STATS205P_Project

Chuqi Wang

2024-06-12

```r
library(rstan)
```

```
## Loading required package: StanHeaders
##
## rstan version 2.32.6 (Stan version 2.32.2)
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
## For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
## change `threads_per_chain` option:
## rstan_options(threads_per_chain = 1)
```

```r
library(rstan)
getwd()
```

```
## [1] "/Users/chuqiwang/Desktop/UCI/STATS205P/Project"
```

```r
data = read.csv("healthcare-dataset-stroke-data.csv", header = TRUE,
                       na.strings = c("N/A", "NA", ""))
summary(data)
```

```
##        id           gender               age         hypertension
##  Min.   :   67   Length:5110        Min.   : 0.08   Min.   :0.00000
##  1st Qu.:17741   Class :character   1st Qu.:25.00   1st Qu.:0.00000
##  Median :36932   Mode  :character   Median :45.00   Median :0.00000
##  Mean   :36518                      Mean   :43.23   Mean   :0.09746
##  3rd Qu.:54682                      3rd Qu.:61.00   3rd Qu.:0.00000
##  Max.   :72940                      Max.   :82.00   Max.   :1.00000
##
##  heart_disease     ever_married        work_type         Residence_type
##  Min.   :0.00000   Length:5110        Length:5110        Length:5110
##  1st Qu.:0.00000   Class :character   Class :character   Class :character
##  Median :0.00000   Mode  :character   Mode  :character   Mode  :character
##  Mean   :0.05401
##  3rd Qu.:0.00000
##  Max.   :1.00000
##
##  avg_glucose_level      bmi        smoking_status         stroke
##  Min.   : 55.12    Min.   :10.30   Length:5110        Min.   :0.00000
##  1st Qu.: 77.25    1st Qu.:23.50   Class :character   1st Qu.:0.00000
##  Median : 91.89    Median :28.10   Mode  :character   Median :0.00000
##  Mean   :106.15    Mean   :28.89                      Mean   :0.04873
##  3rd Qu.:114.09    3rd Qu.:33.10                      3rd Qu.:0.00000
##  Max.   :271.74    Max.   :97.60                      Max.   :1.00000
##                    NA's   :201
```

```r
head(data)
```

```
##      id gender age hypertension heart_disease ever_married     work_type
## 1  9046   Male  67            0             1          Yes       Private
## 2 51676 Female  61            0             0          Yes Self-employed
## 3 31112   Male  80            0             1          Yes       Private
## 4 60182 Female  49            0             0          Yes       Private
## 5  1665 Female  79            1             0          Yes Self-employed
## 6 56669   Male  81            0             0          Yes       Private
##   Residence_type avg_glucose_level  bmi  smoking_status stroke
## 1          Urban            228.69 36.6 formerly smoked      1
## 2          Rural            202.21   NA    never smoked      1
## 3          Rural            105.92 32.5    never smoked      1
## 4          Urban            171.23 34.4          smokes      1
## 5          Rural            174.12 24.0    never smoked      1
## 6          Urban            186.21 29.0 formerly smoked      1
```

```r
dim(data)
```

```
## [1] 5110   12
```

```r
names(data)
```

```
##  [1] "id"                "gender"            "age"
##  [4] "hypertension"      "heart_disease"     "ever_married"
##  [7] "work_type"         "Residence_type"    "avg_glucose_level"
## [10] "bmi"               "smoking_status"    "stroke"
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
data %>% summarise_all(~sum(is.na(.)))
```

```
##   id gender age hypertension heart_disease ever_married work_type
## 1  0      0   0            0             0            0         0
##   Residence_type avg_glucose_level bmi smoking_status stroke
## 1              0                 0 201              0      0
```

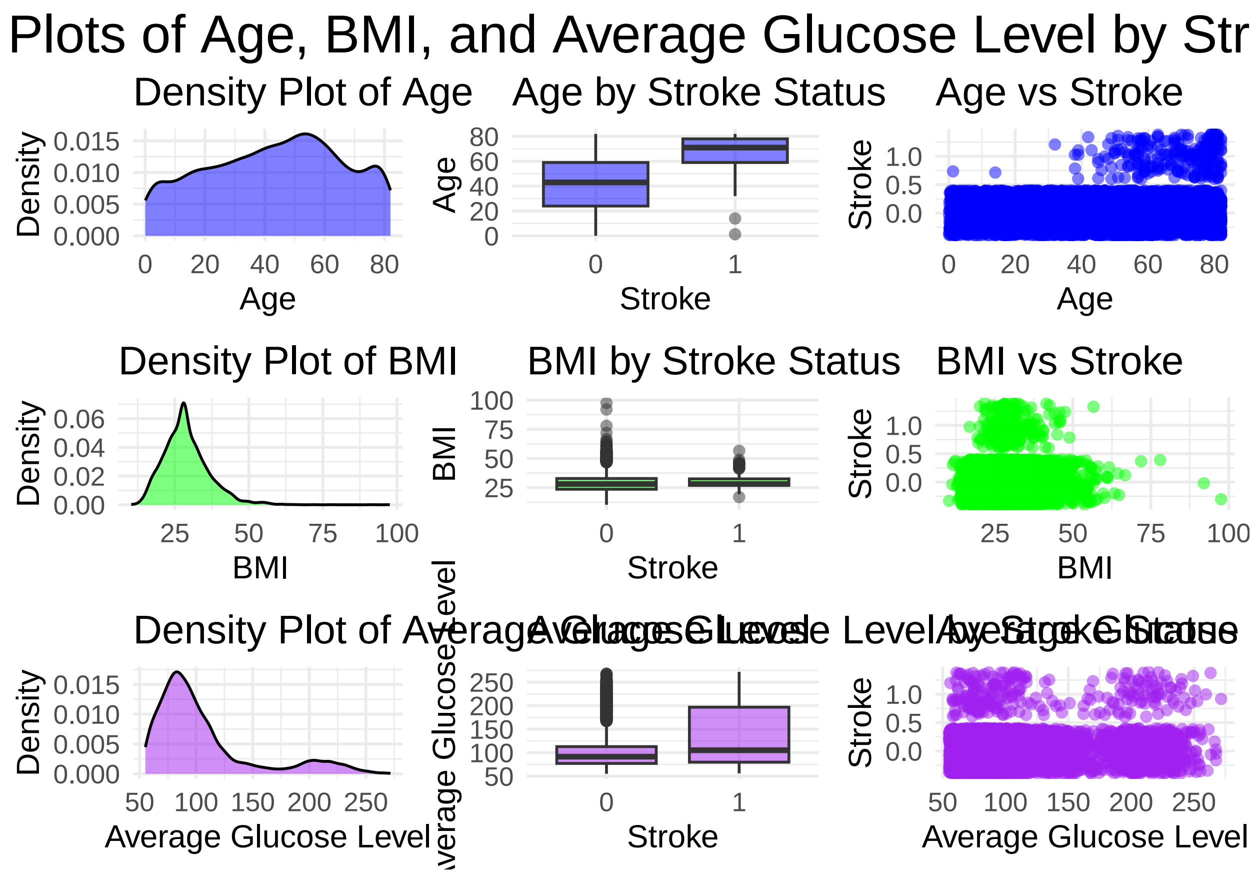
```r
data$bmi[is.na(data$bmi)] <- median(data$bmi, na.rm = TRUE)
```

```r
# Load necessary libraries
# Load necessary libraries
library(ggplot2)
library(gridExtra)
```

```
##
```

```
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

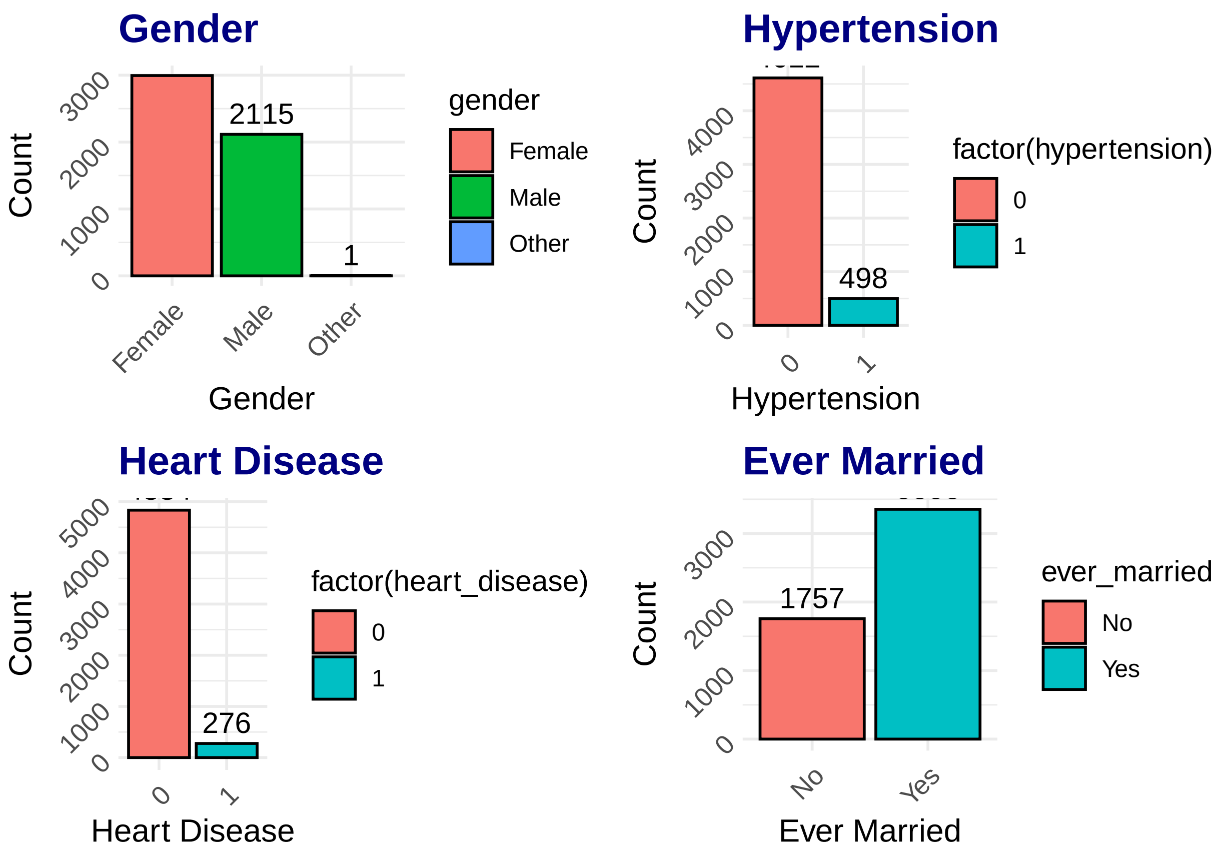
```r
library(grid)
# Create a density plot for age
p1 <- ggplot(data, aes(x = age)) +
  geom_density(fill = "blue", alpha = 0.5) +
  theme_minimal() +
  labs(title = "Density Plot of Age", x = "Age", y = "Density") +
  theme(plot.title = element_text(size = 15),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10))

# Create a density plot for BMI
p2 <- ggplot(data, aes(x = bmi)) +
  geom_density(fill = "green", alpha = 0.5) +
  theme_minimal() +
  labs(title = "Density Plot of BMI", x = "BMI", y = "Density") +
  theme(plot.title = element_text(size = 15),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10))

# Create a density plot for average glucose level
p3 <- ggplot(data, aes(x = avg_glucose_level)) +
  geom_density(fill = "purple", alpha = 0.5) +
  theme_minimal() +
  labs(title = "Density Plot of Average Glucose Level", x = "Average Glucose Level", y = "Density") +
  theme(plot.title = element_text(size = 15),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10))

# Box plot for age by stroke status
p4 <- ggplot(data, aes(x = factor(stroke), y = age)) +
  geom_boxplot(fill = "blue", alpha = 0.5) +
  theme_minimal() +
  labs(title = 'Age by Stroke Status', x = 'Stroke', y = 'Age') +
  theme(plot.title = element_text(size = 15),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10))

# Box plot for bmi by stroke status
p5 <- ggplot(data, aes(x = factor(stroke), y = bmi)) +
  geom_boxplot(fill = "green", alpha = 0.5) +
  theme_minimal() +
  labs(title = 'BMI by Stroke Status', x = 'Stroke', y = 'BMI') +
  theme(plot.title = element_text(size = 15),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10))

# Box plot for avg_glucose_level by stroke status
p6 <- ggplot(data, aes(x = factor(stroke), y = avg_glucose_level)) +
  geom_boxplot(fill = "purple", alpha = 0.5) +
  theme_minimal() +
  labs(title = 'Average Glucose Level by Stroke Status', x = 'Stroke', y = 'Average Glucose Level') +
  theme(plot.title = element_text(size = 15),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10))

# Scatter plot for age vs stroke
p7 <- ggplot(data, aes(x = age, y = stroke)) +
  geom_jitter(color = "blue", alpha = 0.5) +
  theme_minimal() +
  labs(title = 'Age vs Stroke', x = 'Age', y = 'Stroke') +
  theme(plot.title = element_text(size = 15),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10))

# Scatter plot for bmi vs stroke
p8 <- ggplot(data, aes(x = bmi, y = stroke)) +
  geom_jitter(color = "green", alpha = 0.5) +
  theme_minimal() +
  labs(title = 'BMI vs Stroke', x = 'BMI', y = 'Stroke') +
  theme(plot.title = element_text(size = 15),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10))

# Scatter plot for avg_glucose_level vs stroke
p9 <- ggplot(data, aes(x = avg_glucose_level, y = stroke)) +
  geom_jitter(color = "purple", alpha = 0.5) +
  theme_minimal() +
  labs(title = 'Average Glucose Level vs Stroke', x = 'Average Glucose Level', y = 'Stroke') +
  theme(plot.title = element_text(size = 15),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10))

# Combine the three plots into one layout with increased figure size
grid.arrange(p1, p4, p7, p2, p5, p8, p3, p6, p9, ncol = 3, nrow = 3,
             top = textGrob("Combined Plots of Age, BMI, and Average Glucose Level by Stroke Status",
                            gp = gpar(fontsize = 20)))
```



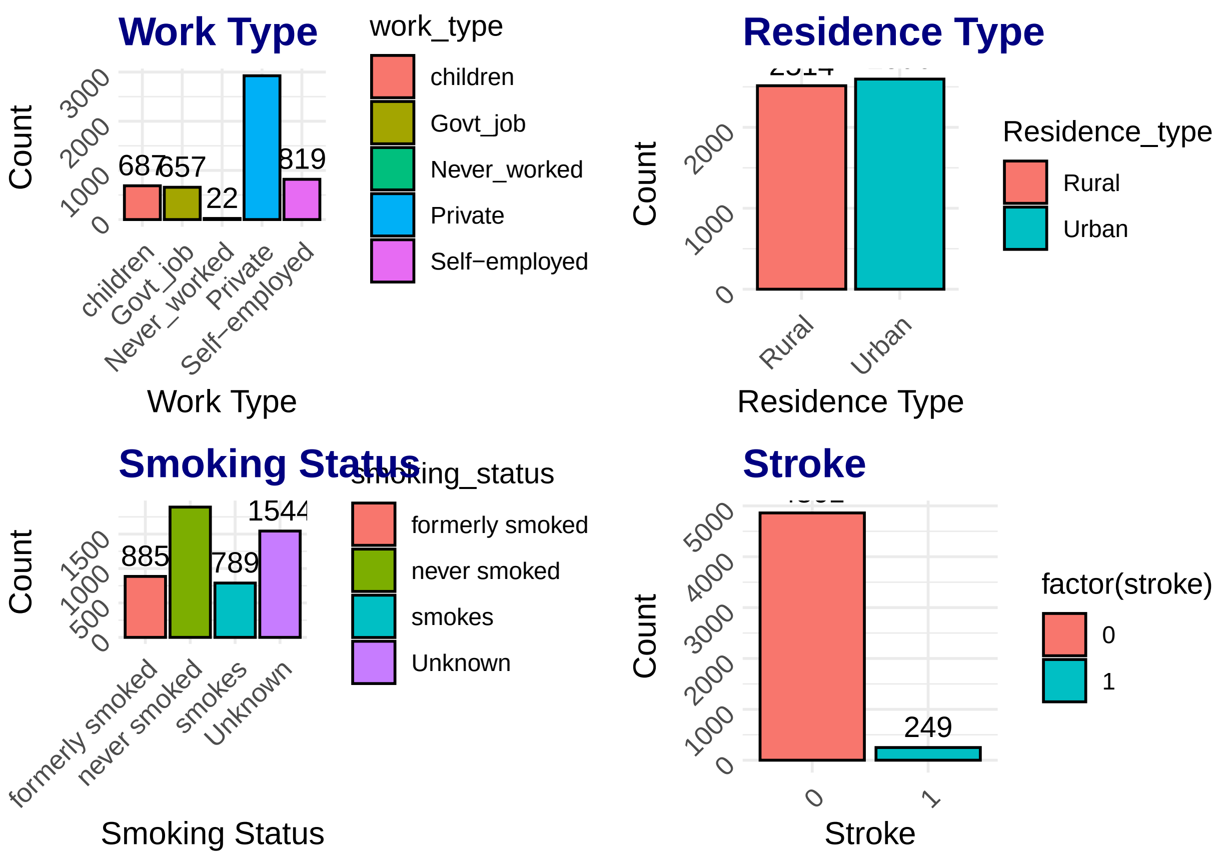
```r
# Create a bar plot for gender
p1 <- ggplot(data, aes(x = gender)) +
  geom_bar(aes(fill = gender), color = "black") +
  theme_minimal() +
  labs(title = "Gender", x = "Gender", y = "Count") +
  geom_text(stat = 'count', aes(label = ..count..), vjust = -0.5) +
  theme(plot.title = element_text(size = 15, face = "bold", color = "navy"),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10, angle = 45, hjust = 1))

# Create a bar plot for hypertension
p2 <- ggplot(data, aes(x = factor(hypertension))) +
  geom_bar(aes(fill = factor(hypertension)), color = "black") +
  theme_minimal() +
  labs(title = "Hypertension", x = "Hypertension", y = "Count") +
  geom_text(stat = 'count', aes(label = ..count..), vjust = -0.5) +
  theme(plot.title = element_text(size = 15, face = "bold", color = "navy"),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10, angle = 45, hjust = 1))

# Create a bar plot for heart_disease
p3 <- ggplot(data, aes(x = factor(heart_disease))) +
  geom_bar(aes(fill = factor(heart_disease)), color = "black") +
  theme_minimal() +
  labs(title = "Heart Disease", x = "Heart Disease", y = "Count") +
  geom_text(stat = 'count', aes(label = ..count..), vjust = -0.5) +
  theme(plot.title = element_text(size = 15, face = "bold", color = "navy"),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10, angle = 45, hjust = 1))

# Create a bar plot for ever_married
p4 <- ggplot(data, aes(x = ever_married)) +
  geom_bar(aes(fill = ever_married), color = "black") +
  theme_minimal() +
  labs(title = "Ever Married", x = "Ever Married", y = "Count") +
  geom_text(stat = 'count', aes(label = ..count..), vjust = -0.5) +
  theme(plot.title = element_text(size = 15, face = "bold", color = "navy"),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10, angle = 45, hjust = 1))

# Create a bar plot for work_type
p5 <- ggplot(data, aes(x = work_type)) +
  geom_bar(aes(fill = work_type), color = "black") +
  theme_minimal() +
  labs(title = "Work Type", x = "Work Type", y = "Count") +
  geom_text(stat = 'count', aes(label = ..count..), vjust = -0.5) +
  theme(plot.title = element_text(size = 15, face = "bold", color = "navy"),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10, angle = 45, hjust = 1))

# Create a bar plot for Residence_type
p6 <- ggplot(data, aes(x = Residence_type)) +
  geom_bar(aes(fill = Residence_type), color = "black") +
  theme_minimal() +
  labs(title = "Residence Type", x = "Residence Type", y = "Count") +
  geom_text(stat = 'count', aes(label = ..count..), vjust = -0.5) +
  theme(plot.title = element_text(size = 15, face = "bold", color = "navy"),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10, angle = 45, hjust = 1))

# Create a bar plot for smoking_status
p7 <- ggplot(data, aes(x = smoking_status)) +
  geom_bar(aes(fill = smoking_status), color = "black") +
  theme_minimal() +
  labs(title = "Smoking Status", x = "Smoking Status", y = "Count") +
  geom_text(stat = 'count', aes(label = ..count..), vjust = -0.5) +
  theme(plot.title = element_text(size = 15, face = "bold", color = "navy"),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10, angle = 45, hjust = 1))

# Create a bar plot for stroke
p8 <- ggplot(data, aes(x = factor(stroke))) +
  geom_bar(aes(fill = factor(stroke)), color = "black") +
  theme_minimal() +
  labs(title = "Stroke", x = "Stroke", y = "Count") +
  geom_text(stat = 'count', aes(label = ..count..), vjust = -0.5) +
  theme(plot.title = element_text(size = 15, face = "bold", color = "navy"),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10, angle = 45, hjust = 1))
```

```
# Combine the bar plots into one layout
grid.arrange(p1, p2, p3, p4, ncol = 2, nrow = 2)
```

```
## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(count)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

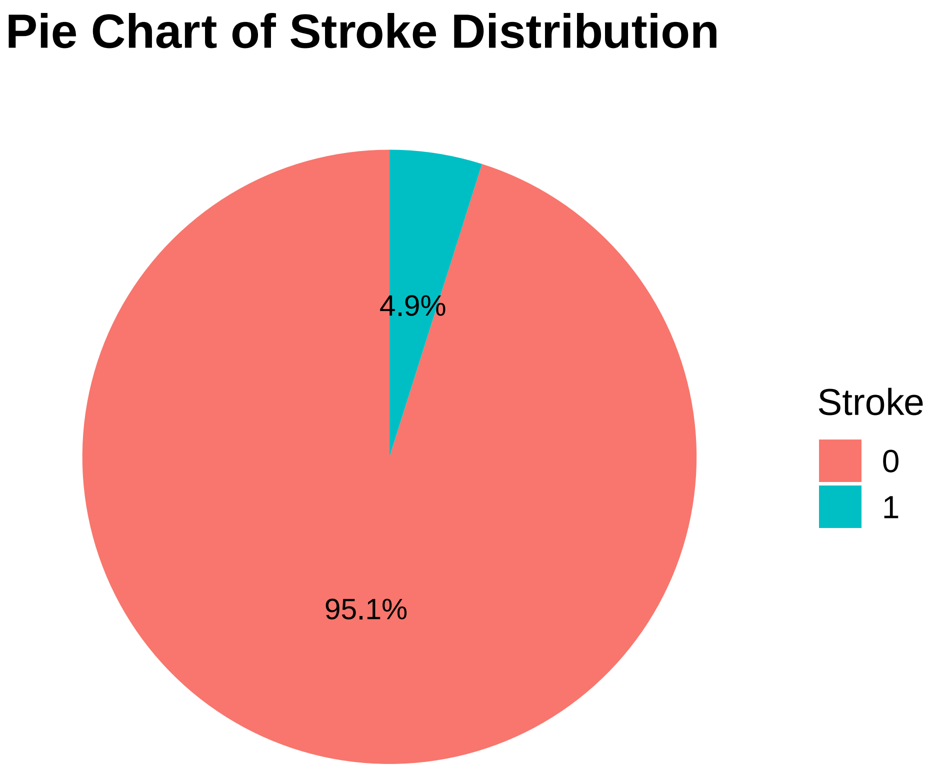


```
grid.arrange(p5, p6, p7, p8, ncol = 2, nrow = 2)
```

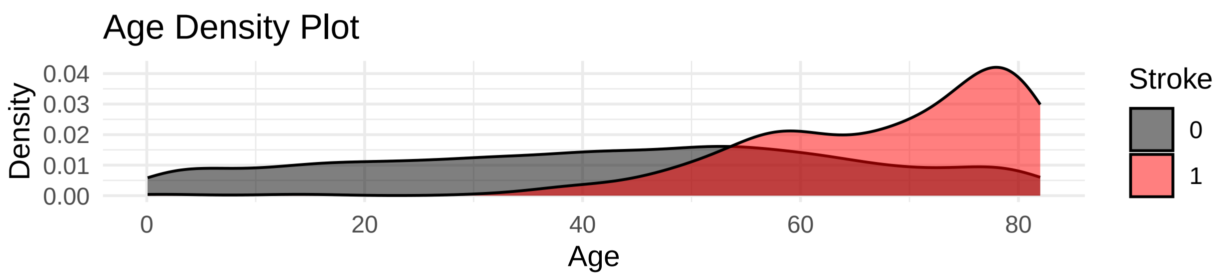


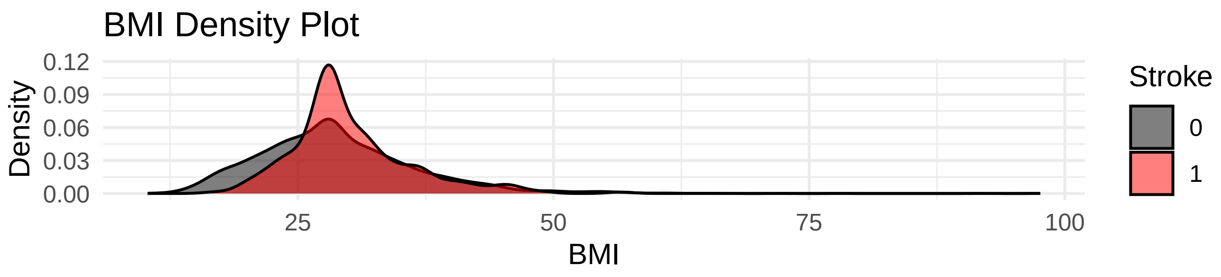
```r
# Load necessary libraries
library(ggplot2)
library(dplyr)

# Prepare the data
stroke_data <- data %>%
  group_by(stroke) %>%
  summarise(count = n()) %>%
  mutate(prop = count / sum(count) * 100,
         label = paste0(round(prop, 1), "%"))

# Create the pie chart with percentage labels
pie_chart <- ggplot(stroke_data, aes(x = "", y = prop, fill = factor(stroke))) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar(theta = "y") +
  labs(title = "Pie Chart of Stroke Distribution", fill = "Stroke") +
  theme_minimal() +
  theme(axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        panel.grid = element_blank(),
        axis.text.x = element_blank(),
        plot.title = element_text(size = 18, face = "bold"),
        legend.title = element_text(size = 14),
        legend.text = element_text(size = 12)) +
  geom_text(aes(label = label), position = position_stack(vjust = 0.5), size = 4)

# Display the pie chart
```

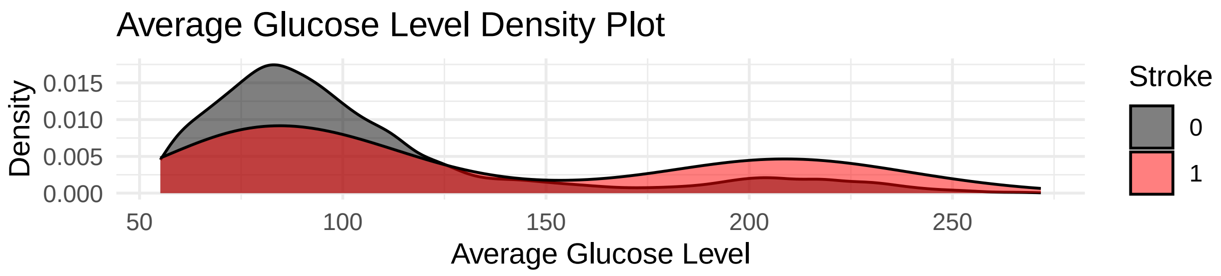
```r
print(pie_chart)
```



```r
# Density plot for age
p1 = ggplot(data, aes(x = age, fill = factor(stroke))) +
  geom_density(alpha = 0.5) +
  theme_minimal() +
  labs(title = 'Age Density Plot', x = 'Age', y = 'Density', fill = 'Stroke') +
  scale_fill_manual(values = c("0" = "black", "1" = "red"))

# Density plot for bmi
p2 = ggplot(data, aes(x = bmi, fill = factor(stroke))) +
  geom_density(alpha = 0.5) +
  theme_minimal() +
  labs(title = 'BMI Density Plot', x = 'BMI', y = 'Density', fill = 'Stroke') +
  scale_fill_manual(values = c("0" = "black", "1" = "red"))

# Density plot for avg_glucose_level
p3 = ggplot(data, aes(x = avg_glucose_level, fill = factor(stroke))) +
  geom_density(alpha = 0.5) +
  theme_minimal() +
  labs(title = 'Average Glucose Level Density Plot', x = 'Average Glucose Level', y = 'Density', fill =
  scale_fill_manual(values = c("0" = "black", "1" = "red"))

grid.arrange(p1, p2, p3, ncol = 1, nrow = 3)
```







```
library(caret)
```

```
## Loading required package: lattice
```

```
data <- data %>%
  mutate(gender = as.factor(gender),
         ever_married = as.factor(ever_married),
         work_type = as.factor(work_type),
         Residence_type = as.factor(Residence_type),
         smoking_status = as.factor(smoking_status))
set.seed(123)  # For reproducibility
train_index <- createDataPartition(data$stroke, p = 0.8, list = FALSE)
train_data <- data[train_index, ]
test_data <- data[-train_index, ]
# Check the dimensions of the split data
dim(train_data)
```

```
## [1] 4088   12
```

```
dim(test_data)
```

```
## [1] 1022   12
```

```
library(brms)
```

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

```
## Loading 'brms' package (version 2.21.0). 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:rstan':
##
##     loo

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

```
priors <- c(set_prior("normal(0, 10)", class = "b"),  # Prior for coefficients
            set_prior("student_t(3, 0, 10)", class = "Intercept"))  # Prior for intercept
formula <- bf(stroke ~ gender + age + hypertension + heart_disease + ever_married +
              work_type + Residence_type + avg_glucose_level + bmi + smoking_status,
              family = bernoulli())

# Fit the model
fit <- brm(formula, data = train_data, prior = priors, iter = 2000, chains = 4, seed = 123)
```

```
## Compiling Stan program...

## Trying to compile a simple C file

## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## using C compiler: ‘Apple clang version 15.0.0 (clang-1500.3.9.4)’
## using SDK: ‘MacOSX14.4.sdk’
## clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Library/Framew
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/StanHeader
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/
## /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/src/Core
## #include <cmath>
##          ^~~~~~~
## 1 error generated.
## make: *** [foo.o] Error 1

## Start sampling

##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000112 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.12 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
```

```
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 3.768 seconds (Warm-up)
## Chain 1:                1.737 seconds (Sampling)
## Chain 1:                5.505 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 9.2e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.92 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 4.354 seconds (Warm-up)
## Chain 2:                1.77 seconds (Sampling)
## Chain 2:                6.124 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 9.4e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.94 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 4.305 seconds (Warm-up)
## Chain 3:                1.794 seconds (Sampling)
## Chain 3:                6.099 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 8.8e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.88 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 5.29 seconds (Warm-up)
## Chain 4:                1.802 seconds (Sampling)
## Chain 4:                7.092 seconds (Total)
## Chain 4:
```

```
summary(fit)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: stroke ~ gender + age + hypertension + heart_disease + ever_married + work_type + Residence
##    Data: train_data (Number of observations: 4088)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Regression Coefficients:
##                            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## Intercept                     -6.76      0.91    -8.78    -5.24 1.00     1755
## genderMale                    -0.01      0.16    -0.33     0.31 1.00     6636
## genderOther                   -5.01      7.02   -20.60     5.54 1.00     4798
## age                            0.08      0.01     0.06     0.09 1.00     3355
## hypertension                   0.57      0.18     0.20     0.92 1.00     6428
## heart_disease                  0.29      0.22    -0.15     0.69 1.00     5742
## ever_marriedYes               -0.13      0.27    -0.64     0.41 1.00     5615
## work_typeGovt_job             -1.10      0.99    -2.74     1.06 1.00     1246
## work_typeNever_worked         -6.63      6.34   -21.67     2.31 1.00     3711
## work_typePrivate              -0.84      0.96    -2.45     1.33 1.00     1289
## work_typeSelfMemployed        -1.30      0.99    -2.97     0.90 1.00     1352
## Residence_typeUrban            0.03      0.16    -0.27     0.34 1.00     5682
## avg_glucose_level              0.00      0.00     0.00     0.01 1.00     4522
## bmi                           -0.00      0.01    -0.03     0.02 1.00     5448
## smoking_statusneversmoked     -0.18      0.20    -0.58     0.21 1.00     3913
## smoking_statussmokes          0.08      0.24    -0.40     0.56 1.00     3852
## smoking_statusUnknown        -0.16      0.24    -0.64     0.33 1.00     3759
##                            Tail_ESS
## Intercept                      1701
## genderMale                     3092
## genderOther                    2556
## age                            2743
## hypertension                   3100
## heart_disease                  2804
## ever_marriedYes                2944
## work_typeGovt_job              1551
## work_typeNever_worked          2587
## work_typePrivate               1487
## work_typeSelfMemployed         1524
## Residence_typeUrban            2814
## avg_glucose_level              2916
## bmi                            3130
## smoking_statusneversmoked      3185
## smoking_statussmokes           3058
## smoking_statusUnknown          2986
##
## 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
predictions <- predict(fit, newdata = test_data, type = "response")

# Convert probabilities to binary predictions
test_data$predicted_stroke <- ifelse(predictions[,1] > 0.5, 1, 0)

# Evaluate the model
conf_matrix <- table(test_data$stroke, test_data$predicted_stroke)
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
cat("Accuracy:", accuracy, "\n")
```

```
## Accuracy: 0.9452055
```

```r
print(conf_matrix)
```

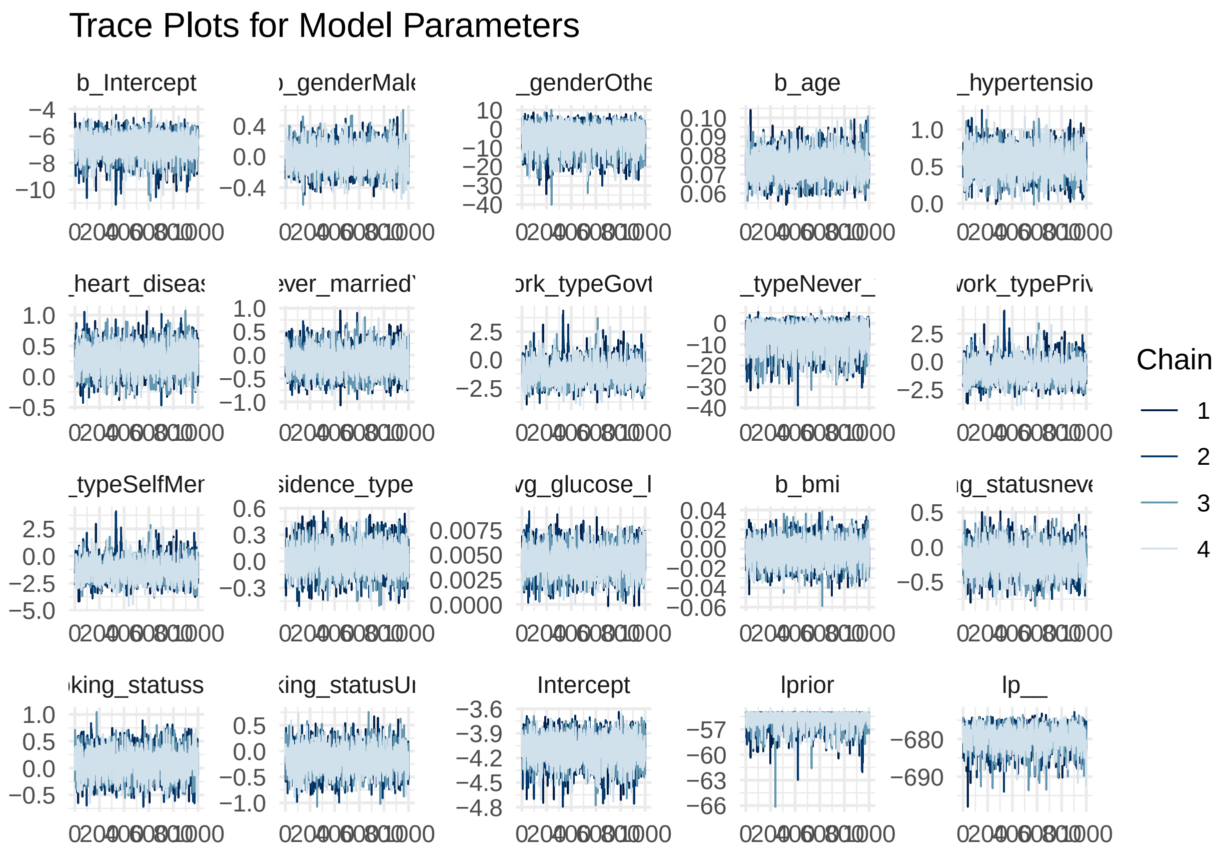
```
##
##       0   1
##   0 965   1
##   1  55   1
```

```r
library(bayesplot)
```

```
## This is bayesplot version 1.11.1

## - Online documentation and vignettes at mc-stan.org/bayesplot

## - bayesplot theme set to bayesplot::theme_default()

##    * Does _not_ affect other ggplot2 plots

##    * See ?bayesplot_theme_set for details on theme setting

##
## Attaching package: 'bayesplot'
## The following object is masked from 'package:brms':
##
##     rhat
```

```
mcmc_trace(as.mcmc(fit)) + ggtitle("Trace Plots for Model Parameters") + theme_minimal()
```

```
## Warning: as.mcmc.brmsfit is deprecated and will eventually be removed.
```



```
# Create individual plots for each parameter
g1 <- mcmc_areas(
  as.mcmc(fit),
  pars = c("b_Intercept"),
  prob = 0.95
) +
  ggtitle("Posterior Distribution of the Intercept") +
  xlab("Intercept") +
  ylab("Density") +
  theme_minimal()
```

```
## Warning: as.mcmc.brmsfit is deprecated and will eventually be removed.
```

```
g2 <- mcmc_areas(
  as.mcmc(fit),
  pars = c("b_genderMale"),
  prob = 0.95
) +
  ggtitle("Posterior Distribution of genderMale") +
  xlab("genderMale") +
  ylab("Density") +
```

```r
  theme_minimal()
```

```
## Warning: as.mcmc.brmsfit is deprecated and will eventually be removed.
```

```r
g3 <- mcmc_areas(
  as.mcmc(fit),
  pars = c("b_genderOther"),
  prob = 0.95
) +
  ggtitle("Posterior Distribution of genderOther") +
  xlab("genderOther") +
  ylab("Density") +
  theme_minimal()
```

```
## Warning: as.mcmc.brmsfit is deprecated and will eventually be removed.
```

```r
# Continue for other parameters
g4 <- mcmc_areas(
  as.mcmc(fit),
  pars = c("b_age"),
  prob = 0.95
) +
  ggtitle("Posterior Distribution of age") +
  xlab("age") +
  ylab("Density") +
  theme_minimal()
```

```
## Warning: as.mcmc.brmsfit is deprecated and will eventually be removed.
```

```r
g5 <- mcmc_areas(
  as.mcmc(fit),
  pars = c("b_hypertension"),
  prob = 0.95
) +
  ggtitle("Posterior Distribution of hypertension1") +
  xlab("hypertension") +
  ylab("Density") +
  theme_minimal()
```

```
## Warning: as.mcmc.brmsfit is deprecated and will eventually be removed.
```

```r
g6 <- mcmc_areas(
  as.mcmc(fit),
  pars = c("b_heart_disease"),
  prob = 0.95
) +
  ggtitle("Posterior Distribution of heart_disease1") +
  xlab("heart_disease") +
  ylab("Density") +
  theme_minimal()
```

```
## Warning: as.mcmc.brmsfit is deprecated and will eventually be removed.
```

```r
g7 <- mcmc_areas(
  as.mcmc(fit),
  pars = c("b_ever_marriedYes"),
  prob = 0.95
) +
  ggtitle("Posterior Distribution of ever_marriedYes") +
  xlab("ever_marriedYes") +
  ylab("Density") +
  theme_minimal()
```

```
## Warning: as.mcmc.brmsfit is deprecated and will eventually be removed.
```

```r
g8 <- mcmc_areas(
  as.mcmc(fit),
  pars = c("b_work_typeGovt_job"),
  prob = 0.95
) +
  ggtitle("Posterior Distribution of work_typeGovt_job") +
  xlab("work_typeGovt_job") +
  ylab("Density") +
  theme_minimal()
```

```
## Warning: as.mcmc.brmsfit is deprecated and will eventually be removed.
```

```r
g9 <- mcmc_areas(
  as.mcmc(fit),
  pars = c("b_work_typeNever_worked"),
  prob = 0.95
) +
  ggtitle("Posterior Distribution of work_typeNever_worked") +
  xlab("work_typeNever_worked") +
  ylab("Density") +
  theme_minimal()
```

```
## Warning: as.mcmc.brmsfit is deprecated and will eventually be removed.
```

```r
g10 <- mcmc_areas(
  as.mcmc(fit),
  pars = c("b_work_typePrivate"),
  prob = 0.95
) +
  ggtitle("Posterior Distribution of work_typePrivate") +
  xlab("work_typePrivate") +
  ylab("Density") +
  theme_minimal()
```

```
## Warning: as.mcmc.brmsfit is deprecated and will eventually be removed.
```

```r
g11 <- mcmc_areas(
  as.mcmc(fit),
  pars = c("b_work_typeSelfMemployed"),
  prob = 0.95
) +
  ggtitle("Posterior Distribution of work_typeSelfEmployed") +
  xlab("work_typeSelfEmployed") +
  ylab("Density") +
  theme_minimal()
```

```
## Warning: as.mcmc.brmsfit is deprecated and will eventually be removed.
```

```r
g12 <- mcmc_areas(
  as.mcmc(fit),
  pars = c("b_Residence_typeUrban"),
  prob = 0.95
) +
  ggtitle("Posterior Distribution of Residence_typeUrban") +
  xlab("Residence_typeUrban") +
  ylab("Density") +
  theme_minimal()
```

```
## Warning: as.mcmc.brmsfit is deprecated and will eventually be removed.
```

```r
g13 <- mcmc_areas(
  as.mcmc(fit),
  pars = c("b_avg_glucose_level"),
  prob = 0.95
) +
  ggtitle("Posterior Distribution of avg_glucose_level") +
  xlab("avg_glucose_level") +
  ylab("Density") +
  theme_minimal()
```

```
## Warning: as.mcmc.brmsfit is deprecated and will eventually be removed.
```

```r
g14 <- mcmc_areas(
  as.mcmc(fit),
  pars = c("b_bmi"),
  prob = 0.95
) +
  ggtitle("Posterior Distribution of bmi") +
  xlab("bmi") +
  ylab("Density") +
  theme_minimal()
```

```
## Warning: as.mcmc.brmsfit is deprecated and will eventually be removed.
```

```r
g15 <- mcmc_areas(
  as.mcmc(fit),
  pars = c("b_smoking_statusneversmoked"),
  prob = 0.95
) +
  ggtitle("Posterior Distribution of smoking_statusneversmoked") +
  xlab("smoking_statusneversmoked") +
  ylab("Density") +
  theme_minimal()
```

```
## Warning: as.mcmc.brmsfit is deprecated and will eventually be removed.
```

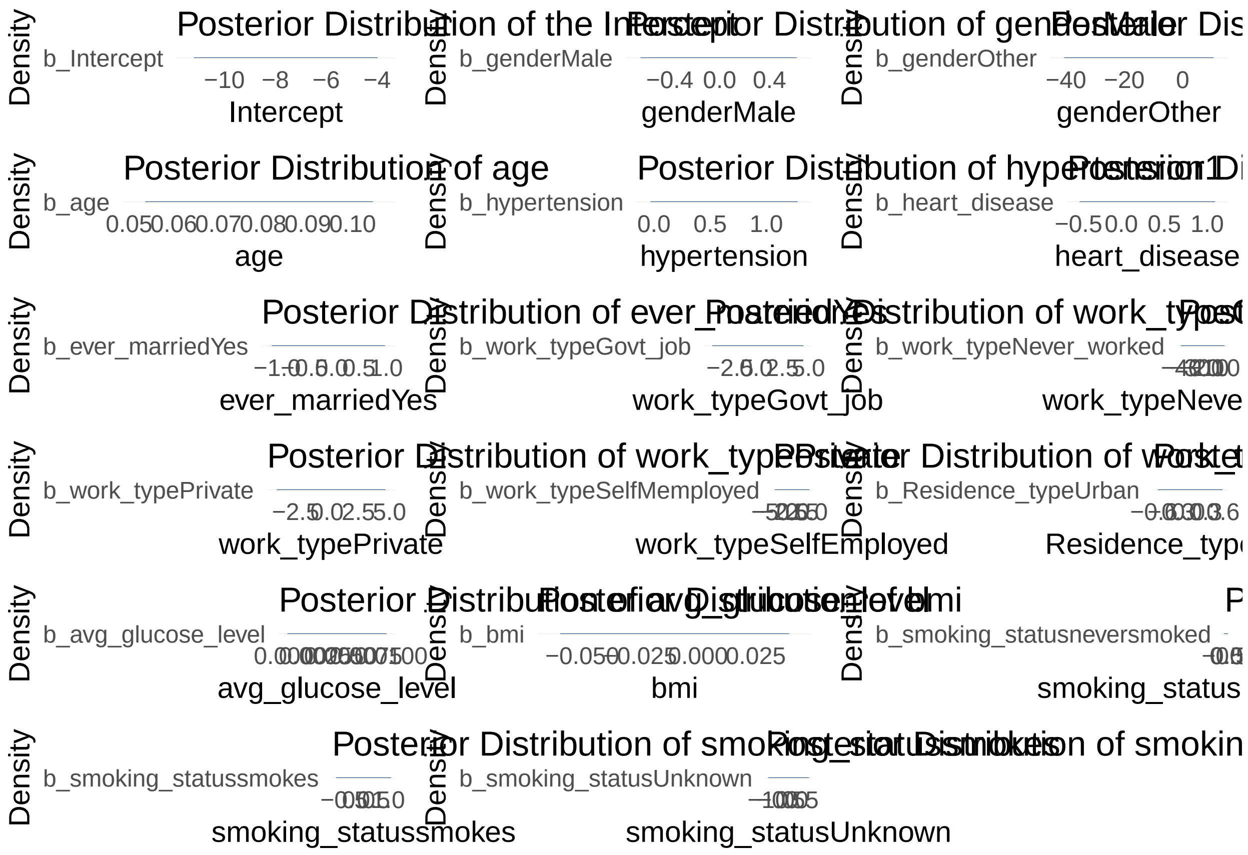
```r
g16 <- mcmc_areas(
  as.mcmc(fit),
  pars = c("b_smoking_statussmokes"),
  prob = 0.95
) +
  ggtitle("Posterior Distribution of smoking_statussmokes") +
  xlab("smoking_statussmokes") +
  ylab("Density") +
  theme_minimal()
```

```
## Warning: as.mcmc.brmsfit is deprecated and will eventually be removed.
```

```
g17 <- mcmc_areas(
  as.mcmc(fit),
  pars = c("b_smoking_statusUnknown"),
  prob = 0.95
) +
  ggtitle("Posterior Distribution of smoking_statusUnknown") +
  xlab("smoking_statusUnknown") +
  ylab("Density") +
  theme_minimal()
```

```
## Warning: as.mcmc.brmsfit is deprecated and will eventually be removed.
```

```
# Combine all plots into a grid
grid.arrange(g1, g2, g3, g4, g5, g6, g7, g8, g9, g10, g11, g12, g13, g14, g15, g16, g17, ncol = 3)
```



```
predictions <- predict(fit, newdata = test_data, type = "response")

# Convert probabilities to binary predictions
test_data$predicted_stroke <- ifelse(predictions[,1] > 0.3, 1, 0)

# Evaluate the model
conf_matrix <- table(test_data$stroke, test_data$predicted_stroke)
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
cat("Accuracy:", accuracy, "\n")
```

```
## Accuracy: 0.9363992
```

```r
print(conf_matrix)
```

```
##
##       0   1
##   0 952  14
##   1  51   5
```

```r
# Load necessary libraries
library(brms)
library(dplyr)

# Assuming 'fit' is your fitted model object
# and 'train_data' is your training dataset

# Generate posterior predictions for the training data
y_rep_train <- posterior_predict(fit, newdata = train_data)

# Calculate the mean predicted probability across the posterior samples
predicted_probs_train <- apply(y_rep_train, 2, mean)

# Set a threshold for stroke prediction (e.g., 0.5)
threshold <- 0.25

# Classify the individual based on the predicted probability
predicted_class_train <- ifelse(predicted_probs_train > threshold, 1, 0)

# Create a confusion matrix
confusion_matrix_train <- table(train_data$stroke, predicted_class_train)
print(confusion_matrix_train)
```

```
##    predicted_class_train
##        0    1
##   0 3812   83
##   1  166   27
```

```r
# Calculate training accuracy
training_accuracy <- sum(diag(confusion_matrix_train)) / sum(confusion_matrix_train)
print(paste("Training Accuracy:", training_accuracy))
```

```
## [1] "Training Accuracy: 0.939090019569472"
```

```r
# Create a data frame for the individual
individual_data <- data.frame(
  gender = "Male",
  age = 68,
  hypertension = 0,
  heart_disease = 1,
  ever_married = "Yes",
  work_type = "Private",
  Residence_type = "Urban",
  avg_glucose_level = 229,
  bmi = 36,
  smoking_status = "formerly smoked"
)
```

```r
# Ensure the levels of categorical variables match the training data
individual_data$gender <- factor(individual_data$gender, levels = levels(train_data$gender))
individual_data$ever_married <- factor(individual_data$ever_married, levels = levels(train_data$ever_ma
individual_data$work_type <- factor(individual_data$work_type, levels = levels(train_data$work_type))
individual_data$Residence_type <- factor(individual_data$Residence_type, levels = levels(train_data$Res
individual_data$smoking_status <- factor(individual_data$smoking_status, levels = levels(train_data$smo
```

```r
# Generate posterior predictions for the individual
predicted_probs <- posterior_predict(fit, newdata = individual_data)

# Calculate the mean predicted probability across the posterior samples
predicted_prob <- mean(predicted_probs)

# Print the predicted probability
print(predicted_prob)
```

```
## [1] 0.2015
```

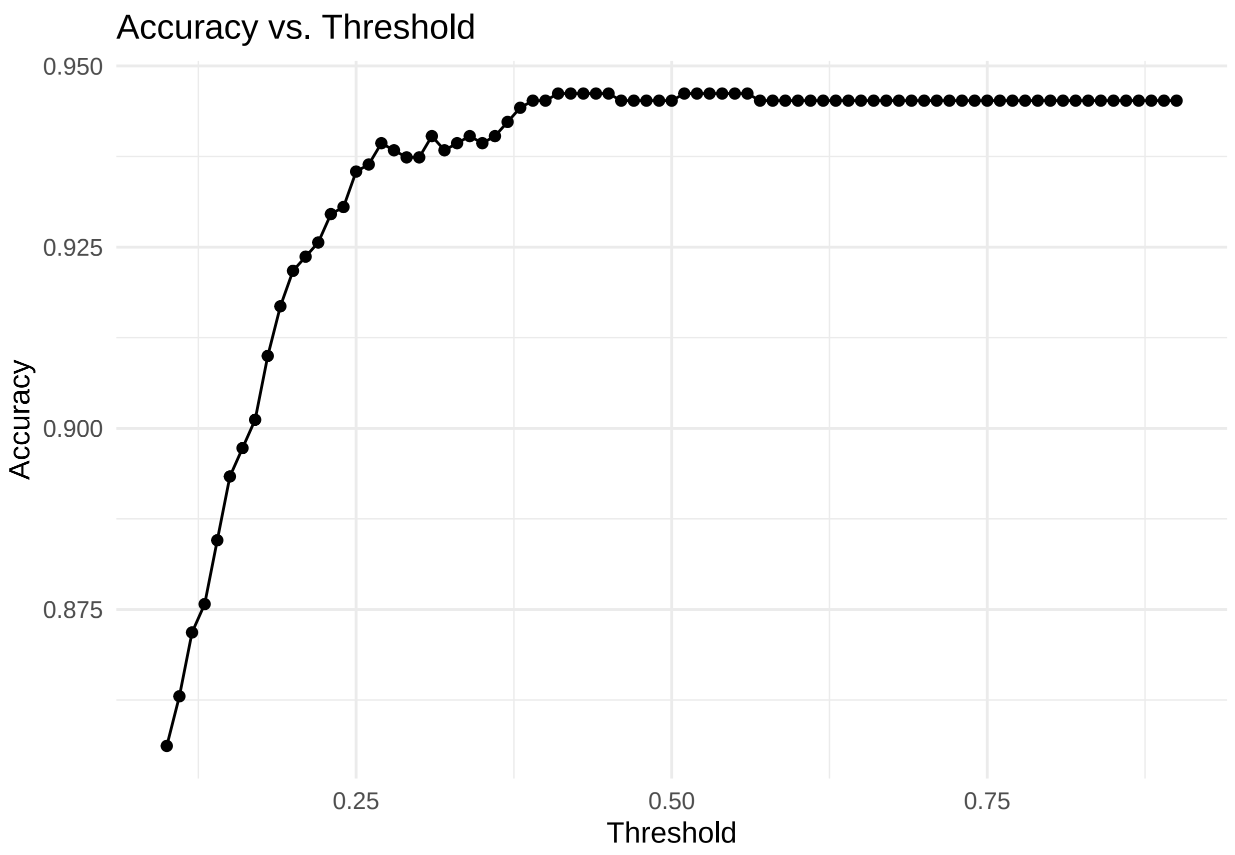
```r
# Define a threshold for classification (e.g., 0.5)
threshold <- 0.5

# Classify the individual based on the predicted probability
predicted_class <- ifelse(predicted_prob > threshold, 1, 0)

# Print the predicted class
if(predicted_class == 1) {
  cat("The model predicts that the individual is at risk of having a stroke.\n")
} else {
  cat("The model predicts that the individual is not at risk of having a stroke.\n")
}
```

```
## The model predicts that the individual is not at risk of having a stroke.
```

```r
# Load necessary libraries
library(brms)
library(dplyr)

# Assuming 'fit' is your fitted model object
# and 'test_data' is your test dataset

# Generate posterior predictions for the test data
y_rep_test <- posterior_predict(fit, newdata = test_data)

# Calculate the mean predicted probability across the posterior samples
predicted_probs <- apply(y_rep_test, 2, mean)

# Define a list of thresholds
thresholds <- seq(0.1, 0.9, by = 0.01)

# Initialize a vector to store accuracy results
accuracies <- numeric(length(thresholds))

# Loop through each threshold and calculate accuracy
for (i in seq_along(thresholds)) {
  threshold <- thresholds[i]
  predicted_class <- ifelse(predicted_probs > threshold, 1, 0)
  confusion_matrix <- table(test_data$stroke, predicted_class)

  # Calculate accuracy
  accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
  accuracies[i] <- accuracy
}

# Combine thresholds and accuracies into a data frame
results <- data.frame(Threshold = thresholds, Accuracy = accuracies)

# Print the results
print(results)
```

```
##    Threshold  Accuracy
## 1       0.10 0.8561644
## 2       0.11 0.8630137
## 3       0.12 0.8718200
## 4       0.13 0.8757339
## 5       0.14 0.8845401
## 6       0.15 0.8933464
## 7       0.16 0.8972603
## 8       0.17 0.9011742
## 9       0.18 0.9099804
## 10      0.19 0.9168297
## 11      0.20 0.9217221
## 12      0.21 0.9236791
## 13      0.22 0.9256360
## 14      0.23 0.9295499
## 15      0.24 0.9305284
## 16      0.25 0.9354207
## 17      0.26 0.9363992
## 18      0.27 0.9393346
## 19      0.28 0.9383562
## 20      0.29 0.9373777
## 21      0.30 0.9373777
## 22      0.31 0.9403131
## 23      0.32 0.9383562
## 24      0.33 0.9393346
## 25      0.34 0.9403131
## 26      0.35 0.9393346
## 27      0.36 0.9403131
## 28      0.37 0.9422701
## 29      0.38 0.9442270
## 30      0.39 0.9452055
## 31      0.40 0.9452055
## 32      0.41 0.9461840
## 33      0.42 0.9461840
## 34      0.43 0.9461840
## 35      0.44 0.9461840
## 36      0.45 0.9461840
## 37      0.46 0.9452055
## 38      0.47 0.9452055
## 39      0.48 0.9452055
## 40      0.49 0.9452055
## 41      0.50 0.9452055
## 42      0.51 0.9461840
## 43      0.52 0.9461840
## 44      0.53 0.9461840
## 45      0.54 0.9461840
## 46      0.55 0.9461840
## 47      0.56 0.9461840
## 48      0.57 0.9452055
## 49      0.58 0.9452055
## 50      0.59 0.9452055
## 51      0.60 0.9452055
## 52      0.61 0.9452055
## 53      0.62 0.9452055
## 54      0.63 0.9452055
## 55      0.64 0.9452055
## 56      0.65 0.9452055
## 57      0.66 0.9452055
## 58      0.67 0.9452055
## 59      0.68 0.9452055
## 60      0.69 0.9452055
## 61      0.70 0.9452055
## 62      0.71 0.9452055
## 63      0.72 0.9452055
## 64      0.73 0.9452055
## 65      0.74 0.9452055
## 66      0.75 0.9452055
## 67      0.76 0.9452055
## 68      0.77 0.9452055
## 69      0.78 0.9452055
## 70      0.79 0.9452055
## 71      0.80 0.9452055
## 72      0.81 0.9452055
## 73      0.82 0.9452055
## 74      0.83 0.9452055
## 75      0.84 0.9452055
## 76      0.85 0.9452055
## 77      0.86 0.9452055
## 78      0.87 0.9452055
## 79      0.88 0.9452055
## 80      0.89 0.9452055
## 81      0.90 0.9452055
```

```r
# Optionally, plot the results
library(ggplot2)
ggplot(results, aes(x = Threshold, y = Accuracy)) +
  geom_line() +
  geom_point() +
  ggtitle("Accuracy vs. Threshold") +
  xlab("Threshold") +
  ylab("Accuracy") +
  theme_minimal()
```



```
# Load necessary libraries
library(brms)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
# Assuming 'fit' is your fitted model object
# and 'test_data' is your test dataset

# Generate posterior predictions for the test data
y_rep_test <- posterior_predict(fit, newdata = test_data)

# Calculate the mean predicted probability across the posterior samples
predicted_probs <- apply(y_rep_test, 2, mean)

# Calculate the ROC curve and AUC
roc_obj <- roc(test_data$stroke, predicted_probs)
```

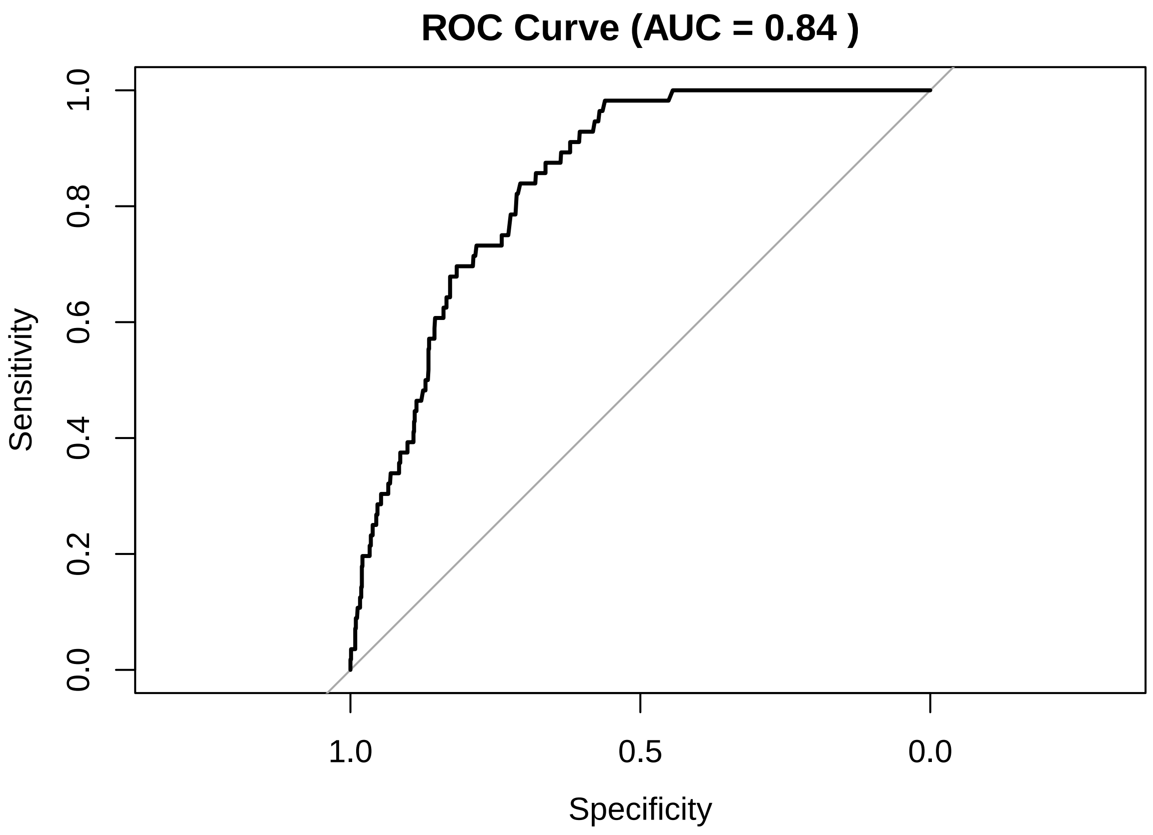
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
# Print the AUC value
auc_value <- auc(roc_obj)
print(paste("ROC-AUC:", auc_value))
```

```
## [1] "ROC-AUC: 0.841162008281574"
```

```r
# Plot the ROC curve
plot(roc_obj, main = paste("ROC Curve (AUC =", round(auc_value, 2), ")"))
```


ROC Curve (AUC = 0.84)
Sensitivity
Specificity