Statistical Distributions

Saneesh

2023-03-17

Statistical distributions are mathematical models that describe the probability of different outcomes in a random variable. In other words, they describe the way that data is distributed or spread out across different values. Understanding statistical distributions is important in many fields, including statistics, economics, biology, and engineering.

Link

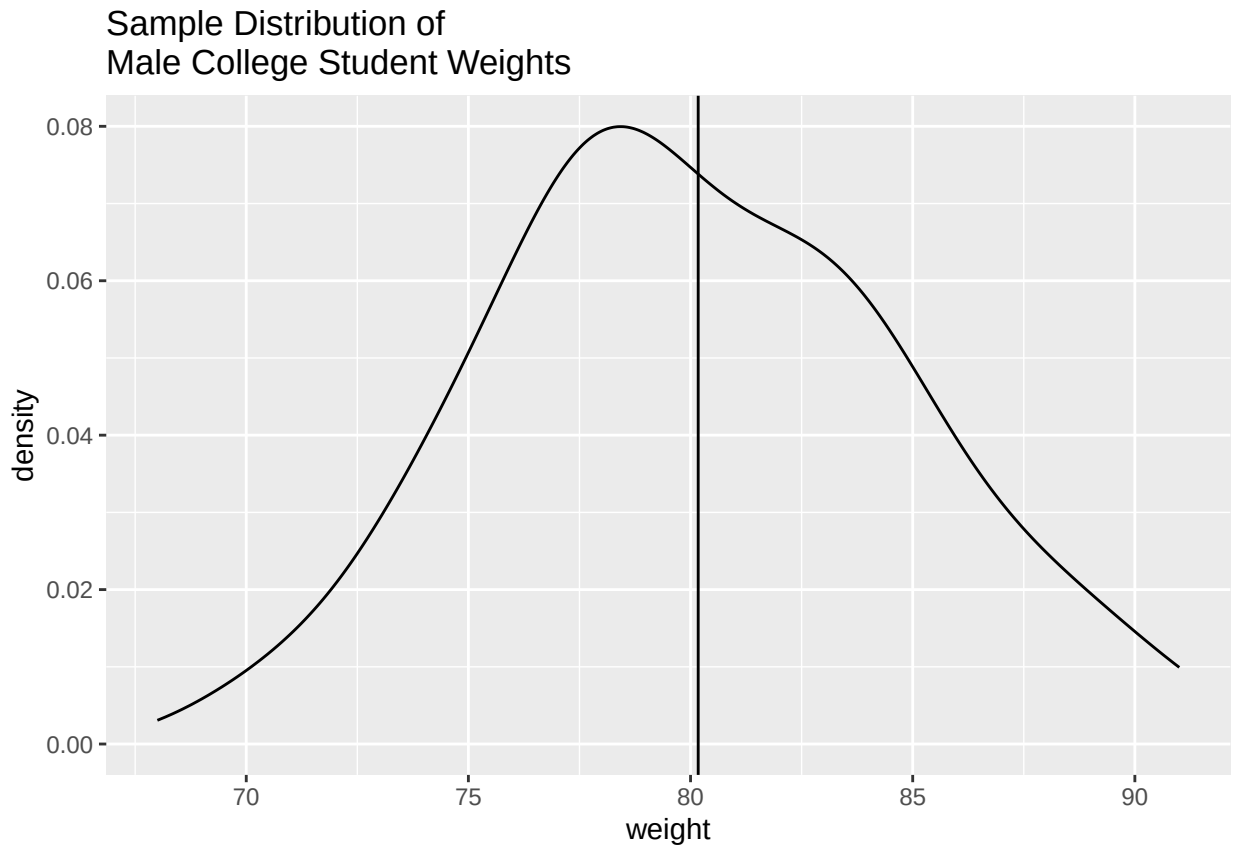
brmsfamilies

Normal distribution:

Also known as the Gaussian distribution, this is a bell-shaped distribution that is symmetrical around the mean. Many natural phenomena follow a normal distribution, such as height, weight, and IQ scores.

For example, if we want to estimate the mean weight of all male college students, we might take a random sample of 50 students and calculate the sample mean and standard deviation. We can then use the normal distribution to construct a confidence interval for the population mean.
To do this, we first calculate the sample mean weight and sample standard deviation, and use these values to calculate the standard error of the mean. We can then construct a confidence interval for the population mean weight using the formula: `sample_mean_weight +/- t*(se_mean_weight)`

```
# Set the random seed for reproducibility

set.seed(123)

# Generate a random sample of 50 weights from a normal distribution with mean
# 80 kg and standard deviation 5 kg
weights <- data.frame(weight = rnorm(n = 50, mean = 80, sd = 5))

# the rnorm() function, which creates values from a normal distribution.

weights %>%
    ggplot(aes(x = weight)) + geom_density() + geom_vline(xintercept = mean(weights$weight)) +
    xlim(68, 91) + ggtitle("Sample Distribution of \nMale College Student Weights")
```




```r
# to construct a confidence interval for the population mean weight Calculate
# the sample mean and standard deviation
mean_weight <- mean(weights$weight)
sd_weight <- sd(weights$weight)
# then calculate the standard error of the mean
se_mean_weight <- sd_weight/sqrt(50)
# Then calculate the critical t-value for a 95% confidence interval with 49
# degrees of freedom (n-1)
t_crit <- qt(0.025, df = 49, lower.tail = FALSE)

conf_int <- c(mean_weight - t_crit * se_mean_weight, mean_weight + t_crit * se_mean_weight)

# Print the results
cat("Sample mean weight:", round(mean_weight, 2), "\n")
```

```
## Sample mean weight: 80.17
```

```r
cat("Standard error of the mean:", round(se_mean_weight, 2), "\n")
```

```
## Standard error of the mean: 0.65
```

```r
cat("Critical t-value for a 95% confidence interval:", round(t_crit, 2), "\n")
```

```
## Critical t-value for a 95% confidence interval: 2.01
```

```r
cat("95% confidence interval for true mean weight:", paste0("(", round(conf_int[1],
    2), ", ", round(conf_int[2], 2), ")"), "\n")
```

```
## 95% confidence interval for true mean weight: (78.86, 81.49)
```

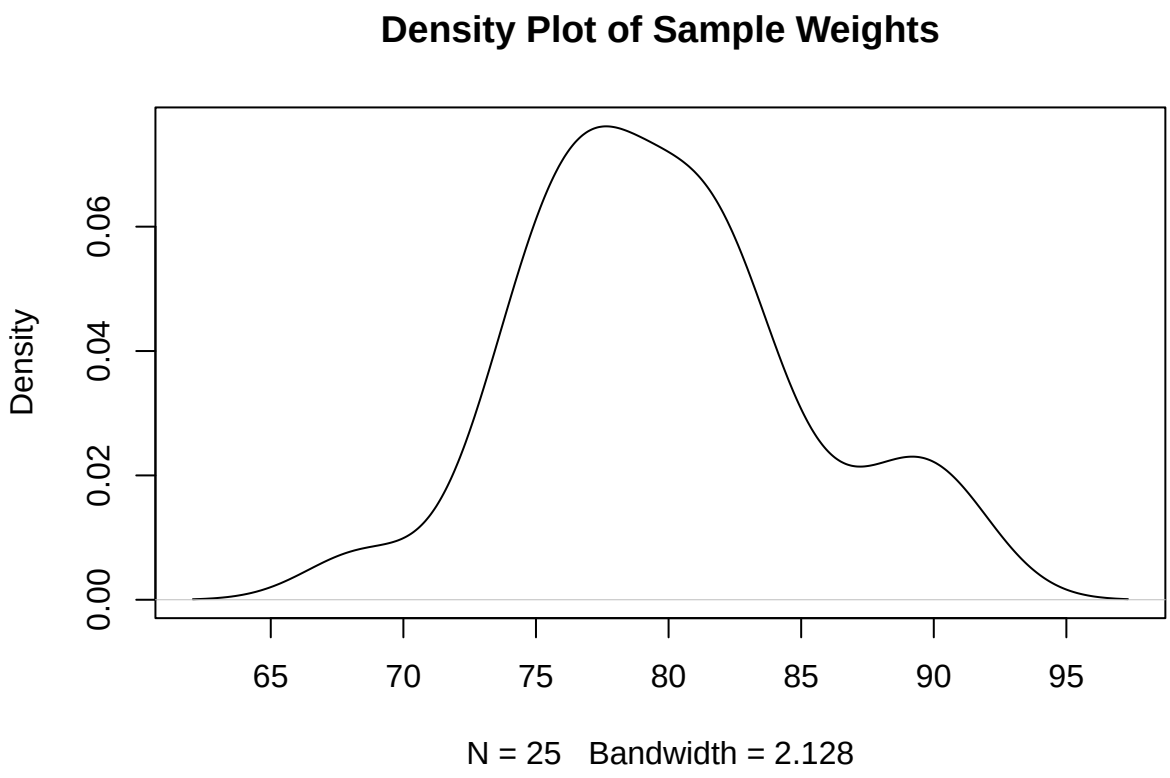
In this example, we generated a random sample of 50 weights from a normal distribution with mean 80 kg and standard deviation 5 kg, and calculated the sample mean weight and standard error of the mean. We then used the `qt()` function to find the critical t-value for a 95% confidence interval with 49 degrees of freedom (n-1). Finally, we used this value to calculate the confidence interval for the true population mean weight using the formula `sample_mean_weight +/- t_crit*se_mean_weight`.

Student's t-distribution:

The Student's t-distribution, often simply referred to as the t-distribution, is a type of probability distribution that arises in statistical inference problems. The t-distribution is similar to the normal distribution, but with heavier tails. It is often used when the sample size is small or when the population variance is unknown. In these situations, the t-distribution provides a way to estimate the population mean with greater accuracy. The shape of the t-distribution depends on a parameter called the degrees of freedom (df). As the degrees of freedom increase, the t-distribution approaches the normal distribution. When the degrees of freedom are small (less than about 30), the t-distribution has fatter tails than the normal distribution, which means that extreme values are more likely to occur. The t-distribution is commonly used in hypothesis testing and confidence interval estimation. For example, if we want to estimate the mean weight of all male college students, we might take a random sample of 25 students and calculate the sample mean and standard deviation. We can then use the t-distribution to construct a confidence interval for the population mean, or to test whether the population mean is equal to a certain value.

Suppose we want to test whether the mean weight of all adult men in a certain city is 80 kilograms, based on a random sample of 25 men. We can use the t-distribution to construct a confidence interval for the true population mean weight and to test whether it differs significantly from 80 kilograms.

```r
# Set the random seed for reproducibility
set.seed(123)

# Generate a random sample of 25 weights from a Student's t-distribution with
# 24 degrees of freedom and mean 80 kg and standard deviation 5 kg
sample_weights <- rt(n = 25, df = 24, ncp = 0) * 5 + 80


# Calculate the sample mean and standard deviation
sample_mean_weight <- mean(sample_weights)
sample_sd_weight <- sd(sample_weights)

# Plot the density of the sample weights
plot(density(sample_weights), main = "Density Plot of Sample Weights")
```




```
# Calculate the standard error of the mean
se_mean_weight <- sample_sd_weight/sqrt(25)

# Calculate the t-statistic and corresponding p-value for a one-sample t-test
# against the null hypothesis that the true population mean weight is 180 lbs
t_stat <- (sample_mean_weight - 180)/se_mean_weight
p_value <- 2 * pt(abs(t_stat), df = 24, lower.tail = FALSE)  # two-sided test

# Calculate a 95% confidence interval for the true population mean weight
conf_int <- sample_mean_weight + qt(c(0.025, 0.975), df = 24) * se_mean_weight

# Print the results
cat("Sample mean weight:", round(sample_mean_weight, 2), "\n")
```

```
## Sample mean weight: 79.8
```

```
cat("Standard error of the mean:", round(se_mean_weight, 2), "\n")
```

```
## Standard error of the mean: 1.06
```

```
cat("t-statistic:", round(t_stat, 2), "\n")
```

```
## t-statistic: -94.14
```

```r
cat("p-value:", format(p_value, scientific = FALSE), "\n")
```

```
## p-value: 0.000000000000000000000000000002429415
```

```r
cat("95% confidence interval for true mean weight:", paste0("(", round(conf_int[1],
    2), ", ", round(conf_int[2], 2), ")"), "\n")
```

```
## 95% confidence interval for true mean weight: (77.6, 82)
```

The 95% confidence interval (77.6, 82) means that, based on the sample of 25 weights, we are 95% confident that the true population mean weight lies between 77.6 and 82 lbs.

In other words, if we were to repeat this sampling process many times and calculate a 95% confidence interval for the true population mean weight from each sample, we would expect 95% of these intervals to contain the true population mean weight, and 5% of them to not contain the true population mean weight.

In this example, we generated a random sample of 25 weights from a normal distribution with mean 80 kilograms and standard deviation 5 kilograms, and assumed that we did not know the true population standard deviation. We then calculated the sample mean weight, standard deviation, and standard error of the mean. We used these values to calculate the t-statistic and p-value for a one-sample t-test against the null hypothesis that the true population mean weight is 80 kilograms. We also calculated a 95% confidence interval for the true population mean weight using the qt() function, which calculates the quantiles of the t-distribution.

There are several situations in ecology where one can observe a Student's t-distribution:
When sample sizes are small: In many ecological studies, sample sizes can be small, especially when dealing with rare or endangered species or when working in remote areas. In these cases, the sample size may be too small for the Central Limit Theorem to apply, and the distribution of the sample mean may be better approximated by a Student's t-distribution.

When working with non-normal data: Many ecological variables, such as population densities or species abundances, are count data or non-normally distributed. In these cases, the sample mean may be better approximated by a Student's t-distribution.

When comparing two groups: In ecological studies, it is often necessary to compare two groups, such as the effects of a treatment versus a control group. If the sample sizes for the two groups are small, the distributions of the sample means may be better approximated by Student's t-distributions.

When working with correlated data: In some ecological studies, data may be correlated, such as when studying the relationships between species or populations in a community. In these cases, the use of a Student's t-distribution can account for the correlation and provide a more accurate estimate of the true population mean.

In the context of linear regression, the t-distribution can be used as a basis for constructing confidence intervals and hypothesis tests for the regression coefficients. When there are outliers in the data, the assumption of normality of the errors may be violated, which can lead to biased or inefficient estimates of the regression coefficients. In this case, the t-distribution can be used as a more robust alternative to the normal distribution.

Skew-normal

The skew-normal distribution is a continuous probability distribution that can be used to model data that are asymmetric or skewed, but still have a bell-shaped appearance. It is a generalization of the normal distribution that allows for the skewness parameter to control the direction and degree of skewness.

In real life, the skew-normal distribution can be used to model various phenomena that exhibit asymmetry or skewness in their distribution, such as financial returns, response times, and reaction rates. For example, in finance, stock returns are often found to be asymmetric and have fat tails, which can be modeled by a skew-normal distribution.

To confirm the distribution, you can plot the density of the sample data using the `density` function and compare it to the density of a theoretical skew normal distribution with the same parameters.

```
library(sn)
```

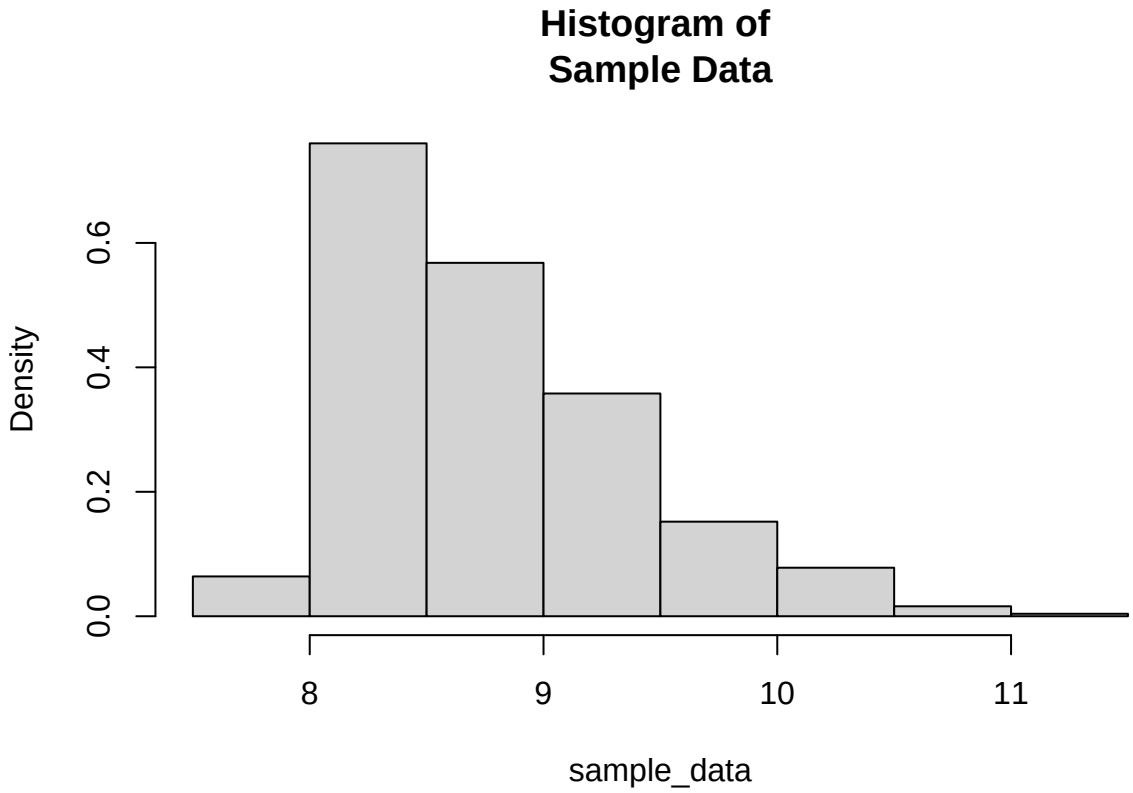
```
## Loading required package: stats4
```

```
##
## Attaching package: 'sn'
```

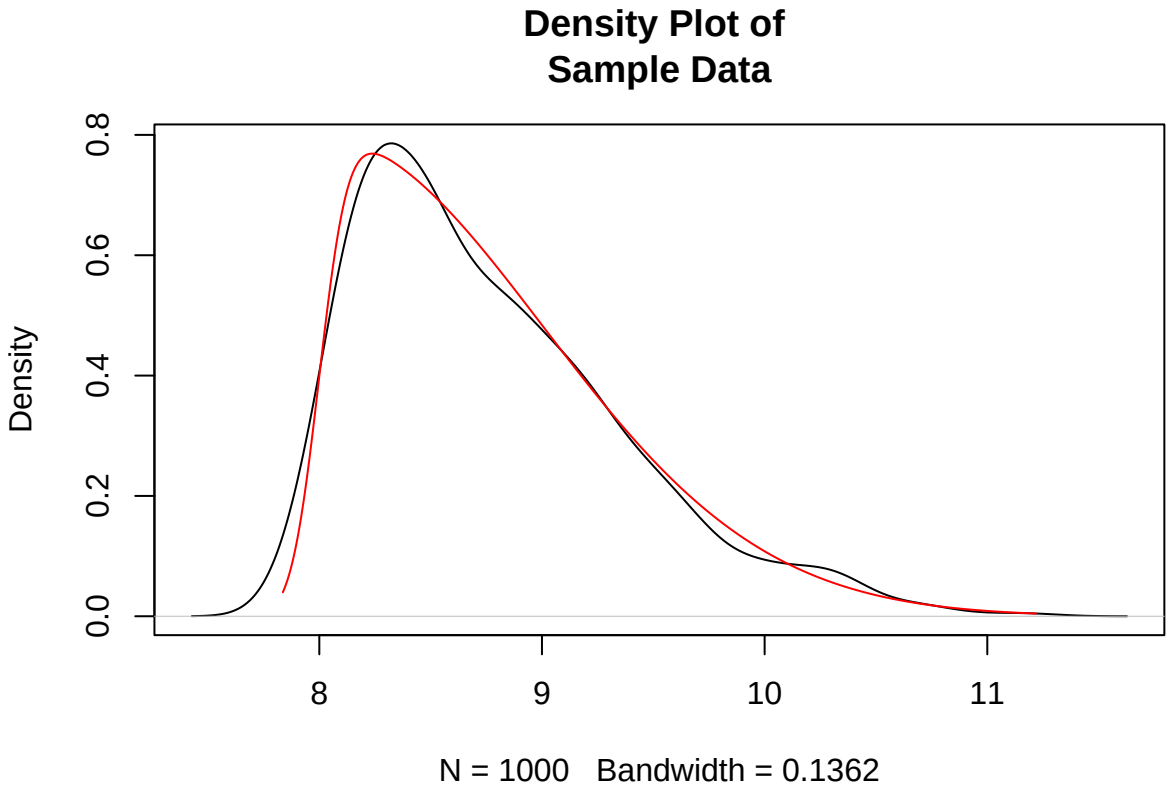
```
## The following object is masked from 'package:lubridate':
##
##     dst
```

```
## The following object is masked from 'package:stats':
##
##     sd
```

```
set.seed(123)
sample_data <- rsn(n = 1000, xi = 8, alpha = 10, omega = 1)

# Plot the sample data
hist(sample_data, freq = FALSE, main = "Histogram of \nSample Data")
```




```
plot(density(sample_data), main = "Density Plot of \nSample Data")



x <- seq(min(sample_data), max(sample_data), length.out = 1000)
lines(x, dsn(x, xi = 8, alpha = 10, omega = 1), col = "red")
```




```
# If the density plot of the sample data closely matches the density of a
# theoretical skew normal distribution, then it is likely that the sample data
# follows a skew normal distribution.
```

`xi`: the location parameter, which determines the location of the peak of the distribution.
`alpha`: the skewness parameter, which determines the degree of skewness in the distribution. A value of 0 indicates a symmetric distribution, while positive and negative values indicate right and left skewness, respectively.
`omega`: the scale parameter, which determines the spread of the distribution. A larger value of omega indicates a wider distribution.

```
# Use the bootstrap method to construct a 95% confidence interval for the
# population mean
n_bootstrap <- 1000
boot_means <- replicate(n_bootstrap, mean(sample_data[sample(1:length(sample_data),
    replace = TRUE)]))
boot_ci <- quantile(boot_means, c(0.025, 0.975))

# Print the results
cat("Sample mean:", round(mean(sample_data), 2), "\n")
```

```
## Sample mean: 8.77
```

```r
cat("Bootstrap 95% confidence interval:", paste0("(", round(boot_ci[1], 2), ", ",
    round(boot_ci[2], 2), ")"), "\n")
```

```
## Bootstrap 95% confidence interval: (8.74, 8.81)
```

Workflow: Building a bayesian linear regression model

Here are two real-life examples to understand the difference between a model with a main effect and a model with an interaction effect.

Let's consider an example where we want to predict the sales of a product based on its price and the type of advertisement used to promote it.

1. Main Effect Model:

If we use a main effect model, we would assume that the effect of `price` and `advertisement type` on sales are independent of each other. We can represent this model using the formula:
`Sales ~ Price + Advertisement Type`
In this model, we assume that the effect of `price` and `advertisement type` on sales are additive and do not depend on each other. For example, if we increase the price of the product by Rupee 1, we would expect the sales to decrease by a certain amount, regardless of the type of advertisement used.

```r
library(brms)
```

```
## Loading required package: Rcpp
```

```
## Loading 'brms' package (version 2.18.0). Useful instructions
## can be found by typing help('brms'). A more detailed introduction
## to the package is available through vignette('brms_overview').
```

```
##
## Attaching package: 'brms'
```

```
## The following object is masked from 'package:stats':
##
##     ar
```

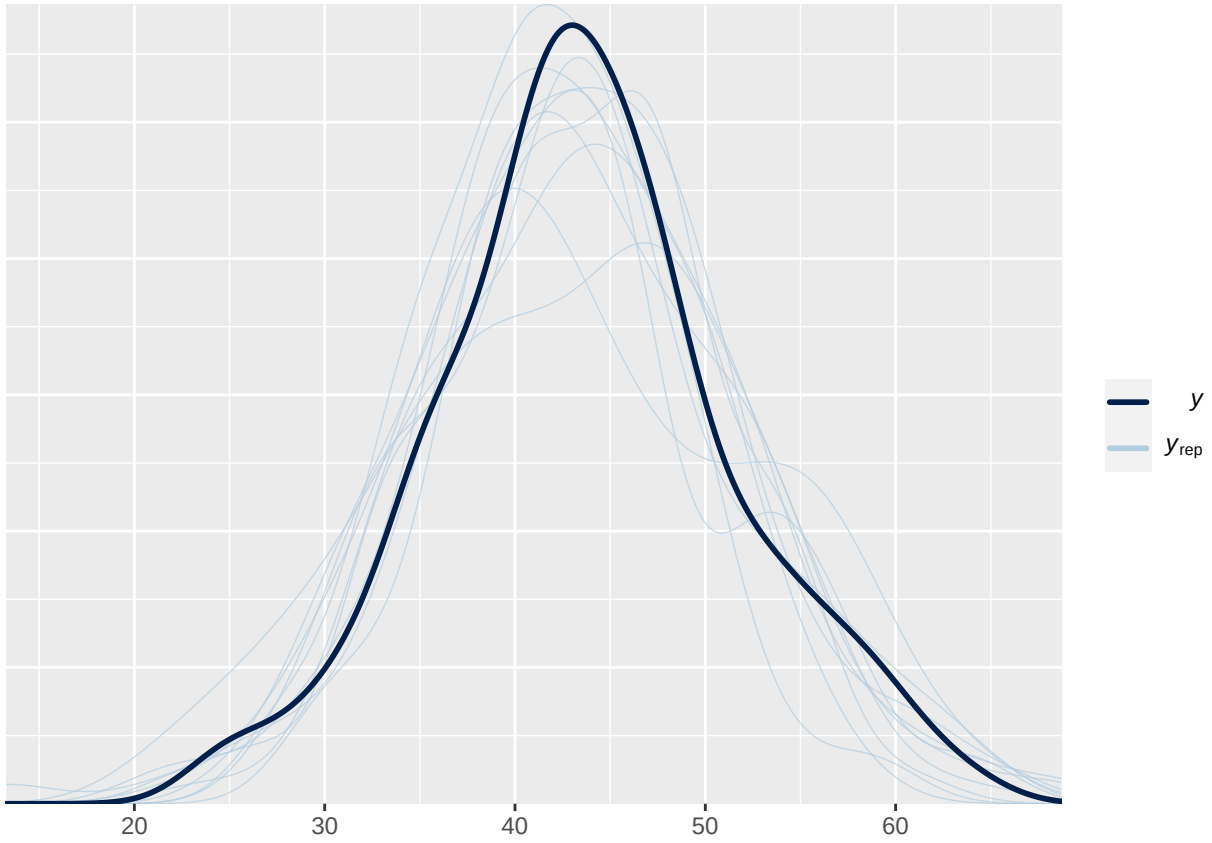
```r
# Set seed for reproducibility
set.seed(1980)

# Simulate data for model 1: Sales ~ Price + Advertisement Type Simulate
# predictor variables
price <- rnorm(100, mean = 50, sd = 10)
ad_type <- factor(sample(c("TV", "Radio", "Online"), 100, replace = TRUE))

# Simulate response variable
sales <- 20 + 0.5 * price - 5 * (ad_type == "Radio") + rnorm(100, mean = 0, sd = 5)

# Create a data frame with simulated data
data1 <- data.frame(sales = sales, price = price, ad_type = ad_type)
```

```r
# mainmodel <- brm(sales ~ price + ad_type, family = gaussian(), data = data1)
# save(mainmodel, file= 'mainmodel.Rdata')

load("mainmodel.Rdata")

summary(mainmodel)
```

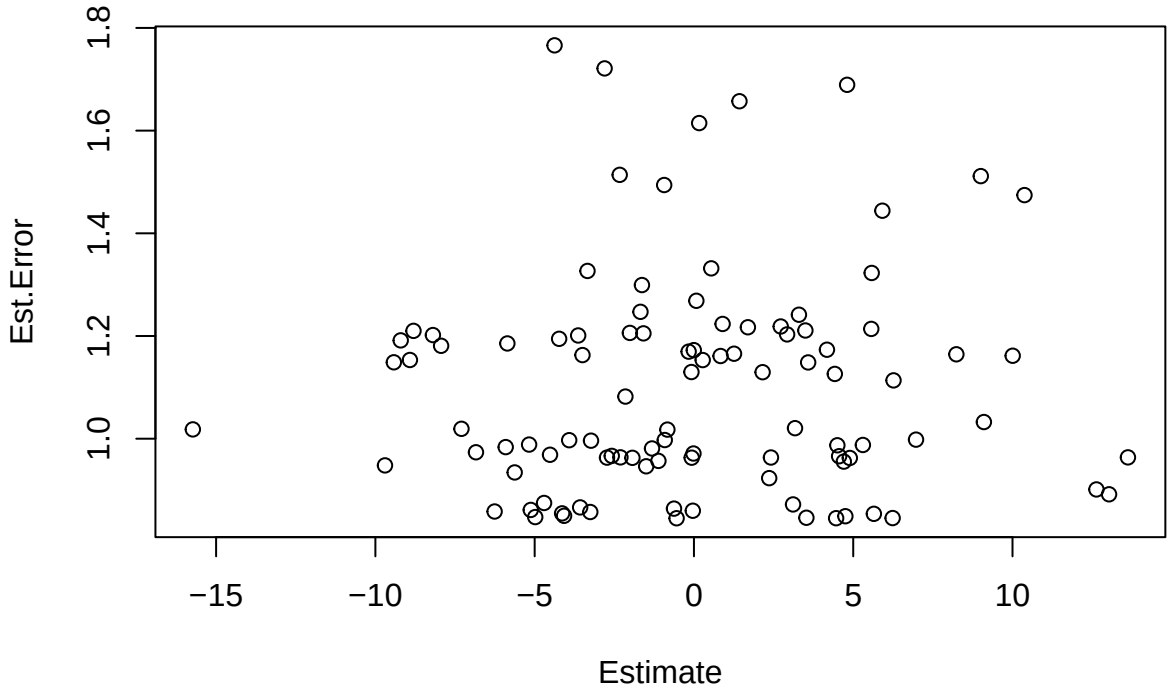
```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: sales ~ price + ad_type
##    Data: data1 (Number of observations: 100)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Population-Level Effects:
##              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept       20.73      3.53    13.73    27.60 1.00     4416     2545
## price            0.50      0.06     0.38     0.64 1.00     4759     2822
## ad_typeRadio    -6.05      1.51    -9.18    -3.12 1.00     3438     2561
## ad_typeTV       -1.76      1.45    -4.56     1.06 1.00     3791     2801
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     5.58      0.41     4.85     6.48 1.00     4270     2488
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```r
pp_check(mainmodel)
```

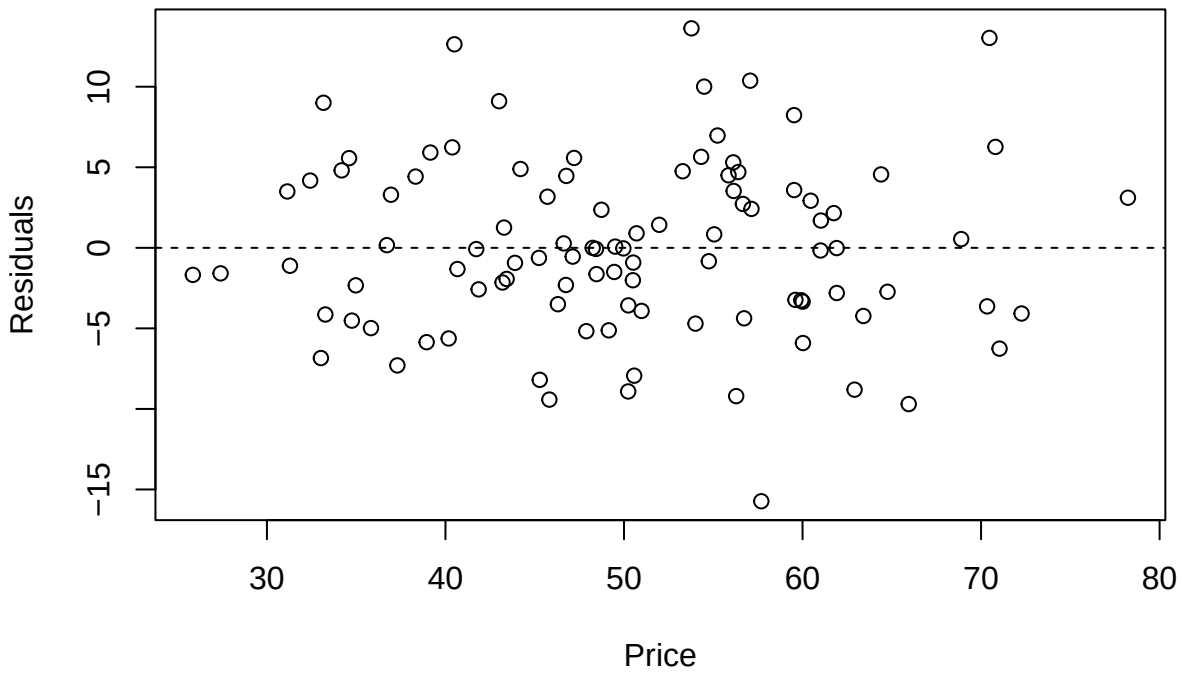
```
## Using 10 posterior draws for ppc type 'dens_overlay' by default.
```



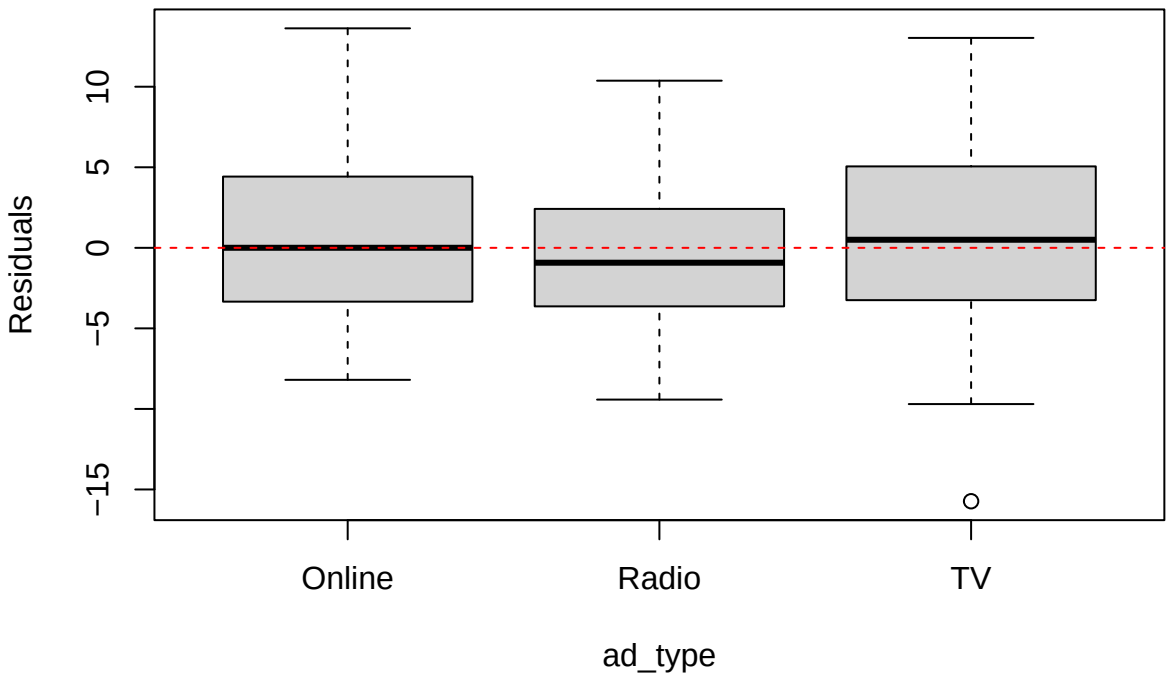

```
# Calculate the posterior distribution of the model residuals, also known as
# the 'posterior predictive distribution of the residuals'. These residuals are
# the difference between the observed response (sales) and predictive values
# from the model.

# Plot the residuals

residual <- residuals(mainmodel)
plot(residual)  # posterior predictive residuals against the predicted values
```



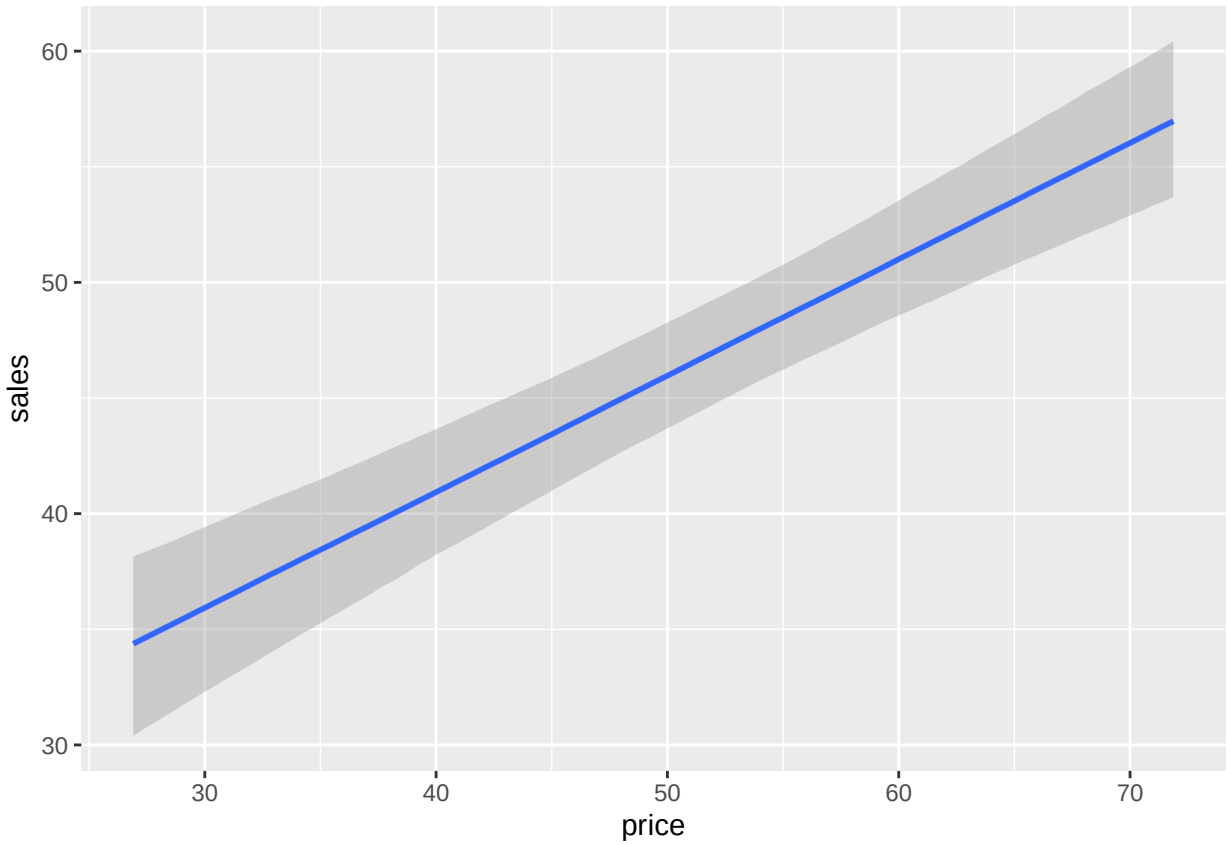

```
# Plot the residuals against price
plot.residuals <- cbind(data1, residuals(mainmodel))

plot(plot.residuals$price, plot.residuals$Estimate, xlab = "Price", ylab = "Residuals")
abline(h = 0, lty = 2)  # to visually evaluate how much the residuals deviate from zero, and whether th
```



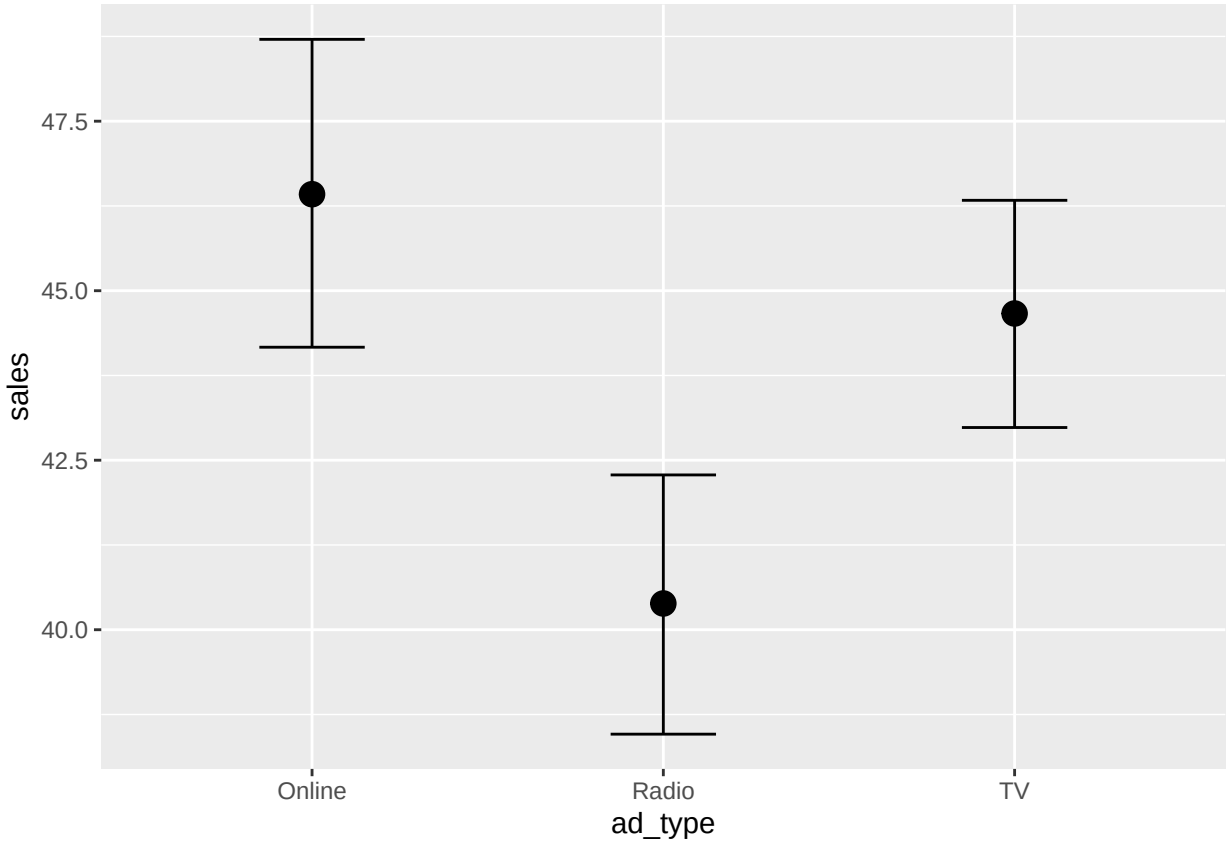

```
# Plot the residuals against ad_type
plot(plot.residuals$ad_type, plot.residuals$Estimate, xlab = "ad_type", ylab = "Residuals")
abline(h = 0, lty = 2, col = "red")
```




```
# Checking the model residuals is an important step in evaluating the goodness
# of fit of a model, as it can help you identify areas where the model is not
# performing well. For example, if the residuals are not normally distributed
# or show patterns in their distribution, this may indicate that the model is
# not capturing some important feature of the data or that there are other
# variables that should be included in the model.


conditional_effects(mainmodel, effects = "price")
```




```
conditional_effects(mainmodel, effects = "ad_type")
```



According to the output, the estimated coefficient for the price variable is 0.51 with a standard error of 0.06. This suggests that for a one unit increase in price, there is a corresponding 0.51 unit increase in sales, on average.

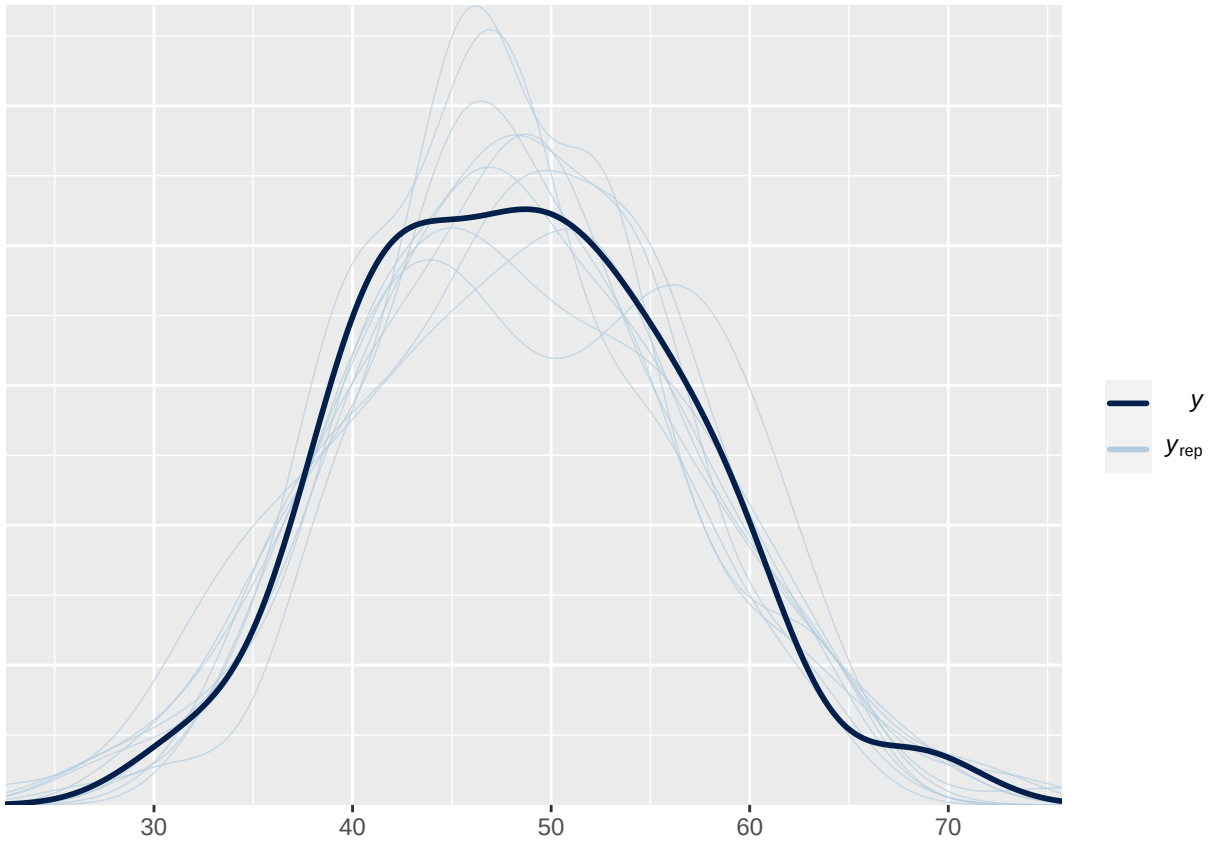
It is also important to note that there are two categorical variables in the model, ad_typeRadio and ad_typeTV. The output shows that compared to the reference level (which is not shown in the output), the estimated coefficient for ad_typeRadio is -6.06, and the estimated coefficient for ad_typeTV is -1.80. This suggests that on average, when ad_type is Radio, there is a decrease of 6.06 units in sales, and when ad_type is TV, there is a decrease of 1.80 units in sales, relative to the reference level.

2. Interaction Effect Model:
   Now let's consider a scenario where the effect of `price` on sales depends on the `type of advertisement` used. In this case, we would use an interaction effect model. We can represent this model using the formula:
   `Sales ~ Price * Advertisement Type`
   In this model, we assume that the effect of `price` on sales is not constant across different types of advertisement. For example, if we increase the price of the product by Rupee 1, the effect on sales may be different depending on the type of advertisement used. If we use a more persuasive advertisement, the effect of price on sales may be less severe compared to a less persuasive advertisement.

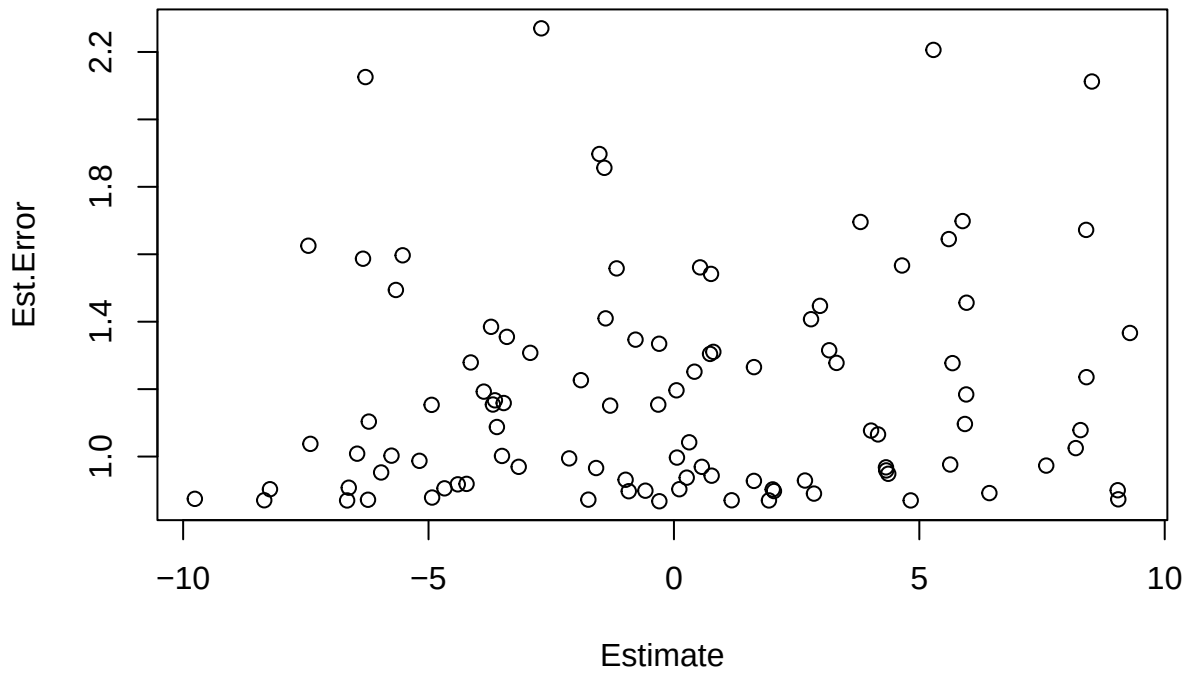
```
set.seed(1980)
# Simulate data for model 2: Sales ~ Price * Advertisement Type Simulate
# predictor variables
price <- rnorm(100, mean = 50, sd = 10)
ad_type <- factor(sample(c("TV", "Radio", "Online"), 100, replace = TRUE))

# Simulate response variable
```

```
sales <- 20 + 0.5 * price + 2 * (ad_type == "TV") + 3 * (ad_type == "Online") + (0.1 *
    price - 0.5) * (ad_type == "Radio") + rnorm(100, mean = 0, sd = 5)

# Create a data frame with simulated data
data2 <- data.frame(sales = sales, price = price, ad_type = ad_type)

# intermodel <- brm(sales ~ price * ad_type, family = gaussian(), data = data2)
# save(intermodel, file= 'intermodel.Rdata')

load("intermodel.Rdata")
pp_check(intermodel)
```

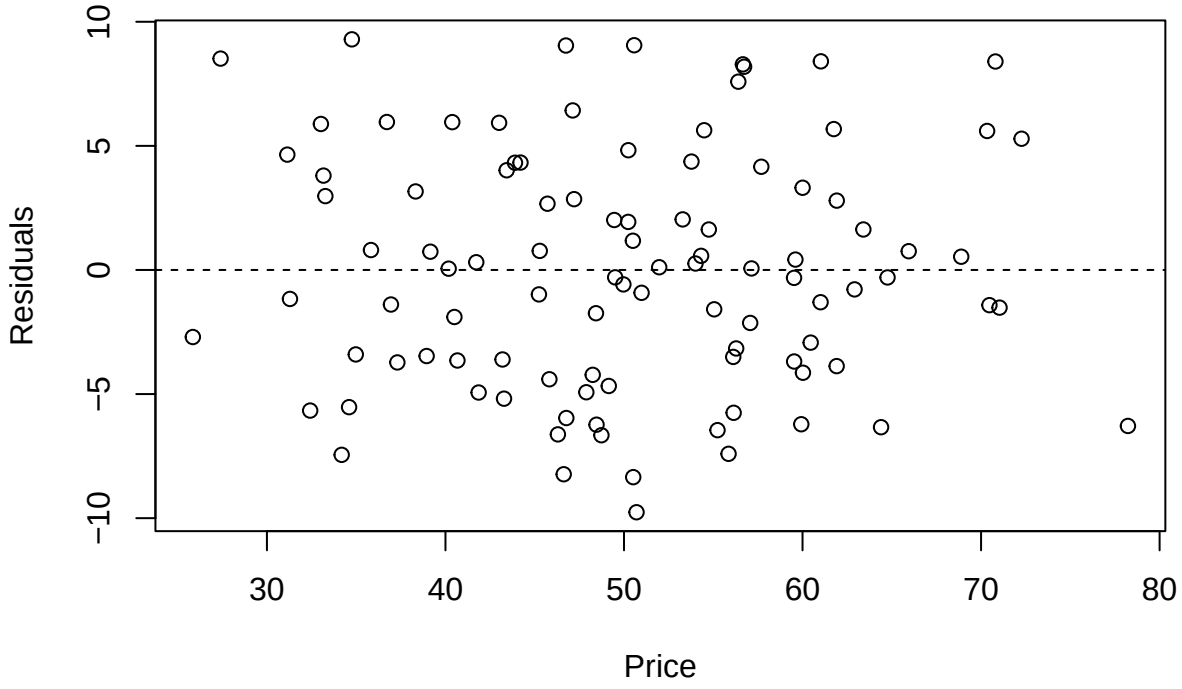
```
## Using 10 posterior draws for ppc type 'dens_overlay' by default.
```



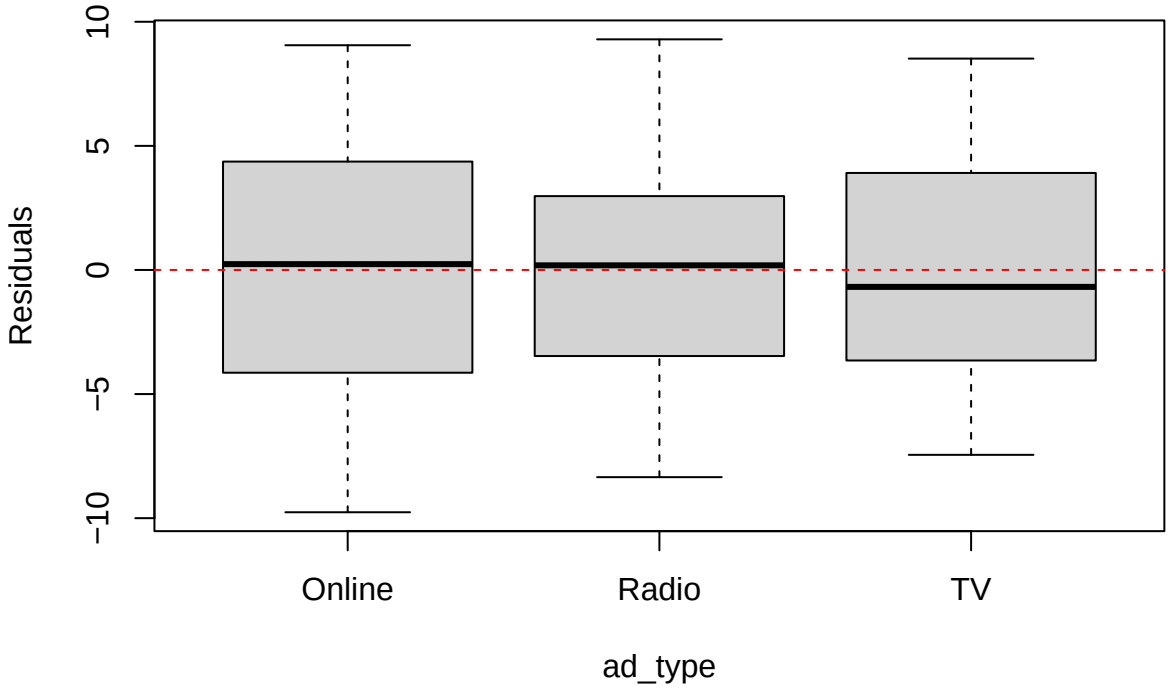

```
# Plot the residuals

residual <- residuals(intermodel)
plot(residual)  # posterior predictive residuals against the predicted values
```




```
# Plot the residuals against price
plot.residuals <- cbind(data2, residuals(intermodel))

plot(plot.residuals$price, plot.residuals$Estimate, xlab = "Price", ylab = "Residuals")
abline(h = 0, lty = 2)  # to visually evaluate how much the residuals deviate from zero, and whether th
```



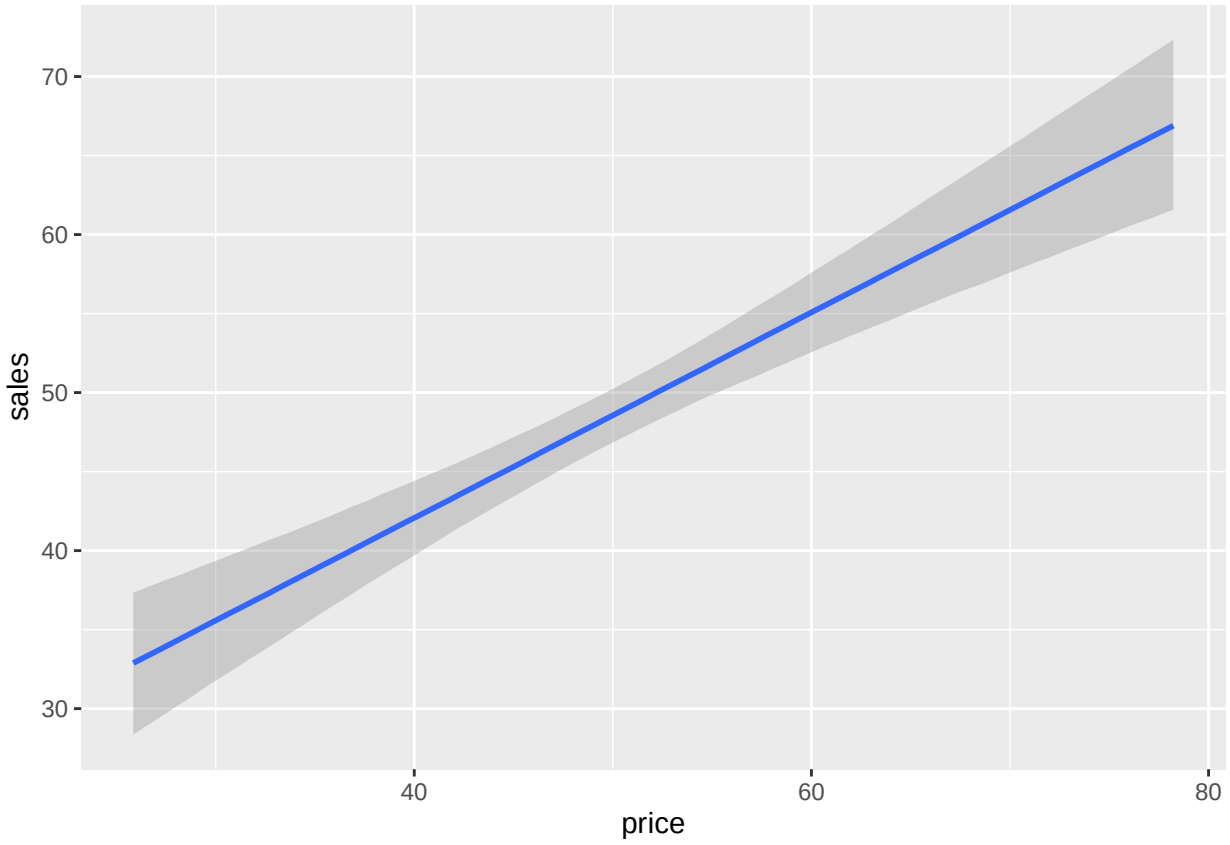

```
# Plot the residuals against ad_type
plot(plot.residuals$ad_type, plot.residuals$Estimate, xlab = "ad_type", ylab = "Residuals")
abline(h = 0, lty = 2, col = "red")
```

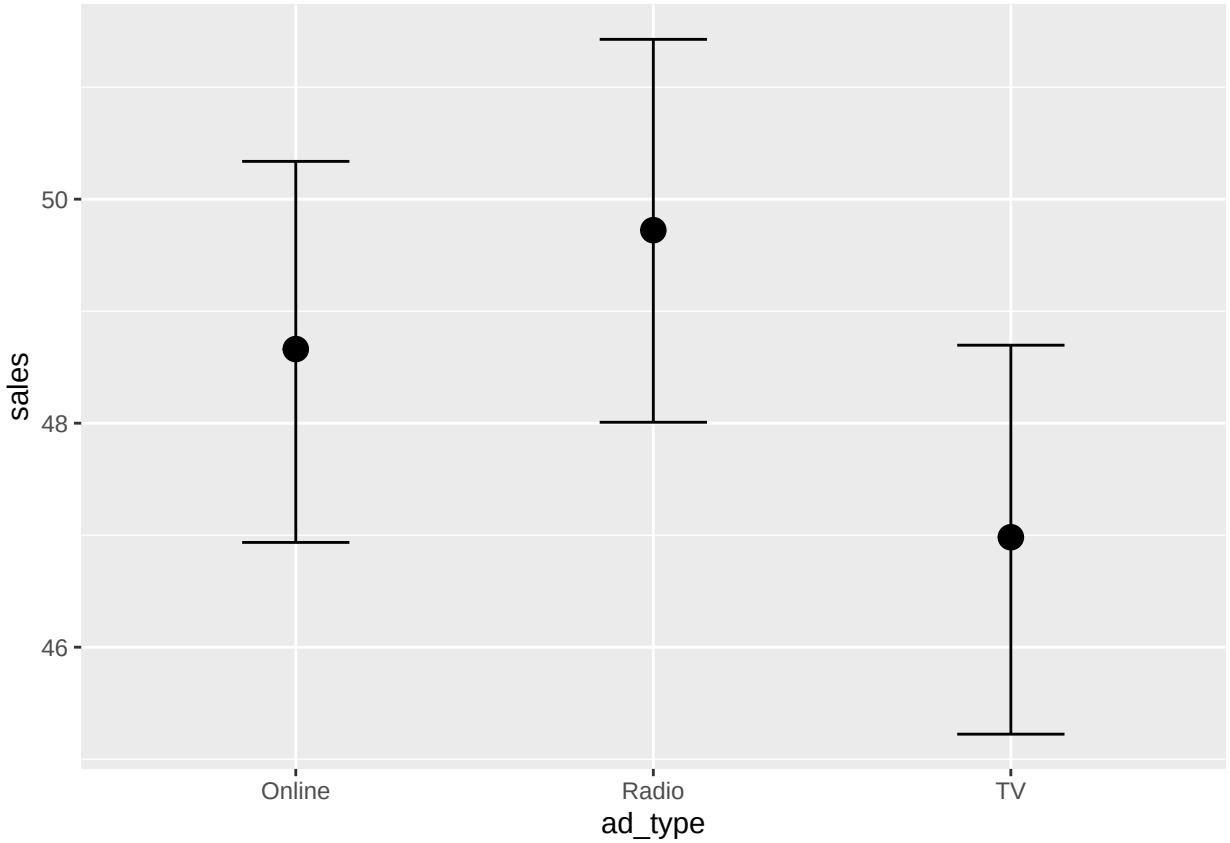


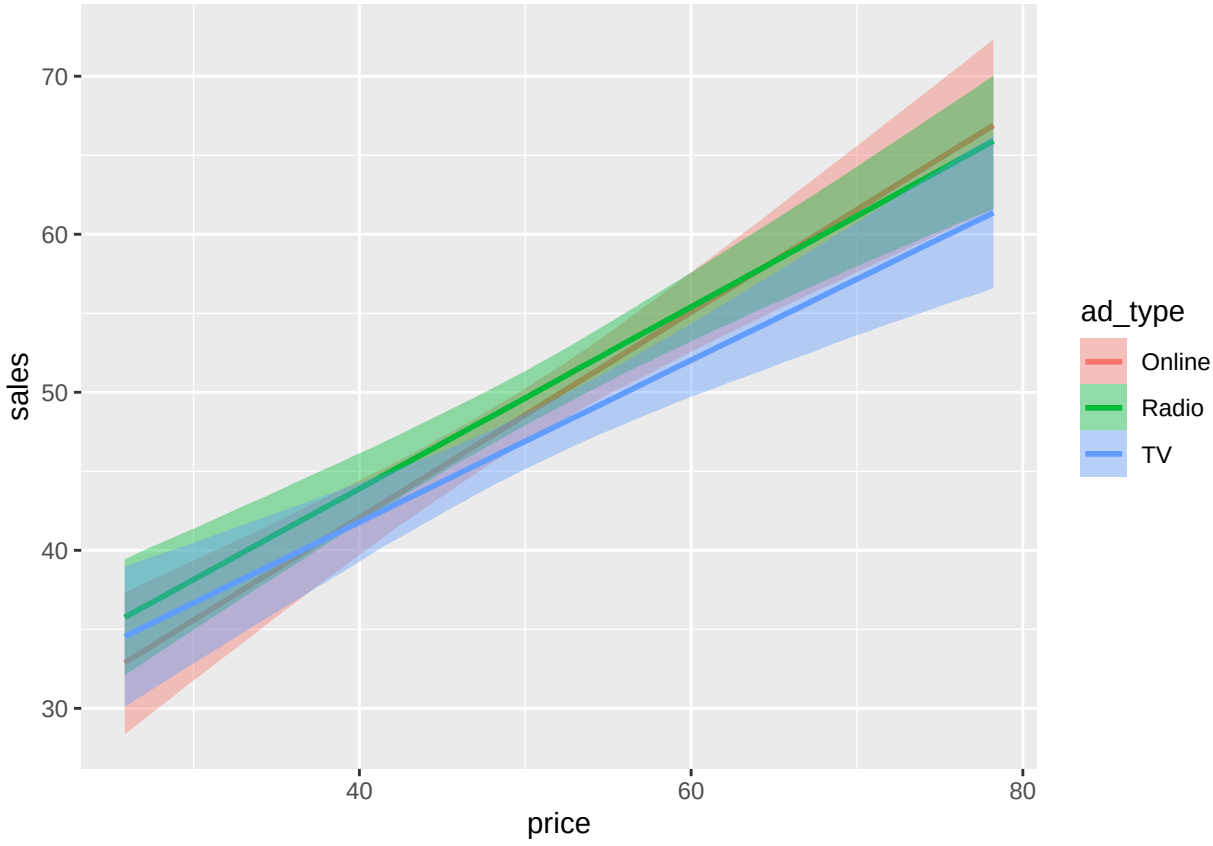

```
summary(intermodel)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: sales ~ price * ad_type
##    Data: data2 (Number of observations: 100)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Population-Level Effects:
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept            16.06      4.48     7.15    24.93 1.00     1182     1557
## price                 0.65      0.09     0.47     0.83 1.00     1170     1649
## ad_typeRadio          4.84      5.73    -6.55    16.77 1.00     1121     1564
## ad_typeTV             5.20      6.09    -6.82    17.59 1.00     1269     1853
## price:ad_typeRadio   -0.07      0.11    -0.31     0.15 1.00     1092     1634
## price:ad_typeTV      -0.14      0.12    -0.37     0.10 1.00     1218     1739
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     4.96      0.37     4.29     5.77 1.00     2971     2550
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

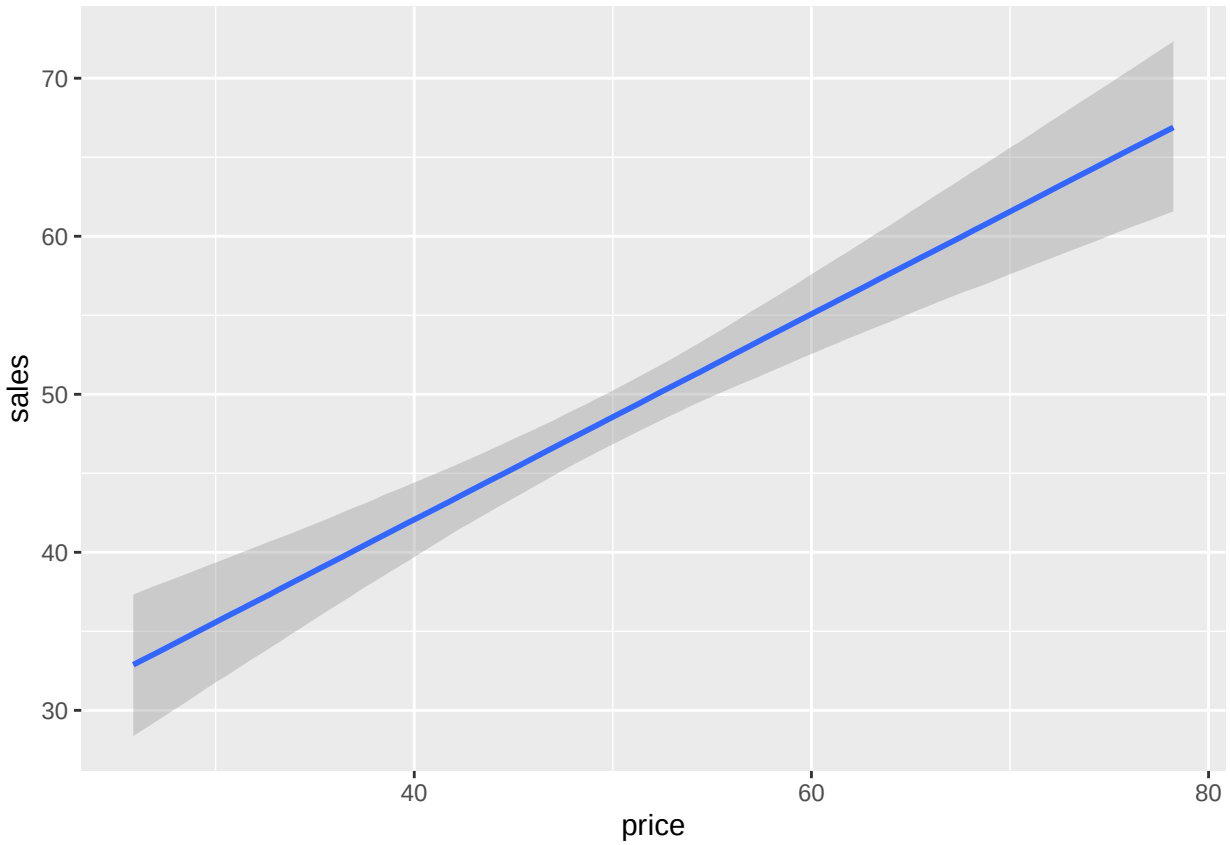
```
conditional_effects(intermodel)
```



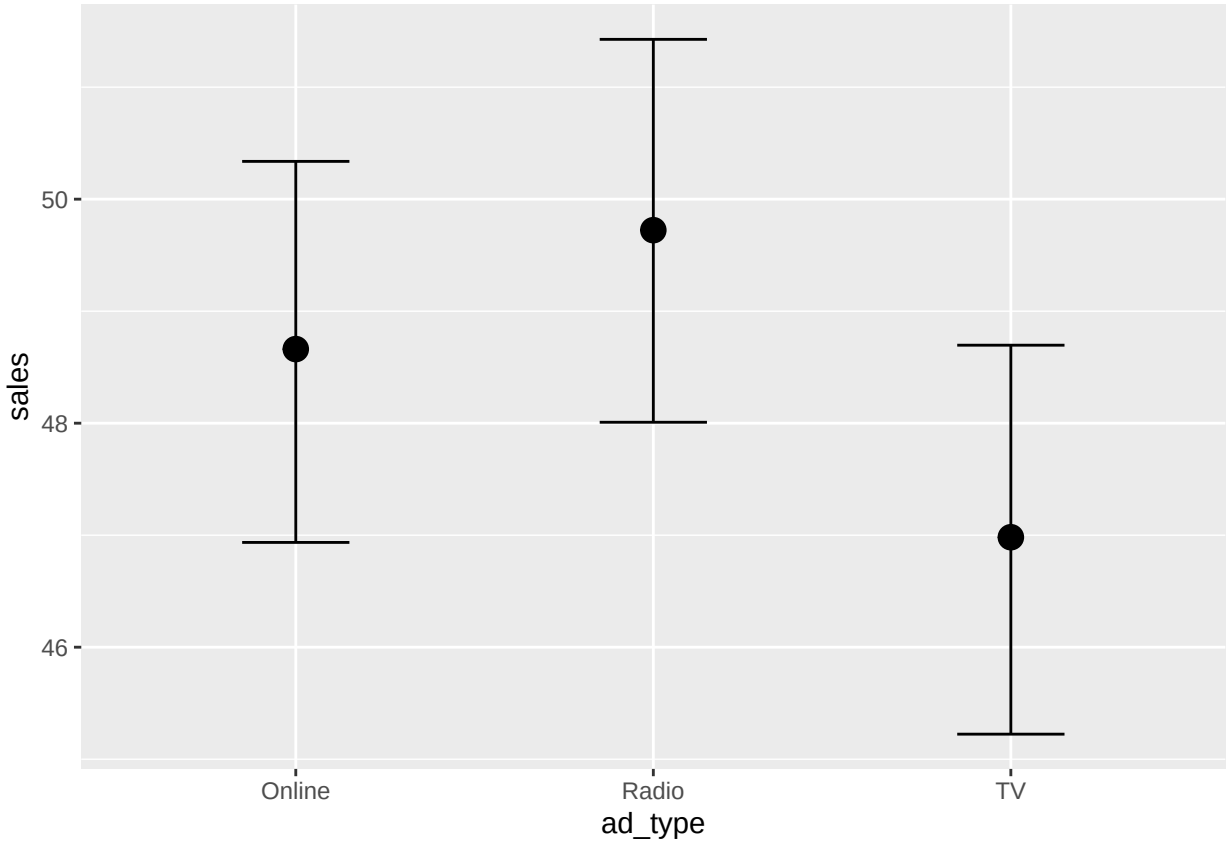

50
48
46
sales
Online
Radio
TV
ad_type




```
conditional_effects(intermodel, effects = "price")
```




```
conditional_effects(intermodel, effects = "ad_type")
```



According to the output, the estimated coefficient for the price variable is 0.65 with a standard error of 0.09. This suggests that for a one unit increase in price, there is a corresponding 0.65 unit increase in sales, on average.

Additionally, the output shows the interaction effects between price and the two levels of ad_type, Radio and TV. The estimates for these interaction terms are -0.08 and -0.14, respectively. These values indicate that the effect of price on sales depends on the level of ad_type, and that the effect is weaker (more negative) for ad_typeRadio and ad_typeTV. However, it is important to note that the confidence intervals for these interaction terms include 0, indicating that these effects may not be statistically significant.

To summarize, a main effect model assumes that the effect of each predictor variable is independent of the other predictor variables, while an interaction effect model assumes that the effect of one predictor variable on the response variable depends on the level of another predictor variable.

Bayesian workflow

1. Build the model.

`model_1 <- brm(x ~ y * z, family = gaussian(), data = data)`. Specify the response variable `y`, predictor variables `y, z`, and any necessary model options (e.g., family, priors, etc.).

2. Save the model.
   `save(model_1, file= 'model_1.Rdata')`

3. `load('model_1.Rdata')`

4. Check model fit.
   Carry out posterior predictive checks using the pp_check() function to assess the fit of the model. This involves simulating data from the posterior predictive distribution and comparing these simulations to the observed data. You can create a variety of plots to assess different aspects of the fit, such as histograms of the observed data overlaid with simulated data, scatterplots of observed vs. simulated data, and plots of the residuals.

```
pp_check(model_1)
```

5. Check residuals

`residual <- residuals(model_1) plot(residual)` # posterior predictive residuals against the predicted values

Plot the residuals against y plot.residuals <- cbind(data, residuals(model_1))

plot(plot.residuals$y, plot.residuals$Estimate, xlab = "y", ylab = "Residuals") abline(h=0, lty= 2) # to visually evaluate how much the residuals deviate from zero, and whether there are any patterns in the residuals.

6. Plot the residuals against z plot(plot.residuals$z, plot.residuals$Estimate, xlab = "z", ylab = "Residuals") abline(h=0, lty= 2, col= 'red')

summary(model_1) conditional_effects(model_1) conditional_effects(model_1, effects = 'x') conditional_effects(intermodel, effects = 'z')

Species rarefaction curve

A species rarefaction curve is a graphical representation of the relationship between the number of individuals sampled and the number of species observed. It is commonly used in ecology to estimate the total number of species in a given area, based on a limited sample of individuals.

To use a species rarefaction curve, you would first sample a certain number of individuals from the area of interest. In this case, let's say that you captured 25 individuals from a list of species present in a plot, where the numbers of individuals of each species are as follows: (a=20, b=10, c=20, d=20, e=10, f=10, g=10).

Next, you would use these data to calculate the observed number of species. One way to do this is to count the number of different species that are represented in the sample. For example, if you captured 5 individuals of species a, 3 individuals of species b, and 2 individuals each of species c, d, and e, then you would have observed 5 different species in your sample.

You would repeat this process for multiple sample sizes (i.e., capturing different numbers of individuals) and plot the number of observed species as a function of the number of individuals sampled. This results in a rarefaction curve that can be used to estimate the number of species present in the entire area.

To estimate the total number of species present, you would extrapolate the curve to the point where it levels off, which represents the estimated number of species that would be observed if an infinite number of individuals were sampled. However, it's important to note that this extrapolation should be done with caution, as there are many factors that can influence the shape of the curve and the accuracy of the estimate.

```
sample1 <- data.frame(species = c("a", "b", "c", "d", "e"), count = c(5, 3, 2, 2,
    2))
sample2 <- data.frame(species = c("a", "c", "f", "g", "h"), count = c(2, 1, 2, 2,
    2))
```