

Sampling posteriors with Bayesian Variable Importance

August Arnstad

05.01.2024

Contents

GENERAL SETUP	1
INSTALLING THE PACKAGE	1
SIMULATE DATA	1
USAGE	2
PLOTTING	3
GELMAN R2	8

GENERAL SETUP

First, we set up the necessary libraries and configure the environment for our analysis. This includes loading essential packages and setting options for chunk output and plot dimensions.”

```
## Loading required package: usethis
```

INSTALLING THE PACKAGE

This section ensures the devtools package is installed, which is required for installing packages from GitHub. We then install the BayesianImportance package directly from GitHub using devtools::install_github(). In the package under the Hello.R file, all functions are defined with corresponding documentation.

```
# If not already installed, install the 'devtools' package
if (!require(devtools)) install.packages("devtools")

# Install BayesianImportance
devtools::install_github("AugustArnstad/BayesianImportance")
```

SIMULATE DATA

In this part, we simulate data to demonstrate the functionality of the BayesianImportance package. We generate random variables with different correlation structures, random effects, and an error term. The data is then structured into data frames for further analysis. If you have a suitable dataset you can use this instead.

```
library(BayesianImportance)
library(INLA)
library(mnormt)
library(ggplot2)
library(reshape2)
library(RColorBrewer)
set.seed(12)

n <- 10000
```

```

nclass_gamma <- 200
nclass_eta <- 100

mu <- c(1, 2, 3)

sigma1 <- matrix(c(1, 0, 0, 0, 1, 0, 0, 0, 1), 3, 3)
sigma2 <- matrix(c(1, 0.1, 0.1, 0.1, 1, 0.1, 0.1, 0.1, 1), 3, 3)
sigma3 <- matrix(c(1, 0.5, 0.5, 0.5, 1, 0.5, 0.5, 0.5, 1), 3, 3)
sigma4 <- matrix(c(1, 0.9, 0.9, 0.9, 1, 0.9, 0.9, 0.9, 1), 3, 3)

# Sample a standardized correlated design matrix
X1 <- rmnorm(n, mu, sigma1)
X2 <- rmnorm(n, mu, sigma2)
X3 <- rmnorm(n, mu, sigma3)
X4 <- rmnorm(n, mu, sigma4)

# Add random effects
gamma <- rep(rnorm(nclass_gamma, 0, sqrt(1)), each = n/nclass_gamma)
eta <- rep(rnorm(nclass_eta, 0, sqrt(1/2)), each = n/nclass_eta)
epsilon = rnorm(n, mean = 0, sd = sqrt(1))
beta <- c(1, sqrt(2), sqrt(3))

# Define some formula
Y1 <- 1 + beta[1] * X1[, 1] + beta[2] * X1[, 2] + beta[3] * X1[, 3] +
  gamma + epsilon # + eta
Y2 <- 1 + beta[1] * X2[, 1] + beta[2] * X2[, 2] + beta[3] * X2[, 3] +
  gamma + epsilon # + eta
Y3 <- 1 + beta[1] * X3[, 1] + beta[2] * X3[, 2] + beta[3] * X3[, 3] +
  gamma + epsilon # + eta
Y4 <- 1 + beta[1] * X4[, 1] + beta[2] * X4[, 2] + beta[3] * X4[, 3] +
  gamma + epsilon # + eta

# Collect as a dataframe
data_bayes1 = data.frame(cbind(Y1, X = X1))
data_bayes1 = data.frame(cbind(data_bayes1, gamma = gamma))
# data_bayes1 = data.frame(cbind(data_bayes1, eta=eta)) data_bayes1

data_bayes2 = data.frame(cbind(Y2, X = X2))
data_bayes2 = data.frame(cbind(data_bayes2, gamma = gamma))
# data_bayes2 = data.frame(cbind(data_bayes, eta=eta)) data_bayes2

data_bayes3 = data.frame(cbind(Y3, X = X3))
data_bayes3 = data.frame(cbind(data_bayes3, gamma = gamma))
# data_bayes = data.frame(cbind(data_bayes, eta=eta)) data_bayes3

data_bayes4 = data.frame(cbind(Y4, X = X4))
data_bayes4 = data.frame(cbind(data_bayes4, gamma = gamma))
# data_bayes = data.frame(cbind(data_bayes, eta=eta)) data_bayes4

```

USAGE

Here we demonstrate the usage of the BayesianImportance package. We fit Bayesian models and sample posterior distributions for different simulated datasets using functions from the package. Some EDA is

required, for example obtaining the number of classes in total across all random effects, which is used as input.

```
set.seed(1234)

model1 <- run_bayesian_imp(Y1 ~ V2 + V3 + V4 + (1 | gamma), data = data_bayes1)
posteriors1 <- sample_posteriors(Y1 ~ V2 + V3 + V4 + (1 | gamma), data = data_bayes1,
                                5000, n)

model2 <- run_bayesian_imp(Y2 ~ V2 + V3 + V4 + (1 | gamma), data = data_bayes2)
posteriors2 <- sample_posteriors(Y2 ~ V2 + V3 + V4 + (1 | gamma), data = data_bayes2,
                                5000, n)

model3 <- run_bayesian_imp(Y3 ~ V2 + V3 + V4 + (1 | gamma), data = data_bayes3)
posteriors3 <- sample_posteriors(Y3 ~ V2 + V3 + V4 + (1 | gamma), data = data_bayes3,
                                5000, n)

model4 <- run_bayesian_imp(Y4 ~ V2 + V3 + V4 + (1 | gamma), data = data_bayes4)
posteriors4 <- sample_posteriors(Y4 ~ V2 + V3 + V4 + (1 | gamma), data = data_bayes4,
                                5000, n)
```

PLOTTING

This code block defines a function `plot_posterior` that visualizes posterior distributions and relative importance from the sampled posteriors. The function creates density plots and overlays line plots for variance marginals. This can be modified for the specific problem at hand, and is not included in the package since it is a problem specific function.

```
plot_posterior <- function(betas, importances, marginals, importance=FALSE, theoretical=FALSE, eta=FALSE) {

  if (importance) {
    mat_long <- melt(as.data.frame(importances))
  } else {
    mat_long <- melt(as.data.frame(betas))
  }
  names(mat_long) <- c("Variable", "Value")

  df_marginals <- do.call(rbind, marginals)
  df_marginals$Effect <- as.factor(df_marginals$Effect)

  if (!importance){
    plot_title = paste("Posterior Distributions")
  }else{
    plot_title = paste("Posterior Relative Importance")
  }

  if (eta) {
    random_effect_labels <- c(expression(sigma[eta]^2), expression(sigma[alpha]^2), expression(sigma[epsilon]^2))
    num=6
  }
  else{
    random_effect_labels <- c(expression(sigma[alpha]^2), expression(sigma[epsilon]^2))
    num=5
  }
}
```

```

fixed_effect_labels <- if (importance) {
  c(expression(beta[1]^2), expression(beta[2]^2), expression(beta[3]^2))
} else {
  c(expression(beta[1]), expression(beta[2]), expression(beta[3]))
}
colors=rainbow(5)

p <- ggplot() +
  geom_density(data = mat_long, aes(x = Value, fill = as.factor(Variable)), alpha = 0.5) +
  geom_line(data = df_marginals, aes(x = x, y = y, color = Effect)) +
  labs(#title = plot_title,
       x = "Importance", y = "Density/Frequency") +
  theme_minimal() +
  theme(legend.position = "none", #USE right FOR LEGEND
        legend.text = element_text(size = 28),
        legend.title = element_text(size = 28),
        plot.title = element_text(size = 28, face = "bold"),
        axis.title.x = element_text(size = 28),
        axis.title.y = element_text(size = 28),
        strip.text = element_text(size = 28),
        axis.text.x = element_text(size = 28),
        axis.text.y = element_text(size = 28)) +
  scale_fill_manual(name = "Fixed effects",
                    values = colors[1:3],
                    labels = fixed_effect_labels) +
  scale_color_manual(name = "Random effects",
                     values = colors[4:num],
                     labels = random_effect_labels)

if (theoretical) {
  if(eta){
    p <- p + geom_vline(xintercept = 1/9, color = "black", linetype = "dashed") +
      geom_vline(xintercept = 1/18, color = "black", linetype = "dashed") +
      geom_vline(xintercept = 2/9, color = "black", linetype = "dashed") +
      geom_vline(xintercept = 3/9, color = "black", linetype = "dashed")
  }
  else{
    p <- p + geom_vline(xintercept = 0.125, color = "black", linetype = "dashed", size=1) +
      geom_vline(xintercept = 0.25, color = "black", linetype = "dashed", size=1) +
      geom_vline(xintercept = 0.375, color = "black", linetype = "dashed", size=1)
  }
}
return(p)
}

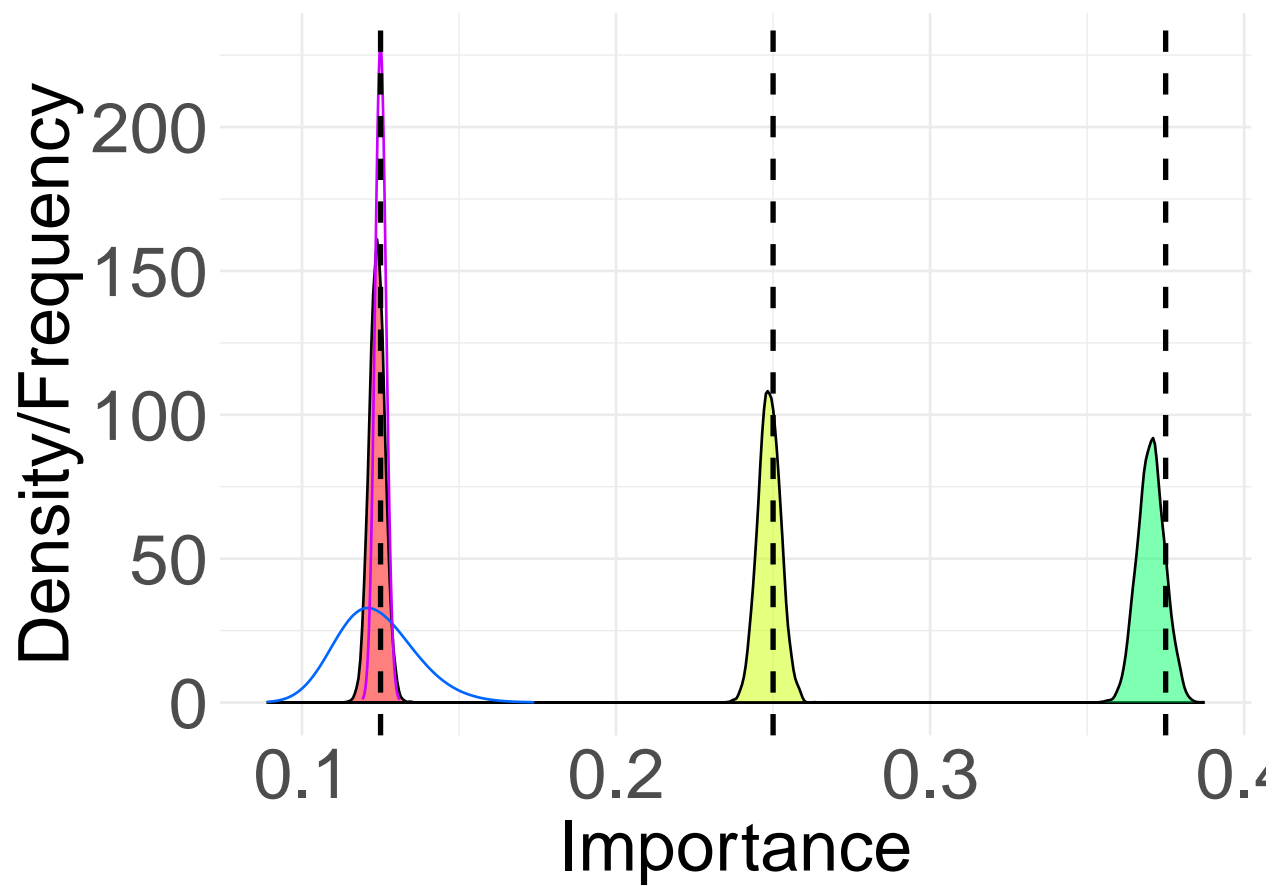
plot1 <- plot_posterior(posterior1$beta, posterior1$importance, posterior1$marginals, importance = T)
plot2 <- plot_posterior(posterior2$beta, posterior2$importance, posterior2$marginals, importance = T)
plot3 <- plot_posterior(posterior3$beta, posterior3$importance, posterior3$marginals, importance = T)
plot4 <- plot_posterior(posterior4$beta, posterior4$importance, posterior4$marginals, importance = T)

```

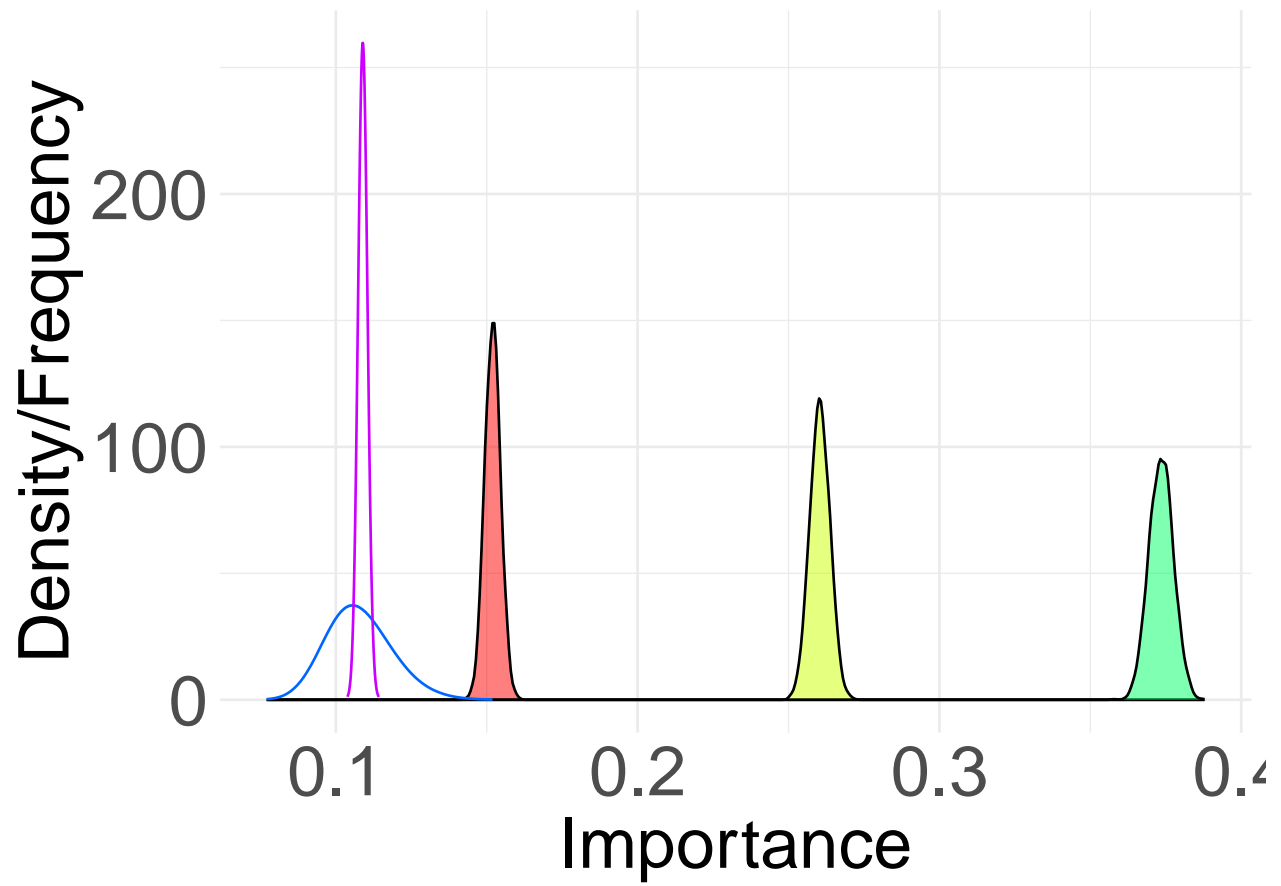
These plots are generated for each model's posteriors. We create and store the plots in variables plot1, plot2,

plot3, and plot4 for the respective models.

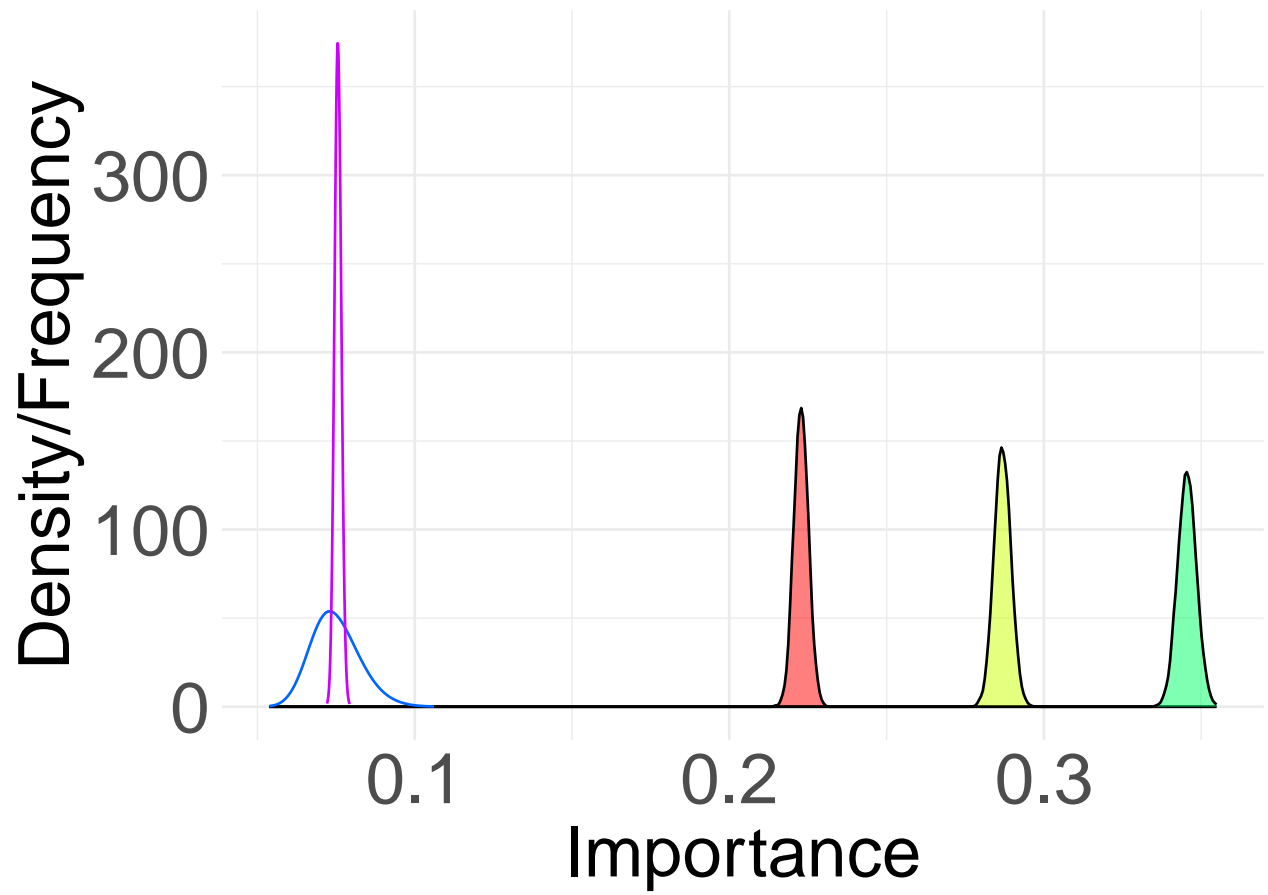
plot1



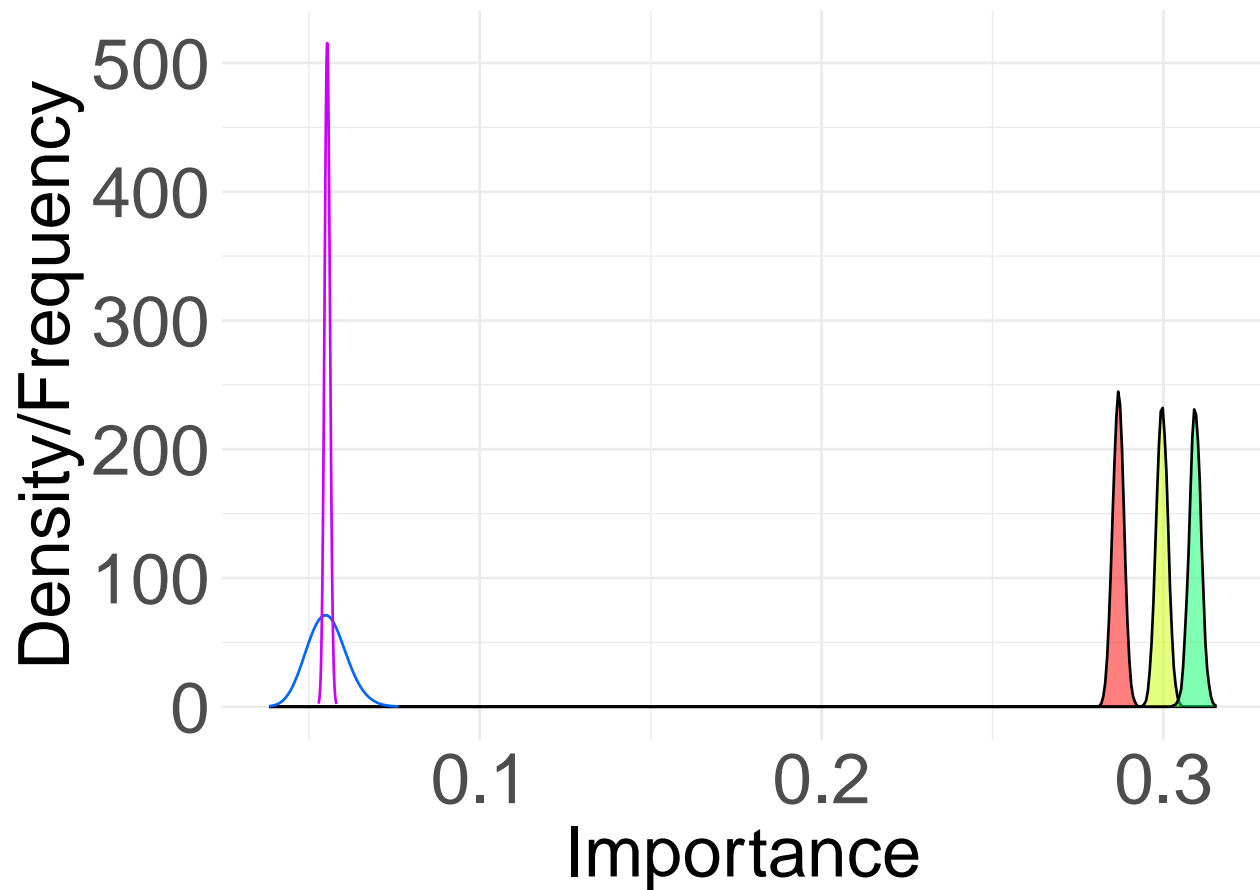
plot2



plot3



plot4



GELMAN R²

Here, we compute and visualize the Gelman conditional R² metrics for each model. This metric gives insight into the proportion of variance explained by the fixed effects in the models.

```
# Calculate theoretical variance
rho_values <- c(0, 0.1, 0.5, 0.9)

# Calculate conditional and marginal R-squared for each covariance
# level
variance_explained <- data.frame(rho = rho_values, R2_conditional = numeric(length(rho_values)),
  R2_marginal = numeric(length(rho_values)))

for (i in 1:length(rho_values)) {
  rho <- rho_values[i]

  # Calculate Var(Y) for the current rho
  var_Y <- (sum((beta^2)) + 2 * (beta[1] * beta[2] * rho + beta[1] *
    beta[3] * rho + beta[3] * beta[2] * rho) + 1 + 1)
  # cat(var_Y)

  # Calculate conditional R-squared
  R2_conditional <- (var_Y - 1)/var_Y

  # Calculate marginal R-squared
  R2_marginal <- (var_Y - 1 - 1)/var_Y
}
```



```

    variance_explained[i, "R2_conditional"] <- R2_conditional
    variance_explained[i, "R2_marginal"] <- R2_marginal
  }

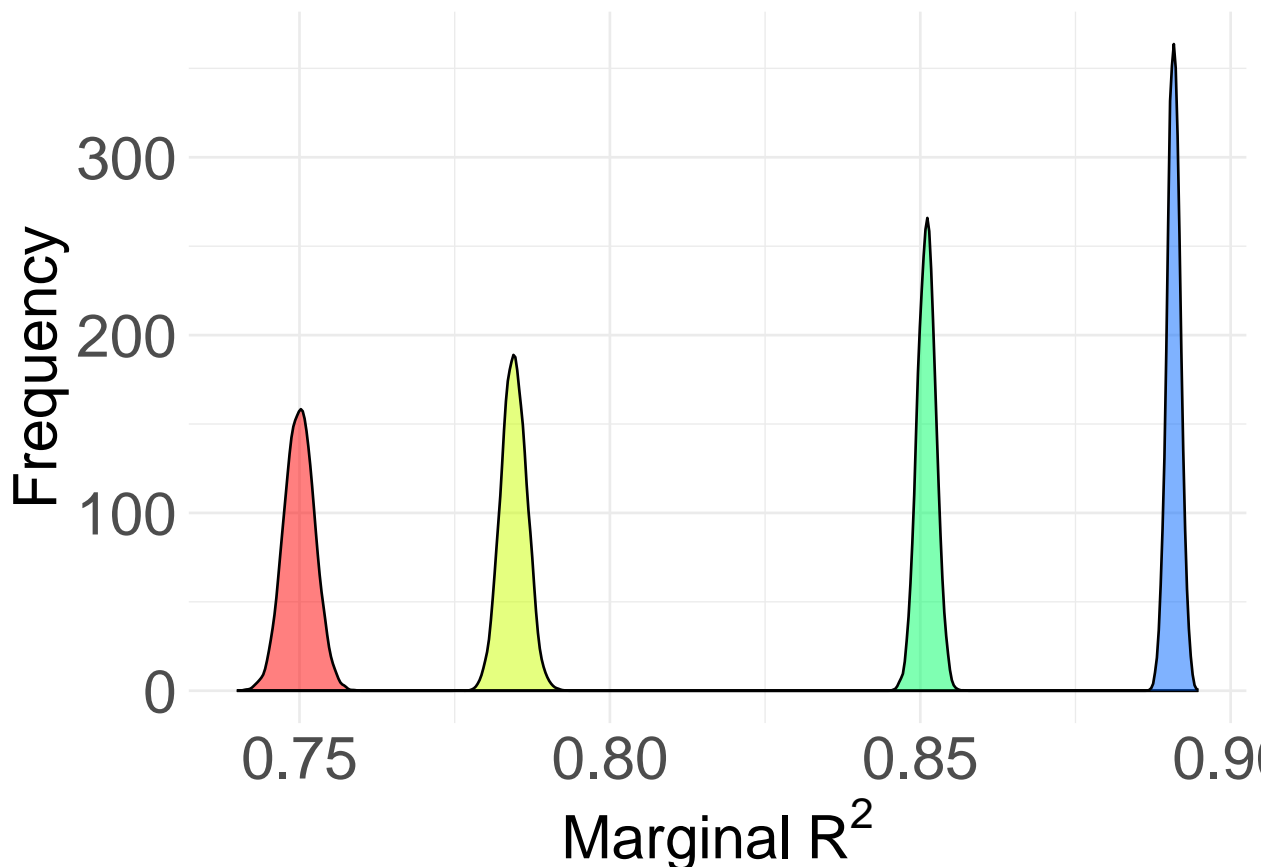
variance_explained

##   rho R2_conditional R2_marginal
## 1 0.0      0.8750000   0.7500000
## 2 0.1      0.8903407   0.7806813
## 3 0.5      0.9264476   0.8528953
## 4 0.9      0.9446669   0.8893338

colors <- rainbow(5)
# Create a combined plot with different colors for each
# distribution
marginal_var <- ggplot() + geom_density(data = as.data.frame(posterior1$r2),
  aes(x = V1), fill = colors[1], alpha = 0.5) + geom_density(data = as.data.frame(posterior2$r2),
  aes(x = V1), fill = colors[2], alpha = 0.5) + geom_density(data = as.data.frame(posterior3$r2),
  aes(x = V1), fill = colors[3], alpha = 0.5) + geom_density(data = as.data.frame(posterior4$r2),
  aes(x = V1), fill = colors[4], alpha = 0.5) + theme_minimal() + theme(legend.position = "none",
  axis.title.x = element_text(size = 24), axis.title.y = element_text(size = 24),
  axis.text.x = element_text(size = 24), axis.text.y = element_text(size = 24)) +
  labs(x = bquote(Marginal ~ R^2), y = "Frequency")

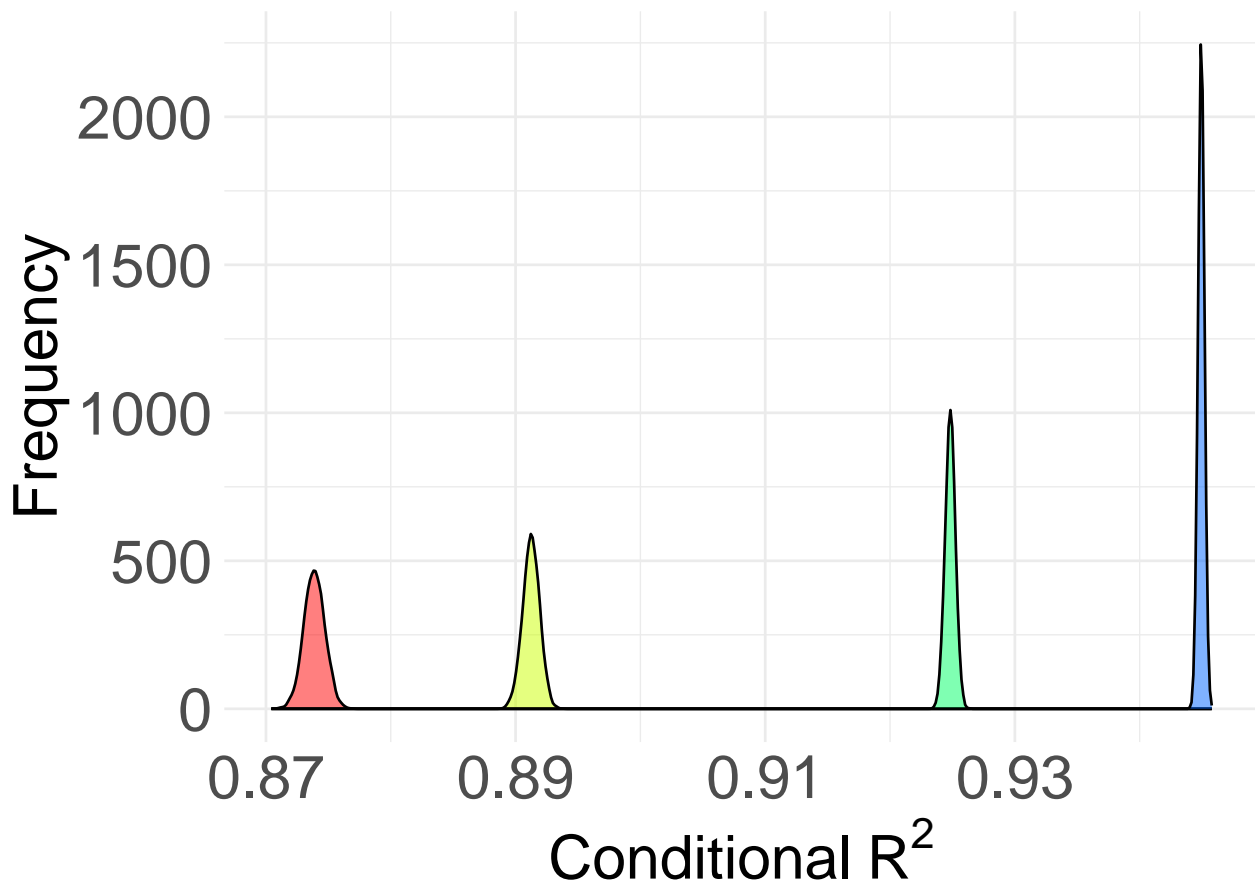
marginal_var

```



```
# Create a combined plot with different colors for each
# distribution
conditional_var <- ggplot() + geom_density(data = as.data.frame(posterior1$r2_cond),
  aes(x = V1), fill = colors[1], alpha = 0.5) + geom_density(data = as.data.frame(posterior2$r2_cond),
  aes(x = V1), fill = colors[2], alpha = 0.5) + geom_density(data = as.data.frame(posterior3$r2_cond),
  aes(x = V1), fill = colors[3], alpha = 0.5) + geom_density(data = as.data.frame(posterior4$r2_cond),
  aes(x = V1), fill = colors[4], alpha = 0.5) + theme_minimal() + theme(legend.position = "none",
  axis.title.x = element_text(size = 24), axis.title.y = element_text(size = 24),
  axis.text.x = element_text(size = 24), axis.text.y = element_text(size = 24)) +
  labs(x = expression(Conditional ~ R2), y = "Frequency")

conditional_var
```



```
output_folder_plots <- "/Users/augustarnstad/Library/CloudStorage/OneDrive-NTNU/Semester_9/Prosjekt/Latent"

# Create the folder if it doesn't exist
if (!dir.exists(output_folder_plots)) {
  dir.create(output_folder_plots, recursive = TRUE)
}

file_name <- file.path(output_folder_plots, paste("Marginal_R2.png",
  sep = ""))
ggsave(file_name, plot = marginal_var, width = 10, height = 6, dpi = 300)

file_name <- file.path(output_folder_plots, paste("Conditional_R2.png",
```

```
    sep = ""))  
ggsave(file_name, plot = conditional_var, width = 10, height = 6, dpi = 300)
```