ADD TITLE

```r
library(tidyverse)
library(janitor)
library(ggplot2)
library(moderndive)
library(gapminder)
library(sjPlot)
library(stats)
library(jtools)
library(GGally)
library(gt)
library(pROC)
library(randomForest)
library(caret)
```

1 Introduction

```r
data<-read.csv('D:/desktop/dataset23.csv')
data$yesno<-as.factor(data$yesno)
data <- data[rowSums(data[, 2:6] > 1) == 0, ] # the percentage of total numbe can not be grea
```

```r
data |>
  summarize(
    crl.tot = mean(crl.tot),
    dollar = mean(dollar),
    bang = mean(bang),
    money = mean(money),
    n000 = mean(n000),
    make = mean(make),
            .by = yesno) |>
```

Table 1: summary of mean

yesno	crl.tot	dollar	bang	money	n000	make
n	157.83	0.01	0.05	0.01	0.00	0.03
y	409.26	0.13	0.31	0.15	0.15	0.11

Table 2: summary of median

yesno	crl.tot	dollar	bang	money	n000	make
n	54.00	0.00	0.00	0.00	0.00	0.00
y	190.50	0.06	0.25	0.00	0.00	0.00

```
  gt() |>
  fmt_number(decimals=2)
```

```
data |>
  summarize(
    crl.tot = median(crl.tot),
    dollar = median(dollar),
    bang = median(bang),
    money = median(money),
    n000 = median(n000),
    make = median(make),
    .by = yesno) |>
  gt() |>
  fmt_number(decimals=2)
```

#Most mean values greater than median values may indicate right skewness.

```
cor_matrix <- cor(data[, c("crl.tot", "dollar", "bang", "money", "n000", "make")])
corrplot::corrplot(cor_matrix, method = "number")
```

Figure 1: Corrplot

#correlation

```
ggplot(data, aes(x=yesno, y=crl.tot, fill=yesno)) +
  geom_boxplot()
```

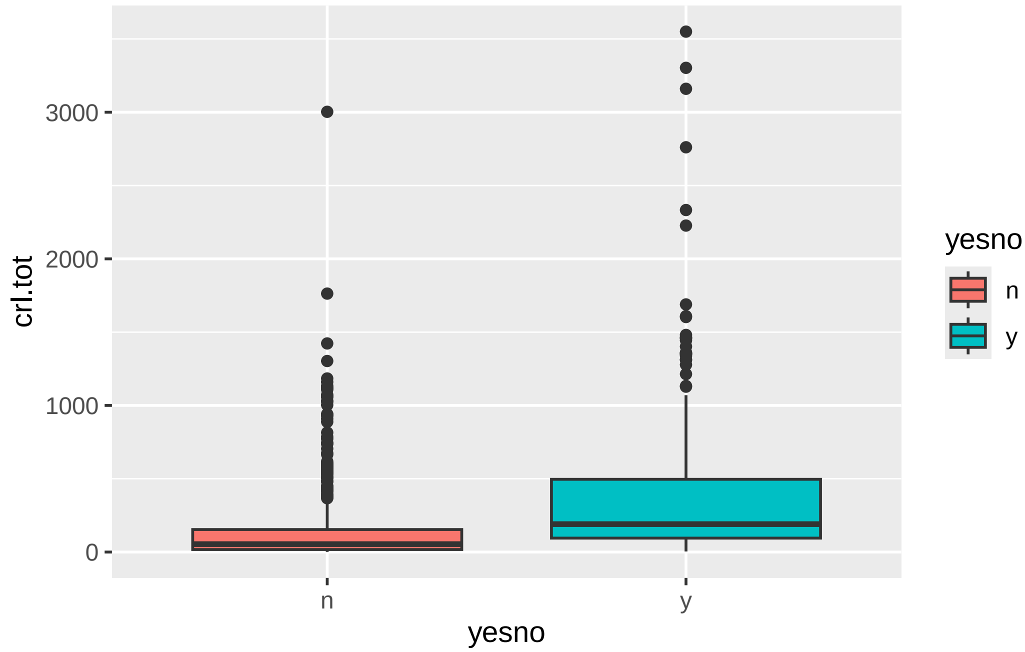

Figure 2: Crl.tot by Class

```
ggplot(data, aes(x=crl.tot, fill=yesno)) +
  geom_density(alpha=0.5)
```

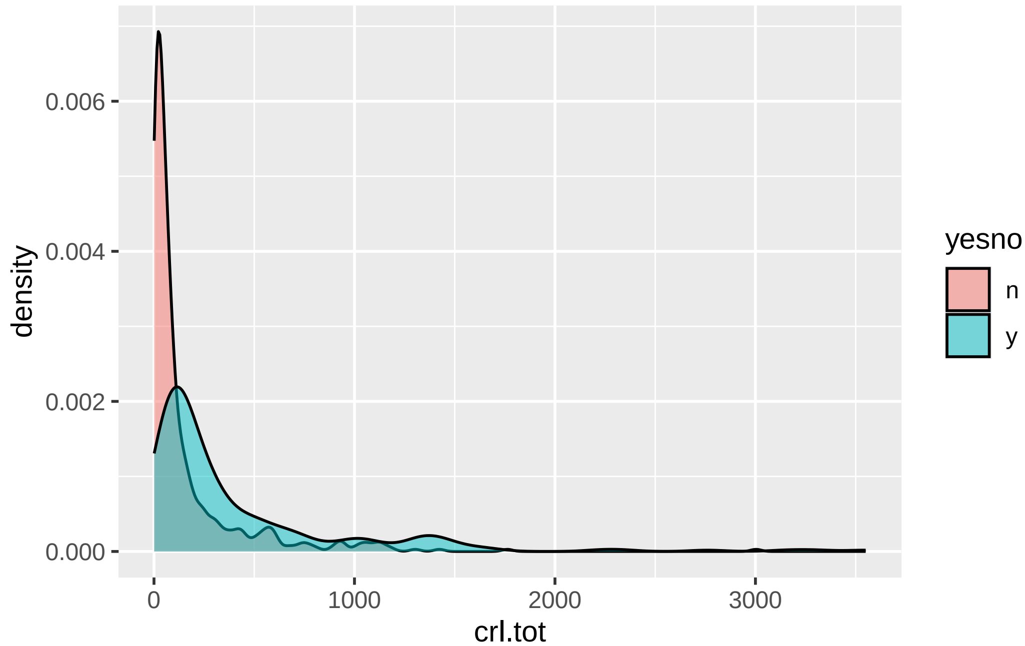

Figure 3: Crl.tot Density by Class

```
ggplot(data, aes(x=yesno, y=bang, fill=yesno)) +
  geom_boxplot()
```

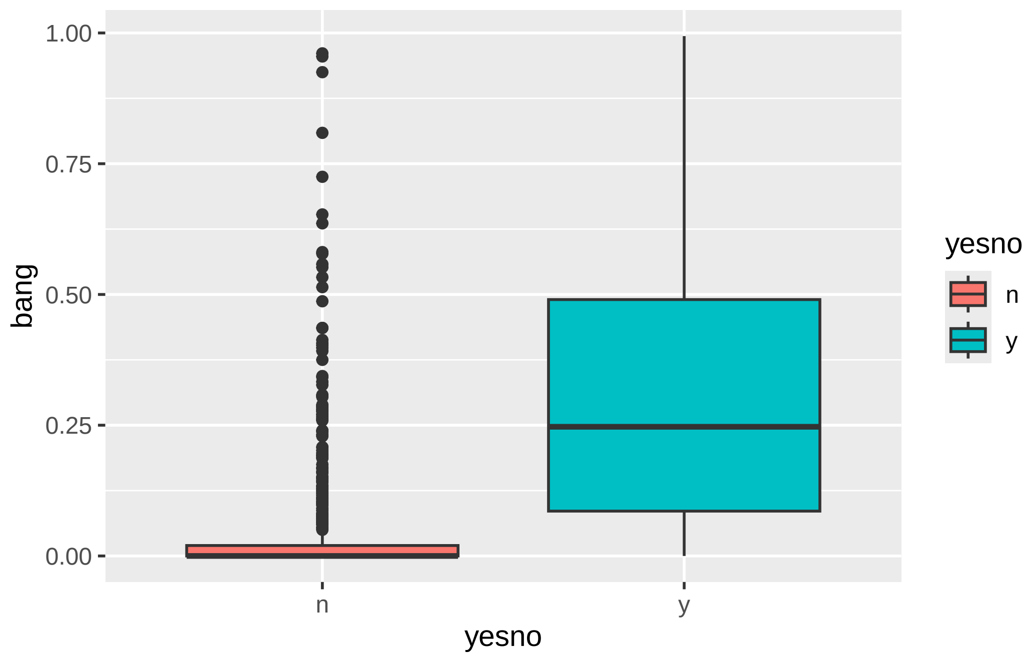

Figure 4: Bang by Class

```
ggplot(data, aes(x=bang, fill=yesno)) +
  geom_density(alpha=0.5)
```

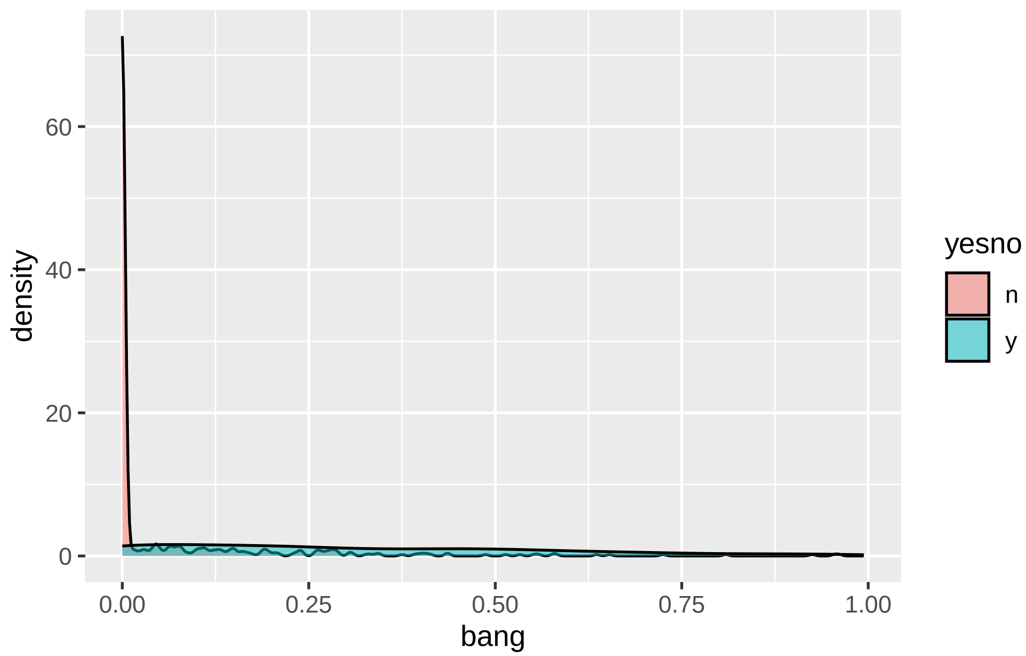

Figure 5: Bang Density by Class

```
ggplot(data, aes(x=yesno, y=money, fill=yesno)) +
  geom_boxplot()
```

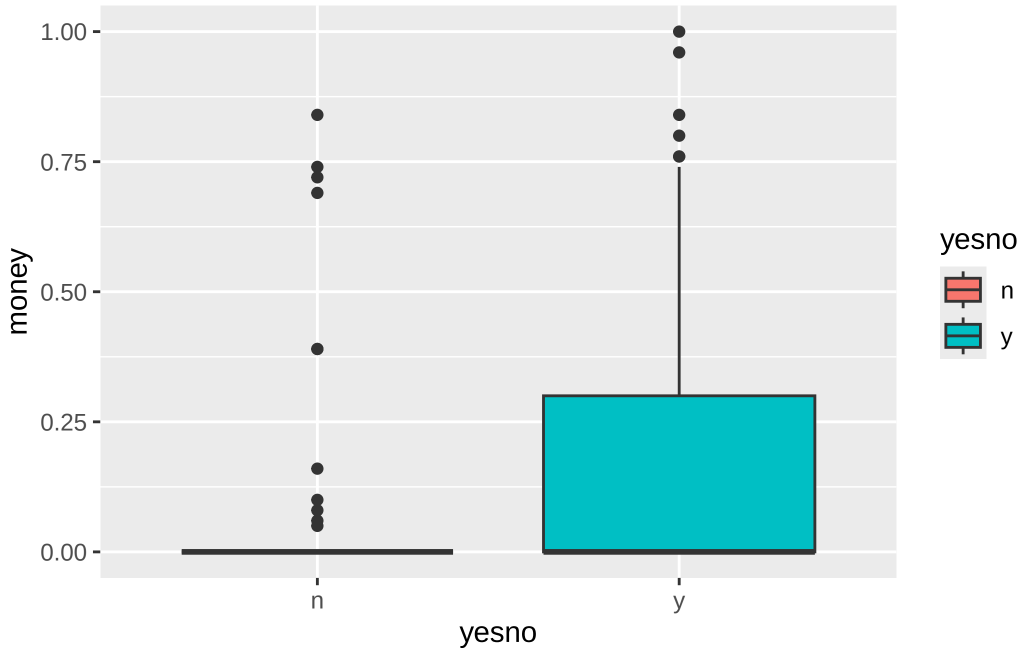

Figure 6: Money by Class

```
ggplot(data, aes(x=money, fill=yesno)) +
  geom_density(alpha=0.5)
```

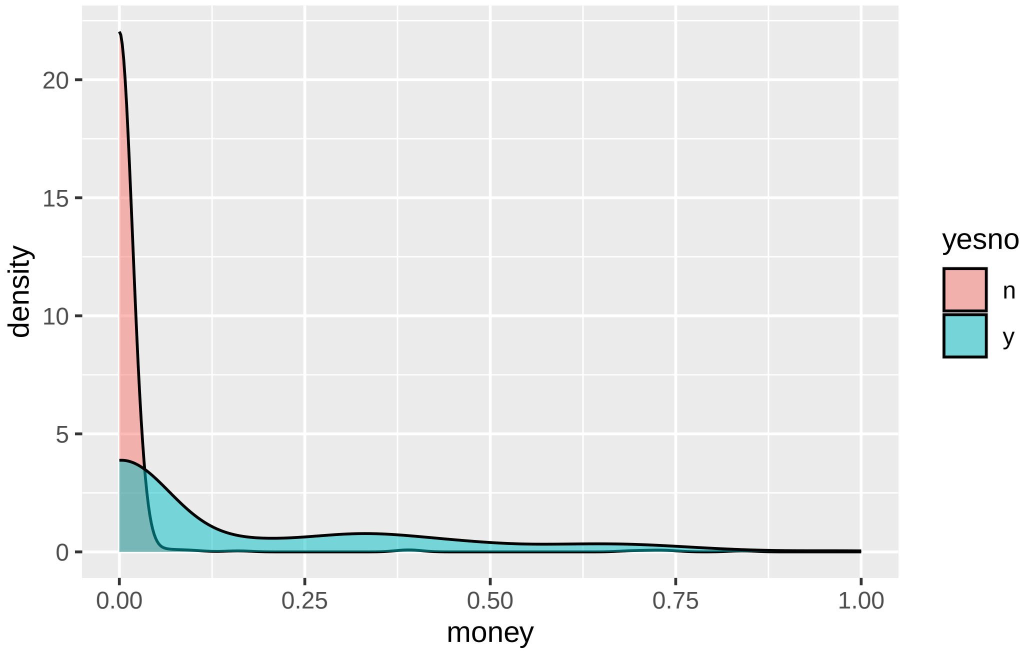

Figure 7: Money Density by Class

```
ggplot(data, aes(x=yesno, y=dollar, fill=yesno)) +
  geom_boxplot()
```

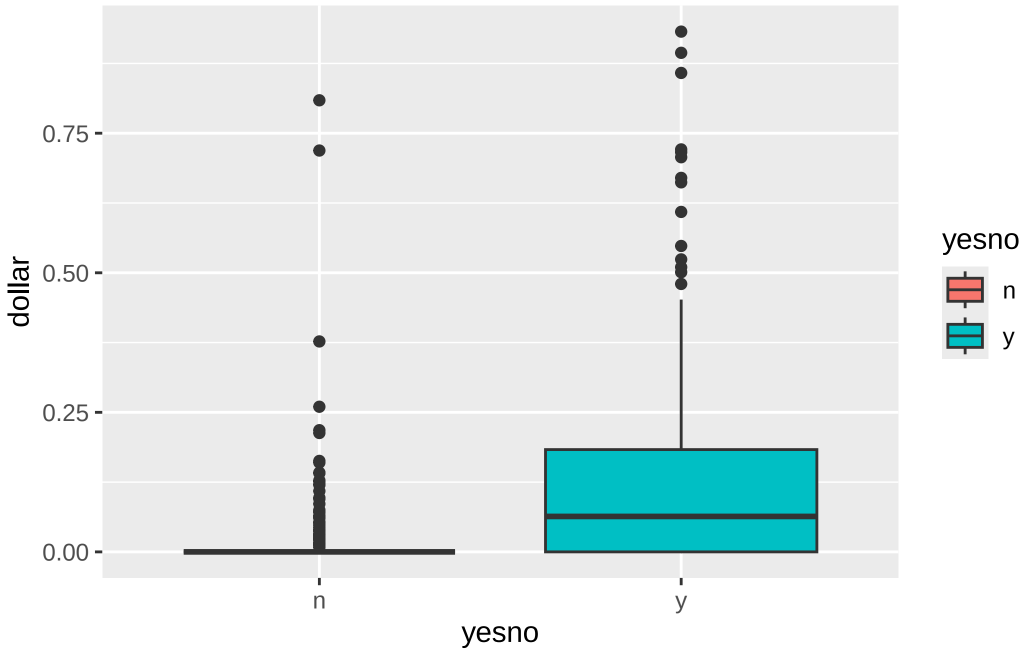

Figure 8: Dollar by Class

```
ggplot(data, aes(x=dollar, fill=yesno)) +
  geom_density(alpha=0.5)
```

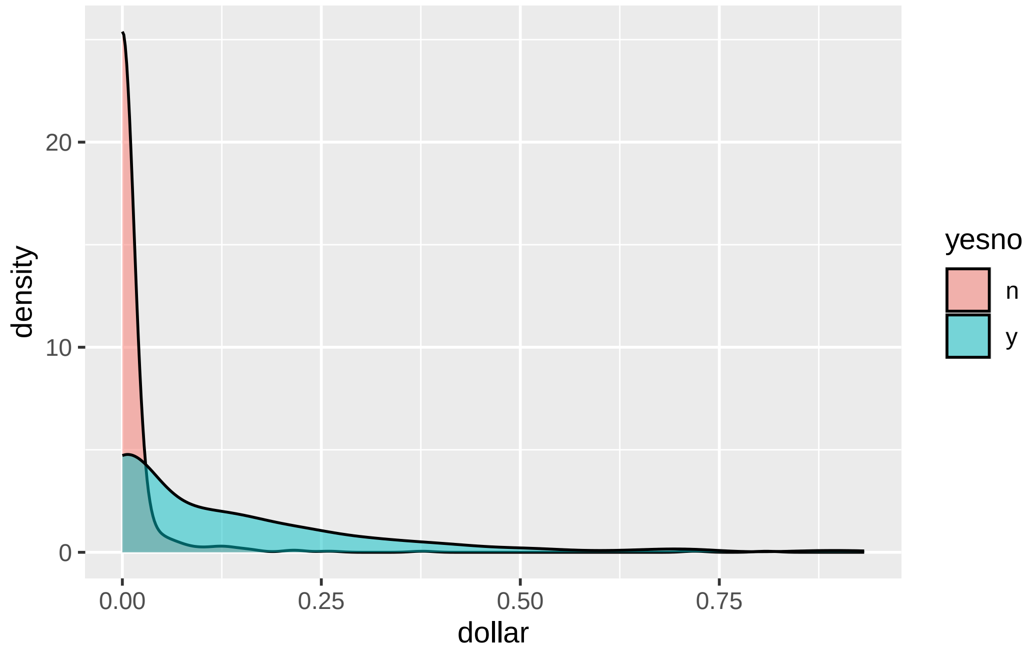

Figure 9: Dollar Density by Class

```
ggplot(data, aes(x=yesno, y=n000, fill=yesno)) +
  geom_boxplot()
```

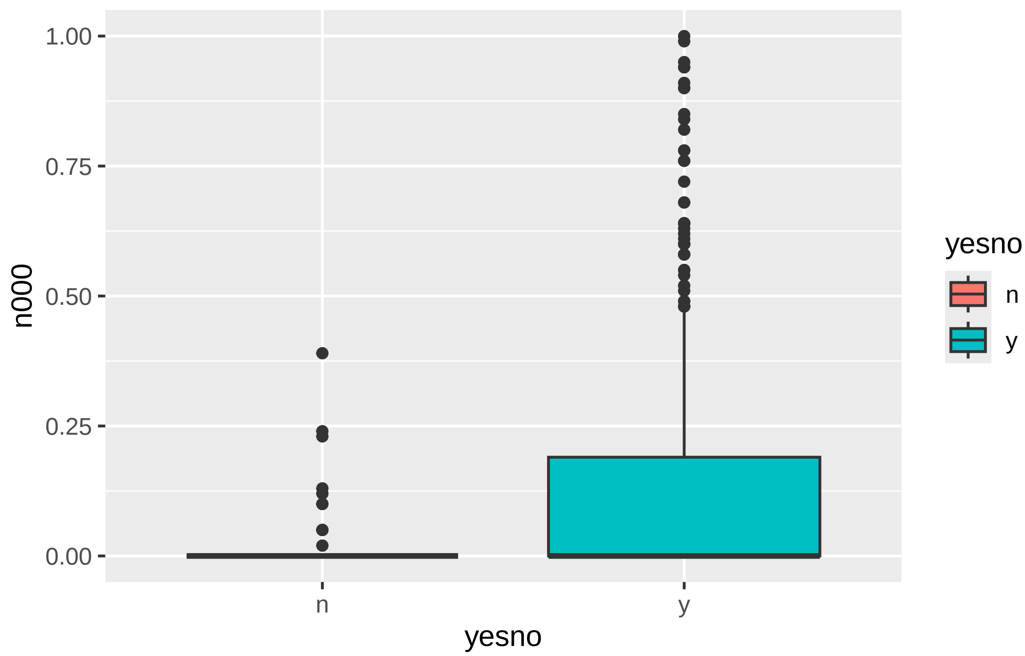

Figure 10: N000 by Class

```
ggplot(data, aes(x=n000, fill=yesno)) +
  geom_density(alpha=0.5)
```

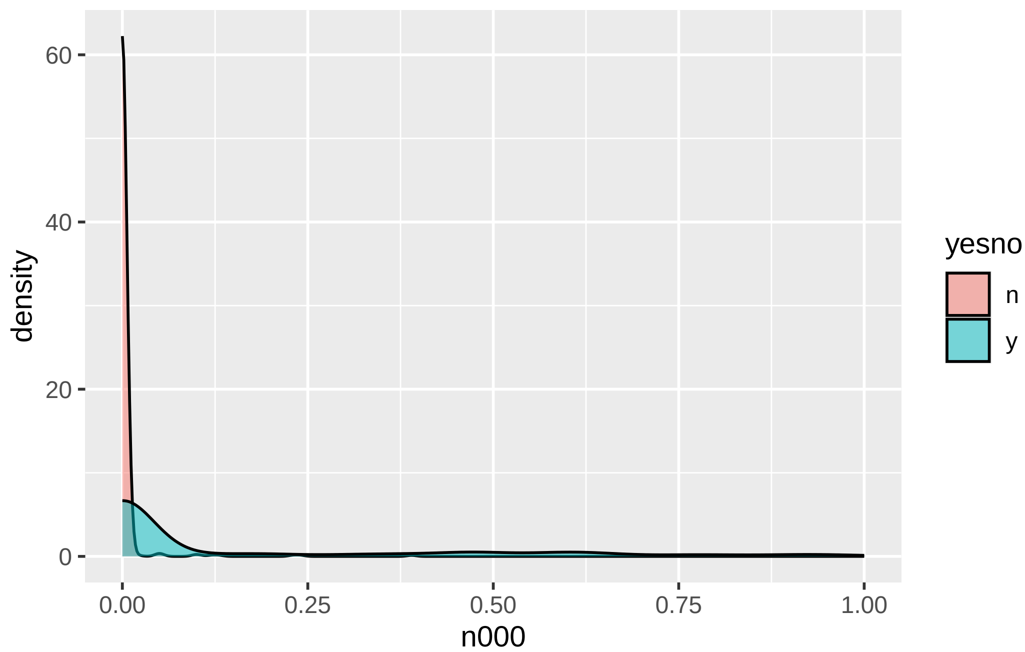

Figure 11: N000 Density by Class

```
ggplot(data, aes(x=yesno, y=make, fill=yesno)) +
  geom_boxplot()
```

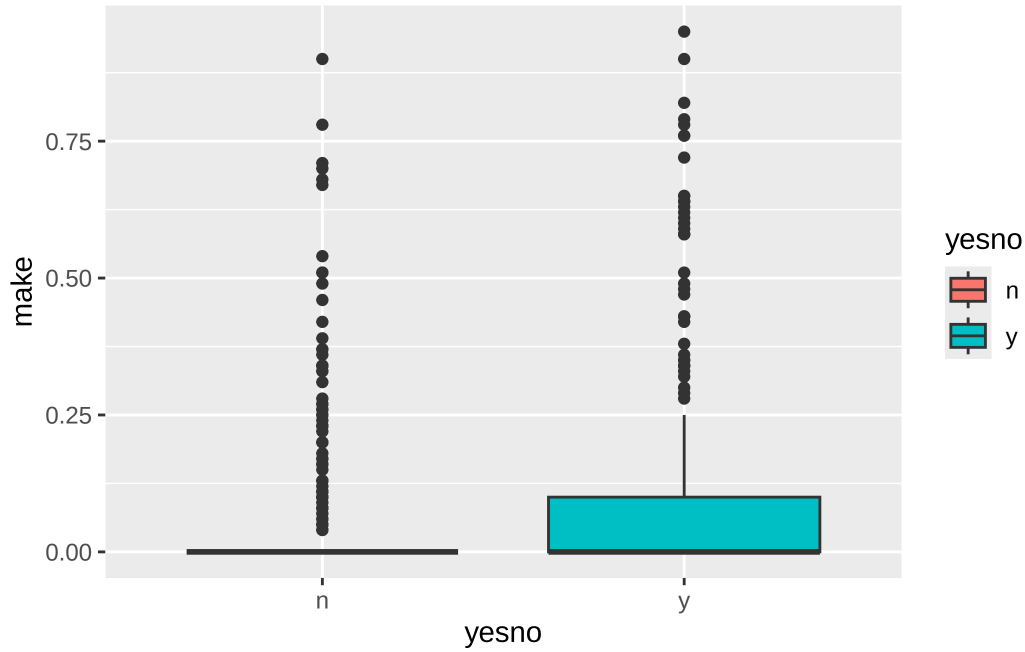

Figure 12: Make by Class

```
ggplot(data, aes(x=make, fill=yesno)) +
  geom_density(alpha=0.5)
```

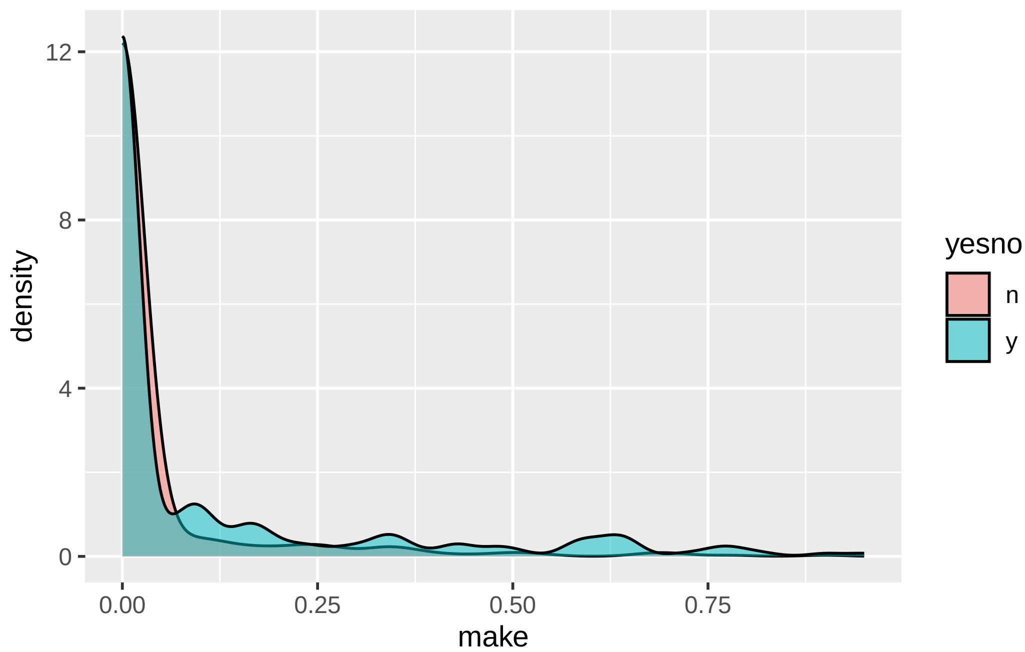

Figure 13: Make Density by Class

#Replace values below the 1st percentile with the 1st percentile value and values over the 99th percentile with the 99th percentile value, then standardize the data to mitigate the effects of outliers and right skewness. Due to the wide distribution and right-skewed nature of crl.tot, we substitute it with log(data$crl.tot + 1).

```
data1<-data
win <- function(x, lower_perc = 0.01, upper_perc = 0.99) {
  x <- as.numeric(x)
  q <- quantile(x, probs = c(lower_perc, upper_perc), na.rm = TRUE)
  x[x < q[1]] <- q[1]
  x[x > q[2]] <- q[2]
  return(x)
}
numeric_vars <- c("crl.tot", "dollar", "bang", "money", "n000", "make")
data[numeric_vars] <- lapply(data[numeric_vars], win)
data[,1:6]<-scale(data[,1:6])
data$crl.tot_log <- log(data$crl.tot+1)
```

$$Y_i \sim \text{Bernoulli}(p_i)$$

$$\log\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1\text{crl.tot}_i + \beta_2\text{dollar}_i + \beta_3\text{bang}_i + \beta_4\text{money}_i + \beta_5\text{n000}_i + \beta_6\text{make}_i$$

-Y_i is ..

-$\mathbf{crl.tot}_i$ is..

-$\mathbf{dollar}_i$ is..

-$\mathbf{bang}_i$ is..

-$\mathbf{money}_i$ is

-$\mathbf{n000}_i$ is

-$\mathbf{make}_i$ is ..

$$\log\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 log(\text{crl.tot}_i) + \beta_2\text{dollar}_i + \beta_3\text{bang}_i + \beta_4\text{money}_i + \beta_5\text{n000}_i + \beta_6\text{make}_i$$

-Y_i is ..

-$log(\mathbf{crl.tot}_i)$ is..

-$\mathbf{dollar}_i$ is..

-$\mathbf{bang}_i$ is..

-$\mathbf{money}_i$ is

-$\mathbf{n000}_i$ is

-$\mathbf{make}_i$ is ..

```r
model_original <- glm(yesno ~ crl.tot + dollar + bang + money + n000 + make,
             family = binomial(link = "logit"),
             data = data1)

summary(model_original)
```

```
Call:
glm(formula = yesno ~ crl.tot + dollar + bang + money + n000 +
    make, family = binomial(link = "logit"), data = data1)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.3209859  0.1563256 -14.847  < 2e-16 ***
crl.tot      0.0007367  0.0002964   2.486 0.012924 *
dollar       7.1037071  1.5468059   4.593 4.38e-06 ***
bang         5.0489232  0.5471349   9.228  < 2e-16 ***
money        4.8483493  1.0004289   4.846 1.26e-06 ***
n000         7.6756922  2.0675084   3.713 0.000205 ***
make        -0.7336087  0.9511158  -0.771 0.440521
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1008.14  on 798  degrees of freedom
Residual deviance:  578.74  on 792  degrees of freedom
AIC: 592.74

Number of Fisher Scoring iterations: 7
```

```r
model_scale <- glm(yesno ~ crl.tot + dollar + bang + money + n000 + make,
             family = binomial(link = "logit"),
             data = data)

summary(model_scale)
```

```
Call:
glm(formula = yesno ~ crl.tot + dollar + bang + money + n000 +
    make, family = binomial(link = "logit"), data = data)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.5182     0.1385  -3.742 0.000183 ***
crl.tot       0.2906     0.1217   2.388 0.016934 *
dollar        0.8696     0.1824   4.767 1.87e-06 ***
bang          1.1251     0.1216   9.250  < 2e-16 ***
```

```
money         0.7472     0.1542   4.844 1.27e-06 ***
n000          1.2408     0.3352   3.702 0.000214 ***
make         -0.1137     0.1449  -0.785 0.432501
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1008.14  on 798  degrees of freedom
Residual deviance:  575.17  on 792  degrees of freedom
AIC: 589.17

Number of Fisher Scoring iterations: 7
```

```r
model_scale_log <- glm(yesno ~ bang + crl.tot_log + dollar+money+n000+make,
               family = binomial(link = "logit"), data = data)
summary(model_scale_log)
```

```
Call:
glm(formula = yesno ~ bang + crl.tot_log + dollar + money + n000 +
    make, family = binomial(link = "logit"), data = data)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.3773     0.1367  -2.761 0.005767 **
bang          1.1332     0.1222   9.277  < 2e-16 ***
crl.tot_log   0.6552     0.1645   3.984 6.78e-05 ***
dollar        0.8414     0.1776   4.737 2.17e-06 ***
money         0.7068     0.1528   4.627 3.71e-06 ***
n000          1.1532     0.3164   3.644 0.000268 ***
make         -0.1511     0.1473  -1.026 0.304915
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1008.14  on 798  degrees of freedom
Residual deviance:  565.11  on 792  degrees of freedom
AIC: 579.11

Number of Fisher Scoring iterations: 7
```

```r
AIC(model_original,model_scale,model_scale_log)
```

```
                df      AIC
model_original   7 592.7427
model_scale      7 589.1703
model_scale_log  7 579.1087
```

```r
BIC(model_original,model_scale,model_scale_log)
```

```
                df      BIC
model_original   7 625.5262
model_scale      7 621.9538
model_scale_log  7 611.8922
```

```r
plot_model(model_scale_log, show.values = TRUE, show.p = TRUE)
```

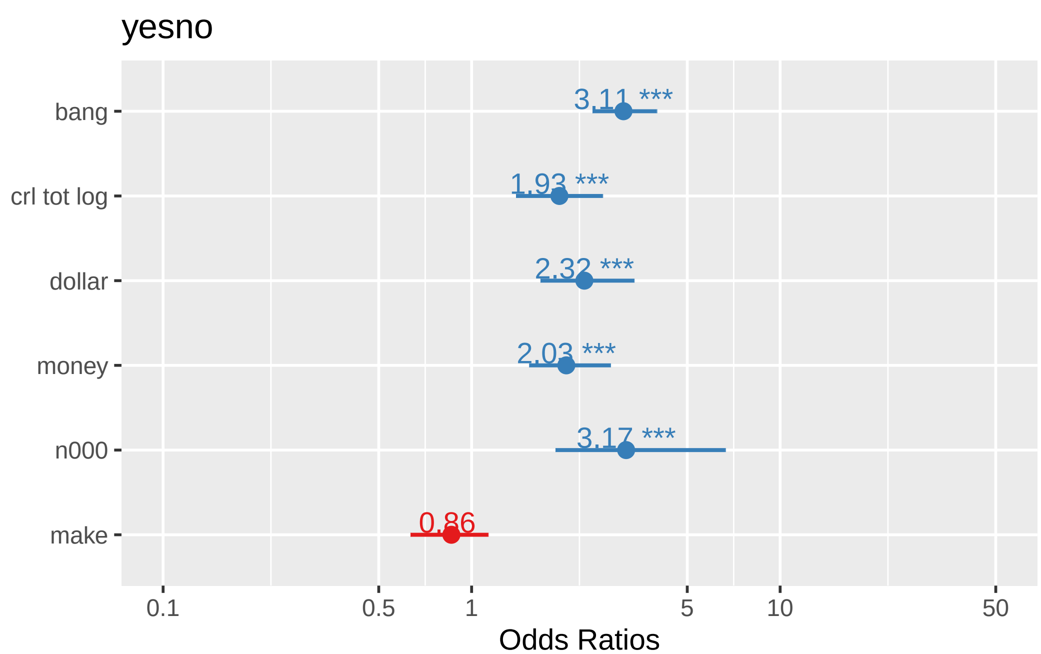

Figure 14: Coefficient Estimates with p-Values

```
plot_model(model_scale_log, type = "pred", title = "",col='steelblue')
```

$bang

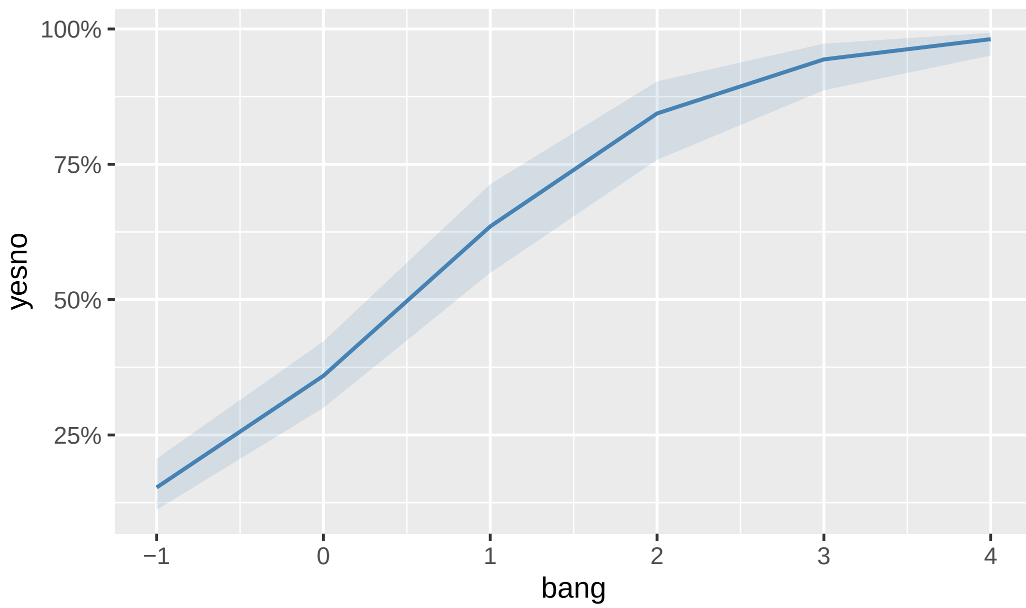

Figure 15: Marginal Predicted Effects

$crl.tot_log

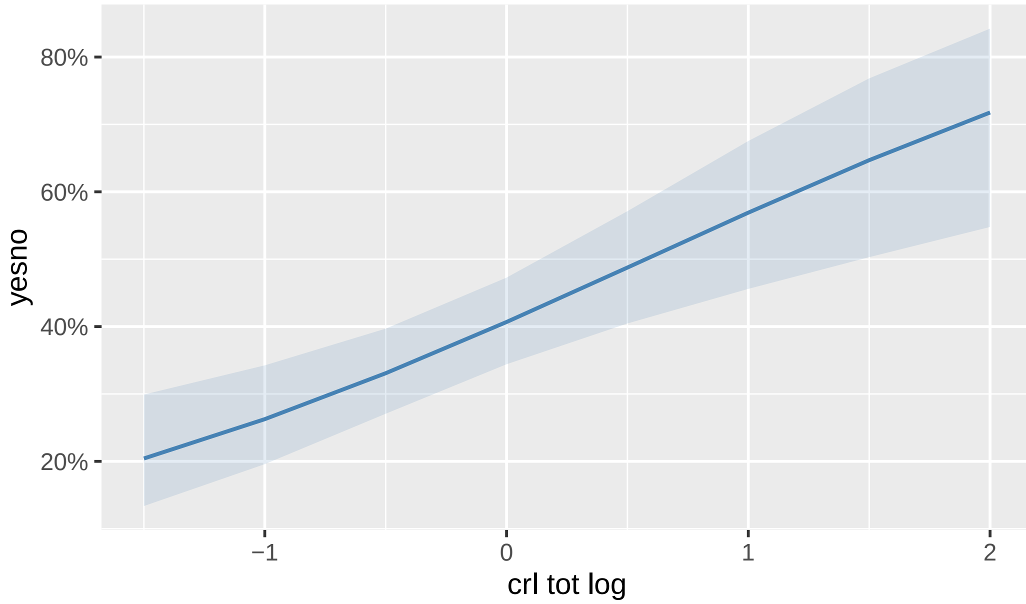

Figure 16: Marginal Predicted Effects

`$dollar`

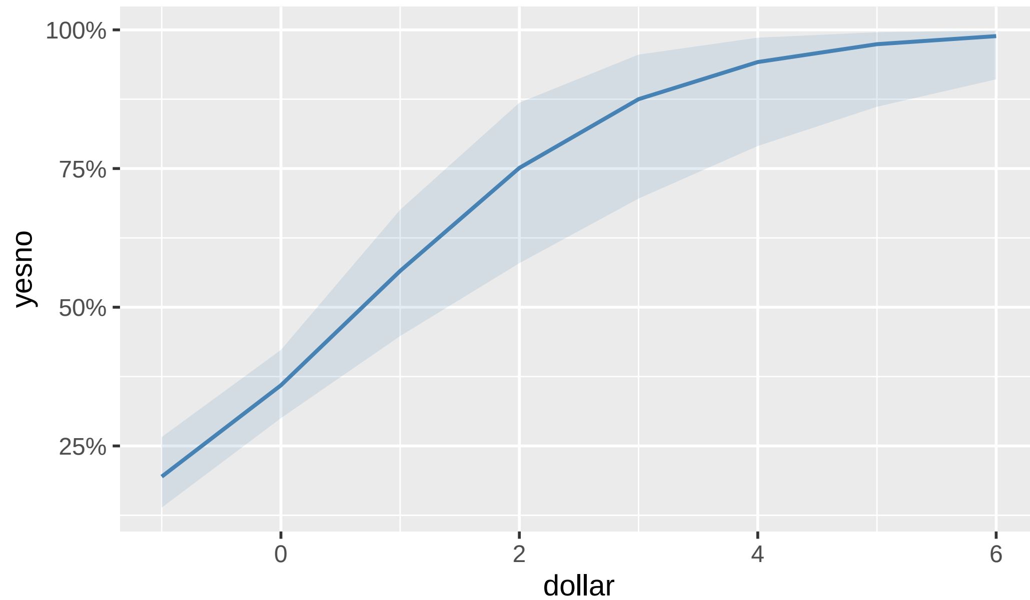

Figure 17: Marginal Predicted Effects

`$money`

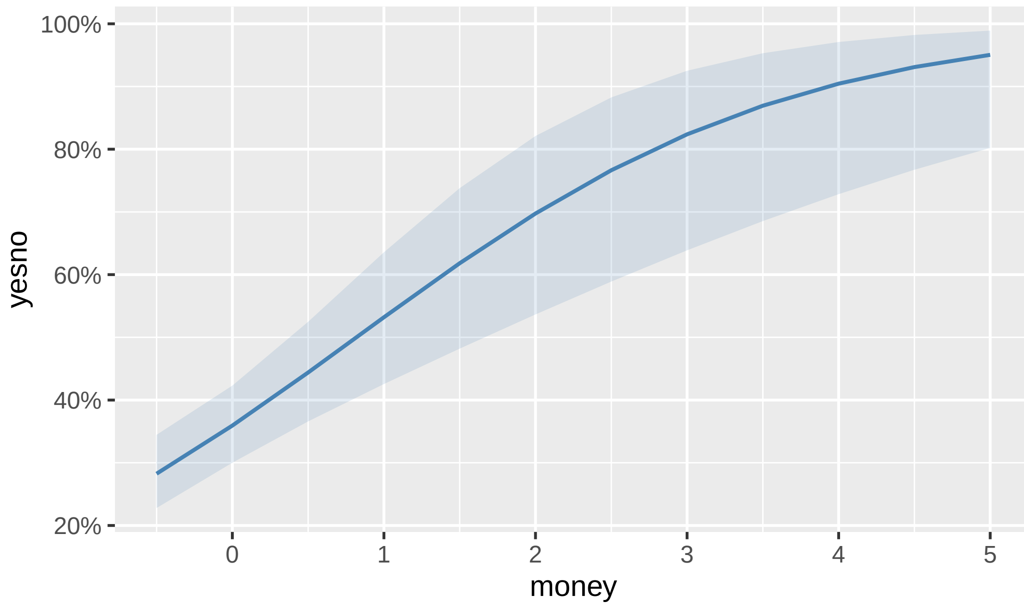

Figure 18: Marginal Predicted Effects

`$n000`

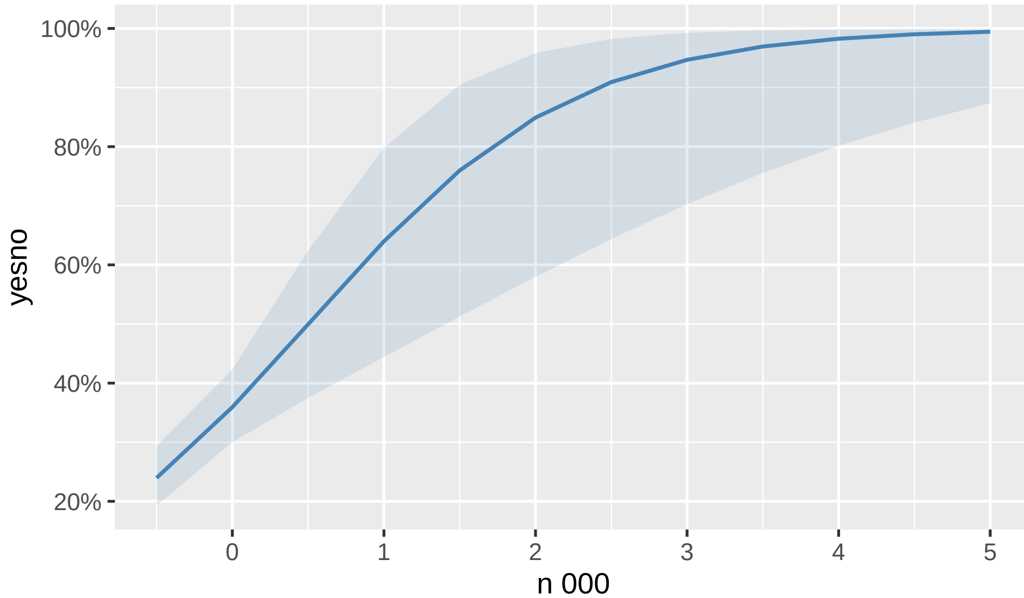

Figure 19: Marginal Predicted Effects

```
$make
```

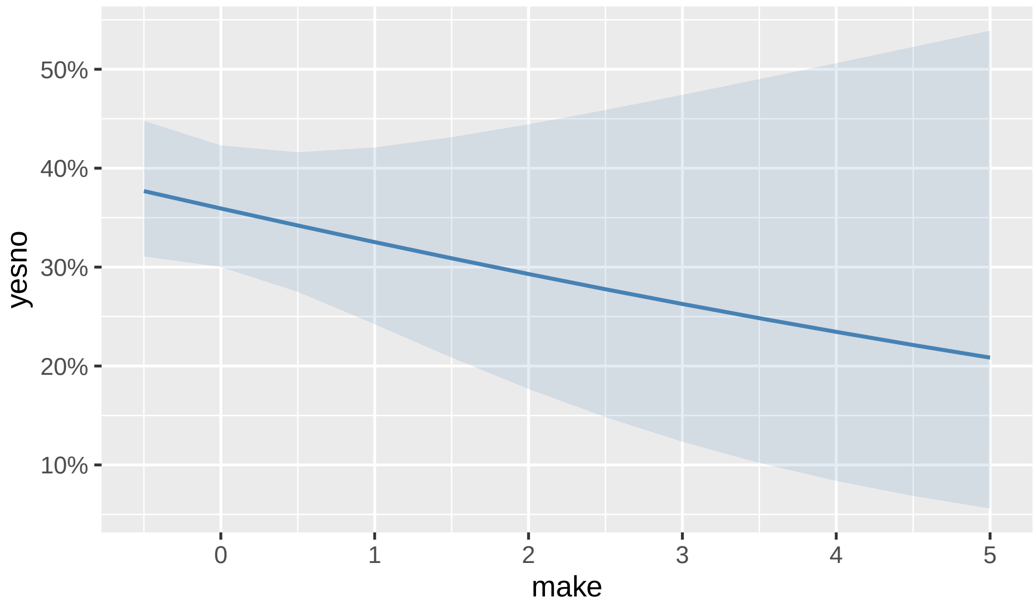

Figure 20: Marginal Predicted Effects

2 Further Work

```
set.seed(123)
index <- createDataPartition(data$yesno, p = 0.7, list = FALSE)
train_data <- data[index, ]
test_data <- data[-index, ]
```

```
glm_model <- glm(yesno ~ bang + crl.tot_log + dollar+money+n000+make, data = train_data, fam

glm_pred_prob <- predict(glm_model, newdata = test_data, type = "response")
glm_pred_class <- ifelse(glm_pred_prob > 0.5,'y','n')
glm_confusion <- confusionMatrix(factor(glm_pred_class), factor(test_data$yesno))
glm_roc <- roc(test_data$yesno, glm_pred_prob)
```

```
set.seed(123)
rf_model <- randomForest(yesno ~ bang + crl.tot_log + dollar+money+n000+make, data = train_da

rf_pred_prob <- predict(rf_model, newdata = test_data, type = "prob")[, 2]
```

```r
rf_pred_class <- ifelse(rf_pred_prob > 0.5,'y','n')

rf_confusion <- confusionMatrix(factor(rf_pred_class), factor(test_data$yesno))
rf_roc <- roc(test_data$yesno, rf_pred_prob)
```

```r
varImpPlot(rf_model)
```

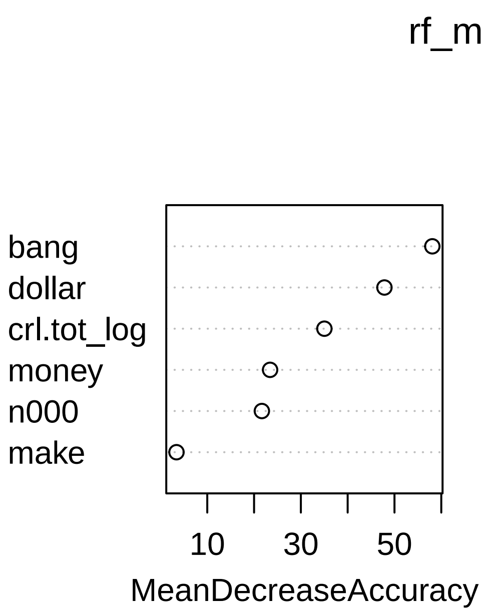

Figure 21: RF Variable Importance Ranking

```r
get_model_metrics <- function(model_name, confusion,k) {
  data.frame(
    Model = model_name,
    Accuracy = confusion$overall["Accuracy"],
    Sensitivity = confusion$byClass["Sensitivity"],
    Specificity = confusion$byClass["Specificity"],
    Precision = confusion$byClass["Precision"],
    AUC=as.numeric(k$auc),
    stringsAsFactors = FALSE
  )
}
results <- bind_rows(
```

```r
  get_model_metrics("Random Forest",
                    confusion = rf_confusion,k=rf_roc),
  get_model_metrics("GLM",
                    confusion = glm_confusion,k=glm_roc)
) %>%
  mutate(across(-Model, ~ round(., 3)))

knitr::kable(results, align = "c")
```

	Model	Accuracy	Sensitivity	Specificity	Precision	AUC
Accuracy...1	Random Forest	0.874	0.919	0.782	0.897	0.932
Accuracy...2	GLM	0.870	0.938	0.731	0.878	0.916

```r
plot(glm_roc, col = "blue")
lines(rf_roc, col = "red")
legend("bottomright", legend = c("GLM", "randomForest"), col = c("blue", "red"), lwd = 2)
```

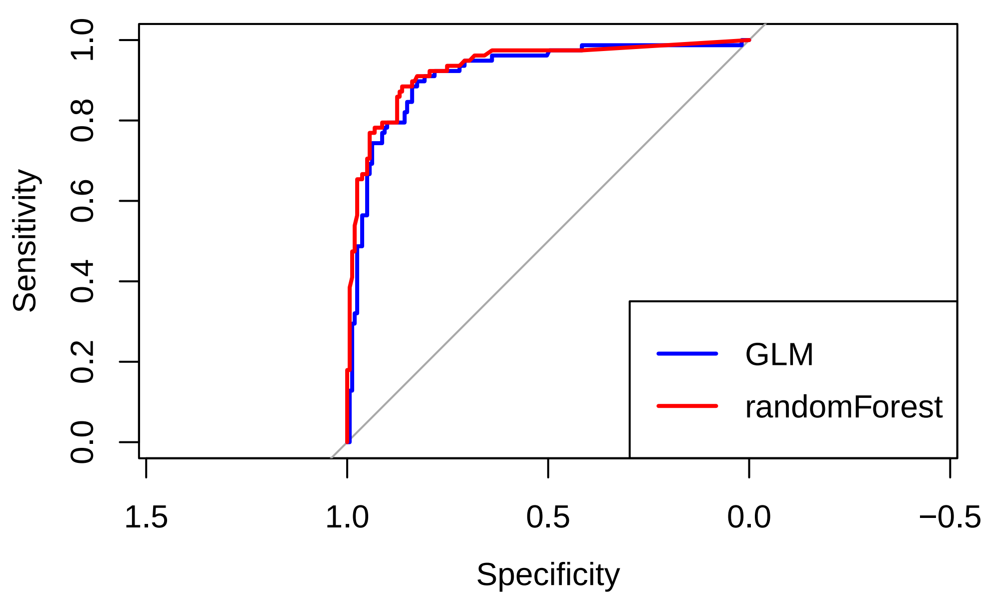

Figure 22: ROC Curve Comparison (GLM & RF)