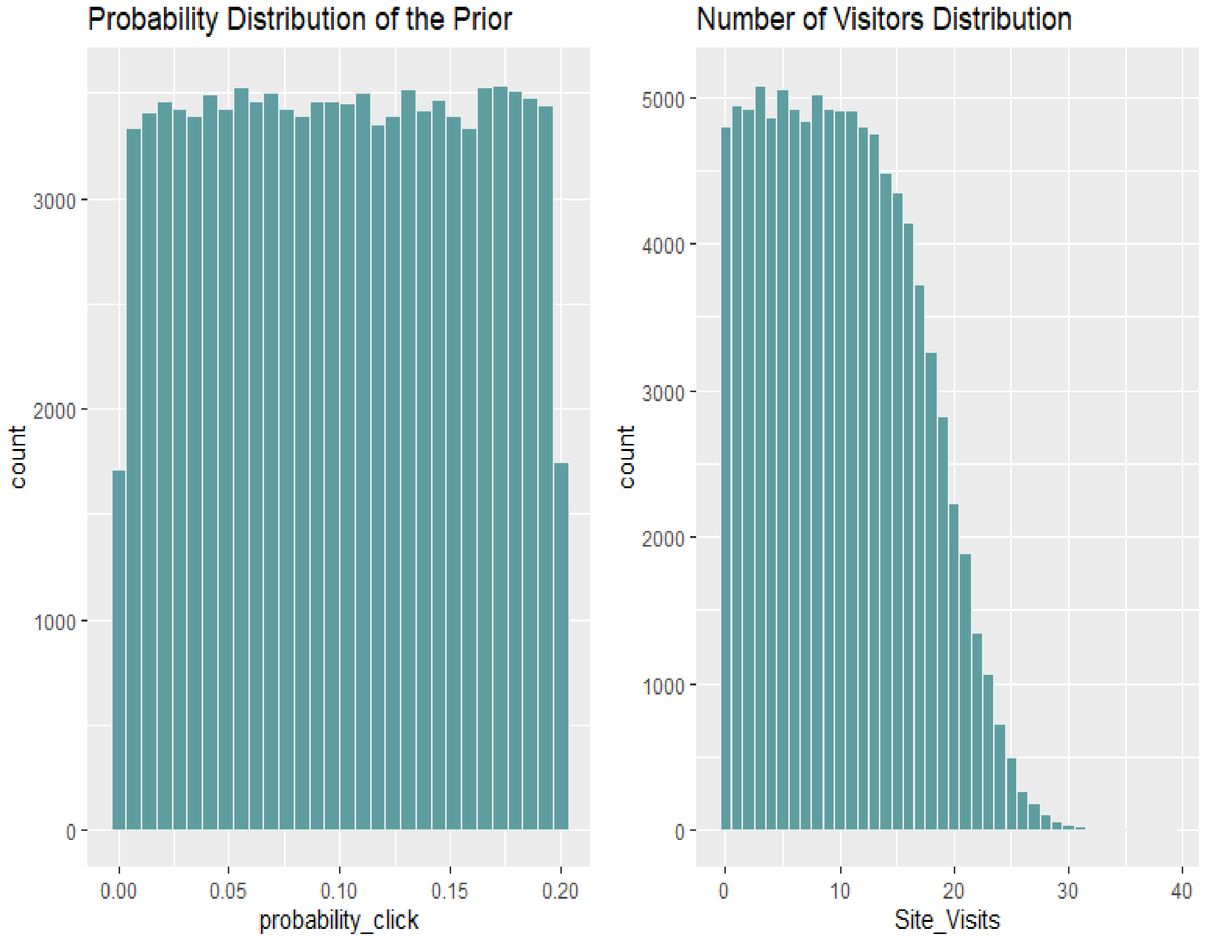
EXAMPLE

Model a uniform probability distribution that ranges from 0 to 0.2 to represent the uncertainty of our value for this prior.

Visualize the probability distribution for the prior, `proportion_clicks`.

How many visitors to a site?

```
n_samples <- 100000
n_ads_shown <- 100
# model a uniform probability distribution that ranges from
the uncertainty of our value for this prior
proportion_clicks <- runif( n = n_samples, min = 0.0, max =
n_visitors <- rbinom( n_samples, n_ads_shown, proportion_cli

#Visualize the probability distribution for the prior, propo
pclick_df <- data.frame( 'probability_click' = proportion_cl

pclick_plot <- ggplot( pclick_df, aes( x = probability_click
  geom_histogram( fill = 'cadetblue', color = "#e9ecef" ) +
  ggtitle( 'Probability Distribution of the Prior' )

nvis_df <- data.frame( 'Site_Visits' = n_visitors )

nvis_plot <- ggplot( nvis_df, aes( x = Site_Visits ) ) +
  geom_histogram( binwidth = 1, fill = 'cadetblue', color =
  ggtitle( 'Number of Visitors Distribution' )

grid.arrange( pclick_plot, nvis_plot, ncol = 2 )
```



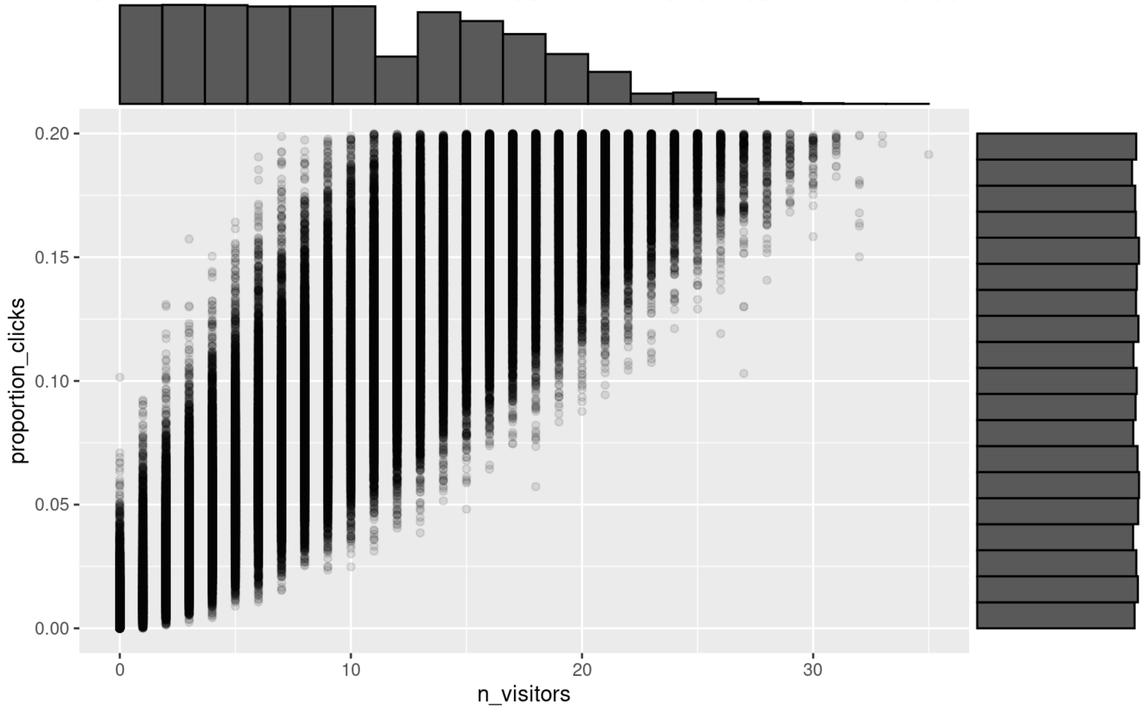
Envelope of the `n-visitors` data from this generative model is much different. The added uncertainty for the prior increases the uncertainty in the number of visitors the site will receive.

EXAMPLE

Bayesian Models and Conditioning

```
#this dataframe represents the joint probability
distribution over proportion_clicks and n_visitors
together.
prior_df <- data.frame( 'proportion_clicks' =
proportion_clicks, 'n_visitors' = n_visitors )

#visualize as a scatterplot....
prior_plot <- ggplot( prior_df, aes( x = n_visitors, y
= proportion_clicks ) ) +
  geom_point( alpha = 1/10 ) +
  #geom_jitter(alpha = 1/10 ) +
  theme( legend.position = 'none' )
prior_plot <- ggMarginal( prior_plot, type =
'histogram',
                          xparams = list(  bins=20 ),
                          yparams = list(  bins=20))
prior_plot
```

Consider if we send our add out into the wilds of the marketplace and get some data on its performance. Let's say 13 out of 100 users



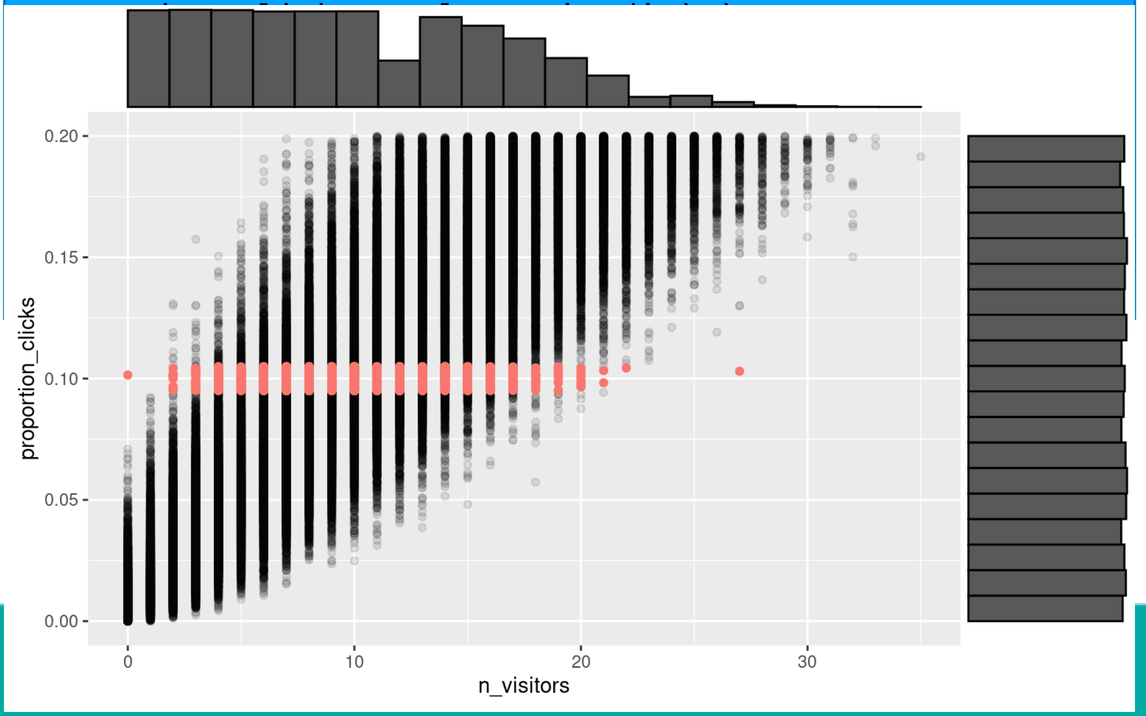
and we ..

ons. her the te

EXAMPLE

Bayesian Models and Conditioning

```
prior10_df <- prior_df %>%
  mutate( p10 = round( proportion_clicks, 2) == 0.1 )
%>%
  filter( p10 == TRUE ) %>%
  select( c( n_visitors, proportion_clicks ) )

prior_plot <- ggplot( prior_df, aes( x = n_visitors, y
= proportion_clicks ) ) +
  geom_point( alpha = 1/10 ) +
  geom_point(data = prior10_df, aes(x = n_visitors, y =
```

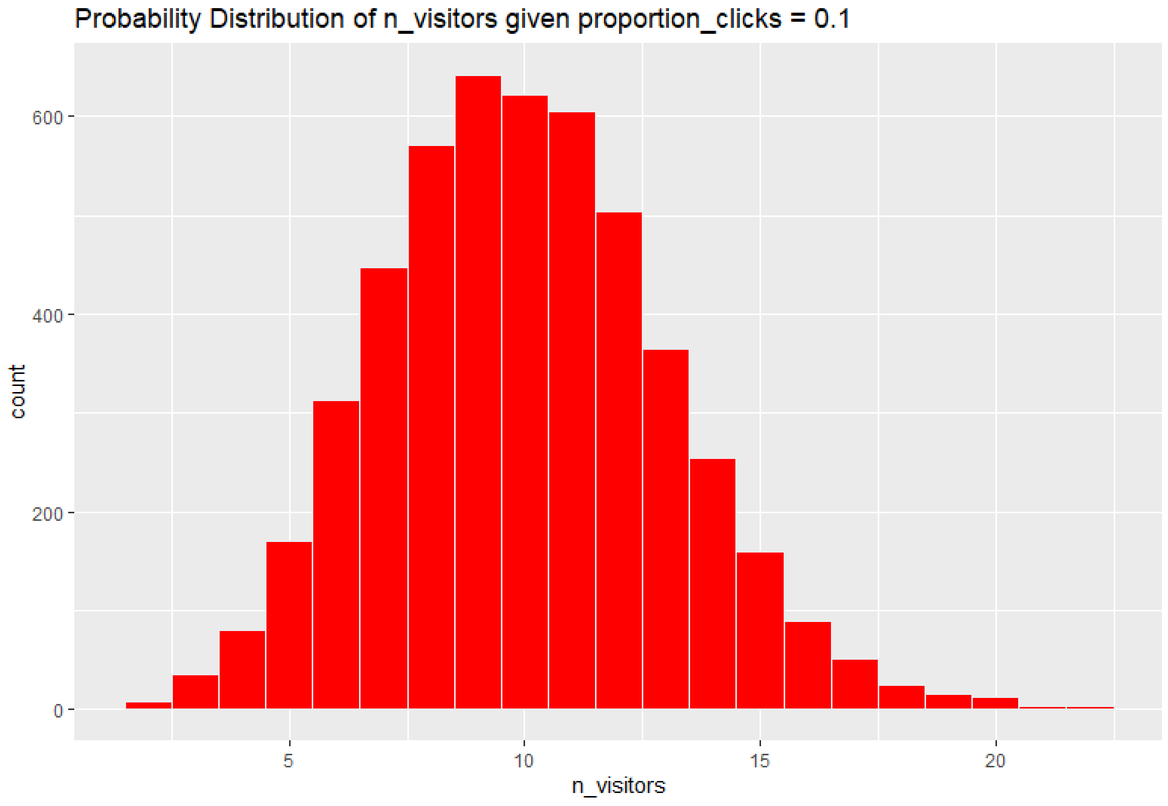


From the joint probability data, we can come up with probability distributions for specific conditional situations. For example, If we know a conditional value for `proportion_clicks = 0.10`, then we can describe the distribution of `n_visitors`.

EXAMPLE

Bayesian Models and Conditioning

```
ggplot( prior10_df, aes( x = n_visitors ) ) +
  geom_histogram( binwidth = 1, fill = 'red', color =
"#e9ecef" ) +
  ggtitle( 'Probability Distribution of n_visitors
given proportion_clicks = 0.1' )
```



Shift the `proportion_clicks`, the distribution of `n_visitors` shifts accordingly.

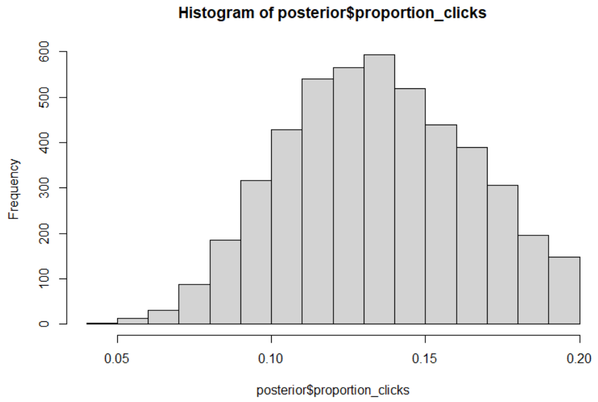
Additionally, we can condition the `proportion_clicks` on a given `n_visitors`.

This shows **the essence of Bayesian Inference**:
Bayesian inference is conditioning on data, in order to learn about parameter values.

EXAMPLE

Bayesian Models and Conditioning

```
head( prior_df )
#condition the joint distribution with the observation
that there were 13 visitors
# Create the posterior data frame
posterior <- prior_df[prior_df$n_visitors == 13, ]

# Visualize posterior proportion clicks
hist( posterior$proportion_clicks )
```



Let's condition on the data we collected were we got 13 visits from 100 presentations of the ad.

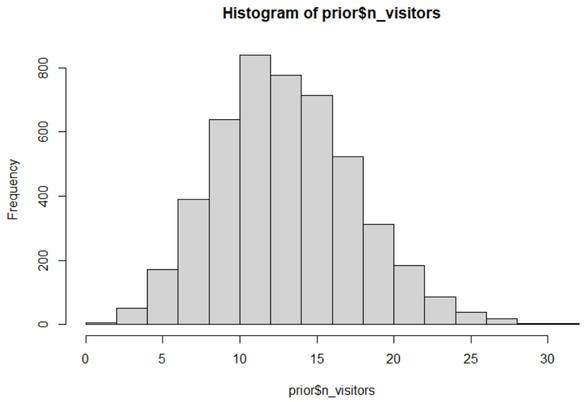
Find the **posterior** distribution for the proportion of clicks given there were 13 visitors.

It is called a posterior, because it represents the uncertainty *after* having included information from the observed data.

EXAMPLE

Bayesian Models and Conditioning

```
# Assign posterior to a new variable called
prior
prior <- posterior

# Take a look at the first rows in prior
head(prior)
# Replace prior$n_visitors with a new sample
and visualize the result
n_samples <-  nrow(prior)
n_ads_shown <- 100
prior$n_visitors <- rbinom(n_samples, size =
n_ads_shown,                              prob =
prior$proportion_clicks)
hist(prior$n_visitors)
# Calculate the probability that you will get
5 or more visitors
sum(prior$n_visitors >= 5) /
length(prior$n_visitors)
```

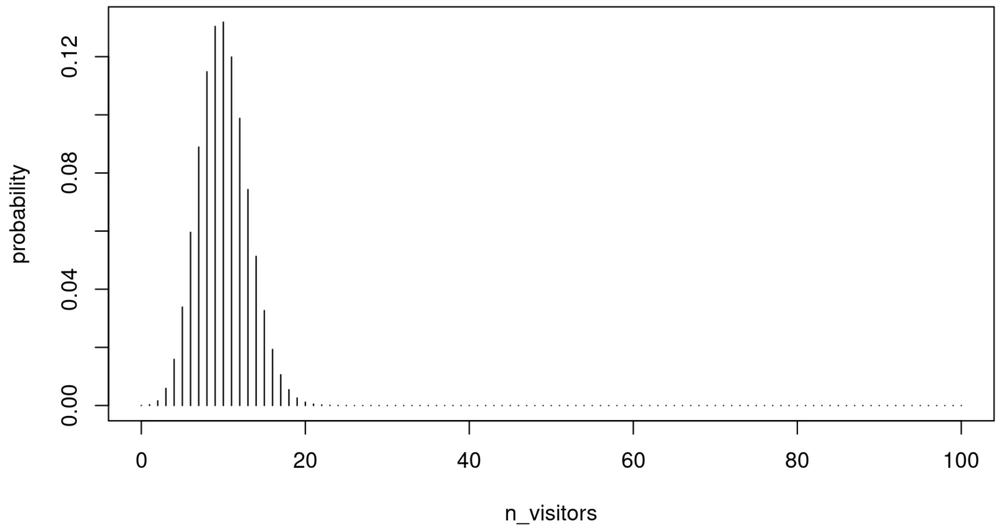


From the observation of 13/100 visits, we find that it is very probable (99% probable) that future add campaigns will generate at least 5 visits.

EXAMPLE

Calculating the likelihood:

```
#calculate P(n_visitors = 13 | prob_success =
10%)
dbinom( 13, size = 100, prob = 0.1)

#calculate P(n_visitors = 13 or n_visitors = 14
| prob_success = 10 )
dbinom( 13, size = 100, prob = 0.1 ) + dbinom(
14, size = 100, prob = 0.1 )

#Calculate Probability Distribution
#calculate P( n_visitors | prob_success = 0.1 )
n_visitors <- seq( 0, 100, by = 1 )
probability <- dbinom( n_visitors, size = 100,
prob = 0.1 )
plot( n_visitors, probability, type = 'h’ )

#Probability Density.. Continuous Distributions
dunif( x = 0.12, min = 0, max = 0.2 )
```

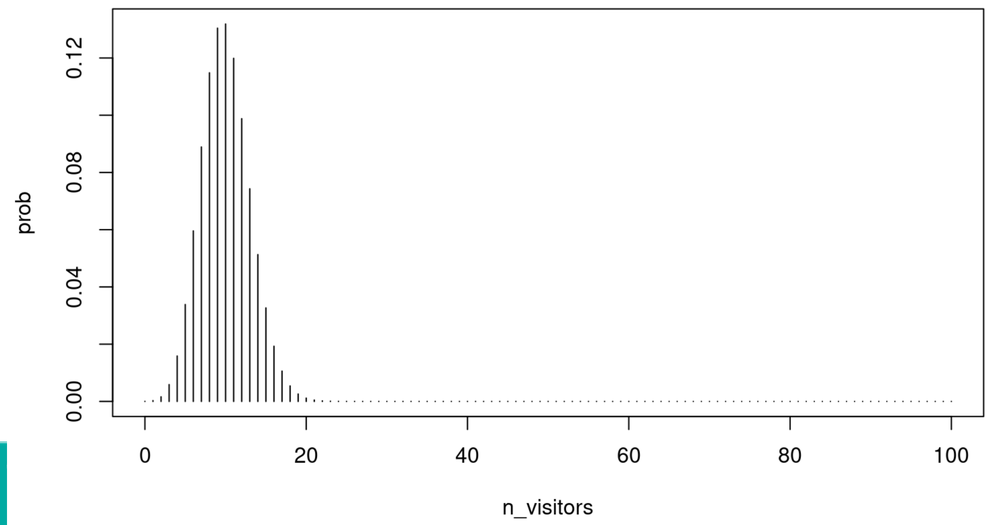


Simulate using the `rbinom`, `rpois` etc. functions. we can calculate using the d functions like `dbinom` and `pois`

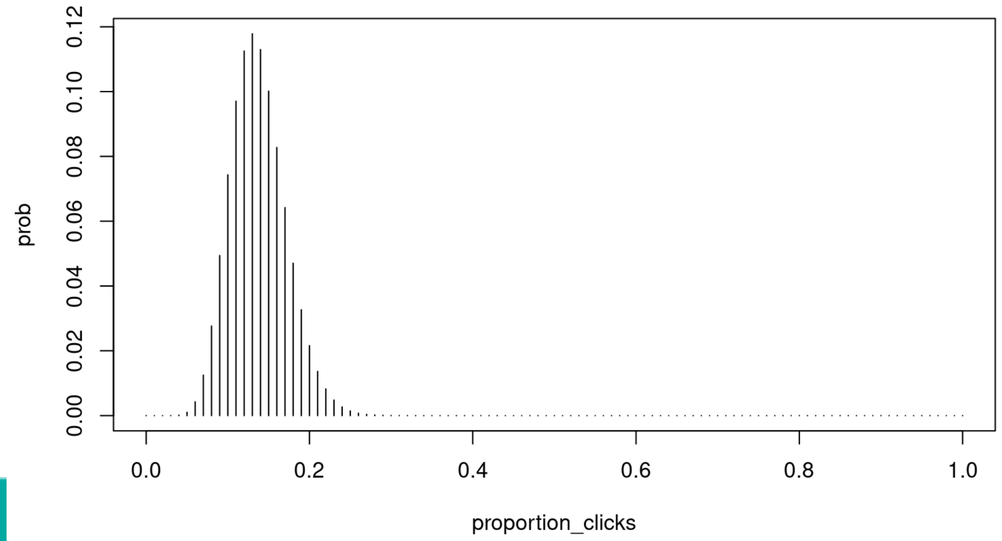
RECALL

Calculating the likelihood:

```
# Explore using dbinom to calculate probability
distributions
n_ads_shown <- 100
proportion_clicks <- 0.1
n_visitors <- seq(0, 100)
prob <- dbinom(n_visitors,
    size = n_ads_shown, prob =
proportion_clicks)

# Plot the distribution
plot( n_visitors, prob, type = 'h')
```




```
# Change the code according to the instructions
n_ads_shown <- 100
proportion_clicks <- seq(0, 1, by = 0.01)
n_visitors <- 13
prob <- dbinom(n_visitors,
    size = n_ads_shown, prob =
proportion_clicks)

plot(proportion_clicks, prob, type = "h")
```

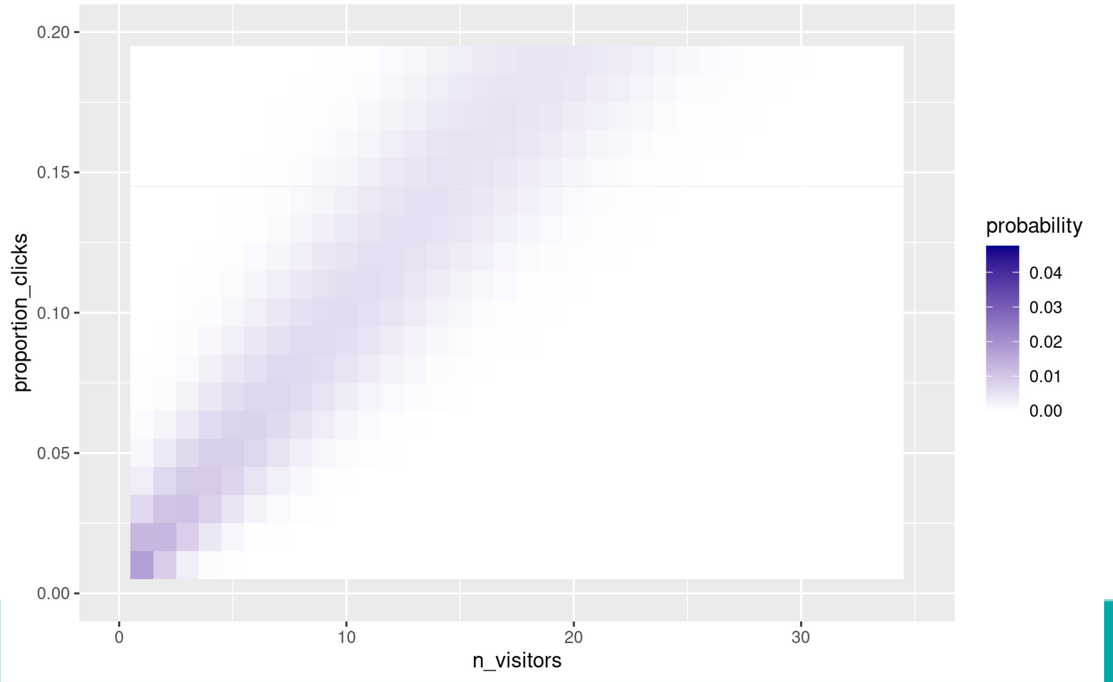


EXAMPLE

Calculating the likelihood: Bayesian Calculation

```
n_ads_shown <- 100
n_visitors <- seq( 0, 100, by = 1 )
proportion_clicks <- seq( 0, 1, 0.01 )
pars <-  expand.grid( proportion_clicks =
proportion_clicks,
                      n_visitors = n_visitors )
pars$prior <- dunif( pars$proportion_clicks, min
= 0, max = 0.2 )
pars$likelihood <- dbinom( pars$n_visitors, size
= n_ads_shown,
                           prob =
pars$proportion_clicks)
pars$probability <- pars$likelihood * pars$prior
pars$probability <- pars$probability / sum(
pars$probability )

glimpse(pars)
```

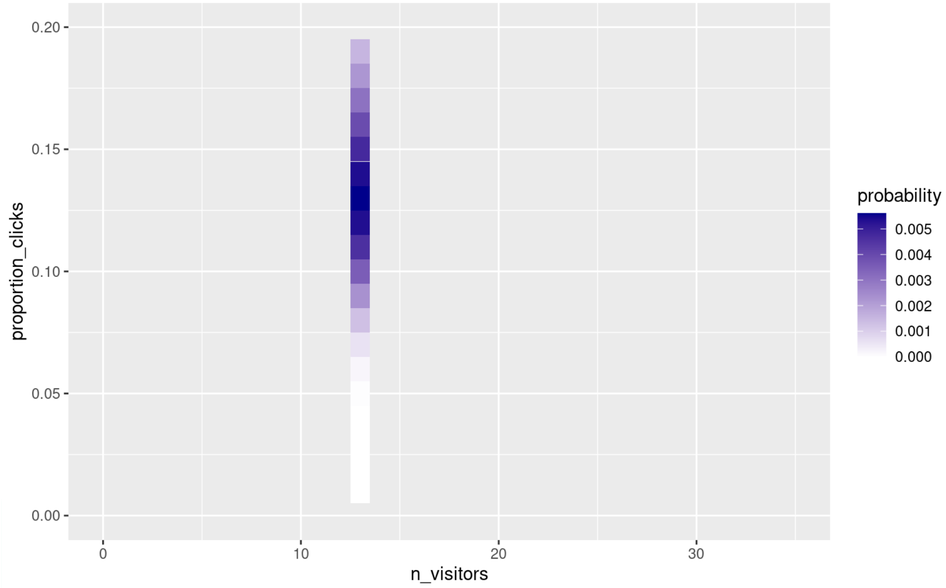
```
ggplot( pars, aes( x = n_visitors, y =
proportion_clicks,
                   fill = probability ) ) +
  geom_tile() +
  xlim( c( 0, 35 ) ) +
  ylim( c( 0, 0.2 ) ) +
  scale_fill_gradient(low="white",
high="darkblue")
```



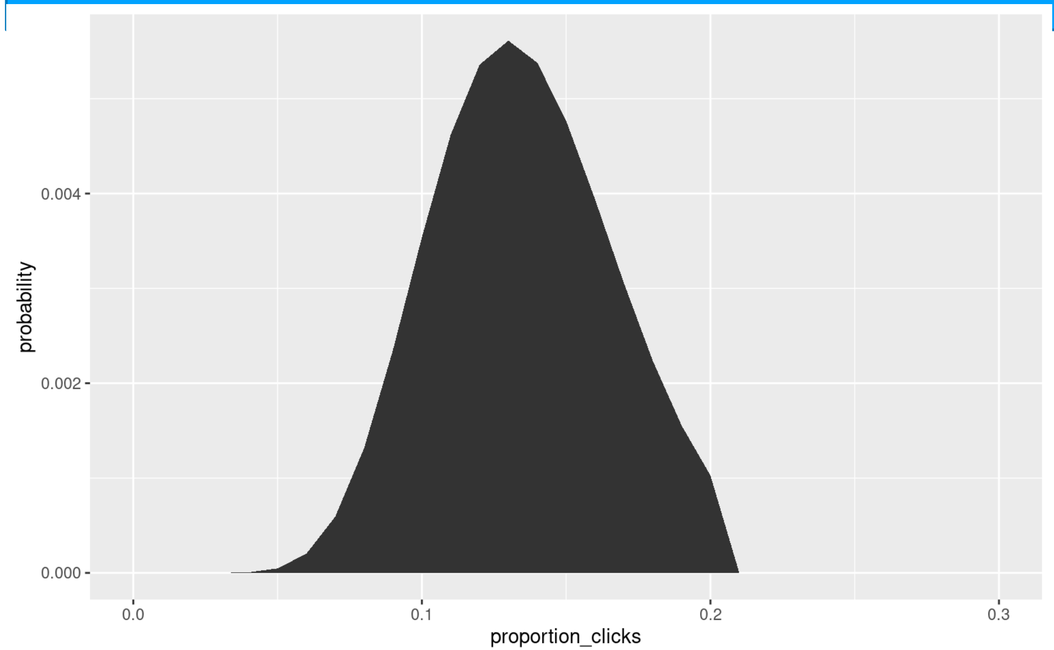
EXAMPLE

Calculating the likelihood: Bayesian Calculation

```
pars13 <- pars[ pars$n_visitors == 13, ]
pars13$probability <- pars13$probability / sum(
pars$ probability )

ggplot( pars13, aes( x = n_visitors, y =
proportion_clicks,
                     fill = probability ) ) +
  geom_tile() +
  xlim( c( 0, 35 ) ) +
  ylim( c( 0, 0.2 ) ) +
  scale_fill_gradient(low="white", high="darkblue")
```



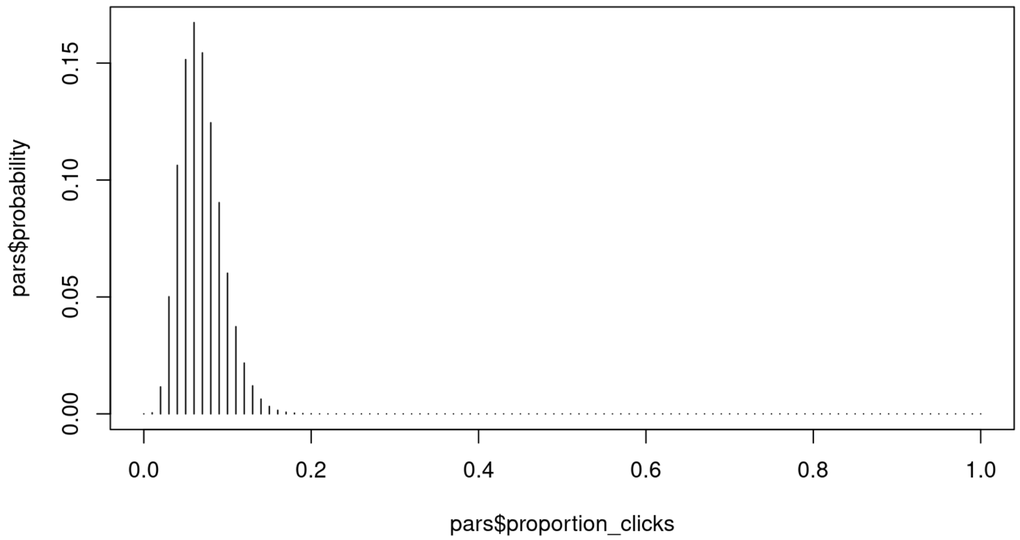

```
ggplot( pars13, aes( x = proportion_clicks, y =
probability ) ) +
  geom_area() +
  xlim( c( 0, 0.3 ) ) )
```



EXAMPLE

Calculating the likelihood: Bayesian Calculation

```
A conditional shortcut: You can directly condition on the
data, no need to first create the joint distribution.

# Simplify the code below by directly conditioning on the
data
n_ads_shown <- 100
proportion_clicks <- seq(0, 1, by = 0.01)
n_visitors <- 6
pars <- expand.grid(proportion_clicks =
proportion_clicks,
                    n_visitors = n_visitors)
pars$prior <- dunif(pars$proportion_clicks, min = 0, max
= 0.2)
pars$likelihood <- dbinom(pars$n_visitors,
    size = n_ads_shown, prob = pars$proportion_clicks)
pars$probability <- pars$likelihood * pars$prior
pars$probability <- pars$probability /
sum(pars$probability)

plot(pars$proportion_clicks, pars$probability, type =
"h")
```



Bayes' Theorem

This is an implementation of Bayes' Theorem

```
pars$probability <- pars$likelihood * pars$prior
pars$probability <- pars$probability / sum( pars$probability )
```

$$P(\theta|D) = \frac{\mathrm{P}(D|\theta) \cdot \mathrm{P}(\theta)}{\sum \mathrm{P}(D|\theta) \cdot \mathrm{P}(\theta)}$$

where:

- θ = paramters
- D = the Data

The probability of the different parameter values given some data equals the likelihood of the relative probability of the data given the different parameter values **multiplied** by the prior probability of the different parameter values **before seeing** the data normalized by the total sum of the likelihood weighted by the prior.

We calculated the probability using an algorithm called **grid approximation**. However, there are many more algorithms to fit Bayesian models, some being more efficient than others.

EXAMPLE

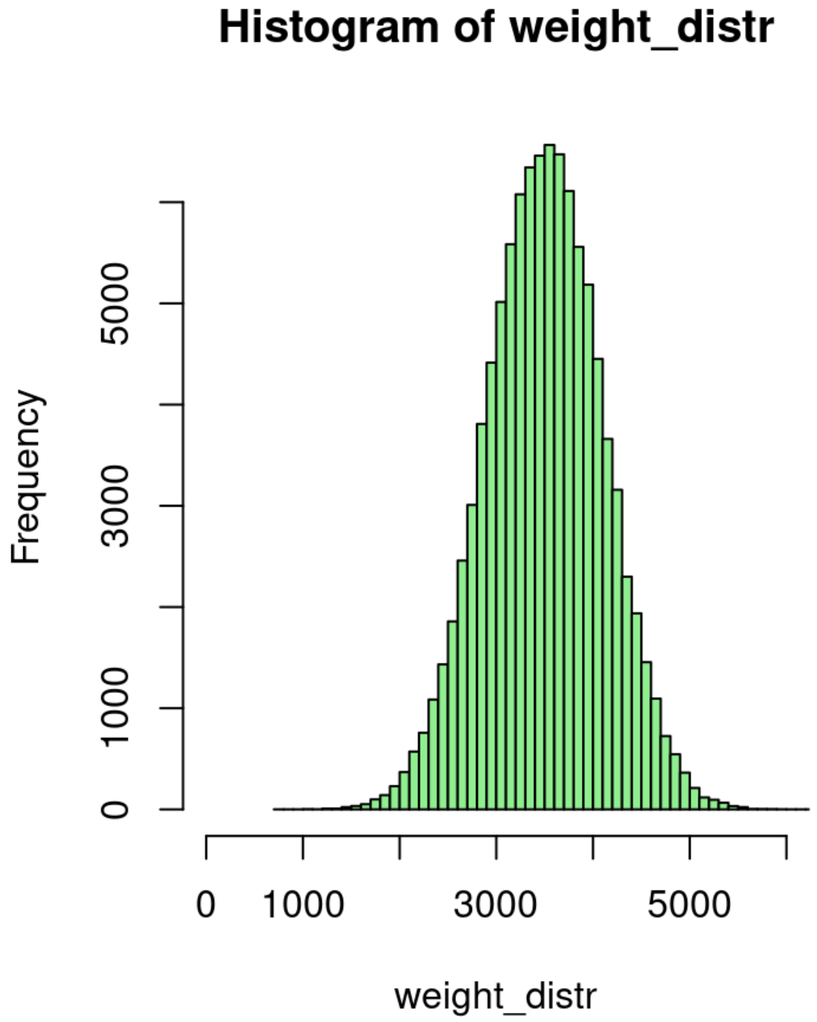
The Temperature in a Normal Lake

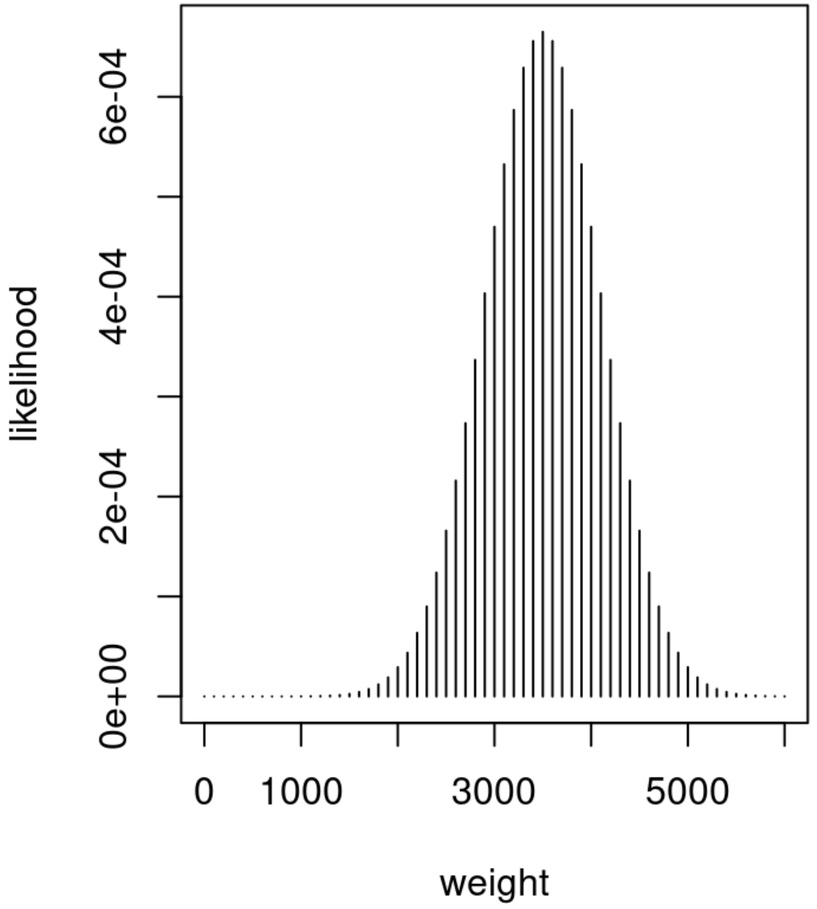
```
temp <- c( 19, 23, 20, 17, 23 )
#f = 66, 73, 68, 63, 73

#Use the Normal distribution as a generative
model for values centered about a mean value.
rnorm( n = 5, mean =  20, sd = 2 )

#use dnorm() to look at how likely an outcome is
#given some fixed parameters
like <- dnorm( x = temp, mean = 20, sd = 2 )
like

#the results are the relative likelihoods of the
data points.
#log likelihood
log(like)
```

```
using rnorm & dnorm

# Assign mu and sigma
mu <- 3500
sigma <- 600

par(mfrow = c( 1,2 ) )
weight_distr <- rnorm(n = 100000, mean = mu, sd
= sigma)
p1 <- hist(weight_distr, 60, xlim = c(0, 6000),
col = "lightgreen")

# Create weight
weight <- seq(0, 6000, by = 100)

# Calculate likelihood
likelihood <- dnorm(weight, mu, sigma)

# Plot the distribution of weight
p2 <- plot( x = weight,
y = likelihood,
type = 'h' )
```

EXAMPLE

The Temperature in a Normal Lake





EXAMPLE

The Temperature in a Normal Lake

A Bayesian Model of Water Temperature
We have out temps: temp = 19, 23, 20, 17, 23 and
we have out generative model: $temp_i \sim Normal(\mu, \sigma)$

Now we need to our prior distributions,
our informed guess about water temperatures is that we can expect the following standard deviation and mean:
$\sigma \sim Uniform(min: 0, max: 10)$ $\mu \sim Normal(mean: 18, sd: 5)$

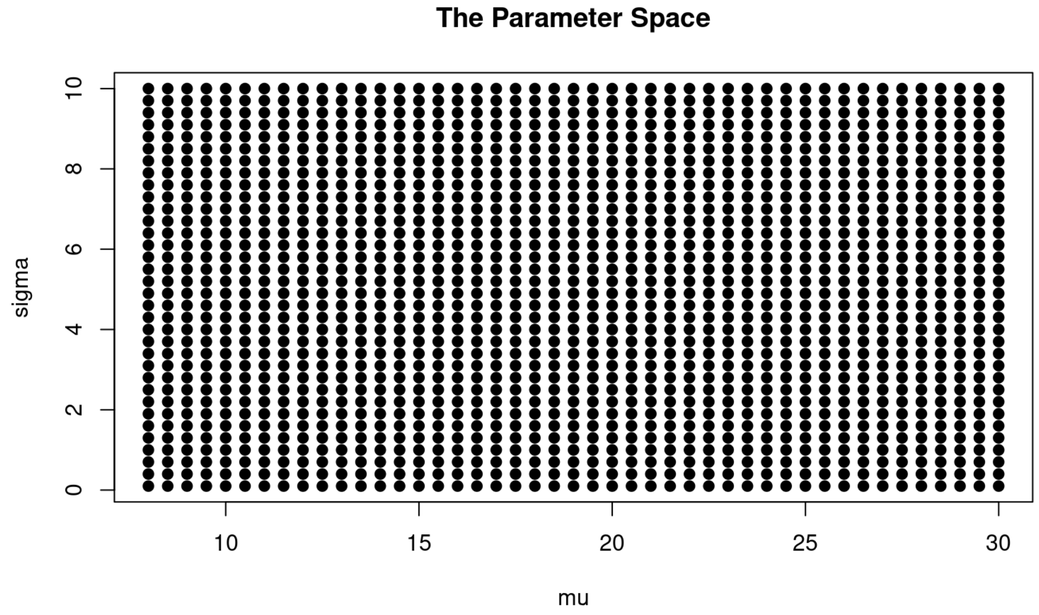
We could have used many other priors, but this is our best guess for now.

EXAMPLE

The Temperature in a Normal Lake

```
temp <- c( 19, 23, 20, 17, 23 )
mu <- seq( 8, 30, by = 0.5 )
sigma <- seq( 0.1, 10, by = 0.3 )
pars <- expand.grid( mu = mu, sigma = sigma )

plot( pars, pch = 19, main = 'The Parameter Space' )

#the priors
pars$mu_prior <- dnorm( pars$mu, mean = 18, sd = 5 )
pars$sigma_prior <- dunif( pars$sigma, min = 0, max = 10 )
pars$prior <- pars$mu_prior * pars$sigma_prior
for( i in 1:nrow( pars ) ) {
  likelihoods <- dnorm( temp, pars$mu[i], pars$sigma[i])
  pars$likelihood[i] <- prod( likelihoods )
}

#calculate the posterior probability
pars$probability <- pars$likelihood * pars$prior
pars$probability <- pars$probability / sum(
pars$probability )

summary( pars )
```

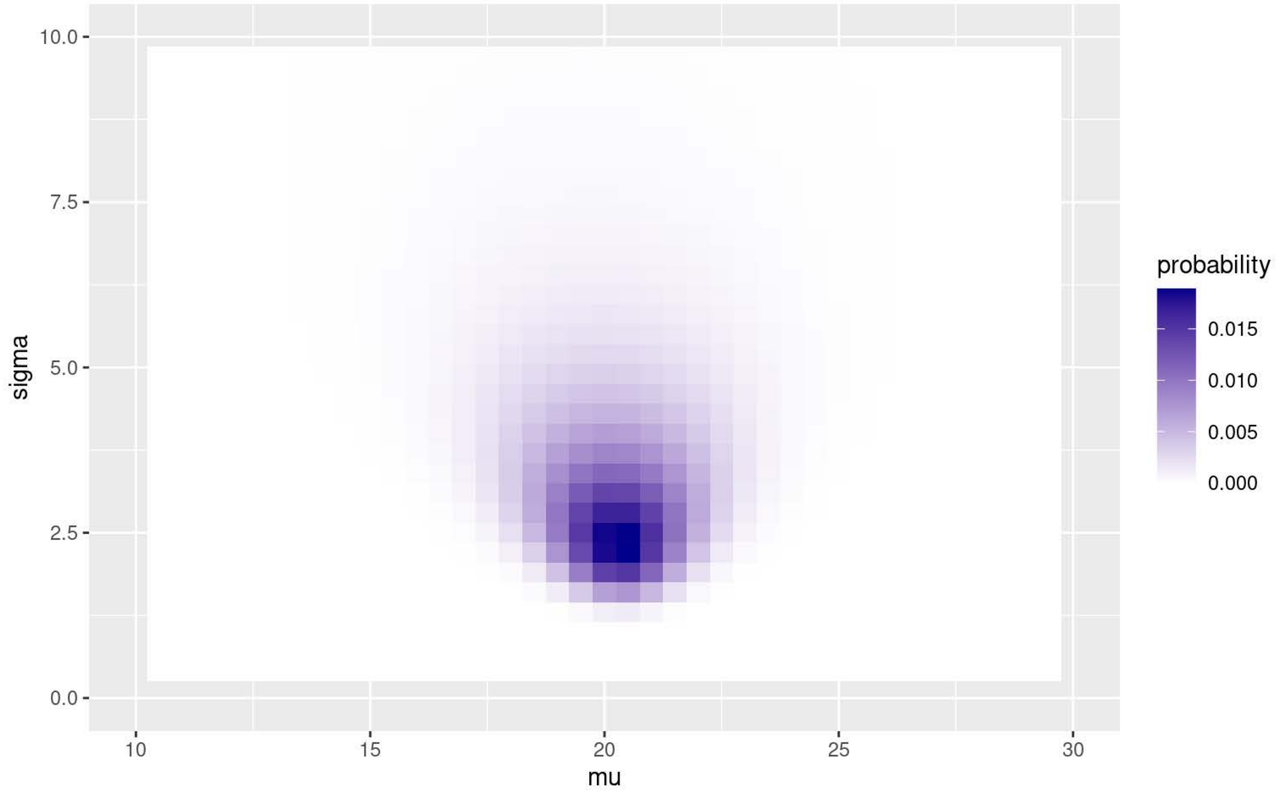
Fit our new normal model using the grid approximation method we used earlier.



EXAMPLE

The Temperature in a Normal Lake

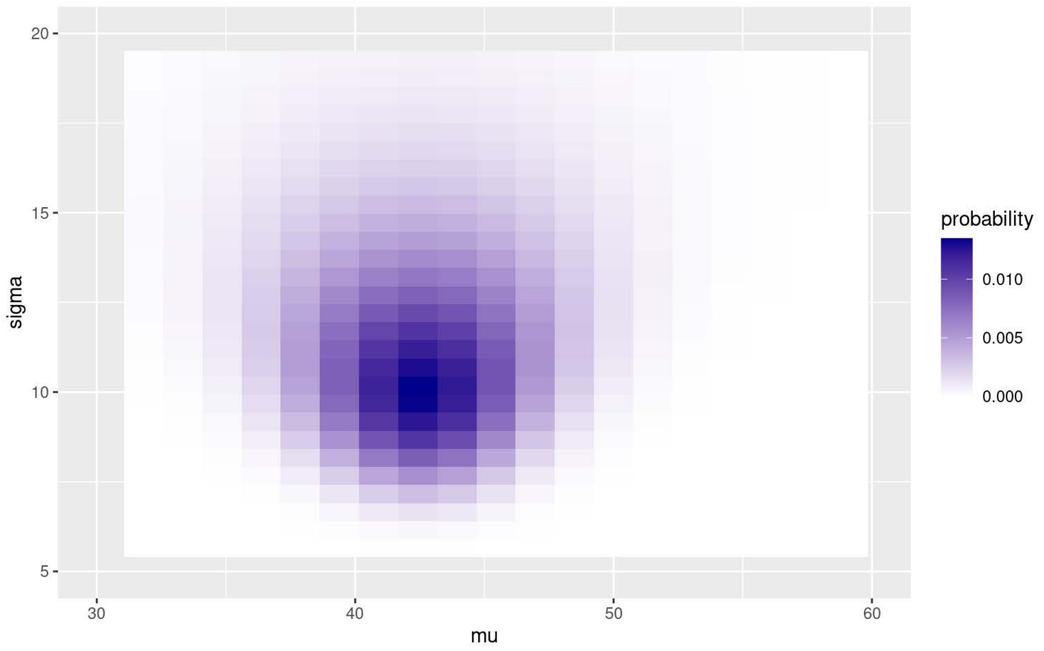
```
#Visualise
ggplot( pars, aes( x = mu, y = sigma,
                   fill = probability ) ) +
  geom_tile() +
  xlim( c( 10, 30 ) ) +
  ylim( c( 0, 10 ) ) +
  scale_fill_gradient(low="white",
high="darkblue")
```



From this result, we would expect the most probable temperature to be around 19-22 degrees C.

EXAMPLE : The zombie example

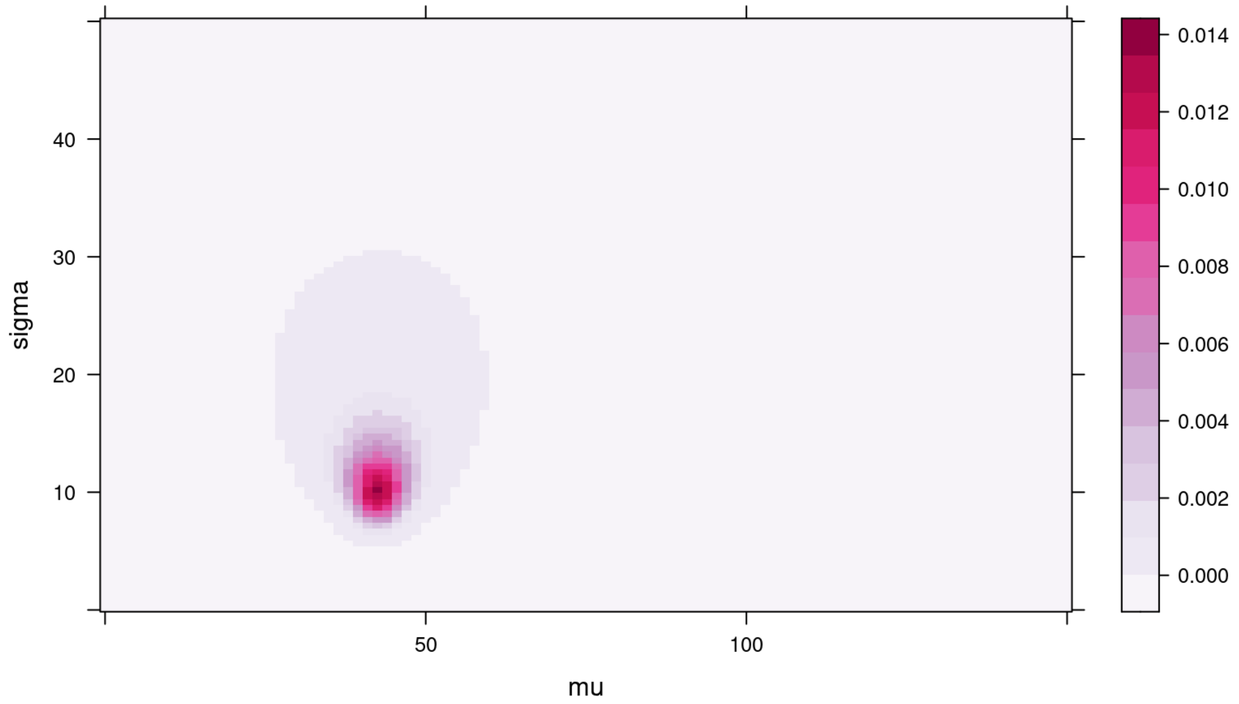
```
# The IQ of a bunch of zombies
iq <- c(55, 44, 34, 18, 51, 40, 40, 49, 48, 46)
# Defining the parameter grid
pars <- expand.grid(mu = seq(0, 150, length.out = 100),
                    sigma = seq(0.1, 50, length.out = 100))
# Defining and calculating the prior density for each
parameter combination
pars$mu_prior <- dnorm(pars$mu, mean = 100, sd = 100)
pars$sigma_prior <- dunif(pars$sigma, min = 0.1, max = 50)
pars$prior <- pars$mu_prior * pars$sigma_prior
# Calculating the likelihood for each parameter combination
for(i in 1:nrow(pars)) {
  likelihoods <- dnorm(iq, pars$mu[i], pars$sigma[i])
  pars$likelihood[i] <- prod(likelihoods)
}
# Calculate the probability of each parameter combination
pars$probability <- pars$likelihood * pars$prior
pars$probability <- pars$probability / sum( pars$probability
)

# Visualize
ggplot( pars, aes( x = mu, y = sigma,
                   fill = probability ) ) +
  geom_tile() +
  xlim( c( 30, 60 ) ) +
  ylim( c( 5, 20 ) ) +
  scale_fill_gradient(low="white", high="darkblue")
```



From this result, we would expect the most probable temperature to be around 19-22 degrees C.

EXAMPLE : The zombie example

```
Generate the same visualization as in the course using the
levelplot from the lattice library

coul <- colorRampPalette(brewer.pal(8, "PuRd"))(25)
levelplot(probability ~ mu * sigma, data = pars, ,
col.regions = coul)
```



From this visualization we see that a credible interval of Zombie IQ is a mean of 42 +/- 10

EXAMPLE

Revisiting the lake: The Temperature in a Normal Lake

Answering the Question: Should I have a beach party?

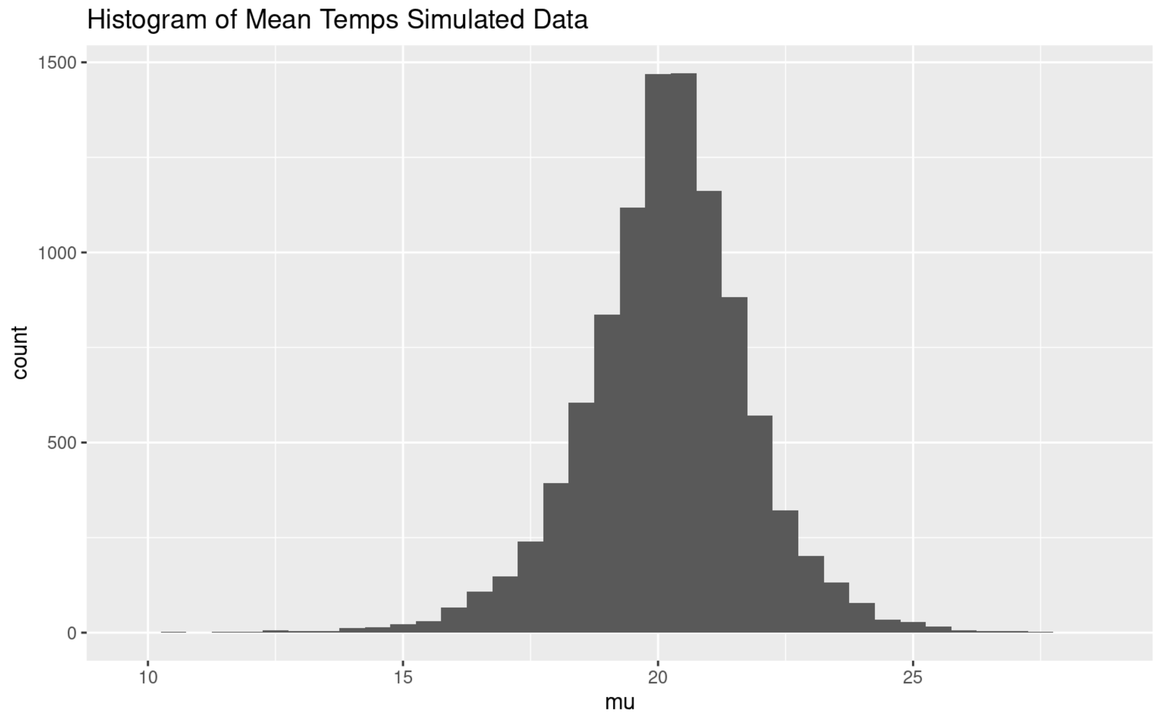
- What's likely the average water temperature on 20th of July?
- What's the probability that the water temperature is going to be 18 or more on the nest 20th of July?

```
temp <- c( 19, 23, 20, 17, 23 )
mu <- seq( 8, 30, by = 0.5 )
sigma <- seq( 0.1, 10, by = 0.3 )
pars <- expand.grid( mu = mu, sigma = sigma )
#the priors
pars$mu_prior <- dnorm( pars$mu, mean = 18, sd = 5 )
pars$sigma_prior <- dunif( pars$sigma, min = 0, max =
10 )
pars$prior <- pars$mu_prior * pars$sigma_prior
for( i in 1:nrow( pars ) ) {
  likelihoods <- dnorm( temp, pars$mu[i],
pars$sigma[i])
  pars$likelihood[i] <- prod( likelihoods )
}

#calculate the posterior probability
pars$probability <- pars$likelihood * pars$prior
pars$probability <- pars$probability / sum(
pars$probability )

summary( pars )
```

EXAMPLE

Revisiting the lake: The Temperature in a Normal Lake

```
sample_indices <- sample( 1:nrow( pars ), size =
10000,
                          replace = TRUE, prob =
pars$probability )
head( sample_indices )

pars_sample <- pars[sample_indices, c( 'mu', 'sigma'
) ]
head( pars_sample )

#plot
ggplot( pars_sample, aes( x = mu ) ) +
  geom_histogram( bins = 38 ) +
  ggtitle( 'Histogram of Mean Temps Simulated Data' )
```



What's a good confidence interval for the temp?

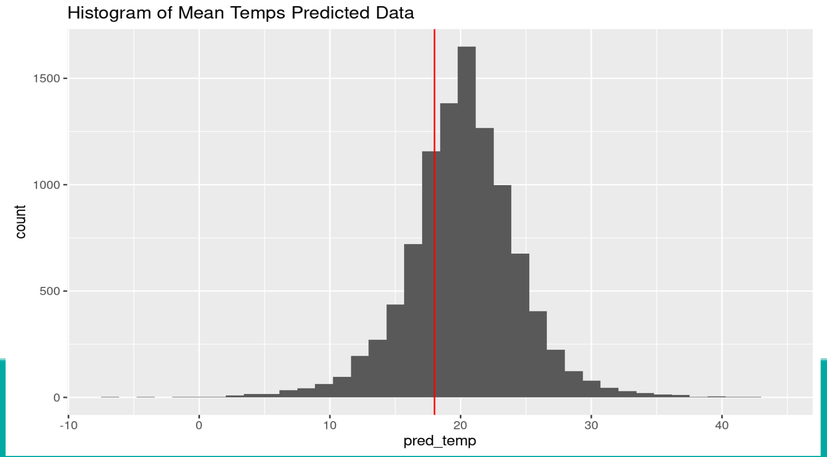
```
quantile( pars_sample$mu, c( 0.05, 0.95 ) )
```

EXAMPLE

Revisiting the lake: The Temperature in a Normal Lake

What is the probability that the temp is 18 or greater on July 20th?

```
#feed in the simulated data
pred_temp <- data.frame(
  'pred_temp' = rnorm( 10000,
                       mean = pars_sample$mu,
                       sd = pars_sample$sigma ) )

ggplot( pred_temp, aes( x = pred_temp ) ) +
  geom_histogram( bins = 38 ) +
  ggtitle( 'Histogram of Mean Temps Predicted Data' )
+
  geom_vline( xintercept = 18, color = 'red' )
```



The probability that the temp is >= 18:

```
sum( pred_temp >= 18 ) /
  length( pred_temp )
```

Summary

- ✓Understand the process in statistical modelling.
- ✓Identify the appropriate model in statistical modelling.
- ✓ Determine the parameter estimation of the model including method of moments, MLE and Bayesian Analysis.
- ✓Understand and use the Bayesian analysis in the statistical modelling.


Thank You!