Homework

Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```
load("C:/Users/Shombit Roy/Downloads/VacData.Rdata")

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
```

```
library(sp)
library(ggplot2)
library(spdep)
```

```
Loading required package: spData

To access larger datasets in this package, install the spDataLarge
package with: `install.packages('spDataLarge',
repos='https://nowosad.github.io/drat/', type='source')`

Loading required package: sf

Linking to GEOS 3.11.2, GDAL 3.7.2, PROJ 9.3.0; sf_use_s2() is TRUE
```

```
library(sf)
library(lmtest)
```

```
Loading required package: zoo
```

```
Attaching package: 'zoo'

The following objects are masked from 'package:base':

    as.Date, as.Date.numeric
```

```
library(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✓ forcats   1.0.0     ✓ stringr   1.5.1
✓ lubridate 1.9.3     ✓ tibble    3.2.1
✓ purrr     1.0.2     ✓ tidyr     1.3.1
✓ readr     2.1.5

── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become
errors
```

```
data <- read.csv("C:/Users/Shombit Roy/Downloads/finalCVD (2).csv")

indices = data %>% select(LocationID) %>% mutate(LocationID = as.character(LocationID)) 

mergedData_2 <- merge(US_Conus_VacSocial, data, by.x = "GEOID", by.y = "LocationID")
my_data_2 <- subset(mergedData_2, select = -c(perc_vac, population, population.persqkm, 

library(randomForest)
```

```
Warning: package 'randomForest' was built under R version 4.3.3

randomForest 4.7-1.1
Type rfNews() to see new features/changes/bug fixes.

Attaching package: 'randomForest'

The following object is masked from 'package:ggplot2':

    margin

The following object is masked from 'package:dplyr':

    combine
```

```
data <- my_data_2[,c("Data_Value", "perc_asian", "perc_white", "perc_hispanic", "perc_af
```

```
train_indices <- sample(1:nrow(data), size = floor(0.8 * nrow(data)))
train_data <- data[train_indices, ]
test_data <- data[-train_indices, ]

# Fitting the Random Forest model
rf_model <- randomForest(Data_Value ~ perc_asian + perc_white + perc_hispanic + perc_afr

predictions <- predict(rf_model, newdata = test_data)

mse <- mean((predictions - test_data$Data_Value)^2)
mse
```

```
[1] 734232.8
```

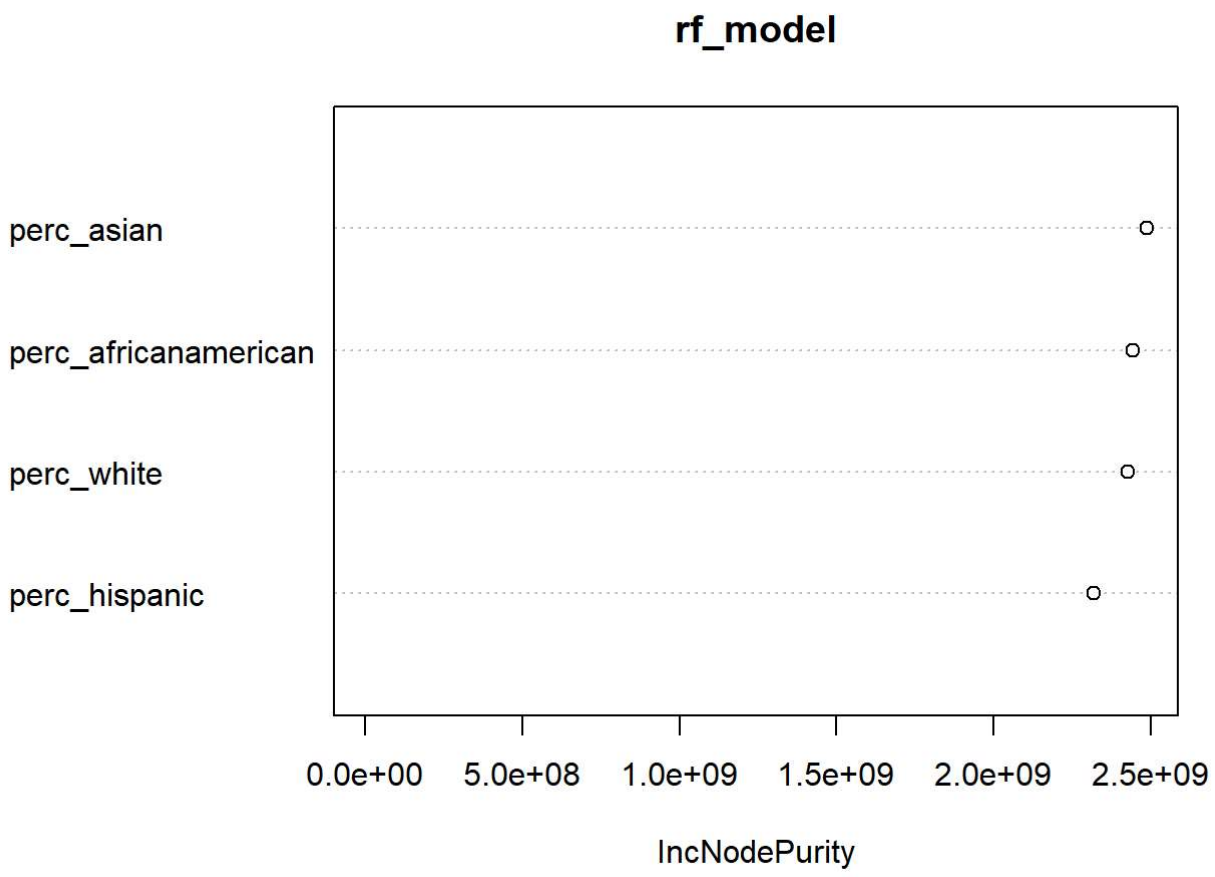
```
rmse <- sqrt(mse)

summary(rf_model)
```

```
               Length Class  Mode
call                4 -none- call
type                1 -none- character
predicted      190771 -none- numeric
mse                10 -none- numeric
rsq                10 -none- numeric
oob.times      190771 -none- numeric
importance          4 -none- numeric
importanceSD        0 -none- NULL
localImportance     0 -none- NULL
proximity           0 -none- NULL
ntree               1 -none- numeric
mtry                1 -none- numeric
forest             11 -none- list
coefs               0 -none- NULL
y              190771 -none- numeric
test                0 -none- NULL
inbag               0 -none- NULL
terms               3 terms  call
```

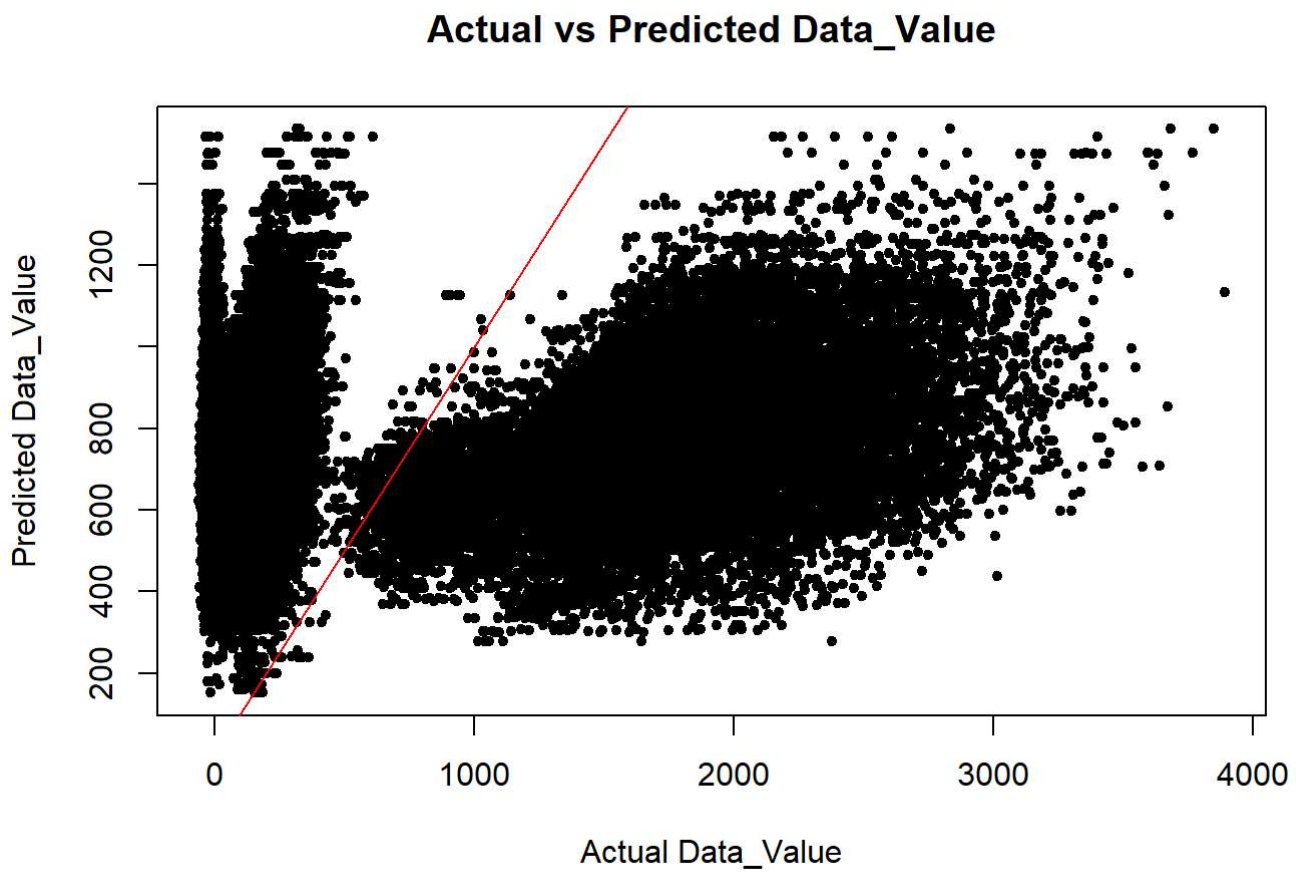
```
importance(rf_model)
```

```
                   IncNodePurity
perc_asian            2487736641
perc_white            2429271315
perc_hispanic         2319816854
perc_africanamerican  2445427097
```

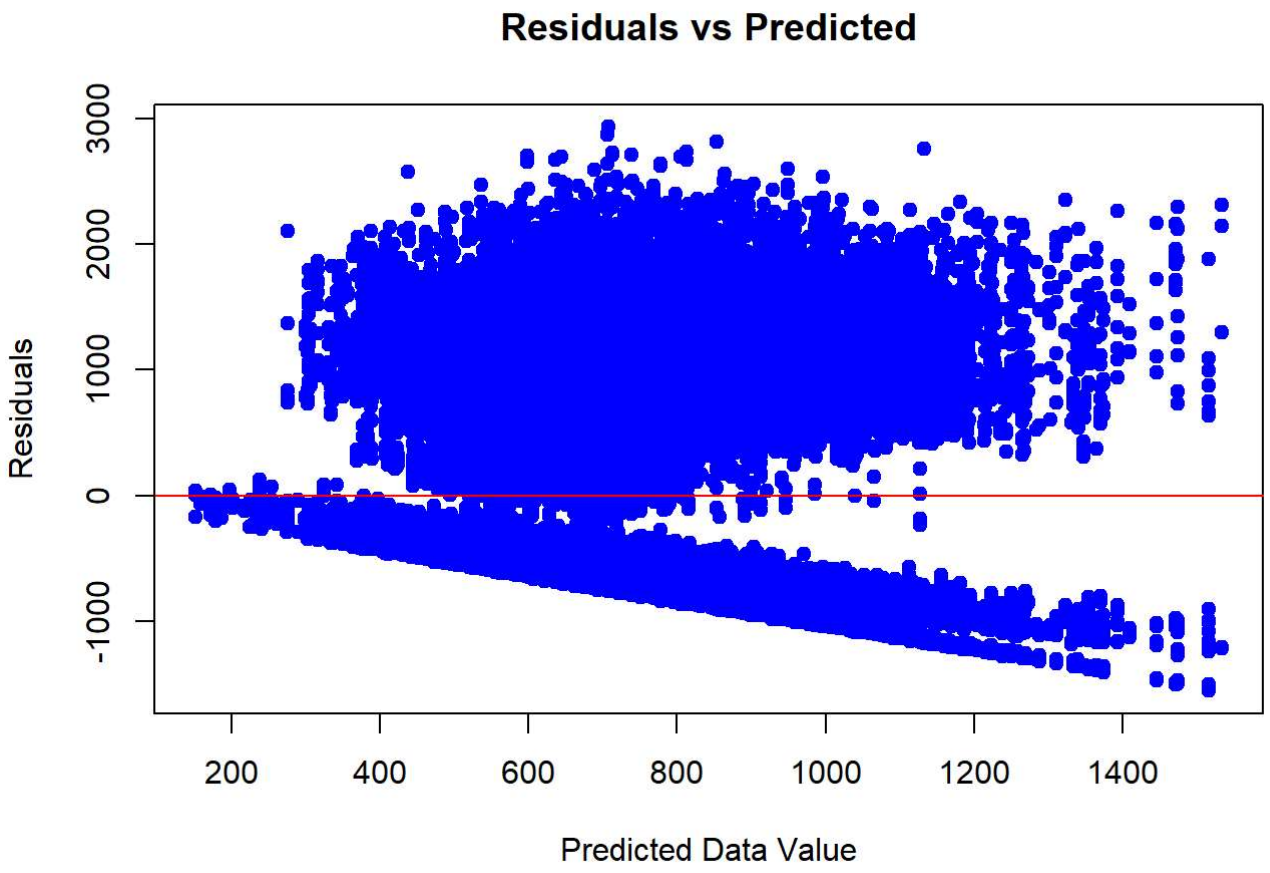
```
varImpPlot(rf_model)
```




```
# Plot of actual vs. predicted values
df_plot = plot(test_data$Data_Value, predictions, main = "Actual vs Predicted Data_Value
     xlab = "Actual Data_Value", ylab = "Predicted Data_Value", pch = 20)
abline(a = 0, b = 1, col = "red")
```




```
# Plot of residuals
residuals <- test_data$Data_Value - predictions
plot(predictions, residuals, main = "Residuals vs Predicted",
     xlab = "Predicted Data Value", ylab = "Residuals", pch = 19, col = 'blue')
abline(h = 0, col = "red")
```



From the model it seems that the perc_asian is the most important factor in play in our data from the Varimpplot function. Also I used 80 -20 split is because this split is known as the Pareto Principle, so i just used that as my rule of thumb.