Comprehensive Data Analysis and Modeling for Glow_Bonemed Dataset from aplore3

Jessica McPhaul, Rafia Mirza, Caleb Thornsbury

2024-04-06

For a detailed exploration and insights from an initial analysis of the Glow_Bonemed dataset, refer to the detailed report on RPubs.

Preliminaries

Load necessary libraries

```
library(ggplot2)
library(dplyr)
library(caret)
library(pROC)
library(car)
library(effects)
## Warning: package 'effects' was built under R version 4.3.3
library(lmtest)
## Warning: package 'lmtest' was built under R version 4.3.3
library(sandwich)
## Warning: package 'sandwich' was built under R version 4.3.3
library(glmnet)
## Warning: package 'glmnet' was built under R version 4.3.3
library(MASS)
library(broom)
library(tidyr)
library(kableExtra)
## Warning: package 'kableExtra' was built under R version 4.3.3
library(aplore3)
```

```
## Warning: package 'aplore3' was built under R version 4.3.3
library(tibble)
library(ranger)
library(pheatmap)
## Warning: package 'pheatmap' was built under R version 4.3.3
library(boot)
## Warning: package 'boot' was built under R version 4.3.3
```

Set seed for reproducibility

```
set.seed(123)
```

Data Overview

Quick data overview

```
str(glow_bonemed)
## 'data.frame':    500 obs. of  18 variables:
##  $ sub_id    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ site_id   : int  1 4 6 6 1 5 5 1 1 4 ...
##  $ phy_id    : int  14 284 305 309 37 299 302 36 8 282 ...
##  $ priorfrac : Factor w/ 2 levels "No","Yes": 1 1 2 1 1 2 1 2 2 1 ...
##  $ age       : int  62 65 88 82 61 67 84 82 86 58 ...
##  $ weight    : num  70.3 87.1 50.8 62.1 68 68 50.8 40.8 62.6 63.5 ...
##  $ height    : int  158 160 157 160 152 161 150 153 156 166 ...
##  $ bmi       : num  28.2 34 20.6 24.3 29.4 ...
##  $ premeno   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ momfrac   : Factor w/ 2 levels "No","Yes": 1 1 2 1 1 1 1 1 1 1 ...
##  $ armassist : Factor w/ 2 levels "No","Yes": 1 1 2 1 1 1 1 1 1 1 ...
##  $ smoke     : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 1 1 1 1 ...
##  $ raterisk  : Factor w/ 3 levels "Less","Same",..: 2 2 1 1 2 2 1 2 2 1
...
##  $ fracscore : int  1 2 11 5 1 4 6 7 7 0 ...
##  $ fracture  : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ bonemed   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 2 1 1 ...
```

```
##  $ bonemed_fu: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 2 1 1 ...
##  $ bonetreat : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 2 1 1 ...
```

```
summary(glow_bonemed)
```

```
##      sub_id         site_id          phy_id       priorfrac      age
##  Min.   :  1.0   Min.   :1.000   Min.   :  1.00   No :374   Min.   :55.00
##  1st Qu.:125.8   1st Qu.:2.000   1st Qu.: 57.75   Yes:126   1st Qu.:61.00
##  Median :250.5   Median :3.000   Median :182.50             Median :67.00
##  Mean   :250.5   Mean   :3.436   Mean   :178.55             Mean   :68.56
##  3rd Qu.:375.2   3rd Qu.:5.000   3rd Qu.:298.00             3rd Qu.:76.00
##  Max.   :500.0   Max.   :6.000   Max.   :325.00             Max.   :90.00
##      weight           height           bmi        premeno   momfrac
armassist
##  Min.   : 39.90   Min.   :134.0   Min.   :14.88   No :403   No :435   No
:312
##  1st Qu.: 59.90   1st Qu.:157.0   1st Qu.:23.27   Yes: 97   Yes: 65
Yes:188
##  Median : 68.00   Median :161.5   Median :26.42
##  Mean   : 71.82   Mean   :161.4   Mean   :27.55
##  3rd Qu.: 81.30   3rd Qu.:165.0   3rd Qu.:30.79
##  Max.   :127.00   Max.   :199.0   Max.   :49.08
##  smoke        raterisk     fracscore      fracture  bonemed   bonemed_fu
##  No :465   Less   :167   Min.   : 0.000   No :375   No :371   No :361
##  Yes: 35   Same   :186   1st Qu.: 2.000   Yes:125   Yes:129   Yes:139
##            Greater:147   Median : 3.000
##                          Mean   : 3.698
##                          3rd Qu.: 5.000
##                          Max.   :11.000
##  bonetreat
##  No :382
##  Yes:118
##
##
##
##
```

Checking for Missing Values

```
missing_values <- sum(is.na(glow_bonemed))

print(paste("Total Missing Values:", missing_values))
```

```
## [1] "Total Missing Values: 0"
```

Data Format and Initial Information

Data Format: A data.frame with 500 rows and 18 variables such as:

priorfrac - If the patient previously had a fracture
age
weight
height
bmi
premeno
momfrac
armassist
smoke
raterisk
fracscore
fracture
bonemed - Bone medications at enrollment (1: No, 2: Yes)
bonemed_fu - Bone medications at follow-up (1: No, 2: Yes)
bonetreat - Bone medications both at enrollment and follow-up (1: No, 2: Yes)

Age vs Weight: As weight increases the average age decreases
Age vs Height: Weak correlation of as height increases age decreases
Age vs BMI: As bmi increases the average age decreases
Age vs fracscore: As age increases the average fracscore increases

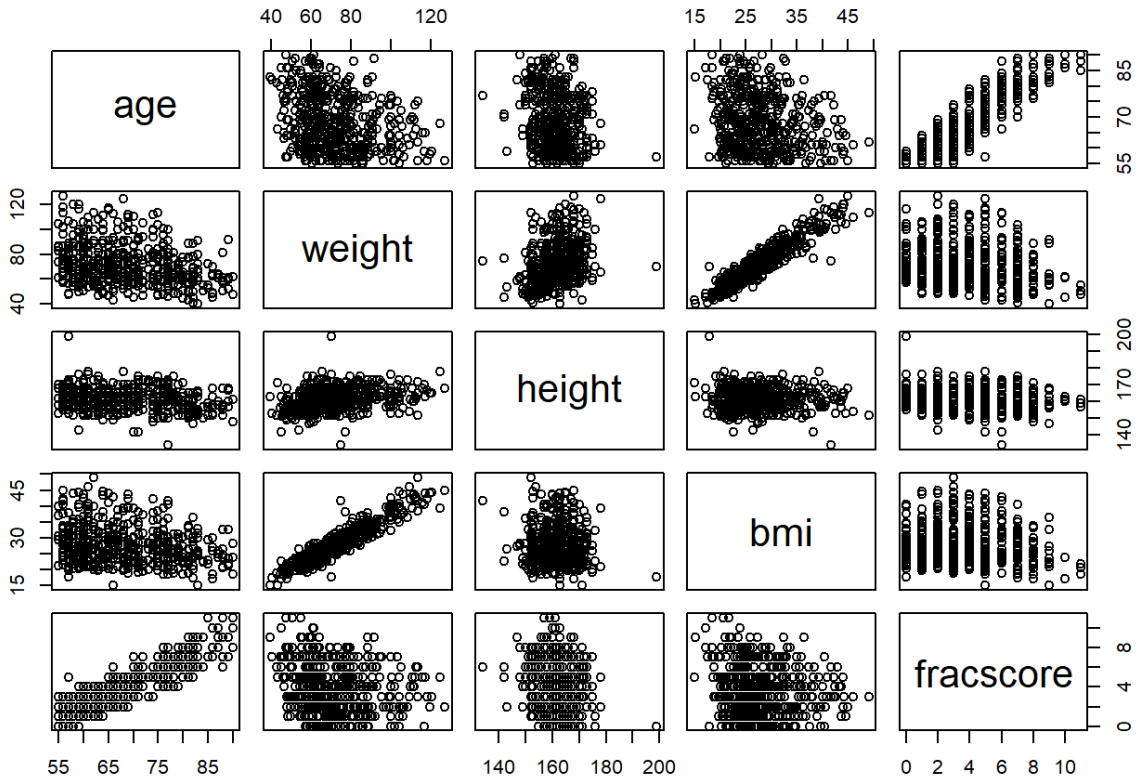
Weight vs Height: As height increases the average weight increases
Weight vs BMI: As bmi increases the average weight increases
Weight vs fracscore: As fracscore increases the average Weight decreases

Height vs BMI: As bmi increases the average height and variance stay the same
Height vs fracscore: As fracscore increases the average height stays the same though variance might decrease

BMI vs fracscore: As fracscore increases the average bmi decreases

Inital Exploration Plot

```
plot(glow_bonemed[, c(5:8, 14)])
```



Data Summarization and Transformation

```r
library(dplyr)

# Summarize numeric variables

# Assuming glow_bonemed is correctly loaded and contains the expected structure

library(dplyr)


# Summarize Numeric Variables

numeric_summary <- glow_bonemed %>%

  summarise(

    Age_Min = min(age, na.rm = TRUE),

    Age_Max = max(age, na.rm = TRUE),

    Weight_Mean = mean(weight, na.rm = TRUE),

    BMI_Median = median(bmi, na.rm = TRUE)

    # Add more as needed
```

```
  )

print(numeric_summary)
```

```
##   Age_Min Age_Max Weight_Mean BMI_Median
## 1      55      90     71.8232   26.41898
```

```
# Categorical summary example using dplyr for a single categorical variable
categorical_summary <- glow_bonemed %>%
  count(priorfrac)

print(categorical_summary)
```

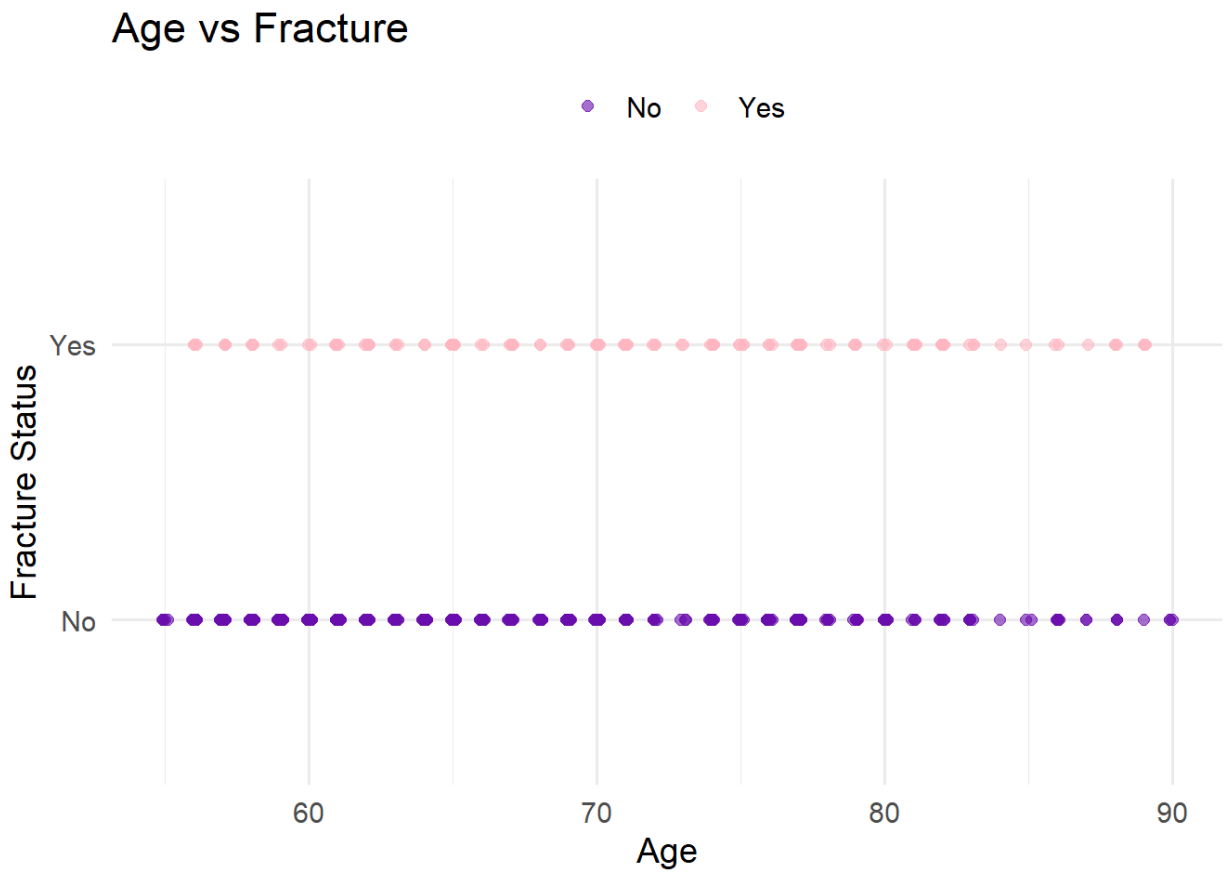
```
##   priorfrac   n
## 1        No 374
## 2       Yes 126
```

```
# Using Base R's table() for a quick look - this works well in markdown documents
print(table(glow_bonemed$priorfrac))
```

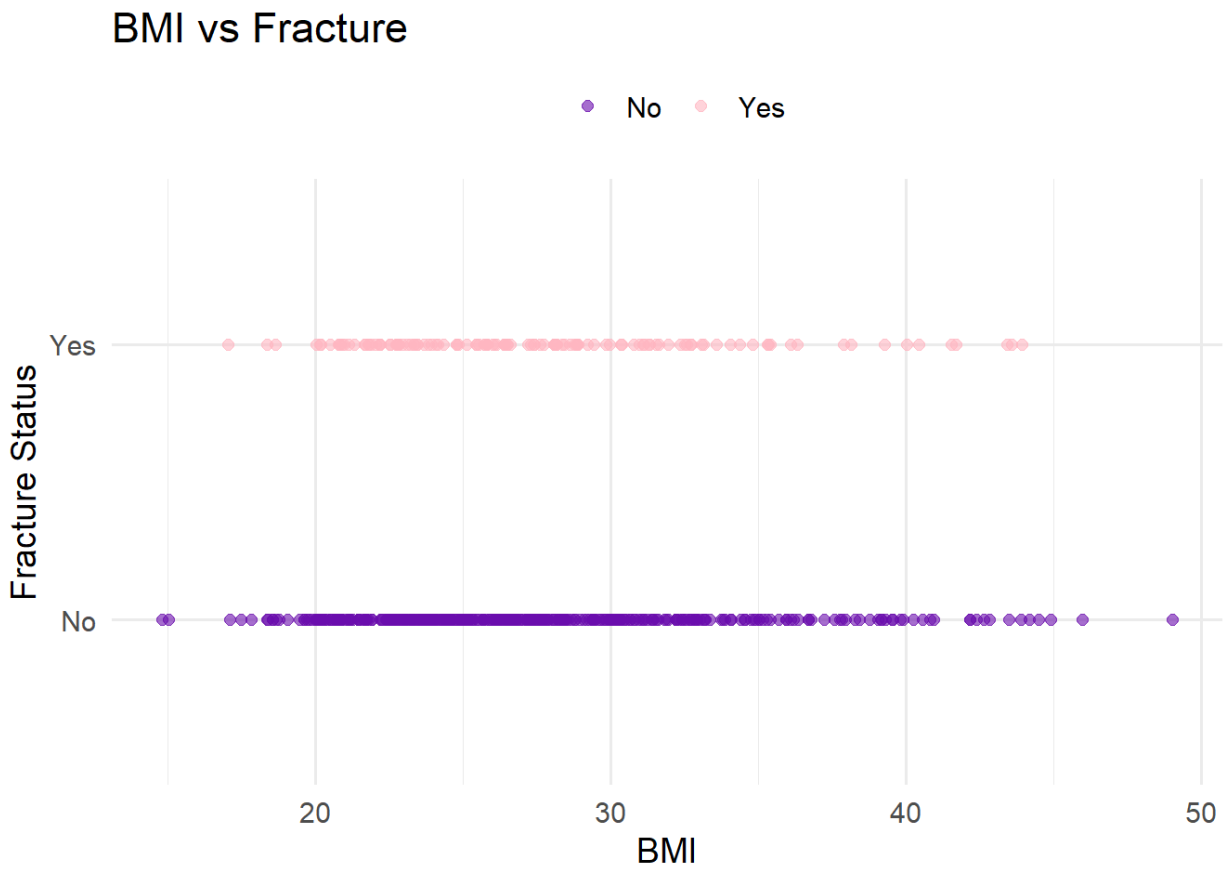
```
##
##  No Yes
## 374 126
```

Exploratory Data Analysis (EDA) with Visualizations

```
ggplot(glow_bonemed, aes(x = age, y = fracture, color = fracture)) +
  geom_jitter(width = 0.1, height = 0, alpha = 0.6, size = 2) +
  scale_color_manual(values = c("#6A0DAD", "lightpink")) +
  theme_minimal(base_size = 14) +
  theme(legend.title = element_blank(),
        legend.position = "top") +
  ggtitle("Age vs Fracture") +
  xlab("Age") +
  ylab("Fracture Status")
```

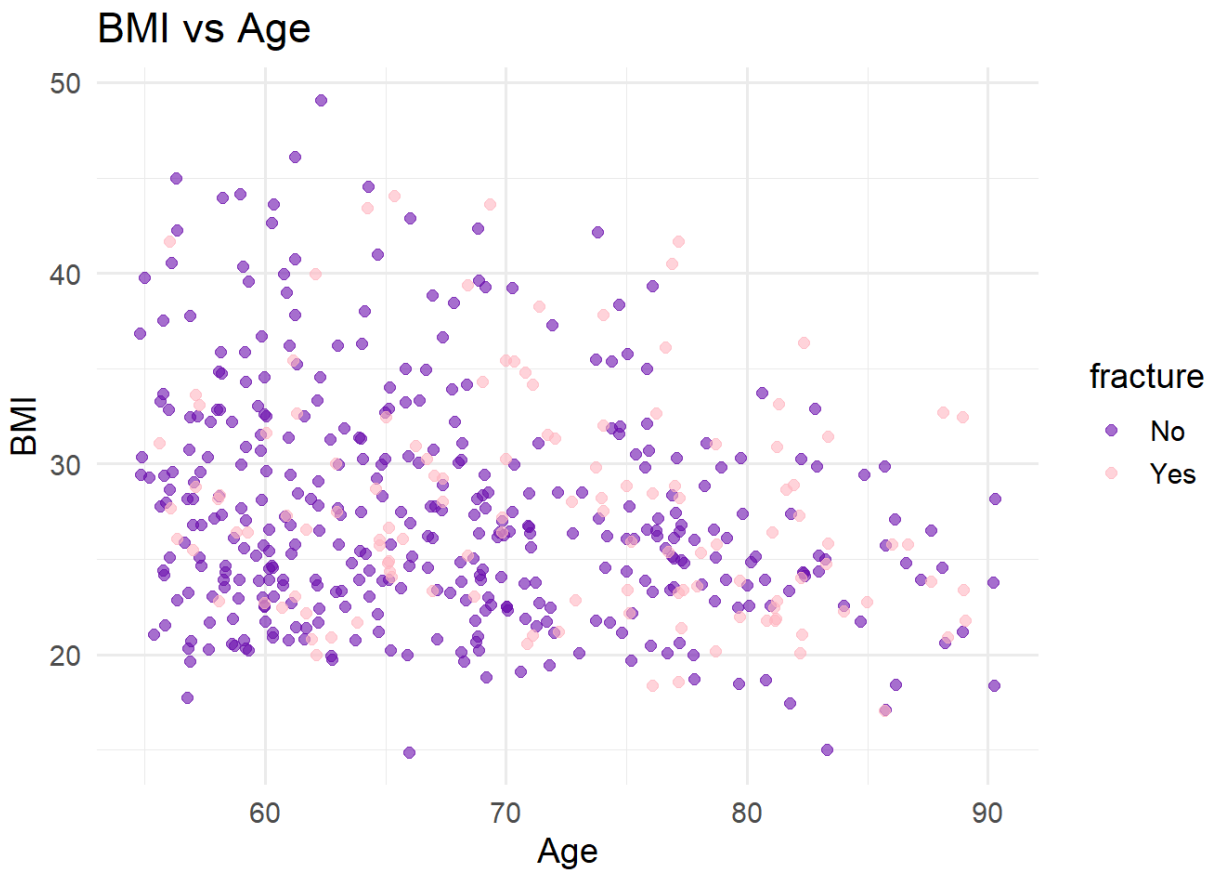



```
ggplot(glow_bonemed, aes(x = bmi, y = fracture, color = fracture)) +
  geom_jitter(width = 0.1, height = 0, alpha = 0.6, size = 2) +
  scale_color_manual(values = c("#6A0DAD", "lightpink")) +
  theme_minimal(base_size = 14) +
  theme(legend.title = element_blank(),
        legend.position = "top") +
  ggtitle("BMI vs Fracture") +
  xlab("BMI") +
  ylab("Fracture Status")
```



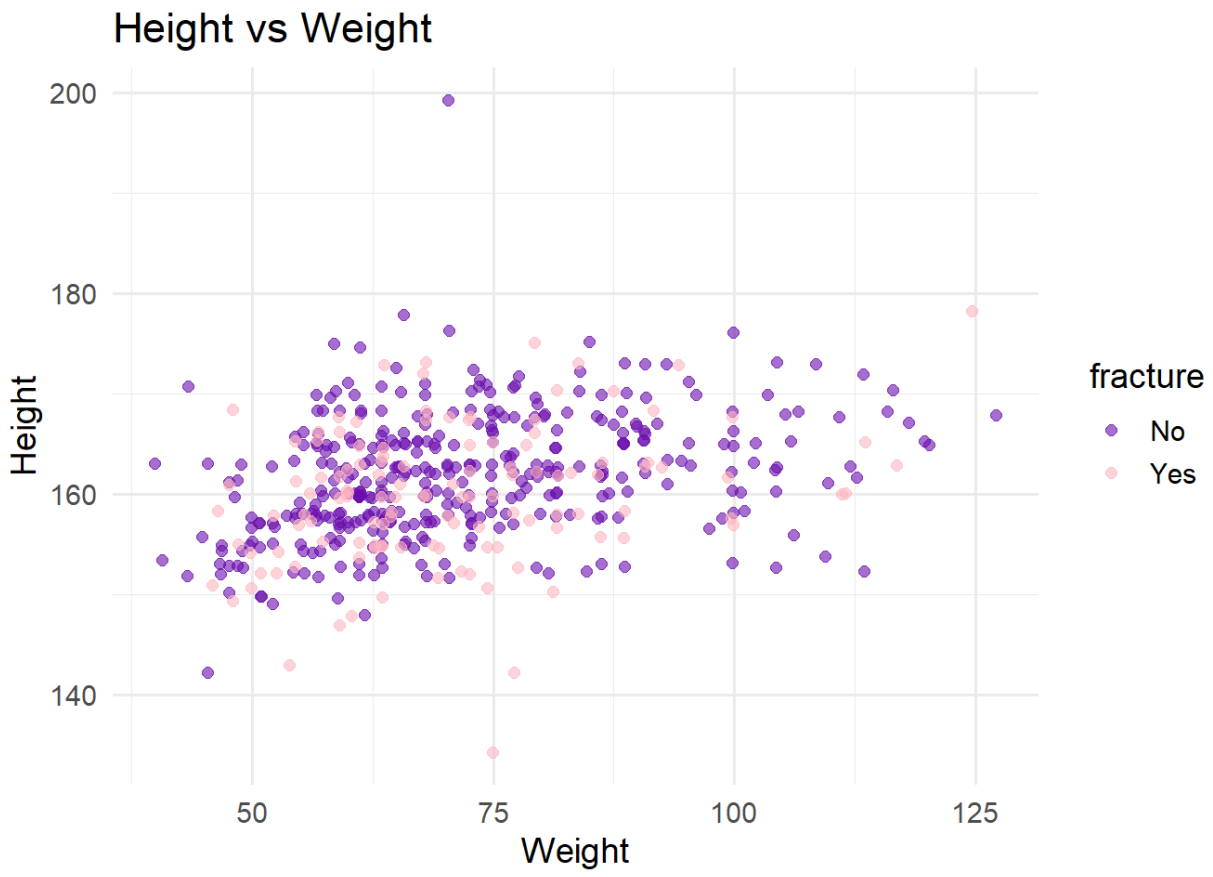

```
ggplot(glow_bonemed, aes(x = age, y = bmi, color = fracture)) +
  geom_jitter(alpha = 0.6, size = 2) +
  scale_color_manual(values = c("No" = "#6A0DAD", "Yes" = "#FFB6C1")) +  # Adjusted the pink color for better visibility
  theme_minimal(base_size = 14) +
  theme(legend.title = element_text("Fracture Status"),
        legend.position = "right") +
  labs(title = "BMI vs Age",
       x = "Age",
       y = "BMI")
```

```
## Warning in grid.Call(C_stringMetric, as.graphicsAnnot(x$label)): font family
## not found in Windows font database
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
```




```
ggplot(glow_bonemed, aes(x = weight, y = height, color = fracture)) +
  geom_jitter(alpha = 0.6, size = 2) +
  scale_color_manual(values = c("No" = "#6A0DAD", "Yes" = "#FFB6C1")) +  # Adjusted the pink color for better visibility
  theme_minimal(base_size = 14) +
  theme(legend.title = element_text("Fracture Status"),
        legend.position = "right") +
  labs(title = "Height vs Weight",
       x = "Weight",
       y = "Height")
```

```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database


## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
```




```
library(plotly)
```

```
##
## Attaching package: 'plotly'
## The following object is masked from 'package:MASS':
##
##     select
## The following object is masked from 'package:ggplot2':
##
##     last_plot
## The following object is masked from 'package:stats':
##
##     filter
## The following object is masked from 'package:graphics':
##
##     layout
```

```
colors = c('#6A0DAD', '#FFB6C1')  # Dark Purple and Light Pink
```

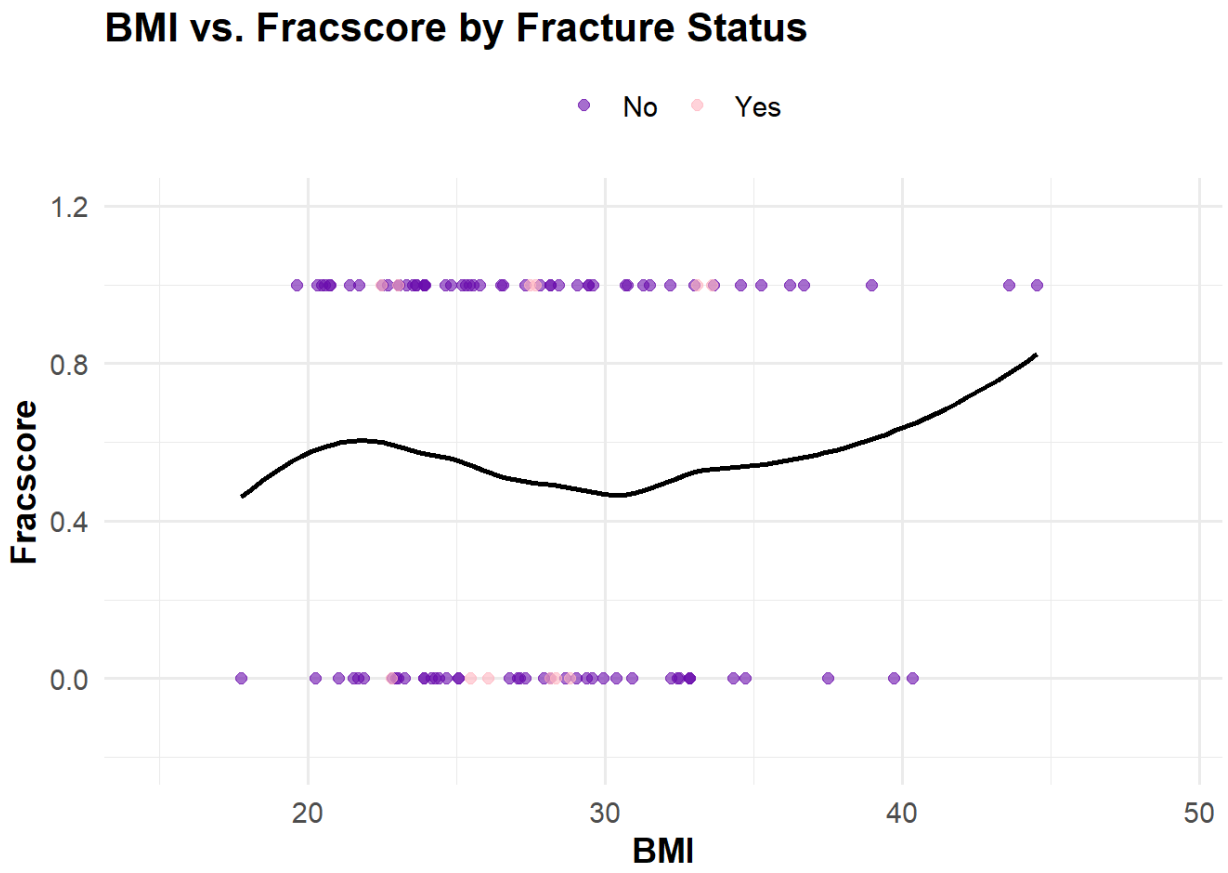
```
fracture3dplot <- plot_ly(glow_bonemed,
                          x = ~age,
                          y = ~height,
                          z = ~bmi,
                          color = ~fracture,
                          colors = c('#6A0DAD', '#FFB6C1'),
                          marker = list(size = 5,
                                        opacity = 0.8),
                          hoverinfo = 'text',
                          text = ~paste('Age:', age,
                                        '<br>Height:', height,
                                        '<br>BMI:', bmi,
                                        '<br>Fracture:', fracture)) %>%
    add_markers() %>%
    layout(title = 'Age, Height, and BMI by Fracture Status',
           scene = list(xaxis = list(title = 'Age'),
                        yaxis = list(title = 'Height'),
                        zaxis = list(title = 'BMI')),
           legend = list(title = list(text = 'Fracture Status')),
           margin = list(l = 0, r = 0, b = 0, t = 50))

# Show the plot
fracture3dplot
```

Fracture StatusNoYesAge, Height, and BMI by Fracture Status

```
ggplot(glow_bonemed, aes(x = bmi, y = fracscore, colour = fracture)) +
  geom_point(alpha = 0.6, size = 2) +  # Adjust transparency and size
  geom_smooth(method = "loess", size = 1, span = 0.75, se = FALSE, color =
"black") +  # se = FALSE removes the confidence interval shading
  ylim(-0.2, 1.2) +
  scale_color_manual(values = c("No" = "#6A0DAD", "Yes" = "lightpink")) +  #
Custom colors
  theme_minimal(base_size = 14) +  # Consistent font size
  theme(
    legend.title = element_blank(),  # Removes the legend title
```

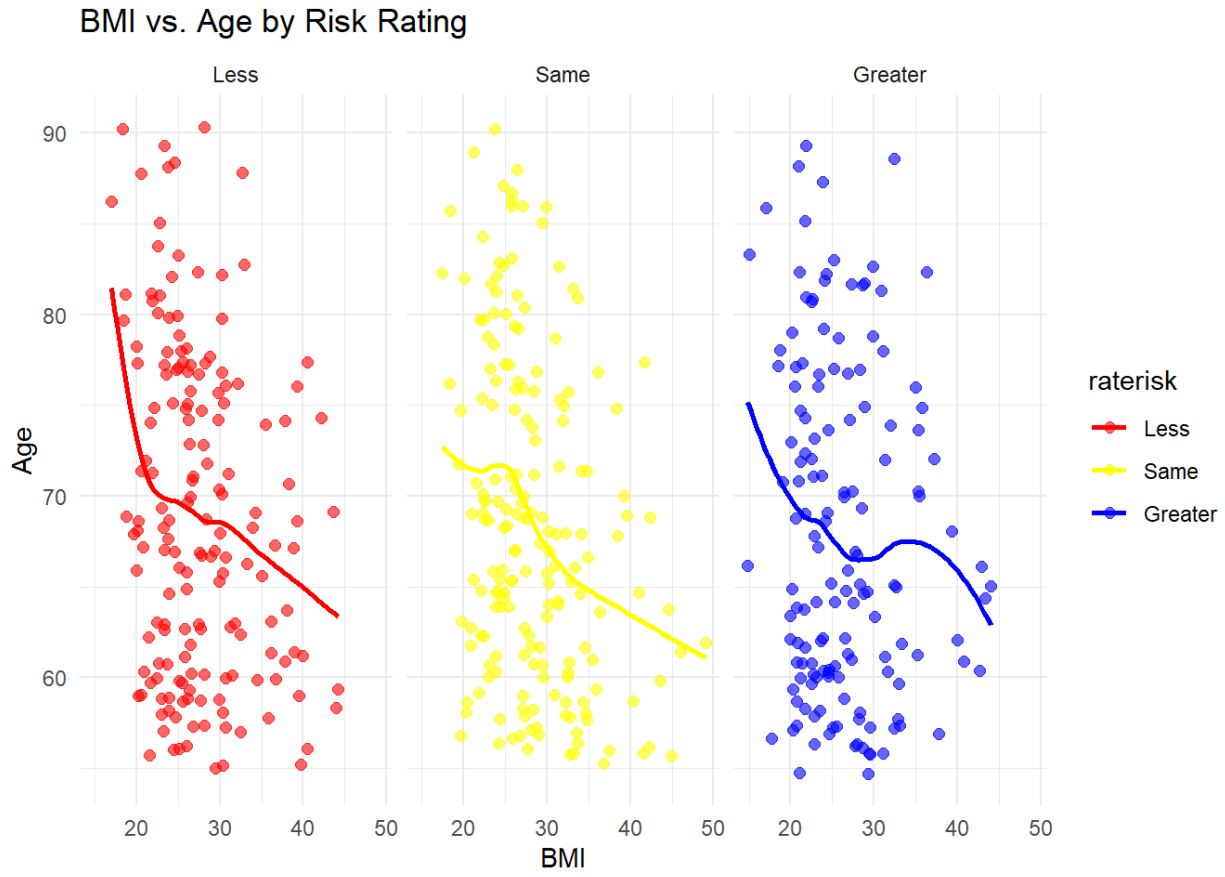
```
    legend.position = "top",  # Moves the legend to the top
    plot.title = element_text(size = 16, face = "bold"),  # Title styling
    axis.title = element_text(size = 14, face = "bold")  # Axis titles
styling
  ) +
  labs(
    title = "BMI vs. Fracscore by Fracture Status",
    x = "BMI",
    y = "Fracscore",
    colour = "Fracture Status"  # Changing the legend label
  )
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
## `geom_smooth()` using formula = 'y ~ x'
## Warning: Removed 394 rows containing non-finite values (`stat_smooth()`).
## Warning: Removed 394 rows containing missing values (`geom_point()`).
```




```
ggplot(glow_bonemed, aes(x = bmi, y = age, color = raterisk)) +
  geom_jitter(alpha = 0.6, size = 2) +
  geom_smooth(method = "loess", se = FALSE, size = 1, span = 0.75) +
  scale_color_manual(values = c("Less" = "red", "Same" = "yellow", "Greater"
= "blue")) +
  theme_minimal() +
  labs(title = "BMI vs. Age by Risk Rating", x = "BMI", y = "Age") +
  facet_wrap(~raterisk)
```

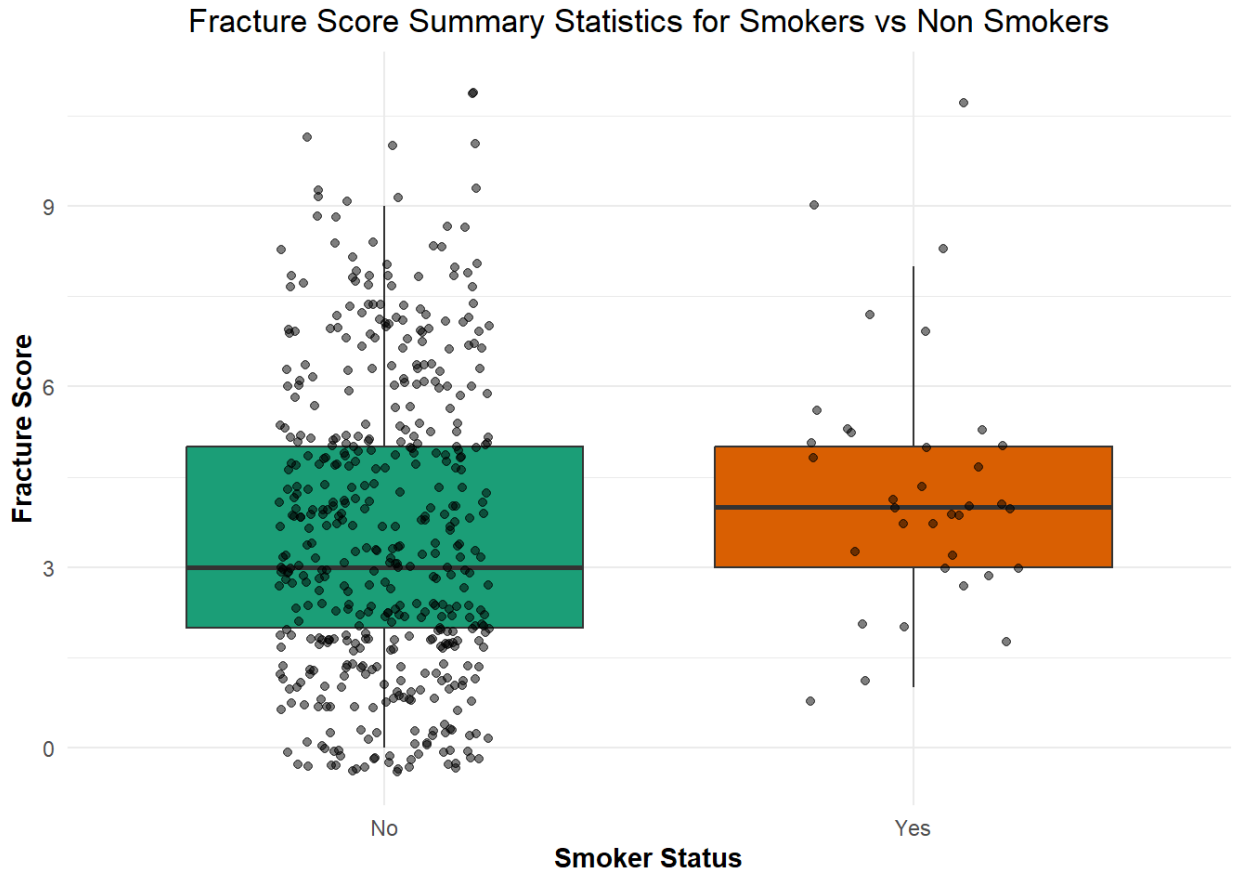
```
## `geom_smooth()` using formula = 'y ~ x'
```



The boxplot shows that the mean fracscore seems to be slightly higher for smokers compared to non smokers

```
ggplot(glow_bonemed, aes(x = smoke, y = fracscore, fill = smoke)) +
  geom_boxplot(outlier.shape = NA) +  # Hide outliers to focus on boxes
  geom_jitter(width = 0.2, alpha = 0.5, color = "black") +  # Add jitter to
show individual data points
  scale_fill_manual(values = c("No" = "#1b9e77", "Yes" = "#d95f02")) +  #
Custom colors for smokers vs non-smokers
  labs(
    title = "Fracture Score Summary Statistics for Smokers vs Non Smokers",
    x = "Smoker Status",
    y = "Fracture Score"
  ) +
  theme_minimal() +  # Minimal theme
  theme(
    plot.title = element_text(hjust = 0.5),  # Center the title
    legend.position = "none",  # Remove legend if not needed
```

```
    axis.title.x = element_text(face = "bold"),   # Bold x-axis title
    axis.title.y = element_text(face = "bold")    # Bold y-axis title
  )
```



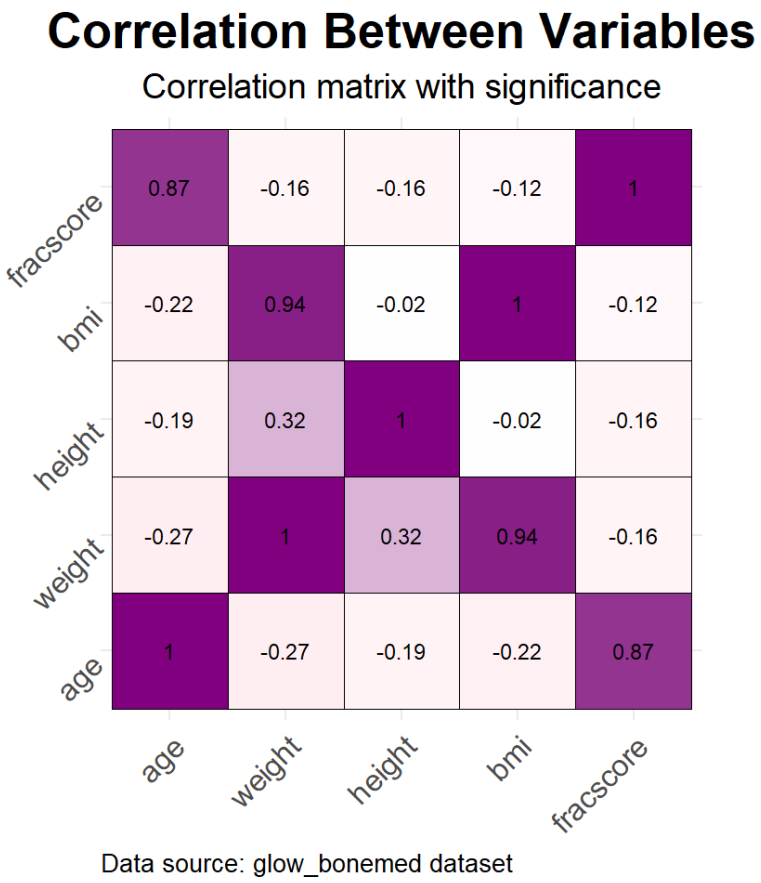
Plot confirms there is a strong correlation between age/fracscore, bmi/weight

```
# Load the required library
library(ggcorrplot)
```

```
## Warning: package 'ggcorrplot' was built under R version 4.3.3
```

```
# Calculate the correlation matrix for selected continuous variables
corr_matrix <- glow_bonemed %>%
  select(age, weight, height, bmi, fracscore) %>% # Select your continuous variables
  na.omit() %>%  # Omit NAs to avoid calculation errors
  cor()  # Compute the correlation matrix


# Use the ggcorrplot function to create a correlation plot
ggcorrplot(corr = corr_matrix, lab = TRUE, lab_size = 3,
```

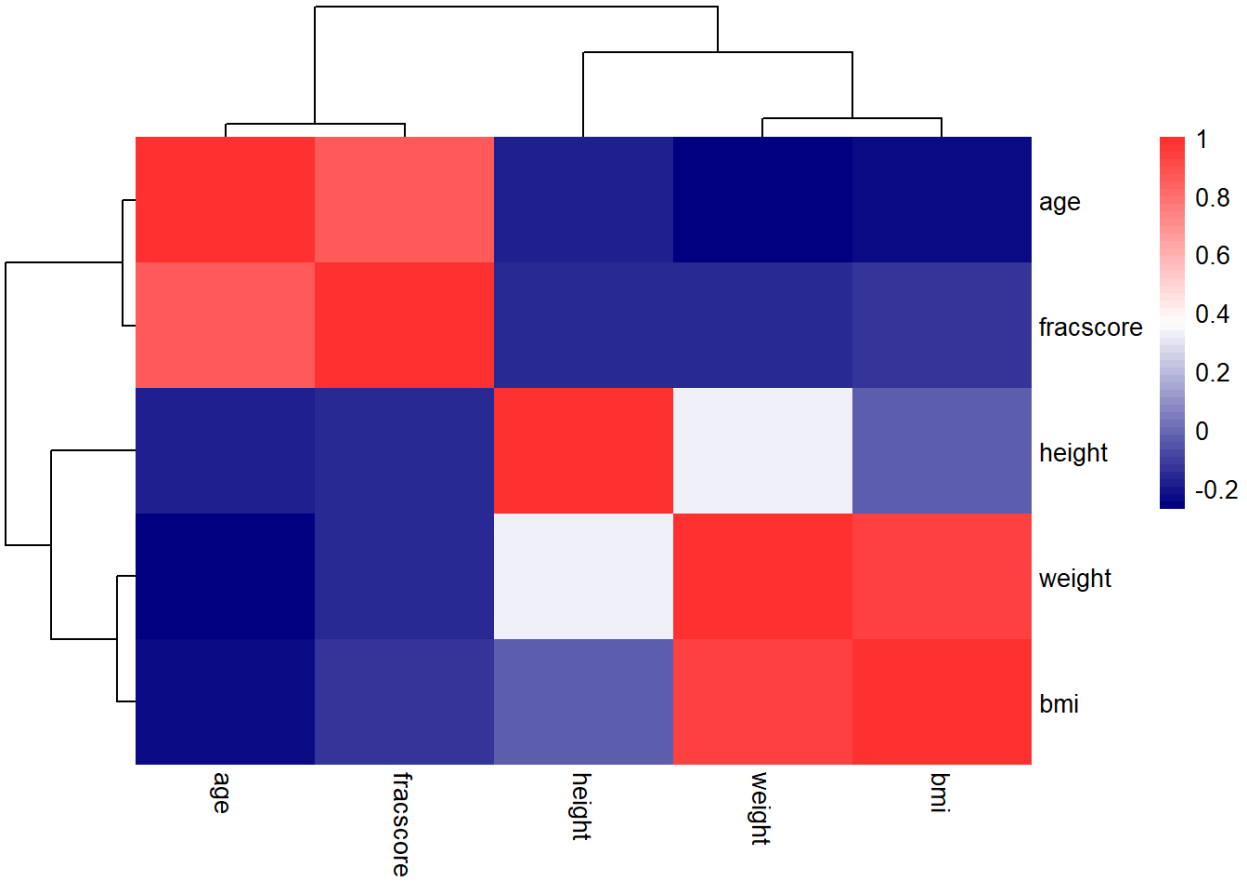
```
            colors = c("#FFC0CB", "white", "#800080"),  # Feminine colors:
light pink, white, purple
            outline.col = "black") +
  labs(title = "Correlation Between Variables",
       subtitle = "Correlation matrix with significance",
       caption = "Data source: glow_bonemed dataset") +
  theme(
    plot.title = element_text(hjust = 0.5, size = 20, face = "bold"),
    plot.subtitle = element_text(hjust = 0.5, size = 14),
    plot.caption = element_text(size = 10, hjust = 0),
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1),
    axis.text.y = element_text(angle = 45, vjust = 1),
    legend.title = element_blank(),
    legend.position = "none"
  )
```



Clustering EDA

```r
# Load necessary library for pheatmap
library(pheatmap)


# Selecting only the numeric columns that are relevant for the heatmap
heatmap_data <- glow_bonemed[, c("age", "weight", "height", "bmi",
"fracscore")]


# Calculate correlation matrix since heatmaps are often used to display
correlations
correlation_matrix <- cor(heatmap_data, use = "complete.obs")  # use complete
cases


# Generate the heatmap
pheatmap(correlation_matrix,
         color = colorRampPalette(c("navyblue", "white", "firebrick1"))(50),
```

```
          border_color = NA,
          cluster_rows = TRUE,
          cluster_cols = TRUE,
          fontsize_row = 10,
          fontsize_col = 10)
```




```
# Load necessary library for Plotly
library(plotly)


heatmap_data <- glow_bonemed[, 5:8]  # replace with actual indices or column
names as needed


# Convert to matrix if it's not already
heatmap_matrix <- as.matrix(heatmap_data)


# Generate the interactive heatmap with Plotly
```

```
plot_ly(x = colnames(heatmap_matrix), y = rownames(heatmap_matrix), z =
heatmap_matrix,
        type = "heatmap", colors = colorRamp(c("navy", "white",
"firebrick")))
```

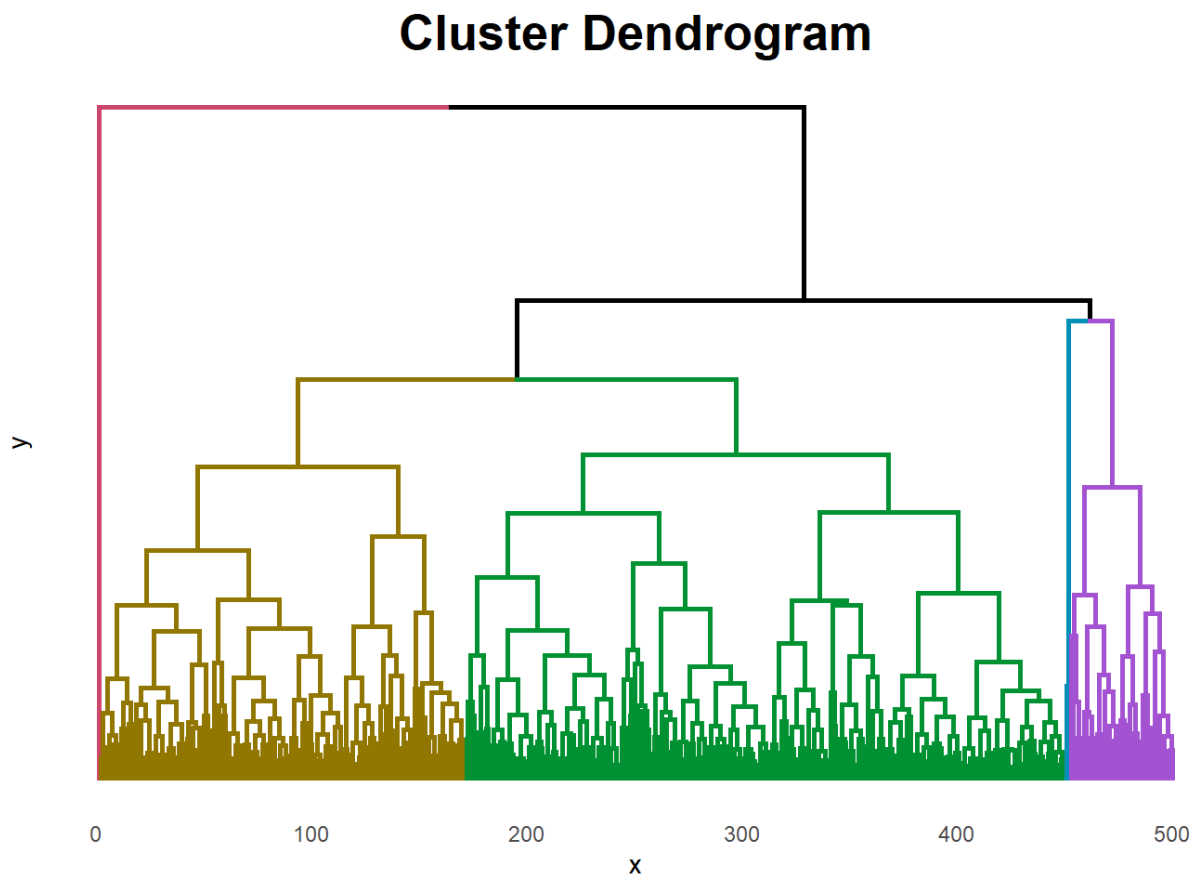
ageweightheightbmi0100200300400
50100150

```
# Load necessary libraries
library(ggplot2)
library(dendextend)
```

```
## Warning: package 'dendextend' was built under R version 4.3.3
##
## ---------------------
## Welcome to dendextend version 1.17.1
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at:
https://github.com/talgalili/dendextend/issues
## You may ask questions at stackoverflow, use the r and dendextend tags:
##   https://stackoverflow.com/questions/tagged/dendextend
##
##  To suppress this message use:
suppressPackageStartupMessages(library(dendextend))
## ---------------------
##
## Attaching package: 'dendextend'
## The following object is masked from 'package:stats':
##
##     cutree
```

```
data_to_cluster <- scale(glow_bonemed[, 5:8])


# Perform hierarchical clustering
continuousVariableClustering <- hclust(dist(data_to_cluster), method =
"complete")
```

```
# Convert to a dendrogram
dend <- as.dendrogram(continuousVariableClustering)

k <- 5  # Choose the number of clusters
dend <- color_branches(dend, k = k)

# Create a ggplot object from the dendrogram
ggdend <- as.ggdend(dend)

# Plot the dendrogram
ggplot(ggdend, labels = FALSE) +
  labs(title = "Cluster Dendrogram") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, size = 20, face = "bold"),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())
```



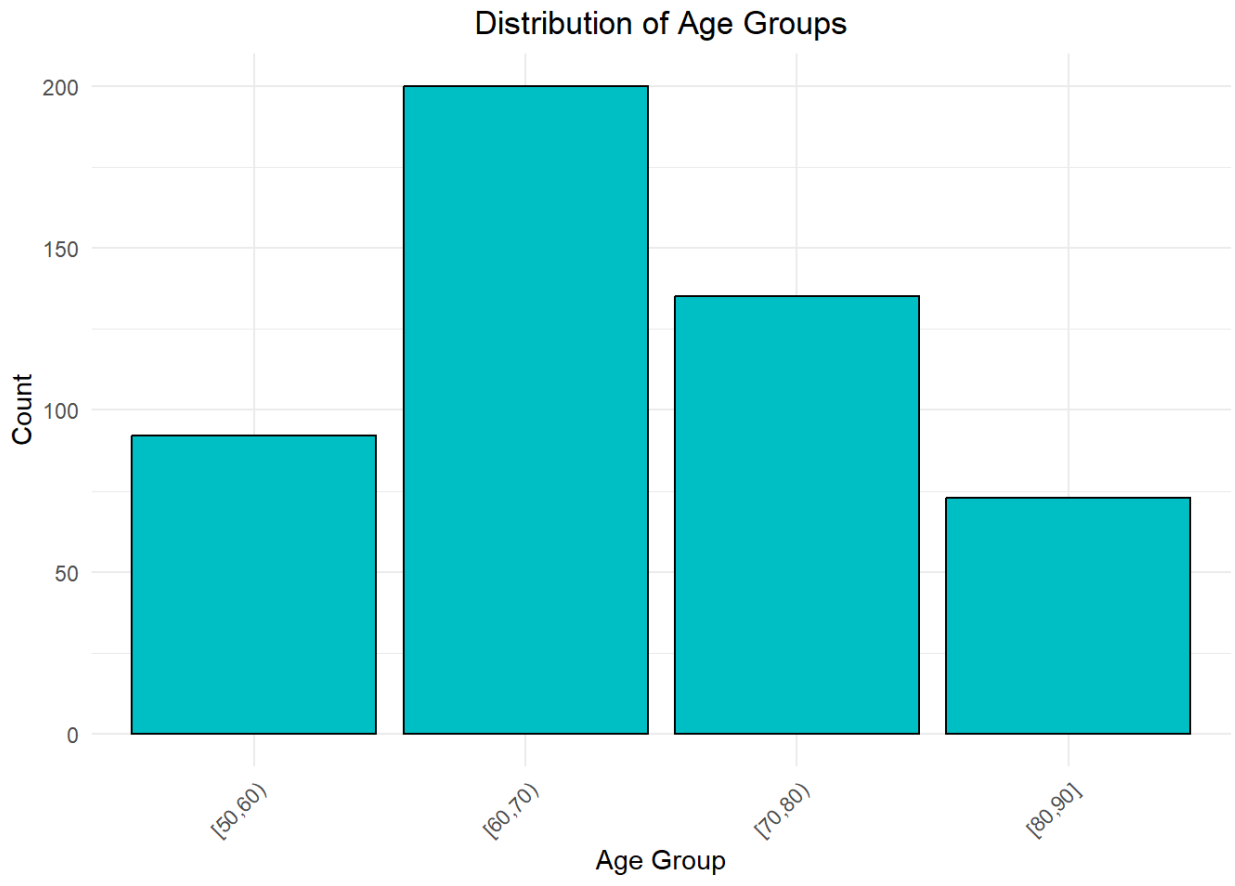

```
# Interactive
library(plotly)

# Convert ggdend object to a ggplot object and then to a plotly object
p <- ggplot(ggdend, labels = FALSE) +
  labs(title = "Cluster Dendrogram") +
  theme_void()  # Remove axes and background

ggplotly(p)
```

Cluster Dendrogram

Feature Engineering

Creating a new feature based on existing data

```r
glow_bonemed$age_group <- cut(glow_bonemed$age, breaks=c(50,60,70,80,90),
include.lowest=TRUE, right=FALSE)


library(ggplot2)


# Plot a bar chart to visualize the count of observations in each age group
ggplot(glow_bonemed, aes(x = age_group)) +
  geom_bar(fill = "#00BFC4", color = "black") +
  labs(title = "Distribution of Age Groups",
       x = "Age Group",
       y = "Count") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_text(angle = 45, hjust = 1))
```



Modeling

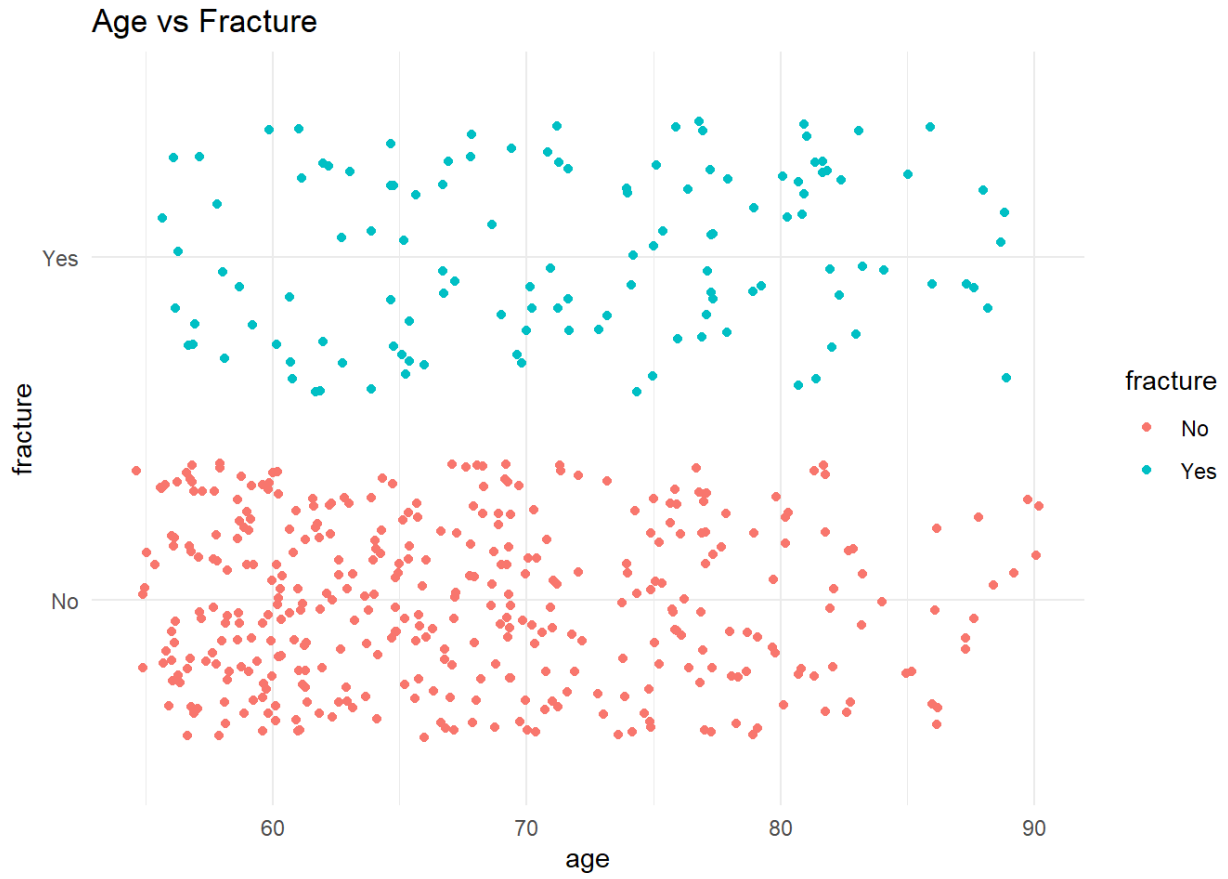
Simple Logistic Regression Model for Interpretability

```
model <- glm(fracture ~ age + bmi, data = glow_bonemed, family = binomial())
summary(model)
```

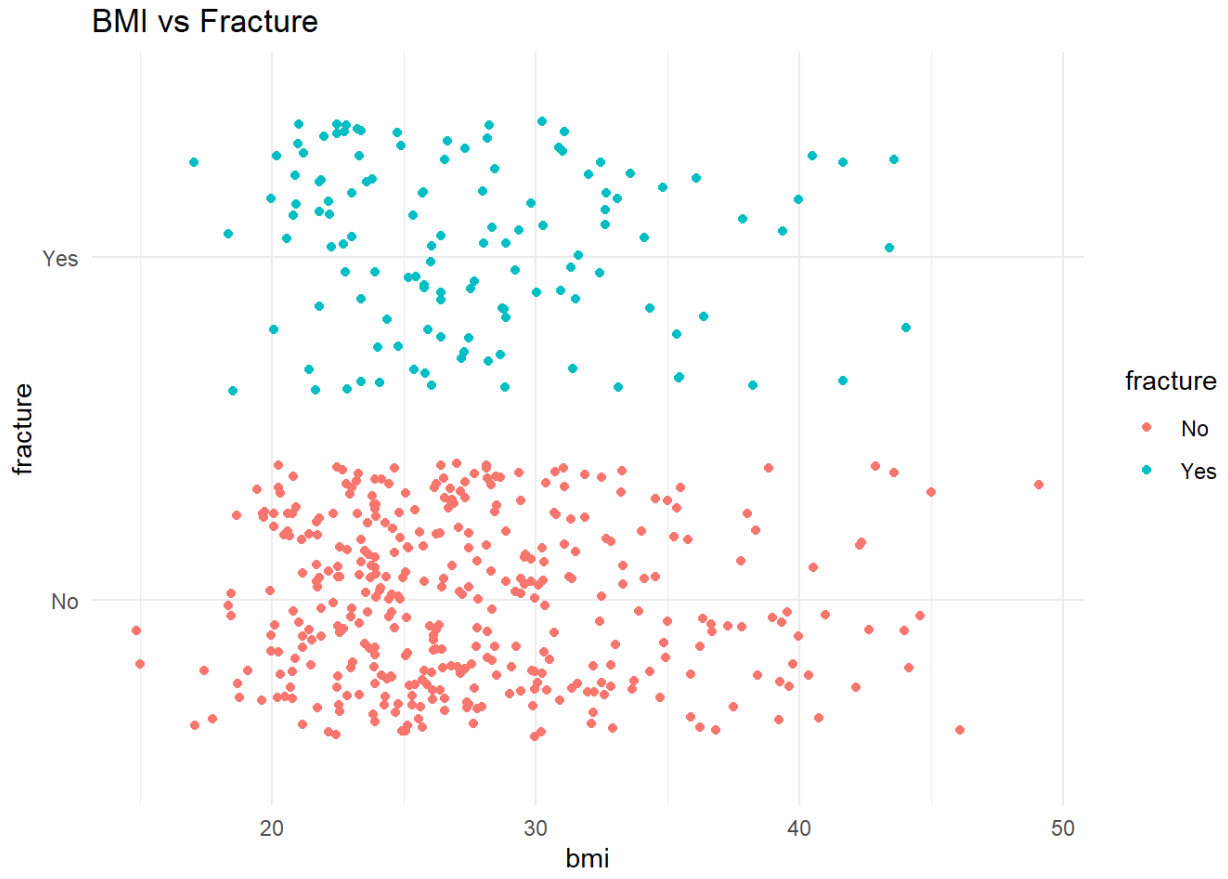
```
##
## Call:
## glm(formula = fracture ~ age + bmi, family = binomial(), data =
glow_bonemed)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.83441    1.10792  -5.266 1.39e-07 ***
## age          0.05736    0.01211   4.735 2.20e-06 ***
## bmi          0.02692    0.01817   1.482    0.138
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 562.34  on 499  degrees of freedom
## Residual deviance: 538.89  on 497  degrees of freedom
## AIC: 544.89
##
## Number of Fisher Scoring iterations: 4
```

Exploratory Data Analysis (EDA) with Visualizations

```
# Age vs Fracture and BMI vs Fracture
ggplot(glow_bonemed, aes(x = age, y = fracture, color = fracture)) +
  geom_jitter() +
  theme_minimal() +
  ggtitle("Age vs Fracture")
```




```
ggplot(glow_bonemed, aes(x = bmi, y = fracture, color = fracture)) +
  geom_jitter() +
  theme_minimal() +
  ggtitle("BMI vs Fracture")
```



Feature Engineering

Creating a new feature based on existing data

```
glow_bonemed$age_group <- cut(glow_bonemed$age, breaks=c(50,60,70,80,90),
include.lowest=TRUE, right=FALSE)
```

Modeling

Simple Logistic Regression Model for Interpretability

```
model <- glm(fracture ~ age + bmi, data = glow_bonemed, family = binomial())

summary(model)
```

```
##
## Call:
```

```
## glm(formula = fracture ~ age + bmi, family = binomial(), data =
glow_bonemed)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.83441    1.10792  -5.266 1.39e-07 ***
## age          0.05736    0.01211   4.735 2.20e-06 ***
## bmi          0.02692    0.01817   1.482    0.138
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 562.34  on 499  degrees of freedom
## Residual deviance: 538.89  on 497  degrees of freedom
## AIC: 544.89
##
## Number of Fisher Scoring iterations: 4
```

Model Validation using Training and Testing sets v 1 with splitIndex (2 versions for varied use that keeps its form)

```
set.seed(123)  # set the seed for reproducibility

splitIndex <- createDataPartition(glow_bonemed$fracture, p = 0.8, list =
FALSE)  # we can adjust p for the percentage split

training_data <- glow_bonemed[splitIndex, ]

test_data <- glow_bonemed[-splitIndex, ]

# Prepare the matrix of predictors for the test set, using the same
transformation as for training

x_test_matrix <- model.matrix(~ age + height + bmi, data = test_data)[, -1]
# Remove intercept
```

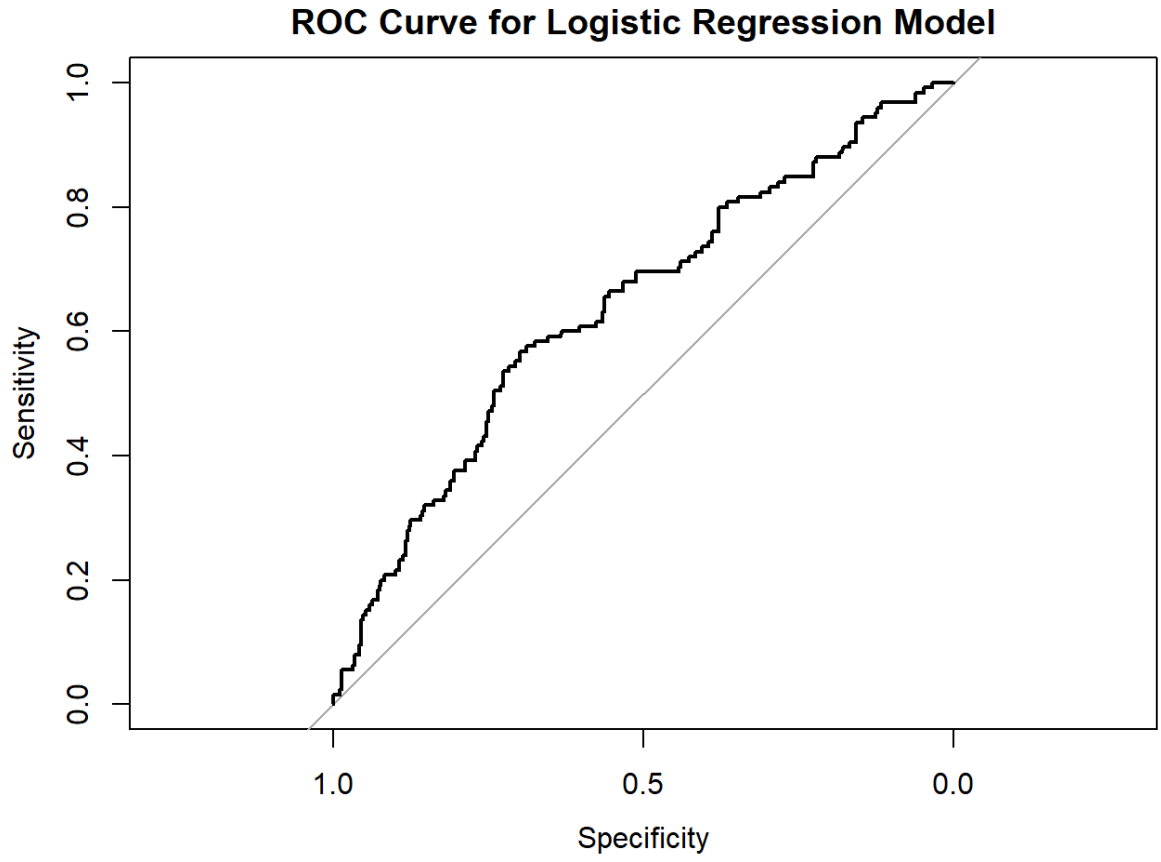
Model Validation using Training and Testing sets

```
training_index <- createDataPartition(glow_bonemed$fracture, p = 0.8, list =
FALSE)

training_data <- glow_bonemed[training_index, ]
```

```
testing_data <- glow_bonemed[-training_index, ]
predictions <- predict(model, newdata = test_data, type = "response")
roc_obj <- roc(test_data$fracture, predictions)
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
auc_value <- auc(roc_obj)
auc_value
## Area under the curve: 0.5691
```

Model Performance Evaluation using ROC Curve and AUC

```
roc_curve <- roc(response = glow_bonemed$fracture, predictor = fitted(model))
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
plot(roc_curve, main="ROC Curve for Logistic Regression Model")
```




```
auc_model <- auc(roc_curve)
```

```
print(paste("AUC for Model:", auc_model))
```

```
## [1] "AUC for Model: 0.639402666666667"
```

VIF Check Model

```
vif(model)
```

```
##      age      bmi
## 1.078522 1.078522
```

Advanced Modeling Techniques

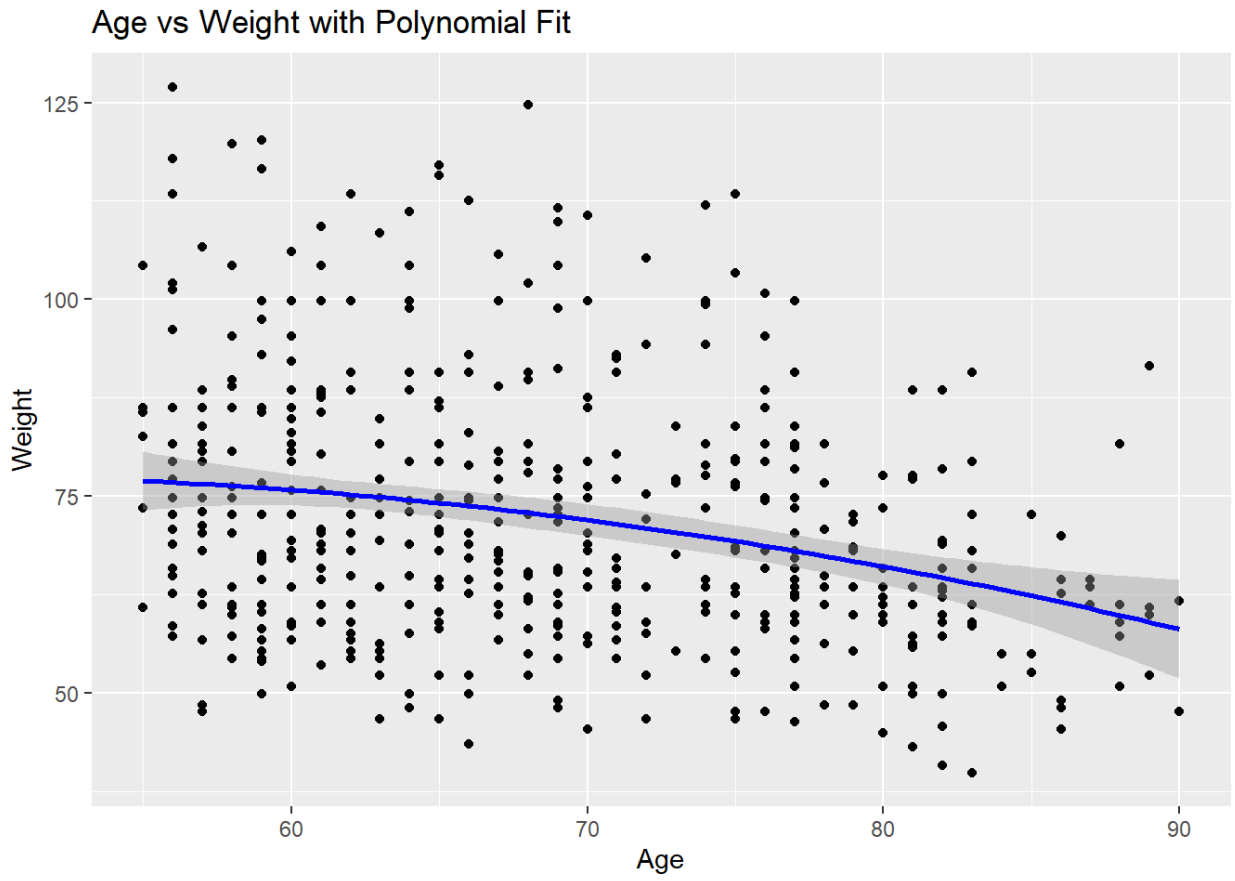
Polynomial Regression Model

```
poly_model <- lm(weight ~ poly(age, 2), data = glow_bonemed)
summary(poly_model)
```

```
##
## Call:
## lm(formula = weight ~ poly(age, 2), data = glow_bonemed)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -30.260 -11.233  -2.640   8.861  51.789
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    71.8232     0.7079 101.466  < 2e-16 ***
## poly(age, 2)1 -99.7172    15.8281  -6.300 6.56e-10 ***
## poly(age, 2)2 -18.5502    15.8281  -1.172    0.242
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.83 on 497 degrees of freedom
## Multiple R-squared:  0.07632,    Adjusted R-squared:  0.0726
## F-statistic: 20.53 on 2 and 497 DF,  p-value: 2.707e-09
```

Visualization for Polynomial Regression

```
ggplot(glow_bonemed, aes(x = age, y = weight)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ poly(x, 2), color = "blue") +
  labs(title = "Age vs Weight with Polynomial Fit", x = "Age", y = "Weight")
```



Model Refinement with Regularization Techniques

```
x_matrix <- model.matrix(weight ~ age + height + bmi, data = glow_bonemed)[,
-1]

y_vector <- glow_bonemed$weight
```

Fit Ridge Regression Model

```
cv_ridge <- cv.glmnet(x_matrix, y_vector, alpha = 0)

optimal_lambda_ridge <- cv_ridge$lambda.min
```

```
ridge_model <- glmnet(x_matrix, y_vector, alpha = 0, lambda =
optimal_lambda_ridge)
```

Fit Lasso Regression Model

```
cv_lasso <- cv.glmnet(x_matrix, y_vector, alpha = 1)

optimal_lambda_lasso <- cv_lasso$lambda.min

lasso_model <- glmnet(x_matrix, y_vector, alpha = 1, lambda =
optimal_lambda_lasso)

# Prepare the matrix of predictors for the test set, using the same
transformation as for training

x_test_matrix <- model.matrix(~ age + height + bmi, data = test_data)[, -1]
# Remove intercept
```

Evaluate the Ridge and Lasso models using the test data

```
predictions_ridge <- predict(ridge_model, s = optimal_lambda_ridge, newx =
x_test_matrix)

predictions_lasso <- predict(lasso_model, s = optimal_lambda_lasso, newx =
x_test_matrix)


ridge_performance <- postResample(predictions_ridge, test_data$weight)

lasso_performance <- postResample(predictions_lasso, test_data$weight)
```

Print the performance metrics for Ridge and Lasso models

```
print(list(Ridge = ridge_performance, Lasso = lasso_performance))
```

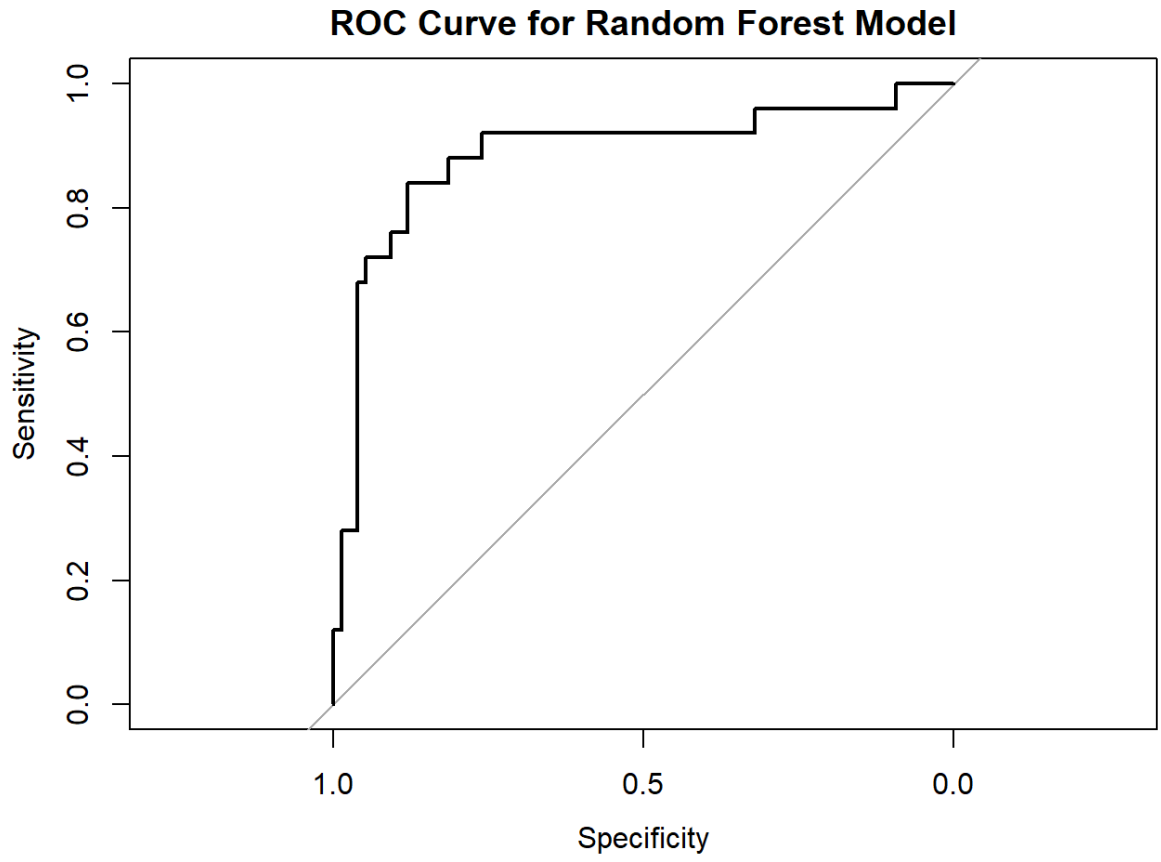
```
## $Ridge

##     RMSE Rsquared      MAE

## 1.726348 0.993411 1.224981

##

## $Lasso

##      RMSE  Rsquared       MAE

## 1.2458963 0.9940341 0.7594917
```

Random Forest Model

```
rf_model <- ranger(fracture ~ age + bmi, data = training_data, probability = TRUE)

rf_predictions <- predict(rf_model, test_data)$predictions[, "Yes"]
```

ROC Curve for Random Forest Model

```
rf_roc_curve <- roc(test_data$fracture, rf_predictions)
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
plot(rf_roc_curve, main="ROC Curve for Random Forest Model")
```




```
auc_rf <- auc(rf_roc_curve)
print(paste("AUC for Random Forest Model:", auc_rf))
## [1] "AUC for Random Forest Model: 0.885866666666667"
# Confirm expected columns
str(glow_bonemed)
## 'data.frame':    500 obs. of  19 variables:
```

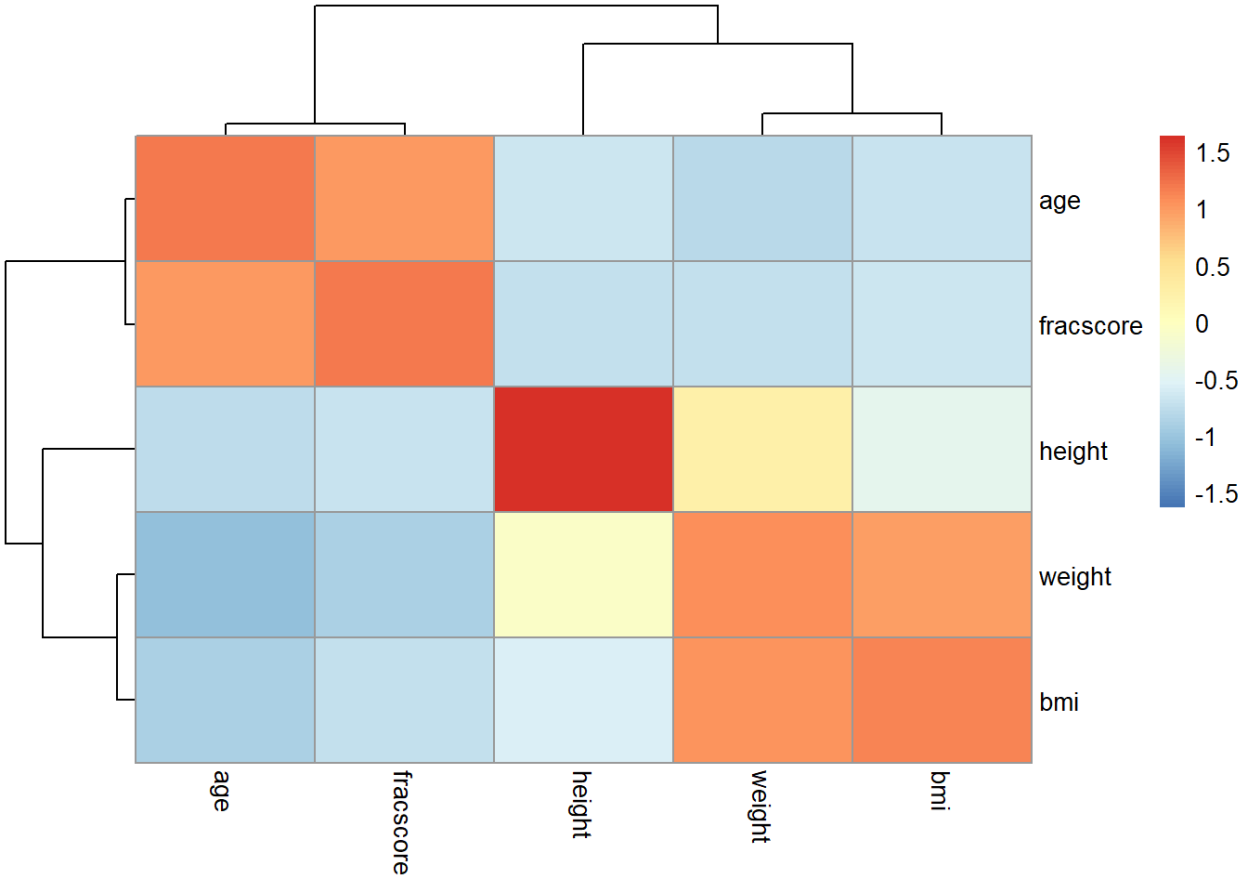
```
##  $ sub_id    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ site_id   : int  1 4 6 6 1 5 5 1 1 4 ...
##  $ phy_id    : int  14 284 305 309 37 299 302 36 8 282 ...
##  $ priorfrac : Factor w/ 2 levels "No","Yes": 1 1 2 1 1 2 1 2 2 1 ...
##  $ age       : int  62 65 88 82 61 67 84 82 86 58 ...
##  $ weight    : num  70.3 87.1 50.8 62.1 68 68 50.8 40.8 62.6 63.5 ...
##  $ height    : int  158 160 157 160 152 161 150 153 156 166 ...
##  $ bmi       : num  28.2 34 20.6 24.3 29.4 ...
##  $ premeno   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ momfrac   : Factor w/ 2 levels "No","Yes": 1 1 2 1 1 1 1 1 1 1 ...
##  $ armassist : Factor w/ 2 levels "No","Yes": 1 1 2 1 1 1 1 1 1 1 ...
##  $ smoke     : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 1 1 1 1 ...
##  $ raterisk  : Factor w/ 3 levels "Less","Same",..: 2 2 1 1 2 2 1 2 2 1
...
##  $ fracscore : int  1 2 11 5 1 4 6 7 7 0 ...
##  $ fracture  : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ bonemed   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 2 1 1 ...
##  $ bonemed_fu: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 2 1 1 ...
##  $ bonetreat : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 2 1 1 ...
##  $ age_group : Factor w/ 4 levels "[50,60)","[60,70)",..: 2 2 4 4 2 2 4 4
4 1 ...
```

```
# Explicitly use dplyr::select to avoid conflicts with other packages
numeric_vars <- glow_bonemed %>%
  dplyr::select(age, weight, height, bmi, fracscore) %>% # Adjust column
names as necessary
  na.omit()


cor_matrix <- cor(numeric_vars)
```

Visualizations (for analysis)

Heatmap for EDA

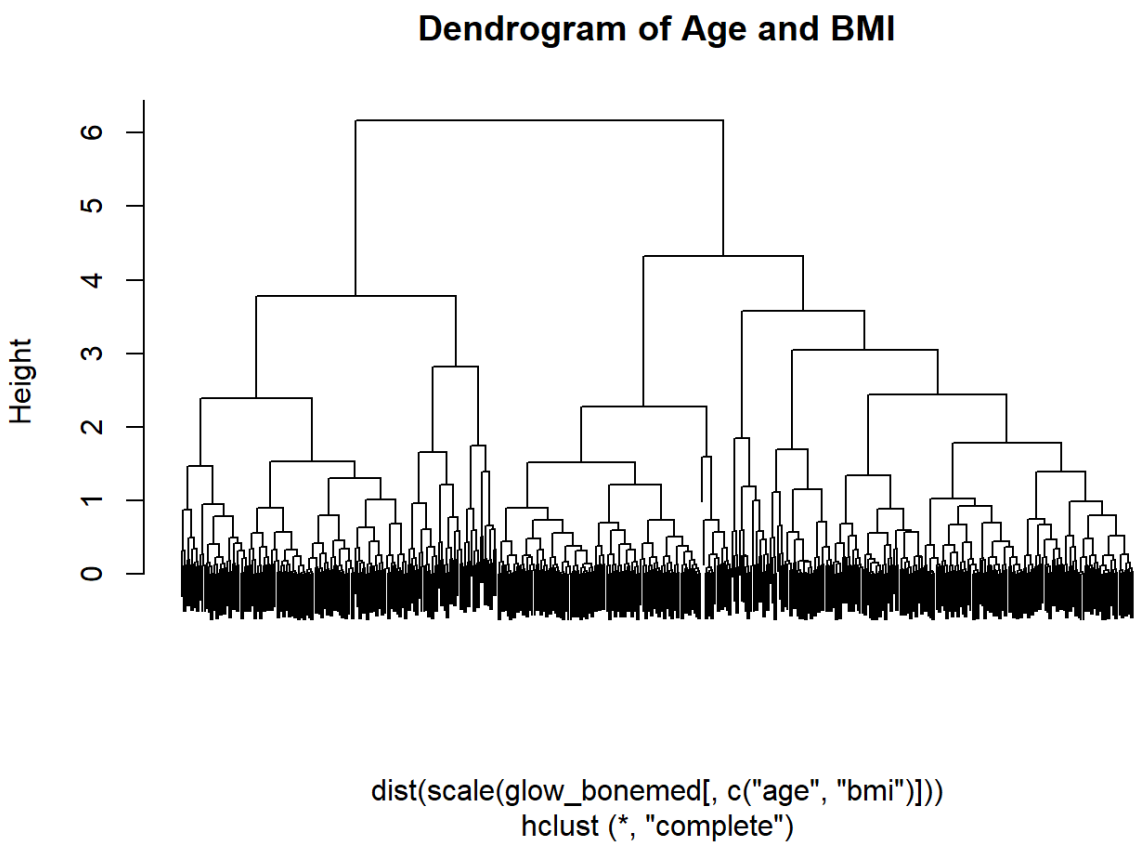
```
pheatmap(cor_matrix, scale = "row", clustering_method = "complete")
```



Enhanced Clustering EDA

Hierarchical clustering with dendrogram

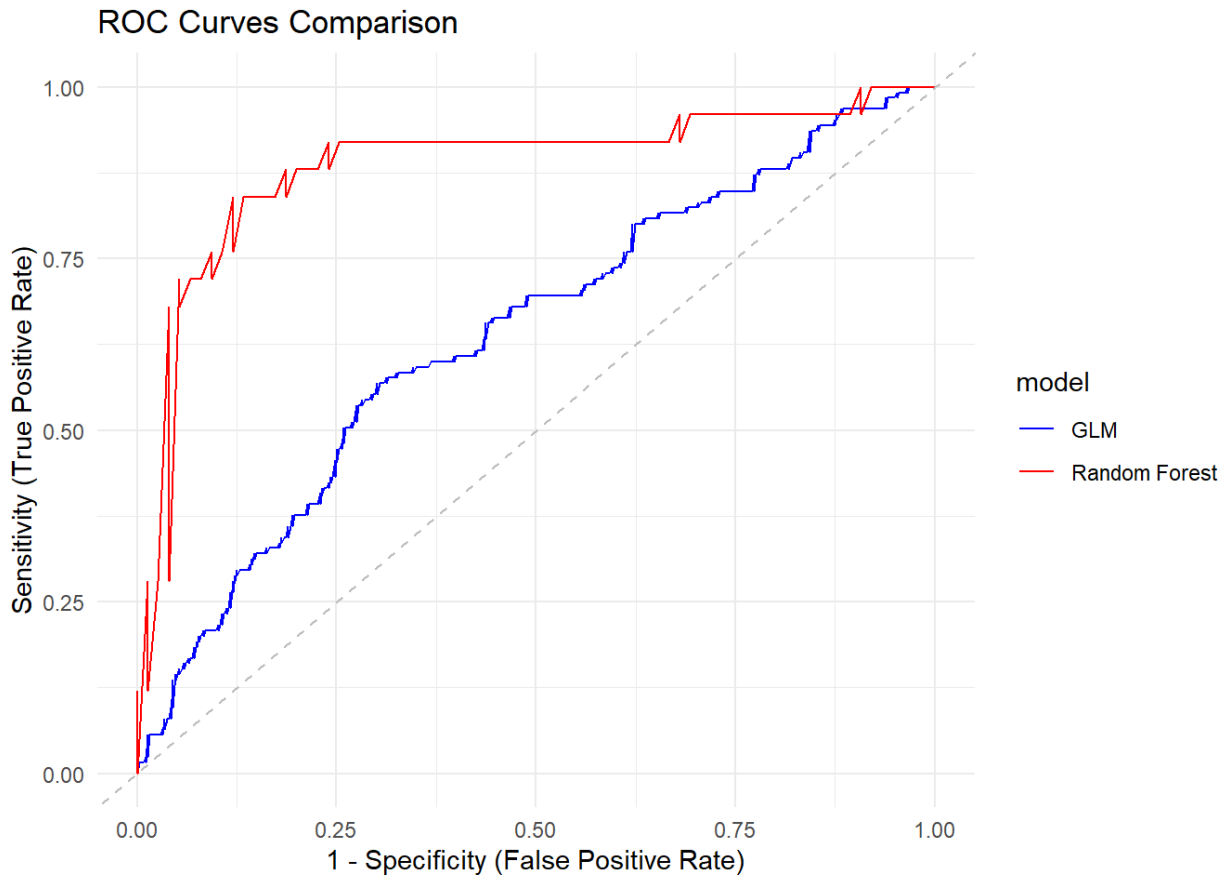
```
hc <- hclust(dist(scale(glow_bonemed[, c("age", "bmi")])), method =
"complete")

plot(hc, labels = FALSE, main = "Dendrogram of Age and BMI")

abline(h = 150, col = "red")
```



Combining ROC Curves for General Model Comparison

```
library(ggplot2)
# First, create a data frame for each model's ROC data
glm_roc_data <- data.frame(
  specificity = 1 - roc_curve$specificities,
  sensitivity = roc_curve$sensitivities,
  model = "GLM"
)

rf_roc_data <- data.frame(
  specificity = 1 - rf_roc_curve$specificities,
  sensitivity = rf_roc_curve$sensitivities,
  model = "Random Forest"
)
```

```r
# Combine the data frames
combined_roc_data <- rbind(glm_roc_data, rf_roc_data)


# Plot using ggplot2
ggplot(combined_roc_data, aes(x = specificity, y = sensitivity, color = model)) +
  geom_line() +
  geom_abline(linetype = "dashed", color = "gray") + # Diagonal line for reference
  labs(title = "ROC Curves Comparison",
       x = "1 - Specificity (False Positive Rate)",
       y = "Sensitivity (True Positive Rate)") +
  theme_minimal() +
  scale_color_manual(values = c("GLM" = "blue", "Random Forest" = "red"))
```

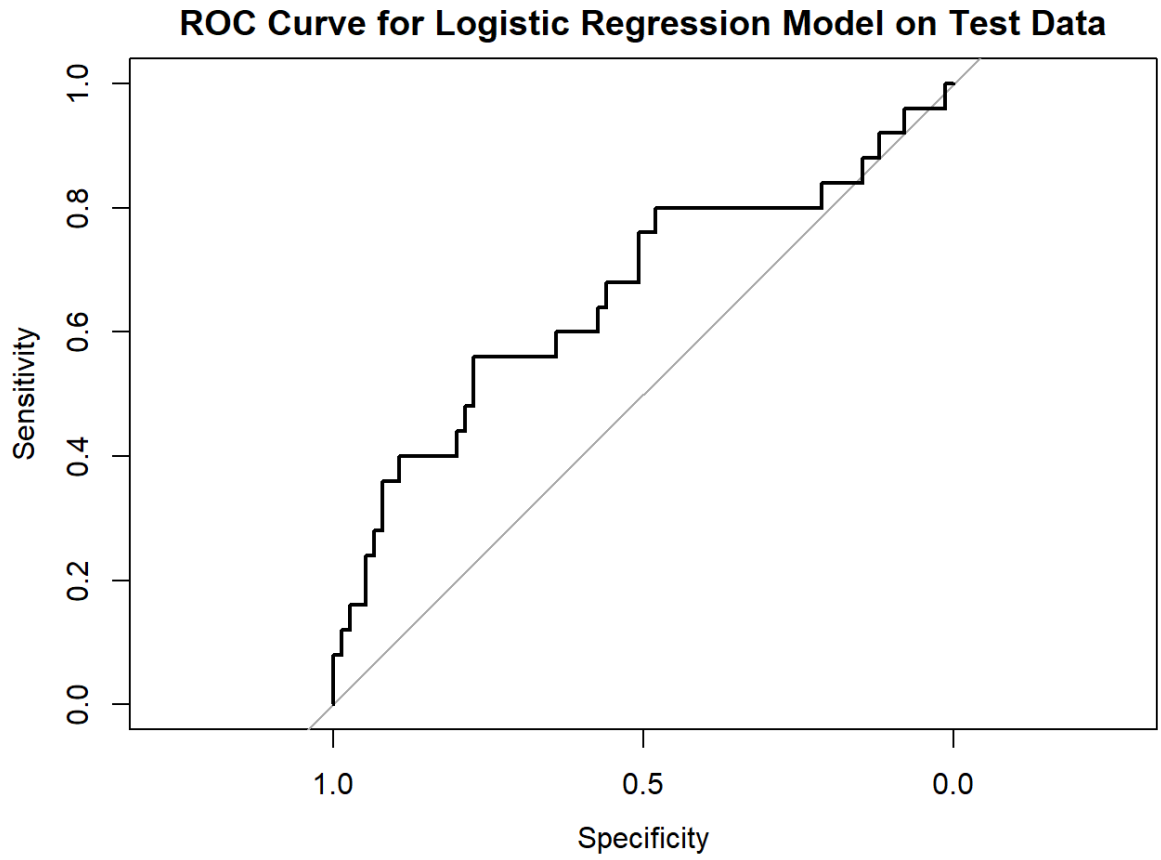


Model Training and Testing with Model

```
model_train <-glm(fracture ~ age + bmi, data = glow_bonemed, family =
binomial())

predictions <- predict(model_train, newdata = testing_data, type =
"response")

roc_curve_test <- roc(response = testing_data$fracture, predictor =
predictions)
```

```
## Setting levels: control = No, case = Yes

## Setting direction: controls < cases
```

```
plot(roc_curve_test, main="ROC Curve for Logistic Regression Model on Test
Data")
```

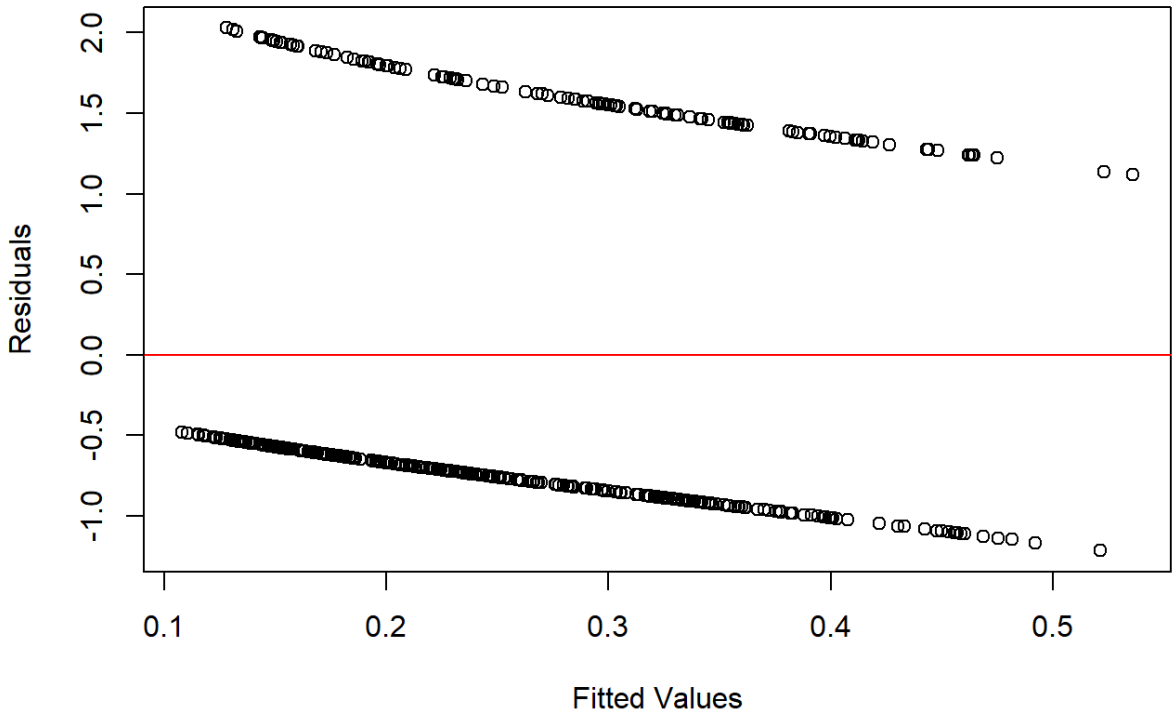



```
auc_model_test <- auc(roc_curve_test)

print(paste("AUC for Model  on Testing Data:", auc_model_test))
```

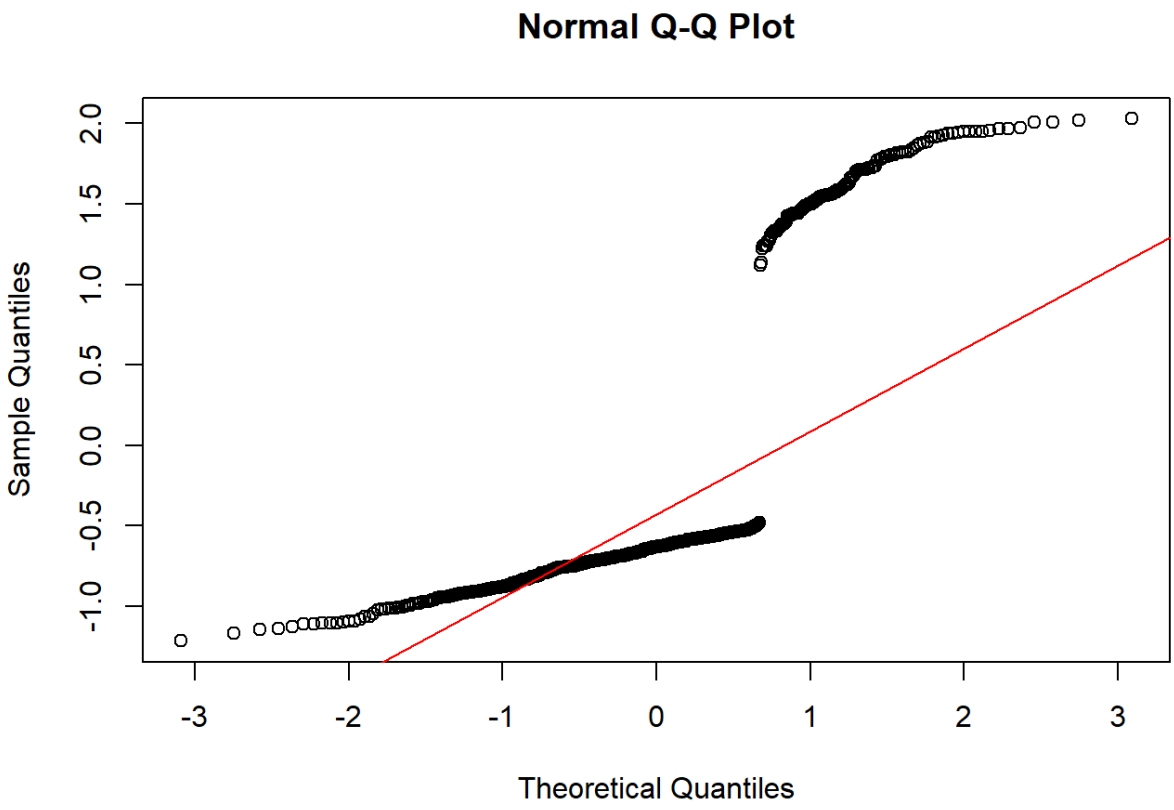
```
## [1] "AUC for Model  on Testing Data: 0.659733333333333"
```

Diagnostic Plots for Model Assumptions with Model

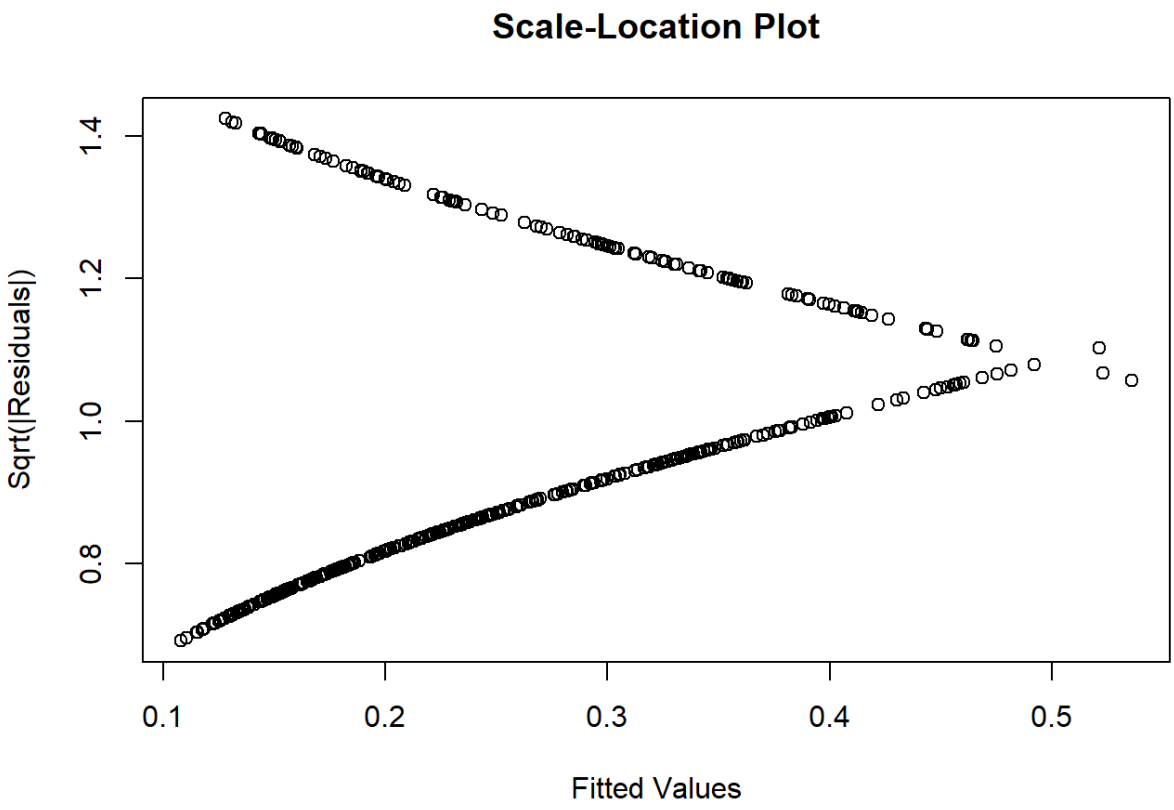
```
plot(fitted(model_train), residuals(model_train), xlab="Fitted Values",
ylab="Residuals")

abline(h=0, col="red")
```




```
qqnorm(residuals(model_train))

qqline(residuals(model_train), col="red")
```

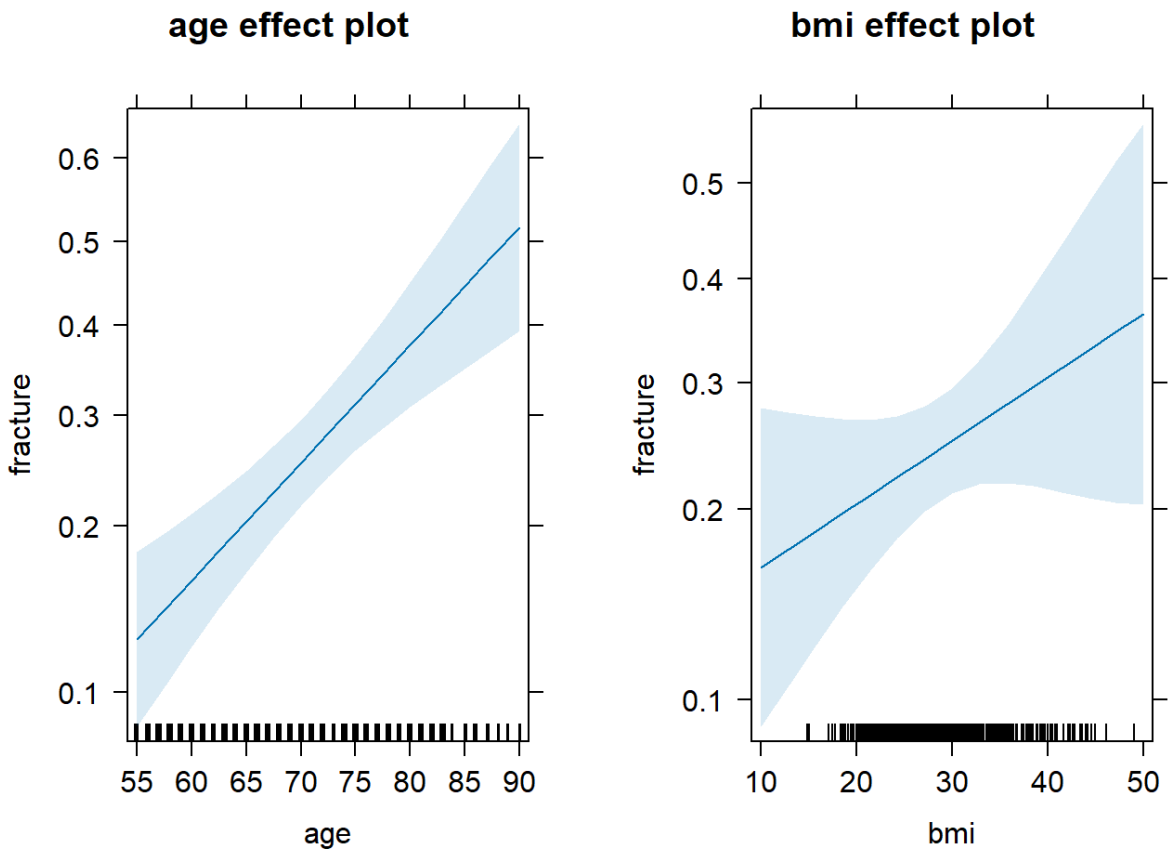



```
plot(fitted(model_train), sqrt(abs(residuals(model_train))), xlab="Fitted
Values", ylab="Sqrt(|Residuals|)", main="Scale-Location Plot")

abline(h = 0, col="red")
```



Effect Plots for Model

```
plot(allEffects(model))
```



Robust Standard Errors for Model

```
model_robust <- coeftest(model, vcov = vcovHC(model, type = "HC1"))
print(model_robust)
```

```
##
## z test of coefficients:
##
##              Estimate Std. Error z value  Pr(>|z|)
## (Intercept) -5.834409   1.118467 -5.2164 1.824e-07 ***
## age          0.057359   0.012240  4.6862 2.784e-06 ***
## bmi          0.026924   0.017563  1.5330    0.1253
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Model 2

Model with Additional Predictors including the new feature age_group

```
model2 <- glm(fracture ~ age_group + bmi + priorfrac + smoke + raterisk, data
= glow_bonemed, family = binomial())
summary(model2)
```

```
##
## Call:
## glm(formula = fracture ~ age_group + bmi + priorfrac + smoke +
##     raterisk, family = binomial(), data = glow_bonemed)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -3.20516    0.67241  -4.767 1.87e-06 ***
## age_group[60,70)   0.30592    0.35629   0.859 0.390544
## age_group[70,80)   0.94260    0.36732   2.566 0.010283 *
## age_group[80,90]   1.37895    0.41749   3.303 0.000957 ***
## bmi                0.02875    0.01884   1.526 0.127125
## priorfracYes       0.69424    0.24282   2.859 0.004249 **
## smokeYes          -0.26158    0.45599  -0.574 0.566201
## rateriskSame       0.53169    0.27594   1.927 0.054001 .
## rateriskGreater    0.88414    0.28799   3.070 0.002140 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 562.34  on 499  degrees of freedom
## Residual deviance: 514.74  on 491  degrees of freedom
## AIC: 532.74
##
## Number of Fisher Scoring iterations: 4
```

```
predictions <- predict(model2, newdata = test_data, type = "response")
roc_obj <- roc(test_data$fracture, predictions)
```

```
## Setting levels: control = No, case = Yes
```

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

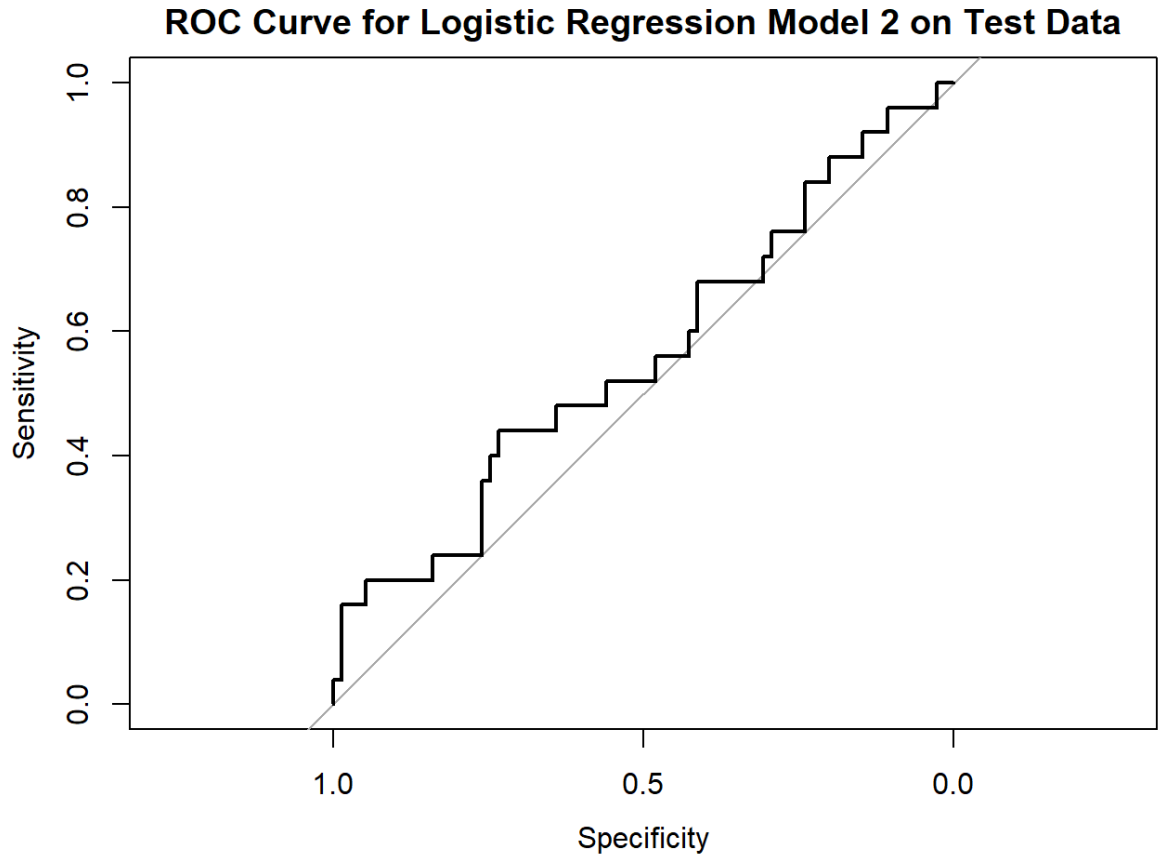
```
auc_value <- auc(roc_obj)
```

Model Training and Testing with Model 2

```
model2_train <- glm(fracture ~ age_group + bmi + priorfrac + smoke + 
raterisk, data = training_data, family = binomial())

predictions2 <- predict(model2_train, newdata = testing_data, type = 
"response")

roc_curve_test2 <- roc(response = testing_data$fracture, predictor = 
predictions2)
```

```
## Setting levels: control = No, case = Yes

## Setting direction: controls < cases
```

```
plot(roc_curve_test2, main="ROC Curve for Logistic Regression Model 2 on Test 
Data")
```

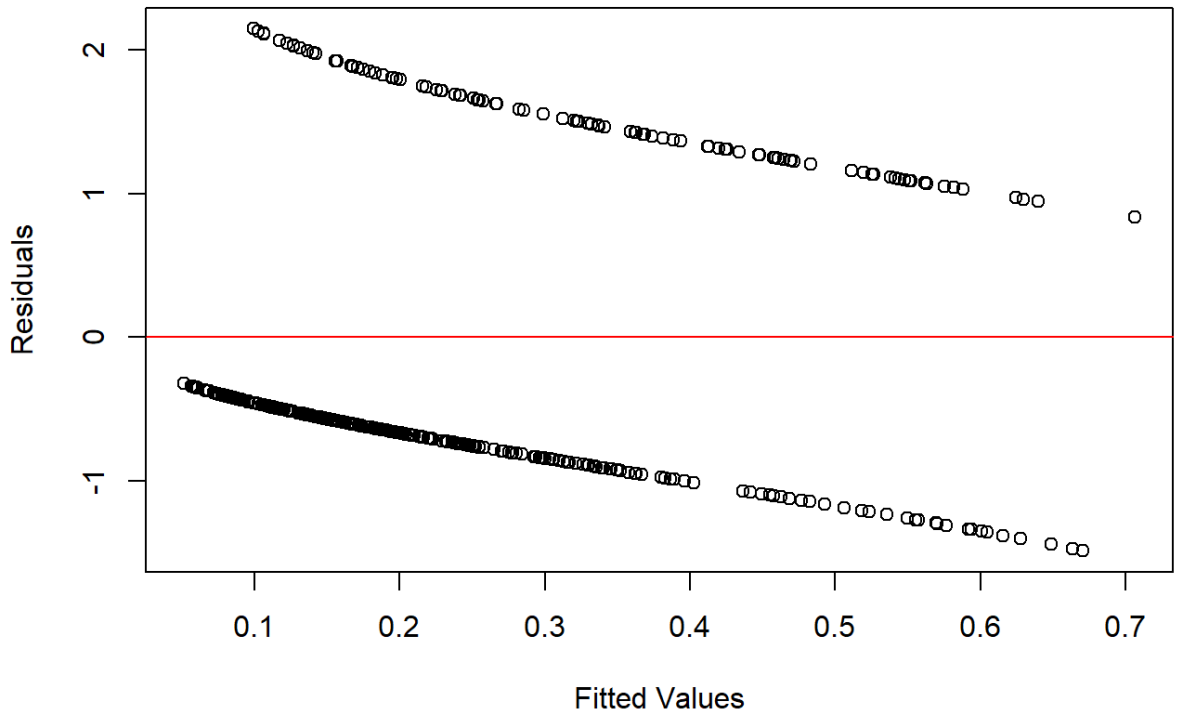



```
auc_model2_test <- auc(roc_curve_test2)

print(paste("AUC for Model 2 on Testing Data:", auc_model2_test))
```

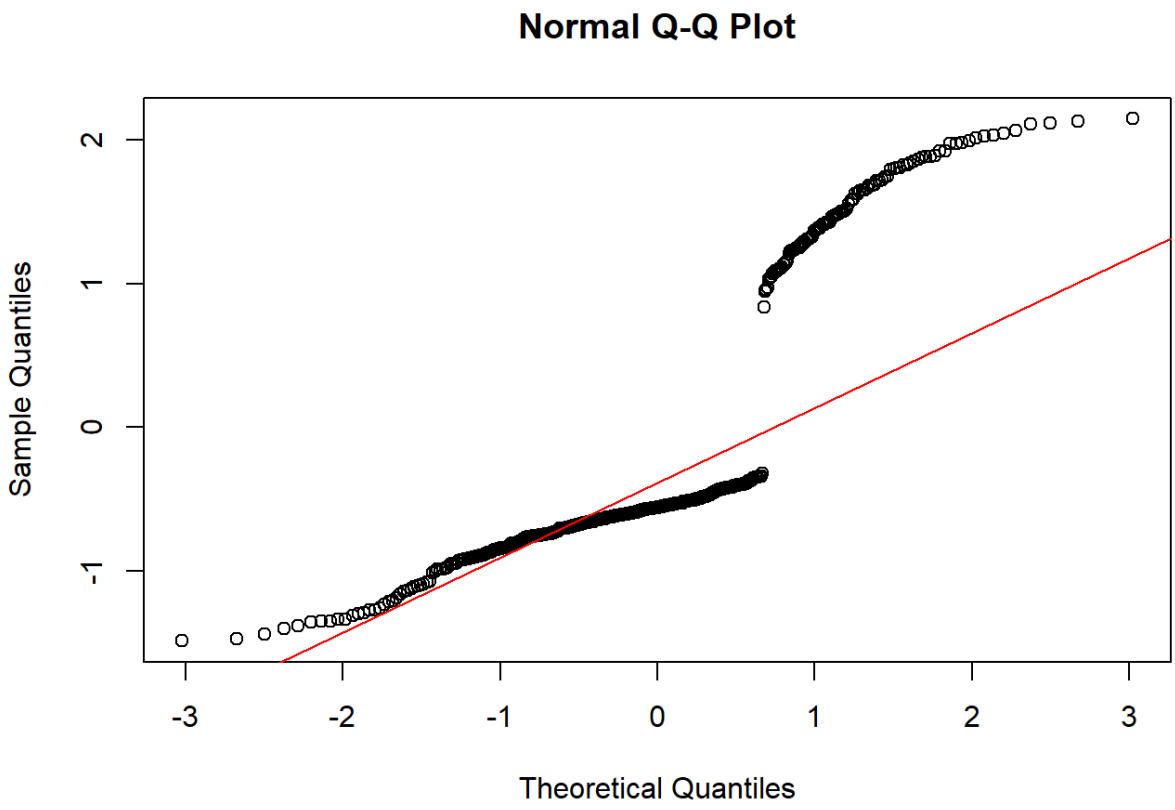
```
## [1] "AUC for Model 2 on Testing Data: 0.56"
```

Diagnostic Plots for Model Assumptions with Model 2

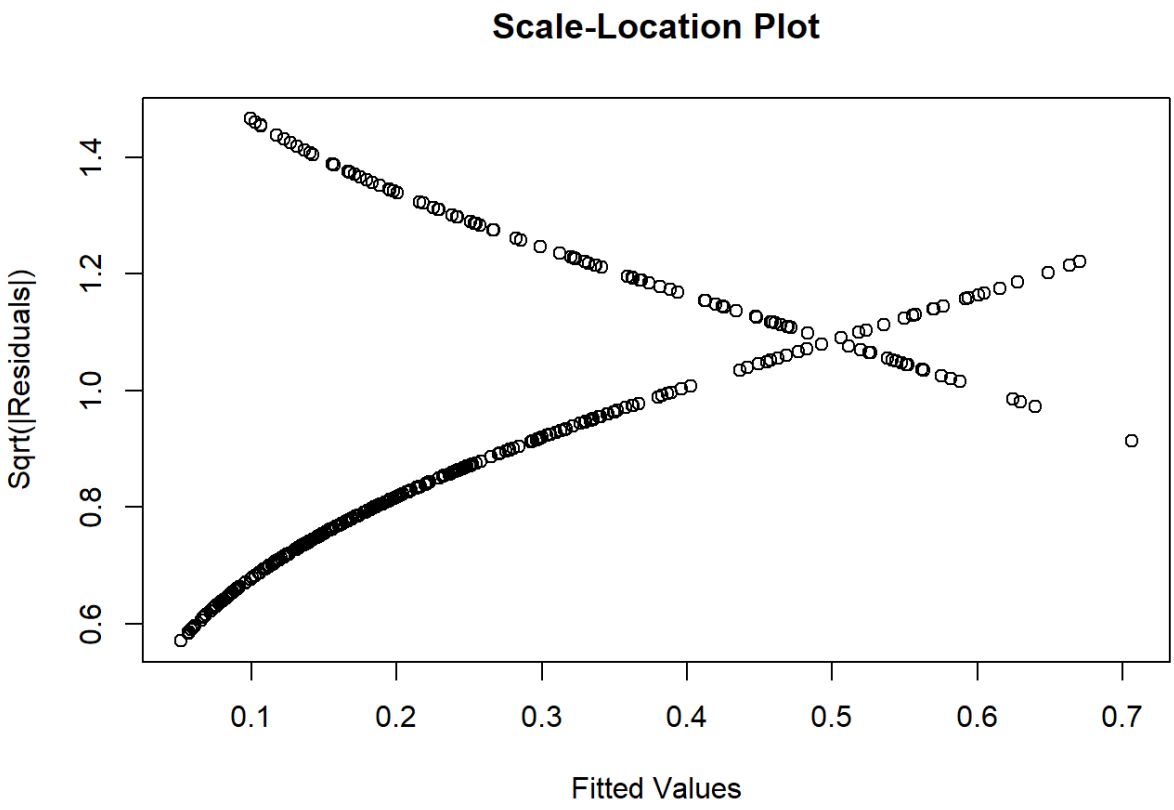
```
plot(fitted(model2_train), residuals(model2_train), xlab="Fitted Values",
ylab="Residuals")

abline(h=0, col="red")
```




```
qqnorm(residuals(model2_train))

qqline(residuals(model2_train), col="red")
```




```
plot(fitted(model2_train), sqrt(abs(residuals(model2_train))), xlab="Fitted
Values", ylab="Sqrt(|Residuals|)", main="Scale-Location Plot")

abline(h = 0, col="red")
```



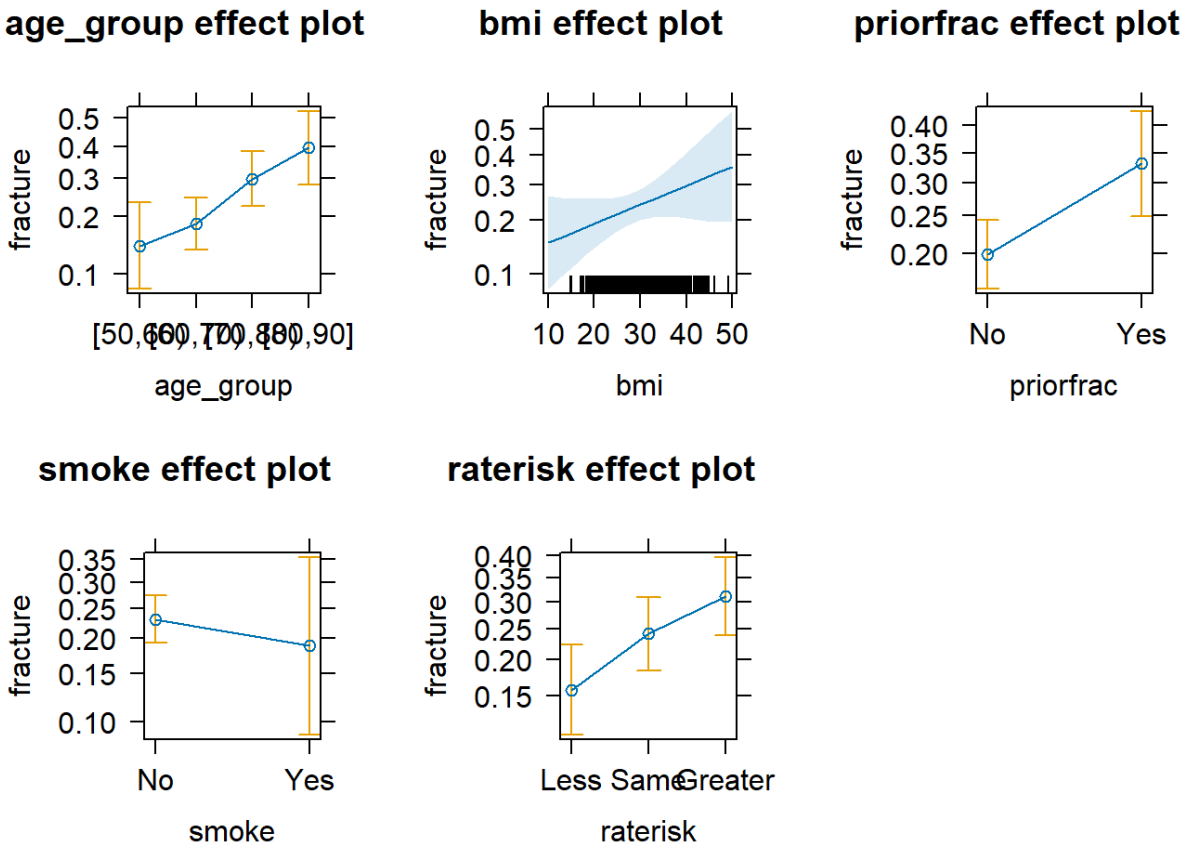
VIF Check for Model 2

```
vif(model2)
```

```
##                GVIF Df GVIF^(1/(2*Df))
## age_group 1.194966  3        1.030131
## bmi       1.112903  1        1.054942
## priorfrac 1.113189  1        1.055078
## smoke     1.016564  1        1.008248
## raterisk  1.067620  2        1.016492
```

Effect Plots for Model 2

```
plot(allEffects(model2))
```



Robust Standard Error for Model 2

```
model2_robust <- coeftest(model2, vcov = vcovHC(model2, type = "HC1"))
print(model2_robust)
```

```
##
## z test of coefficients:
##
##                   Estimate Std. Error z value  Pr(>|z|)
## (Intercept)      -3.205165   0.651828 -4.9172 8.779e-07 ***
## age_group[60,70)  0.305923   0.352708  0.8674  0.385747
## age_group[70,80)  0.942600   0.372691  2.5292  0.011433 *
## age_group[80,90]  1.378945   0.425707  3.2392  0.001199 **
## bmi               0.028746   0.017968  1.5998  0.109641
## priorfracYes      0.694237   0.247996  2.7994  0.005120 **
## smokeYes         -0.261581   0.423374 -0.6178  0.536676
## rateriskSame      0.531693   0.277662  1.9149  0.055506 .
```

```
## rateriskGreater   0.884136   0.286103  3.0903  0.002000 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Cross-validation for Generalizability of Both Models

```
cv_model1 <- cv.glm(glow_bonemed, model, K = 10)
cv_model2 <- cv.glm(glow_bonemed, model2, K = 10)
print(paste("CV Error for Model 1:", cv_model1$delta[1]))
## [1] "CV Error for Model 1: 0.179734181089004"
print(paste("CV Error for Model 2:", cv_model2$delta[1]))
## [1] "CV Error for Model 2: 0.177011822124299"
```

Final Model Evaluation and Selection

```
# Compute the prediction probabilities for the training dataset
glm_probs <- predict(model, newdata = training_data, type = "response")
rf_probs <- predict(rf_model, data = training_data, type =
"response")$predictions[, "Yes"]


# Compute the ROC curve and AUC for the GLM model
glm_roc <- roc(response = training_data$fracture, predictor = glm_probs)
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
auc_train1 <- auc(glm_roc)


# Compute the ROC curve and AUC for the Random Forest model
rf_roc <- roc(response = training_data$fracture, predictor = rf_probs)
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
auc_train2 <- auc(rf_roc)
library(pROC)
# For a GLM model
glm_probs_test <- predict(model, newdata = testing_data, type = "response")
```

```
# Compute the ROC curve
roc_test_glm <- roc(response = testing_data$fracture, predictor =
glm_probs_test)
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
# Compute the AUC
auc_model1_test <- auc(roc_test_glm)
```

Consolidate AUC Values for All Models

```
auc_values <- data.frame(
  Model = c("GLM", "Model with Age Group", "Ridge", "Lasso", "Random
Forest"),
  AUC_Training = c(auc_train1, auc_train2, NA, NA, NA), ### Add actual AUC
values
  AUC_Testing = c(auc_model1_test, auc_model2_test, ridge_performance['AUC'],
lasso_performance['AUC'], auc_rf)
)
# AIC for the GLM model
aic_model1 <- AIC(model)

# BIC for the GLM model, using BIC function
bic_model1 <- BIC(model)
# 'model' and 'model2' are already fitted
aic_model1 <- AIC(model)
bic_model1 <- BIC(model)

aic_model2 <- AIC(model2)
bic_model2 <- BIC(model2)
# For GLM models
num_predictors_model1 <- length(coef(model)) - 1  # Excluding intercept
num_predictors_model2 <- length(coef(model2)) - 1

# Extract number of non-zero coefficients (excluding intercept)
num_predictors_ridge <- sum(coef(ridge_model, s = optimal_lambda_ridge) != 0)
- 1
```

```
num_predictors_lasso <- sum(coef(lasso_model, s = optimal_lambda_lasso) != 0)
- 1

# `fracture` is the response variable

response <- glow_bonemed$fracture


# `model` is fitted glm model

predictor_glm <- predict(model, newdata = glow_bonemed, type = "response")


# `model2` is fitted glm model2

predictor_glm2 <- predict(model2, newdata = glow_bonemed, type = "response")


# calculate the ROC curves and AUCs

library(pROC)

roc_curve_glm <- roc(response, predictor_glm)

## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

roc_curve_glm2 <-roc(response, predictor_glm2)

## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

# Calculate the AUC

auc_glm <- auc(roc_curve_glm)

auc_glm2 <- auc(roc_curve_glm2)


# Print the AUC value

print(auc_glm)

## Area under the curve: 0.6394

print(auc_glm2)

## Area under the curve: 0.6998
```

Simplify the Data Frame by Including Only Necessary Metrics

```
model_performance <- data.frame(

  Model = c("GLM", "Model with Age Group", "Ridge", "Lasso", "Random
Forest"),
```

```
  AIC = c(aic_model1, aic_model2, NA, NA, NA), ### Replace NAs with actual AIC if available
  BIC = c(bic_model1, bic_model2, NA, NA, NA), ### Replace NAs with actual BIC if available
  Number_of_Predictors = c(num_predictors_model1, num_predictors_model2, NA, NA, NA) ### Calculate number of predictors for Ridge and Lasso
)
```

Final Model Comparison

```
print("AUC Values for Model Comparison")
```

```
## [1] "AUC Values for Model Comparison"
```

```
print(auc_values)
```

```
##                   Model AUC_Training AUC_Testing
## 1                   GLM    0.6327833   0.6597333
## 2 Model with Age Group    0.9732167   0.5600000
## 3                 Ridge           NA          NA
## 4                 Lasso           NA          NA
## 5         Random Forest           NA   0.8858667
```

```
print("Performance Metrics for Model Comparison")
```

```
## [1] "Performance Metrics for Model Comparison"
```

```
print(model_performance)
```

```
##                   Model      AIC      BIC Number_of_Predictors
## 1                   GLM 544.8936 557.5375                     2
## 2 Model with Age Group 532.7363 570.6677                     8
## 3                 Ridge       NA       NA                    NA
## 4                 Lasso       NA       NA                    NA
## 5         Random Forest       NA       NA                    NA
```

Cross-validation Results

```
cross_validation_results <- data.frame(
  Model = c("GLM", "Model with Age Group"),
  CV_Error = c(cv_model1$delta[1], cv_model2$delta[1]) ### CV errors from boot::cv.glm
)
```

Include insights from cross-validation

```
print("Cross-validation Results for Model Robustness")
## [1] "Cross-validation Results for Model Robustness"
print(cross_validation_results)
##                   Model  CV_Error
## 1                   GLM 0.1797342
## 2 Model with Age Group 0.1770118
```

REFERENCES: #Sources:
Hosmer, D.W., Lemeshow, S. and Sturdivant, R.X. (2013) Applied Logistic Regression, 3rd ed., New York: Wiley

https://cran.r-project.org/web/packages/aplore3/aplore3.pdf#page=11&zoom=100,132,90

Notes for Partners' Review

WHAT WE NEED TO ADD/WORK ON.
JESSICA 1-4 CALEB 5-7 RAFIA 8-9

1. **Detailed Variable Descriptions**: Ensure each variable is explicitly described, including how they might impact the study's outcome or what they represent in the context of bone health and fractures. This will help readers unfamiliar with your dataset to understand the relevance of each variable to your analysis.
2. **Data Quality Assessment**: More explicit details on data cleaning and preprocessing steps. For example:
    - Handling of missing values beyond just noting their presence. Did you impute, remove, or ignore them? What strategy was used for imputation if applied?
    - Detection and treatment of outliers, if any were present.
    - Future iterations should include explicit strategies for identifying and handling outliers, considering their potential impact on the analysis and model performance.

3. **Statistical Summary and Interpretation**:

- While summaries for numeric and categorical variables are provided, deeper interpretation of these summaries could enrich our understanding. For instance:
    - What implications do the distributions of `age`, `weight`, `height`, and `bmi` have on our study population?
    - How do proportions within `priorfrac`, `smoke`, or `bonetreat` categories potentially affect our outcomes?

4. **Comprehensive Exploratory Data Analysis (EDA)**:

- In addition to initial plots and 3D interactive plots, incorporating histograms, density plots, and box plots could offer more detailed distribution insights.
- Our observations (e.g., the relationship between weight and age) warrant further exploration with statistical tests to affirm or challenge these initial findings.

5. **Correlation Analysis**:

- A thorough discussion on the correlation between variables, especially key ones, could offer additional insights. Questions to consider include:
    - How do these correlations influence our modeling choices?
    - What does the presence of multicollinearity imply for our models?

6. **Modeling Details**:

- Each model's selection should be justified, with a clear interpretation of coefficients and a discussion on model fit and diagnostics.
- For models employing regularization, the process for selecting hyperparameters like lambda should be detailed.

7. **Validation and Testing**:

- The rationale behind the division of training and testing sets needs clarification, as does the impact of this division on model performance.
- The choice of performance metrics (e.g., AUC, accuracy) should be justified.

8. **Comparative Analysis**:

- A direct comparison between models, focusing on strengths and weaknesses, would be beneficial. This could be supported by a table summarizing key metrics for each model.

9. **Conclusion and Step to Future Work – How we are getting to Objective 2**:

- Summarizing key findings and their implications is crucial for providing clear takeaways from our analysis.

Code chunks eliminated:

- Sections on model training, testing, and evaluation have been streamlined for clarity. However, detailed review and replication of these processes remain essential for validating our findings and preparing for Objective 2.

Code chunks eliminated:

Modeling

Split the data into a training/testing set

set.seed(4) trainingIndices = sample(c(1:dim(glow_bonemed)[1]), dim(glow_bonemed)[1]*0.8) trainingDataframe = glow_bonemed[trainingIndices,] testingDataframe = glow_bonemed[-trainingIndices,]

Age is the only statistically significant continuous variable at the alpha = 0.2 level (p < 0.0001

model = glm(fracture ~ age + weight + height + bmi, data = glow_bonemed, family = “binomial”) summary(model) AIC(model)

library(ResourceSelection) hoslem.test(model$y,fitted(model)) # shows non-significant test result which means this is a decent model fit

get odds ratio for model

exp((model$coefficients))

get confidence intervals

exp(confint(model))

trying to figure out how to use sjPlot to mimic what we did in unit12 prelive

#plot_model(model, type = “pred”, terms = c(“age”, “smoke”))

corr_vars <- c(“age”, “weight”, “height”, “bmi”, “fracscore”) pc.result<-prcomp(glow_bonemed[, corr_vars],scale.=TRUE) #Eigen Vectors pc.result rotation #Eigen Values eigenvals<-pc.result rotation #Eigen Valueseigenvals<-pc.resultsdev^2 eigenvals #Scree plot par(mfrow = c(1,2)) plot(eigenvals,type = “l”, main = “Scree Plot”, ylab = “Eigen Values”, xlab = “PC #”) plot(eigenvals / sum(eigenvals), type = “l”, main = “Scree Plot”, ylab = “Prop. Var. Explained”, xlab = “PC #”, ylim = c(0, 1)) cumulative.prop = cumsum(eigenvals / sum(eigenvals)) lines(cumulative.prop, lty = 2)

Loess curve for fracscore by bonetreatment group showing fracture or not

glow_bonemedbonetreat.num<−ifelse(glowbonemed$bonetreat.num<-ifelse(glowbonemedbonetreat == “No”, 0, 1) ggplot(glow_bonemed, aes(x = fracscore, y = bonetreat.num, color = fracture)) + geom_jitter()+ geom_smooth(method=“loess”,size=1,span=1)+ ylim(-.2,1.2) + xlab(“Fracture

Score”) + ylab(“Received bonetreatment at both iterations”) + ggtitle(“Difference in Fracture Score vs bonetreatment at both time points”) # shows that there is not an increased risk, i.e. no changes, in likelihood of fracture, whereas only receiving one or no treatments trends to increase the likelihood of a fracture as the fracture score goes up

plot breaking down to see if there is any separation

ggplot(glow_bonemed, aes(x = fracscore, y = bonetreat, color = fracture)) + geom_jitter()

in bonetreat, i.e. bone meds at both time points, in the no group, there appear to be higher fracture rates with increased fracscore, which would be predicted, i.e. if you received treatment at both times there doesn’t appear to be a correlation in fracscore and breaking a bone (fracture), vs the group that did not receive both treatments appears to be a correlation with a higher likelihood correlating to likelihood of fracture

Loess curve for fracscore by physician group showing fracture or not

table(glow_bonemed$priorfrac) # show table of prior fractures

glow_bonemedpriorfrac.num<−ifelse(glowbonemed$priorfrac.num <- ifelse(glowbonemedpriorfrac == “No”, 0, 1) # create numeric variable

glow_bonemedfracture.num<−ifelse(glowbonemed$fracture.num <- ifelse(glowbonemedfracture == “No”, 0, 1) # create numeric variable

levels(glow_bonemed$fracture)

ggplot(glow_bonemed, aes(x = fracscore, y = fracture.num, color = priorfrac)) + geom_jitter()+ geom_smooth(method="loess",size=1,span=1)+ ylim(-.2,1.2) # shows that there is not an increased risk associated with higher fracscore, i.e. no changes, in likelihood of an increased fracture if you previous had a fracture whereas the group that has never had a fracture tends to increase the likelihood of a fracture as the fracture score goes up

plot breaking down to see if there is any separation

ggplot(glow_bonemed, aes(x = bonetreat, y = priorfrac, color = fracture)) + geom_jitter()

library(caret) # plot ggplot(glow_bonemed, aes(x = age, y = ifelse(glow_bonemed$smoke == "No", 0, 1), color = fracture)) + geom_jitter()+ geom_smooth(method="loess",size=1,span=1)+ ylim(-.2,1.2)

library(pROC) set.seed(4)

#note CV and error metric are not really used here, but logLoss is reported for the final model. # set tuning parameters using logloss fitControl<- trainControl(method="repeatedcv",number=10,repeats=1,classProbs=TRUE, summaryFunction=mnLogLoss)

build glmnet model

glmnet.fit<-train(fracture ~ . - sub_id, data=trainingDataframe, method="glmnet", trControl=fitControl, metric="logLoss")
coef(glmnet.fitfinalModel,glmnet.fit$finalModel$lambdaOpt)
#Getting predictions for glmnet for Complex model glmnetfit.predprobs<-predict(glmnet.fit, trainingDataframe ,type="prob")

glmnet ROC

glmnet.roc<-roc(response= trainingDataframefracture,predictor=glmnetfit.predprobsNo,levels=c("No","Yes"))
plot(glmnet.roc,col="steelblue")

Save for later

plot(glmnet.roc,add=T,col="steelblue")

legend("bottomright",

legend=c(“Simple”, “Complex”,“GLMNET”),

col=c(“black”, “red”,“steelblue”),

lwd=4, cex =1, xpd = TRUE, horiz = FALSE)

Build complex model with interactions and/or polynomials

model1.2 = glm(fracture ~ age + weight + height + bmi + bonetreat + fracscore + armassist + bonetreat:fracscore, data = glow_bonemed, family = “binomial”) summary(model1.2) AIC(model1.2)

library(ResourceSelection) hoslem.test(model1.2$y,fitted(model1.2)) # shows non-significant test result which means this is a decent model fit

get odds ratio for model

exp((model1.2$coefficients))

get confidence intervals

exp(confint(model1.2))

skeleton code

library(caret) fitControl<-trainControl(method=“repeatedcv”,number=5,repeats=1,classProbs=TRUE, summaryFunction=mnLogLoss)

set.seed(4)

#Version 1 lda.fit<-train(fracture ~ ., data=trainingDataframe, method=“lda”, trControl=fitControl, metric=“logLoss”)

#Computing predicted probabilities on the training data predictions <- predict(lda.fit, trainingDataframe, type = “prob”)[,“Yes”]

summary(predictions)

#Getting confusion matrix threshold=0.0468 lda.preds<- factor(ifelse(predictions>threshold,“Yes”,“No”),levels=c(“Yes”,“No”)) confusionMatrix(data = lda.preds, reference = trainingDataframe$fracture)

library(ranger) # set tuning parameters using logloss fitControl<- trainControl(method=“repeatedcv”,number=5,repeats=1,classProbs=TRUE, savePredictions = T)

names(trainingDataframe) randomForestModel<-train(fracture ~ . - sub_id, data=trainingDataframe, method="ranger", trControl=fitControl, preProc = c("center", "scale"))

summary(randomForestModel) randomForestModel$results

library(MLeval) result <- evalm(randomForestModel)

#get AUROC result$roc

skeleton code for models using glm function

library(sjPlot) library(sjmisc) names(glow_bonemed) #plot_model(glmnet.fit,type="pred", terms = c("fracscore")) #plot_model(complex1,type="pred",terms=c("Pclass","Age[5,15,30,45]")) #plot_model(complex1,type="pred",terms=c("Age","Sex","Pclass"))

skeleton code for models using caret package (train) function

library(pROC) #Predicting probabilities on the training data glmnet.predprobs<- predict(glmnet.fit,Rose,type="prob") #note if we were using a caret model type="raw" glmnet.roc<- roc(response=RoseSurvived2,predictor=glmnet.predprobs$Survived2, predictor=glmnet.pred probsSurvived,levels=c("Perished","Survived"))
plot(simple.roc) plot(complex1.roc,print.thres="best",col="red",add=T)
plot(glmnet.roc,add=T,col="lightblue") legend("bottomright", legend=c("Simple", "Complex","GLMNET"), col=c("black", "red","lightblue"), lwd=4, cex =1, xpd = TRUE, horiz = FALSE)

plot(log.roc,print.thres="best") #This graph is nice because the x axis is plotted in terms of specificity rather than FPR

auc(log.roc)