Data Analysis on Diabetes

Sai Mulagan

2023-12-11

This project focuses on the analysis of diabetes-related data to gain insights into factors affecting diabetes management. The dataset comprises various variables, including stable glucose levels (stab.glu), HDL cholesterol levels (hdl), glycosylated hemoglobin levels (glyhb), and a target variable glyhb_star representing a measure of diabetes control. The primary objective is to identify the most influential factors associated with glyhb_star and develop a predictive model for assessing diabetes control.

Data Analysis:

Data exploration and data splitting.

1.

In the dataset:

Quantitative Variables: cholesterol levels (chol), stable glucose levels (stab.glu), HDL cholesterol levels (hdl), ratio, glycosylated hemoglobin levels (glyhb), age, height, weight, systolic blood pressure (bp.1s), diastolic blood pressure (bp.1d), waist circumference, hip circumference, and time spent on physical/nutritional planning (time.ppn).

Qualitative Variables: location, gender, and body frame size (frame).

```
library(ggplot2)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
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
```

```
diabetes_data <- read.table("diabetes.txt", header = TRUE)
str(diabetes_data)
```

```
## 'data.frame':    366 obs. of  16 variables:
##  $ chol    : int  203 165 228 78 249 248 195 177 263 242 ...
##  $ stab.glu: int  82 97 92 93 90 94 92 87 89 82 ...
```

```
##  $ hdl     : int  56 24 37 12 28 69 41 49 40 54 ...
##  $ ratio   : num  3.6 6.9 6.2 6.5 8.9 ...
##  $ glyhb   : num  4.31 4.44 4.64 4.63 7.72 ...
##  $ location: chr  "Buckingham" "Buckingham" "Buckingham" "Buckingham" ...
##  $ age     : int  46 29 58 67 64 34 30 45 55 60 ...
##  $ gender  : chr  "female" "female" "female" "male" ...
##  $ height  : int  62 64 61 67 68 71 69 69 63 65 ...
##  $ weight  : int  121 218 256 119 183 190 191 166 202 156 ...
##  $ frame   : chr  "medium" "large" "large" "large" ...
##  $ bp.1s   : int  118 112 190 110 138 132 161 160 108 130 ...
##  $ bp.1d   : int  59 68 92 50 80 86 112 80 72 90 ...
##  $ waist   : int  29 46 49 33 44 36 46 34 45 39 ...
##  $ hip     : int  38 48 57 38 41 42 49 40 50 45 ...
##  $ time.ppn: int  720 360 180 480 300 195 720 300 240 300 ...
```

```r
summary(diabetes_data)
```

```
##       chol          stab.glu          hdl             ratio
##  Min.   : 78.0   Min.   : 48.0   Min.   : 12.00   Min.   : 1.500
##  1st Qu.:179.0   1st Qu.: 81.0   1st Qu.: 38.00   1st Qu.: 3.200
##  Median :203.5   Median : 90.0   Median : 46.00   Median : 4.200
##  Mean   :207.5   Mean   :107.4   Mean   : 50.27   Mean   : 4.536
##  3rd Qu.:228.8   3rd Qu.:108.0   3rd Qu.: 59.00   3rd Qu.: 5.400
##  Max.   :443.0   Max.   :385.0   Max.   :120.00   Max.   :19.300
##      glyhb          location              age           gender
##  Min.   : 2.680   Length:366         Min.   :19.00   Length:366
##  1st Qu.: 4.393   Class :character   1st Qu.:34.00   Class :character
##  Median : 4.860   Mode  :character   Median :45.00   Mode  :character
##  Mean   : 5.607                      Mean   :46.69
##  3rd Qu.: 5.630                      3rd Qu.:60.00
##  Max.   :16.110                      Max.   :92.00
##      height          weight         frame               bp.1s
##  Min.   :52.00   Min.   : 99.0   Length:366         Min.   : 90.0
##  1st Qu.:63.00   1st Qu.:151.0   Class :character   1st Qu.:121.2
##  Median :66.00   Median :174.0   Mode  :character   Median :136.0
##  Mean   :66.05   Mean   :178.1                      Mean   :137.2
##  3rd Qu.:69.00   3rd Qu.:200.0                      3rd Qu.:148.0
##  Max.   :76.00   Max.   :325.0                      Max.   :250.0
##      bp.1d            waist            hip           time.ppn
##  Min.   : 48.00   Min.   :26.00   Min.   :30.00   Min.   :   5.00
##  1st Qu.: 75.00   1st Qu.:33.00   1st Qu.:39.00   1st Qu.:  93.75
##  Median : 82.00   Median :37.00   Median :42.00   Median : 240.00
##  Mean   : 83.36   Mean   :37.93   Mean   :43.05   Mean   : 339.04
##  3rd Qu.: 92.00   3rd Qu.:41.75   3rd Qu.:46.00   3rd Qu.: 480.00
##  Max.   :124.00   Max.   :56.00   Max.   :64.00   Max.   :1560.00
```

```r
# Identifying quantitative and qualitative variables
quantitative_vars <- names(diabetes_data)[sapply(diabetes_data, is.numeric)]
qualitative_vars <- names(diabetes_data)[sapply(diabetes_data, function(x) is.factor(x) | is.character(x
quantitative_vars
```

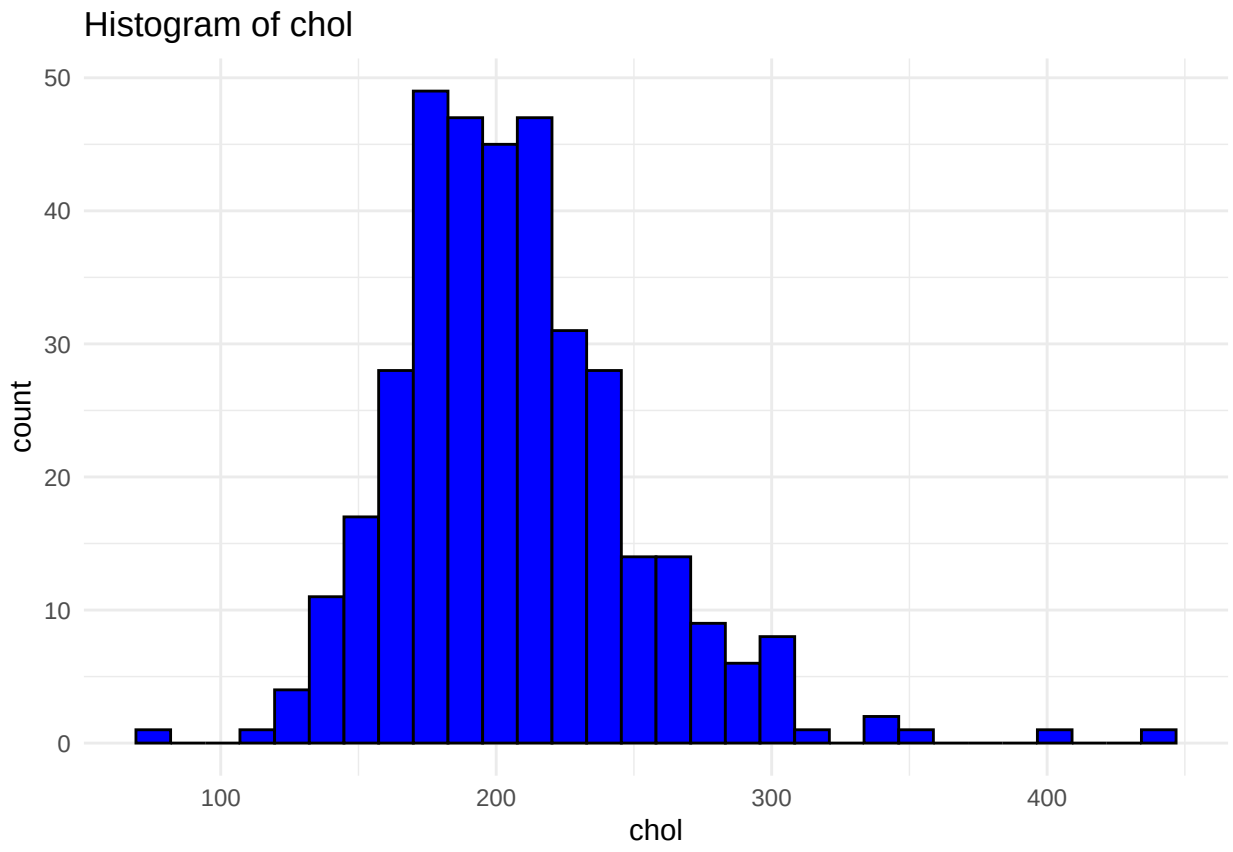
```
##  [1] "chol"     "stab.glu" "hdl"      "ratio"    "glyhb"    "age"
##  [7] "height"   "weight"   "bp.1s"    "bp.1d"    "waist"    "hip"
## [13] "time.ppn"
```

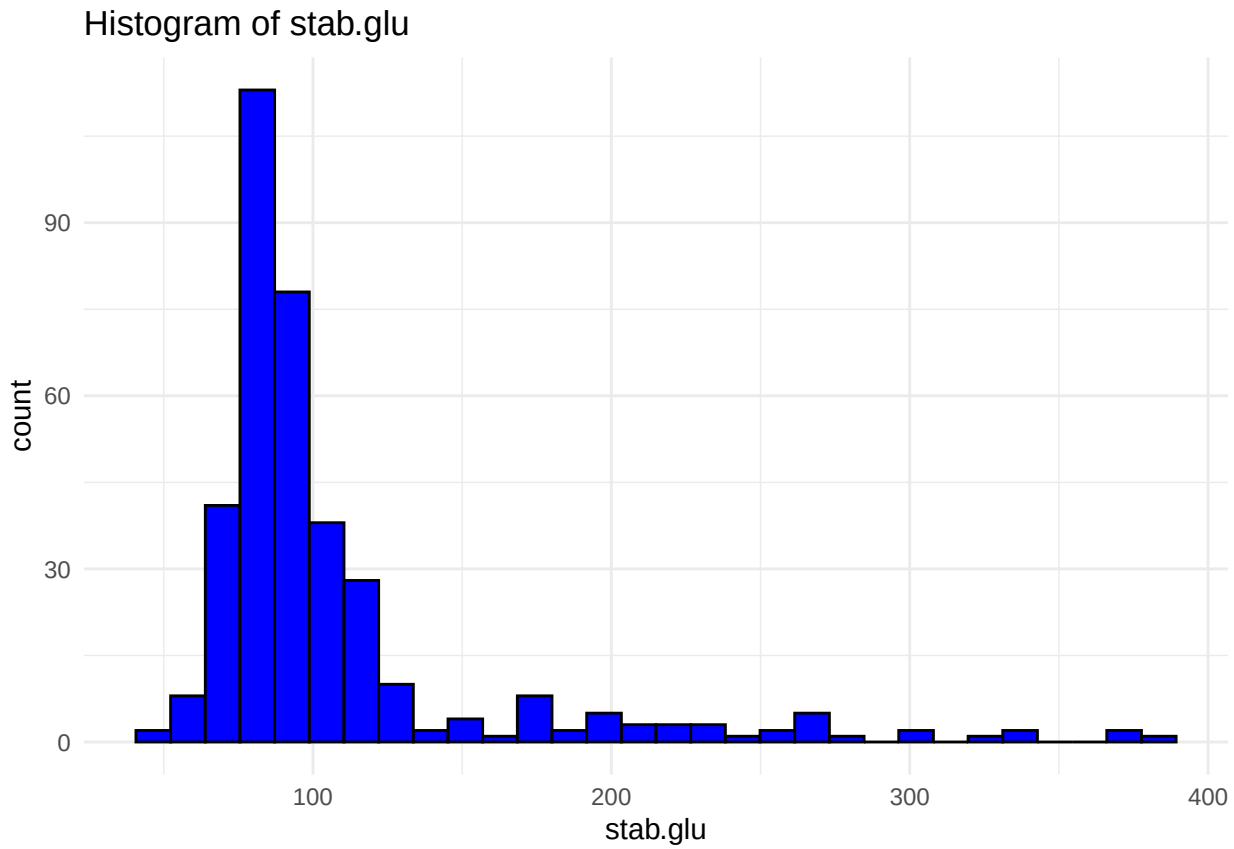
```
qualitative_vars
```

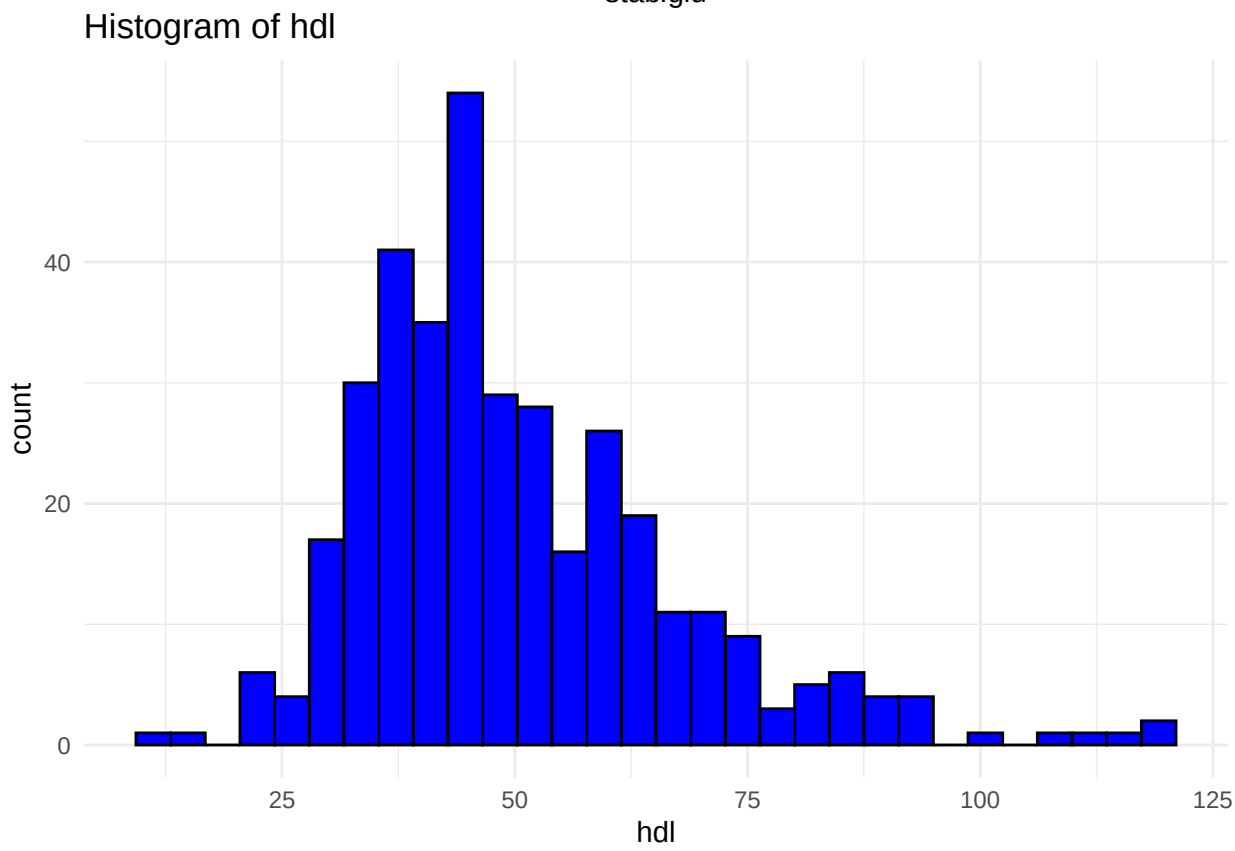
```
## [1] "location" "gender"   "frame"
```

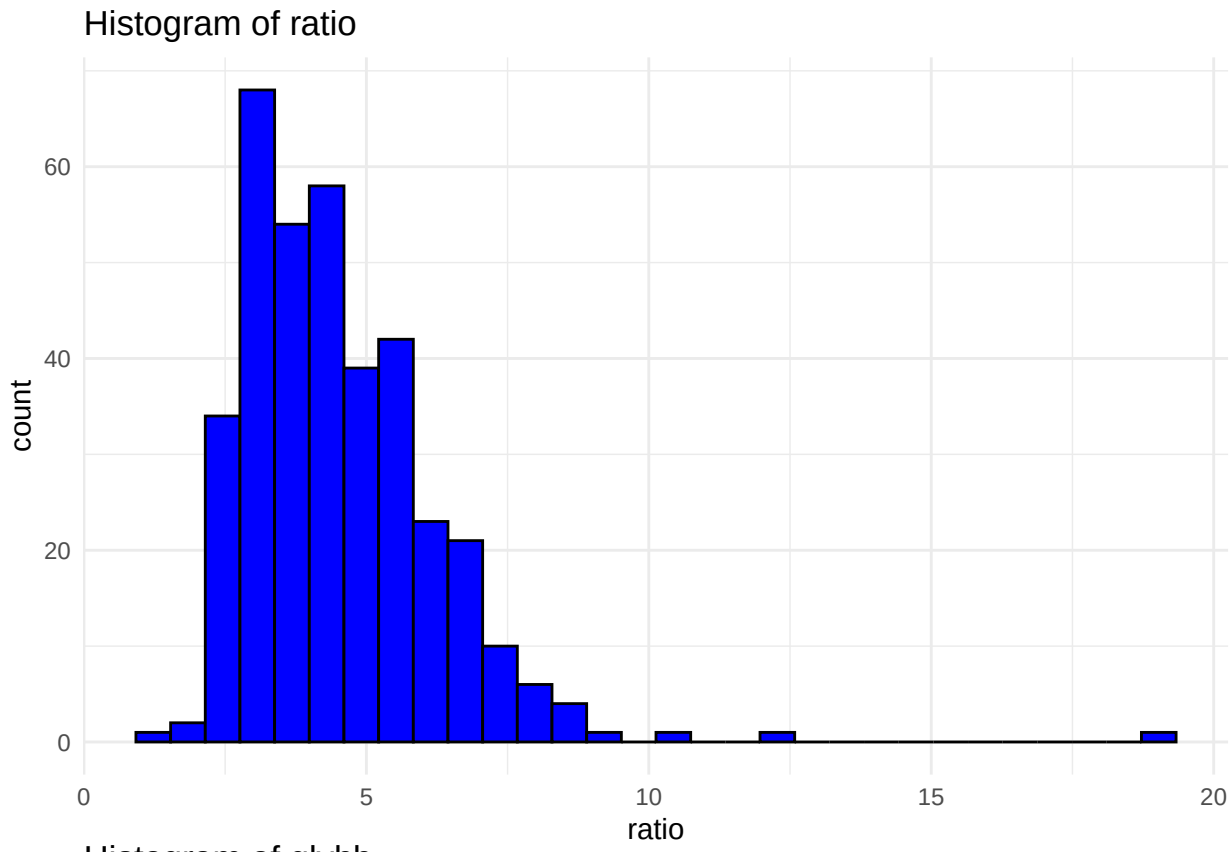
```
# Plot histograms for each quantitative variable
for (var in quantitative_vars) {
  p <- ggplot(diabetes_data, aes_string(x = var)) +
       geom_histogram(bins = 30, fill = "blue", color = "black") +
       theme_minimal() +
       ggtitle(paste("Histogram of", var))
  print(p)
}
```

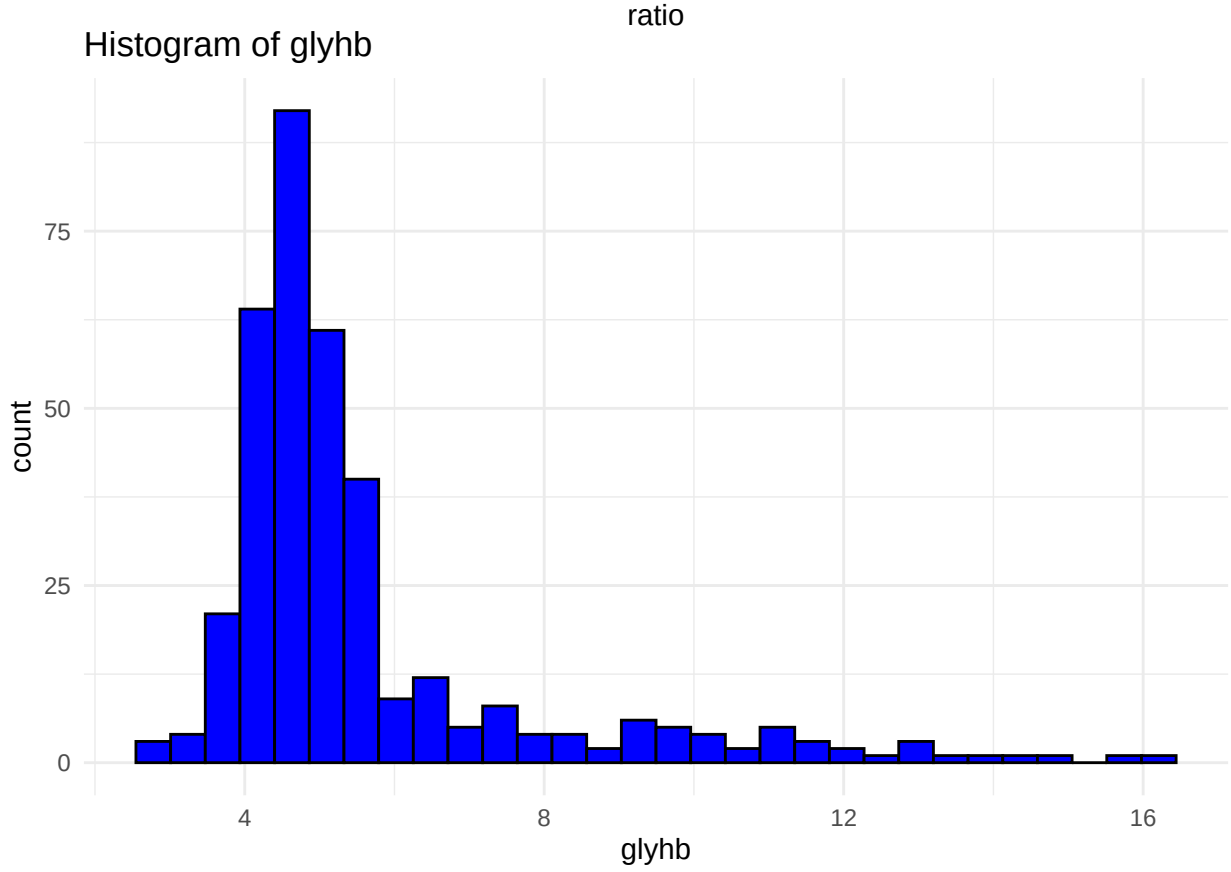
```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

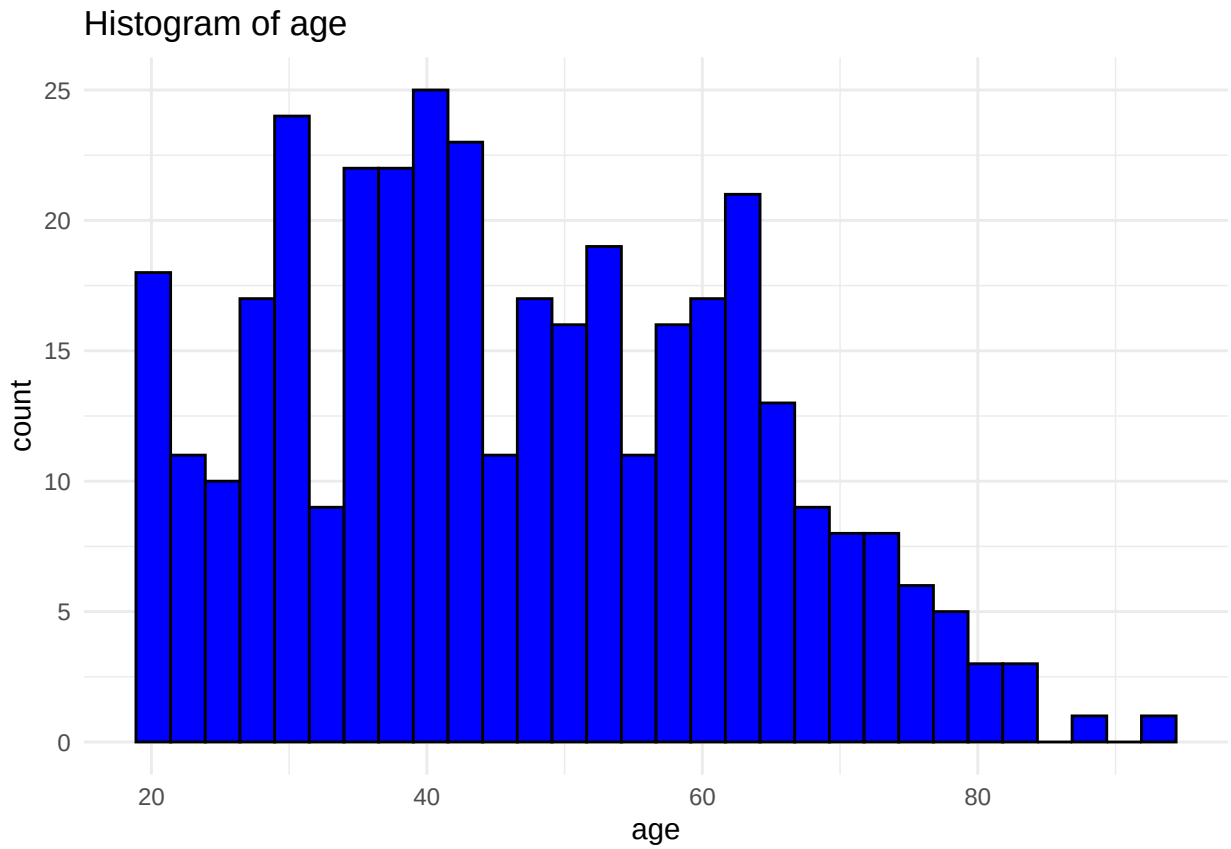


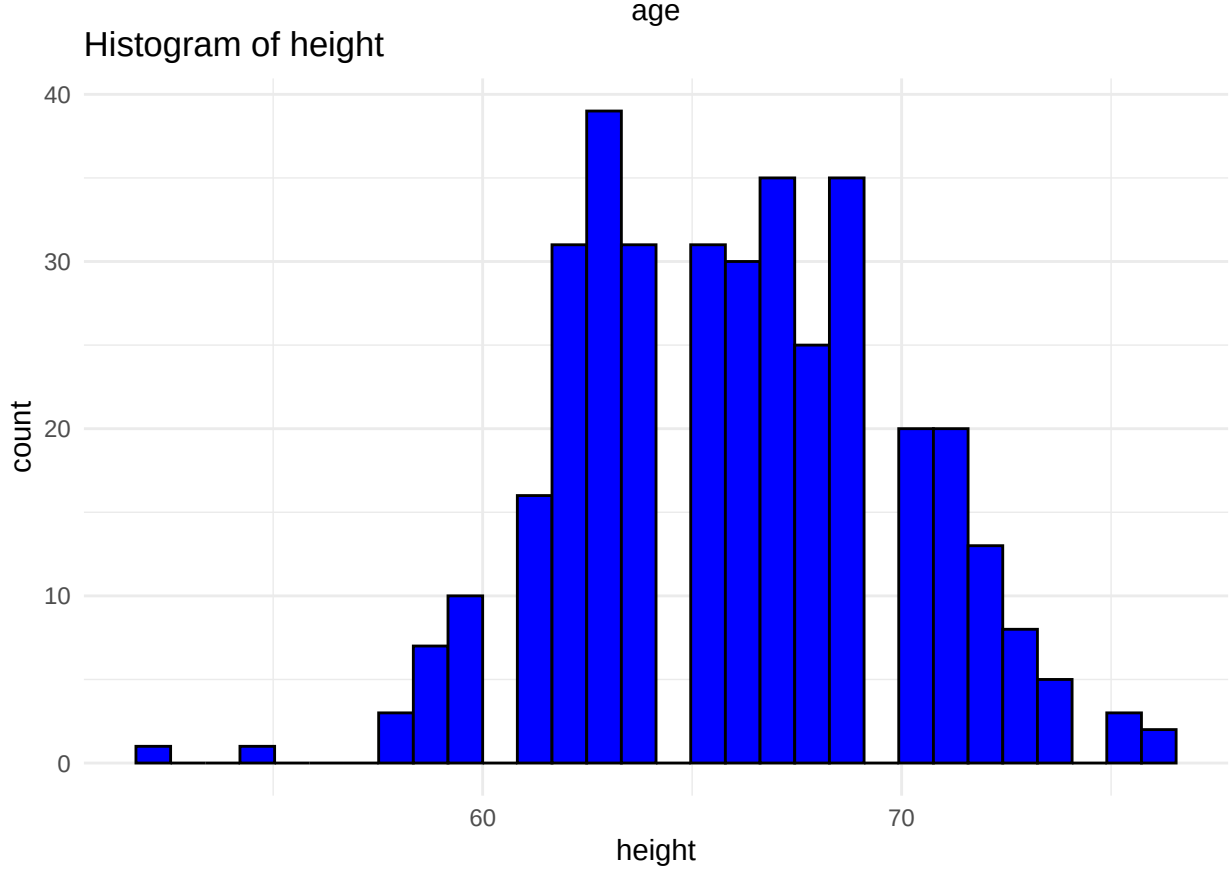

Histogram of stab.glu
count
90
60
30
0
100
200
300
400
stab.glu

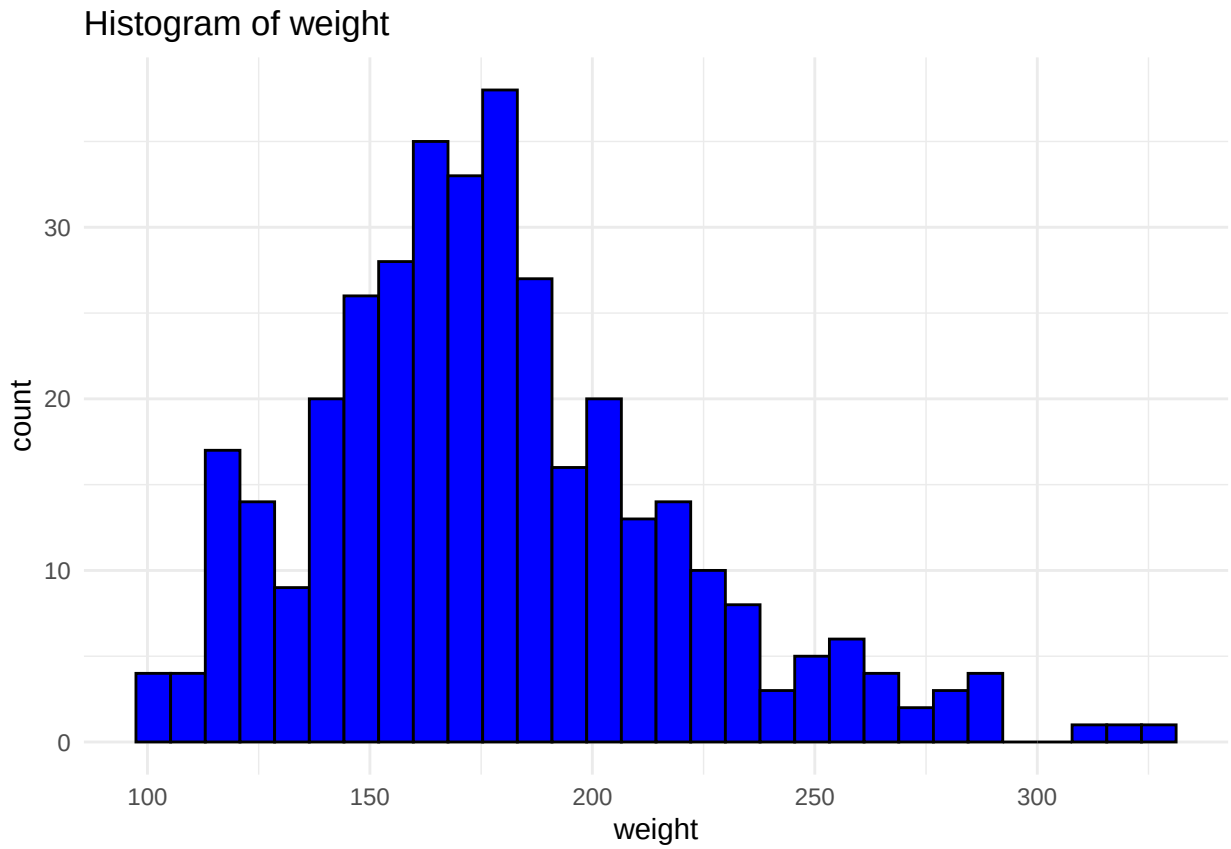

Histogram of hdl
count
40
20
0
25
50
75
100
125
hdl

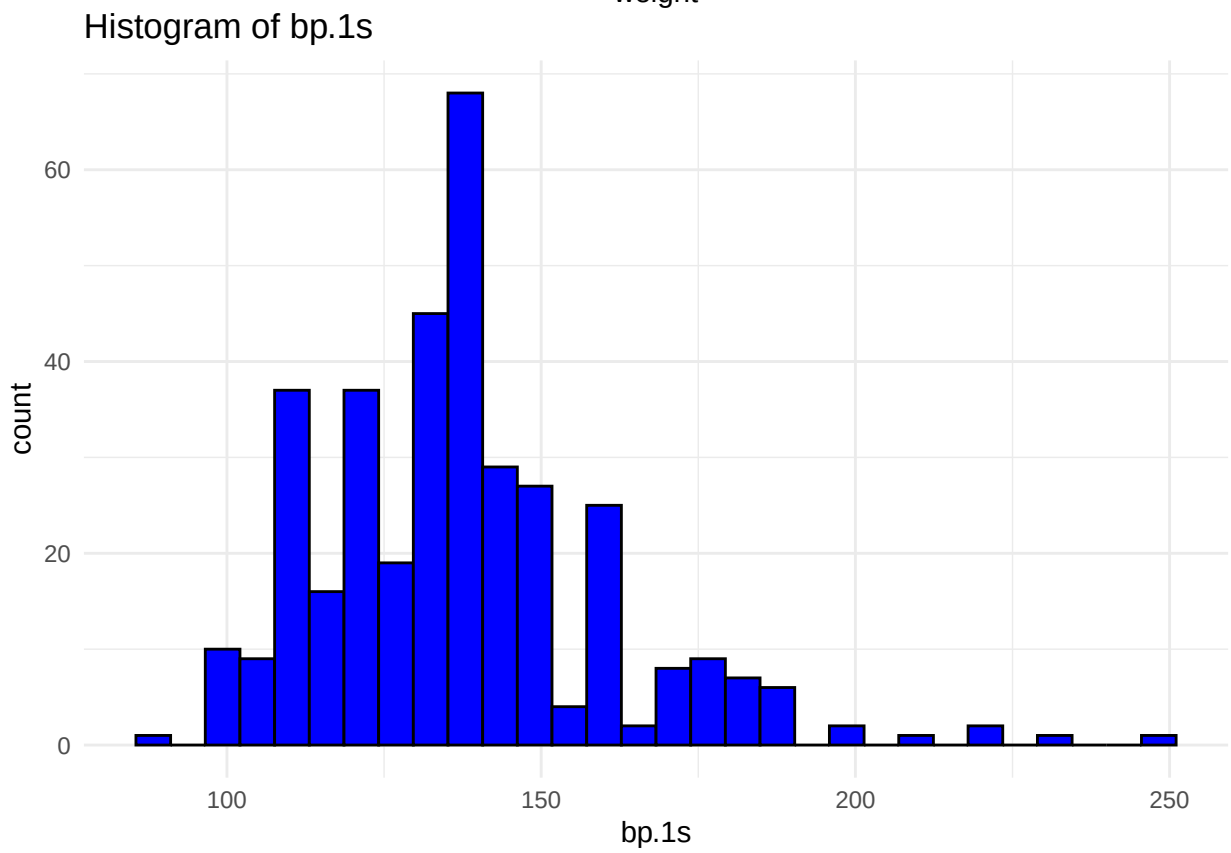

Histogram of ratio
count
ratio
0
20
40
60
0
5
10
15
20

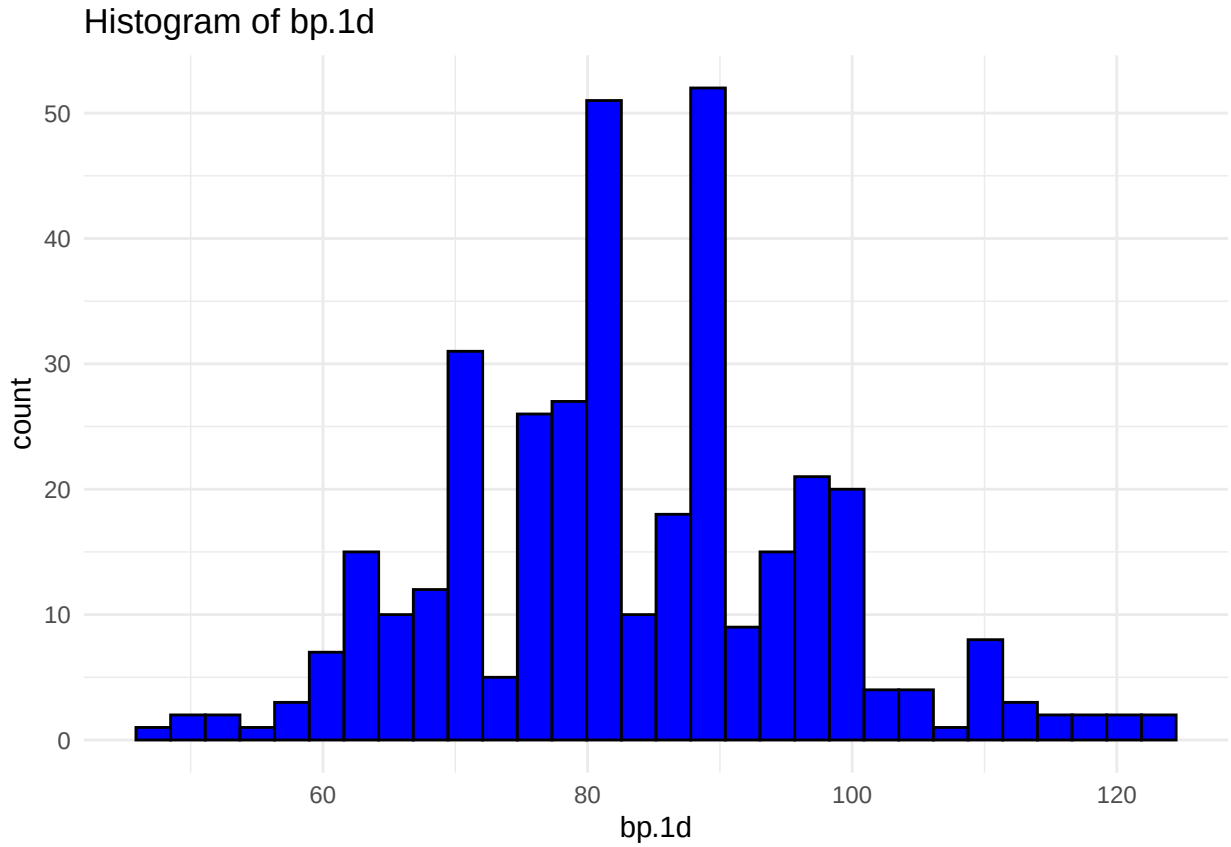

Histogram of glyhb
count
glyhb
0
25
50
75
4
8
12
16

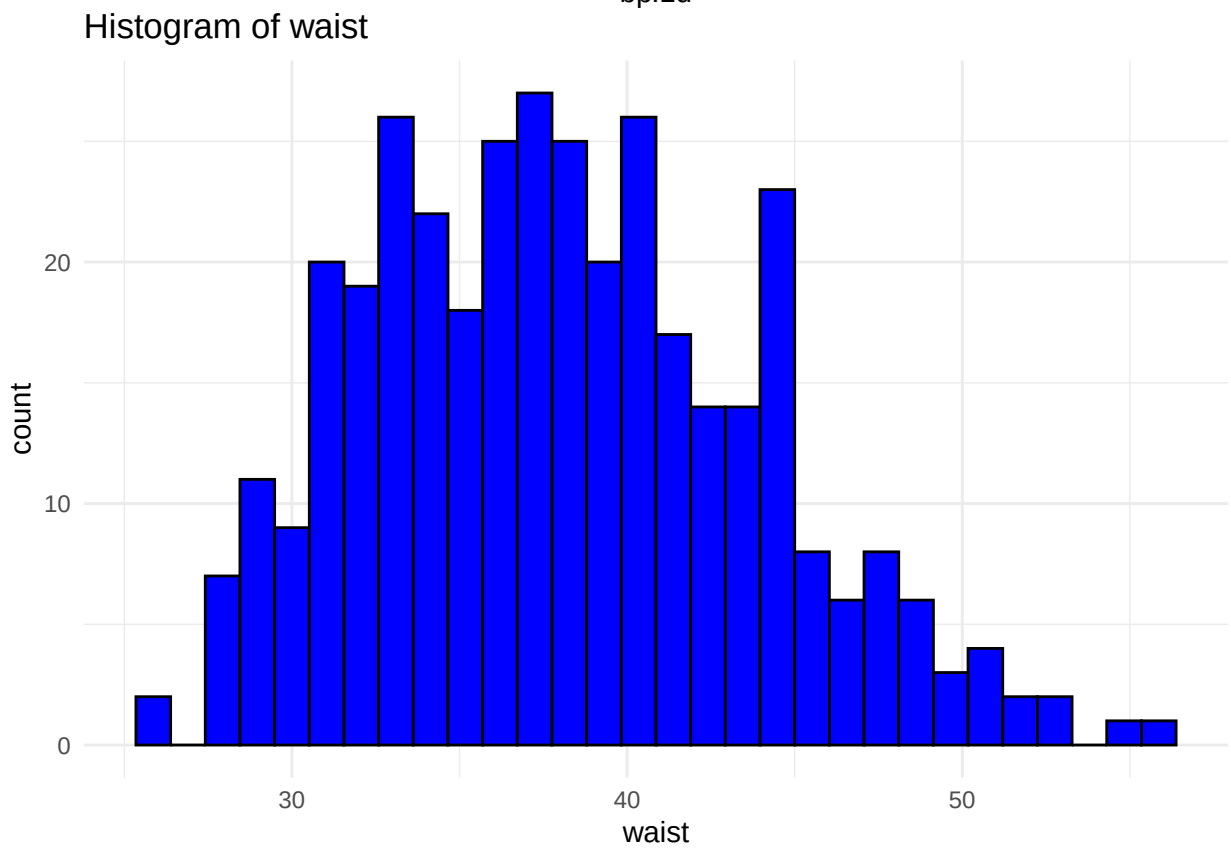

Histogram of age
count
age
0
5
10
15
20
25
20
40
60
80

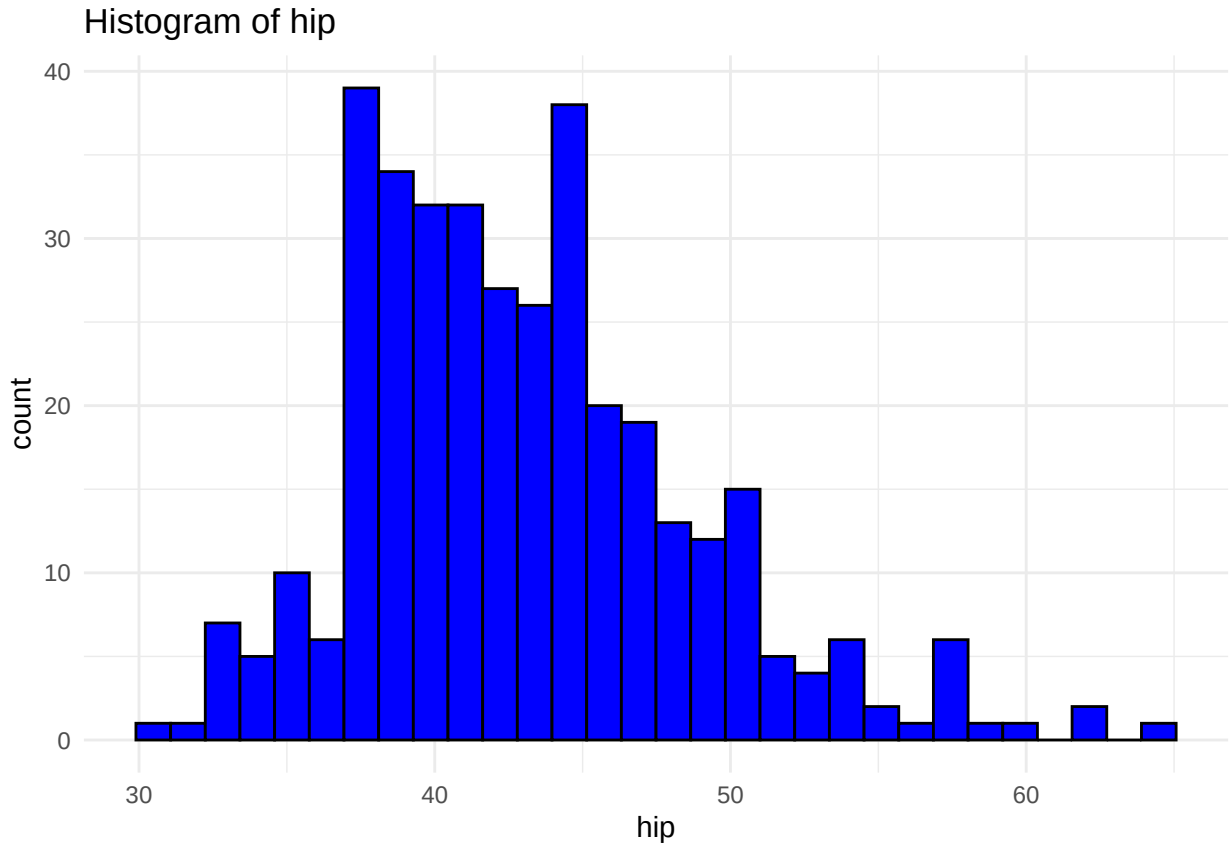

Histogram of height
count
height
0
10
20
30
40
60
70

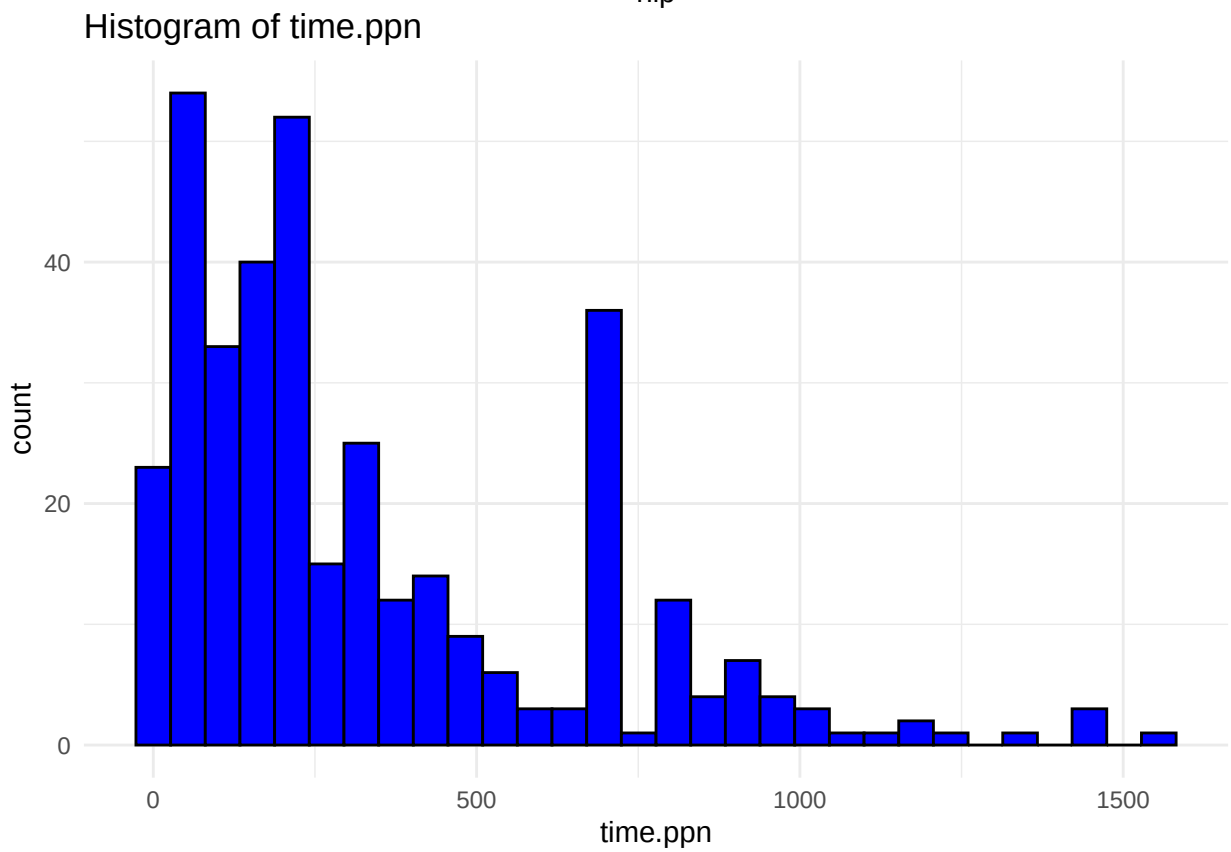

Histogram of weight
count
30
20
10
0
100
150
200
250
300
weight


Histogram of bp.1s
count
60
40
20
0
100
150
200
250
bp.1s


Histogram of bp.1d
count
bp.1d
0
10
20
30
40
50
60
80
100
120


Histogram of waist
count
waist
0
10
20
30
40
50





Cholesterol (chol): Appears to have a roughly normal distribution.

Stable Glucose (stab.glu): Skewed to the right, indicating higher values in fewer subjects.

High-Density Lipoprotein (hdl): Right-skewed, similar to stable glucose.

Ratio: Slightly right-skewed.

Glycosolated Hemoglobin (glyhb): Shows a right-skewed distribution, important for diabetes analysis.

Age: Seems fairly uniformly distributed across the range.

Height: Shows a roughly normal distribution.

Weight: Right-skewed, indicating that fewer subjects have higher weight.

Systolic Blood Pressure (bp.1s): Somewhat normal distribution with a slight right skew.

Diastolic Blood Pressure (bp.1d): Also slightly right-skewed.

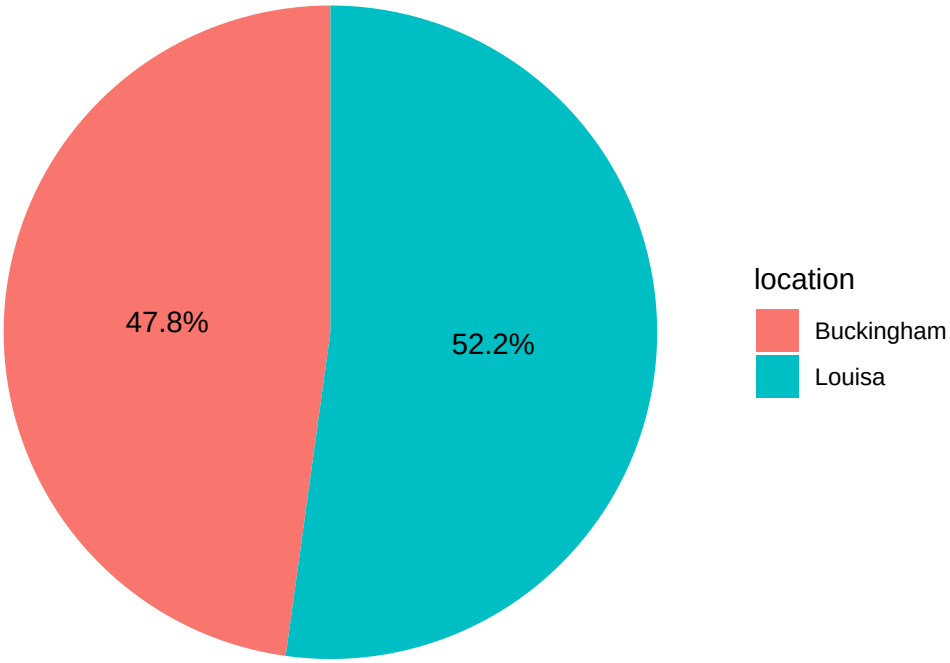
Waist: Right-skewed, similar to weight.

Hip: Appears roughly normal but with some skewness to the right.

Time in Physical/Physician's Care (time.ppn): Highly right-skewed.

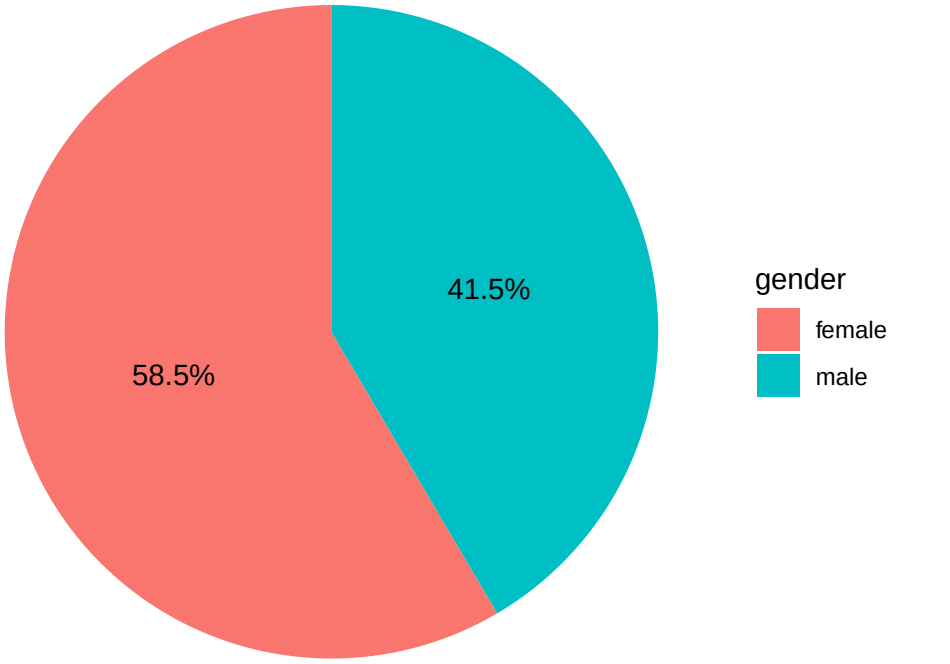
```r
library(ggplot2)
library(dplyr)

# Function to create a pie chart with percentages using ggplot2
create_pie_chart <- function(data, var_name) {
    data %>%
        count(!!sym(var_name)) %>%
        mutate(perc = n / sum(n) * 100) %>%
        ggplot(aes(x = "", y = n, fill = !!sym(var_name))) +
        geom_bar(stat = "identity", width = 1) +
        coord_polar("y", start = 0) +
        theme_void() +
        geom_text(aes(label = paste0(round(perc, 1), "%")), position = position_stack(vjust = 0.5)) +
        labs(fill = var_name, title = paste("Pie Chart of", var_name))
}

# Creating pie charts for each qualitative variable
create_pie_chart(diabetes_data, "location")
```

Pie Chart of location

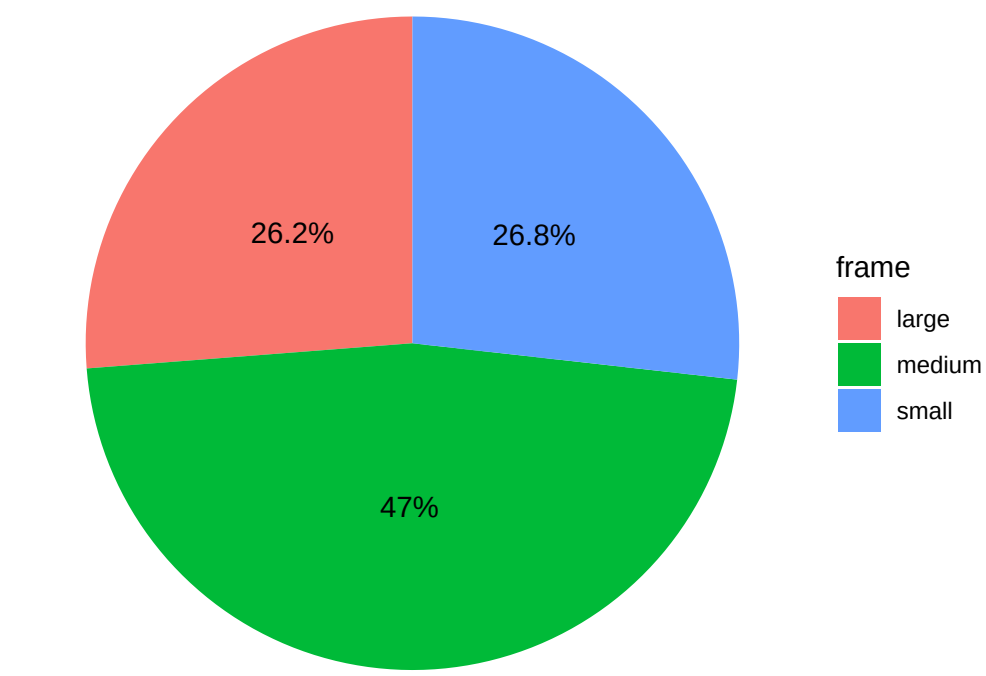



```
create_pie_chart(diabetes_data, "gender")
```

Pie Chart of gender




```
create_pie_chart(diabetes_data, "frame")
```

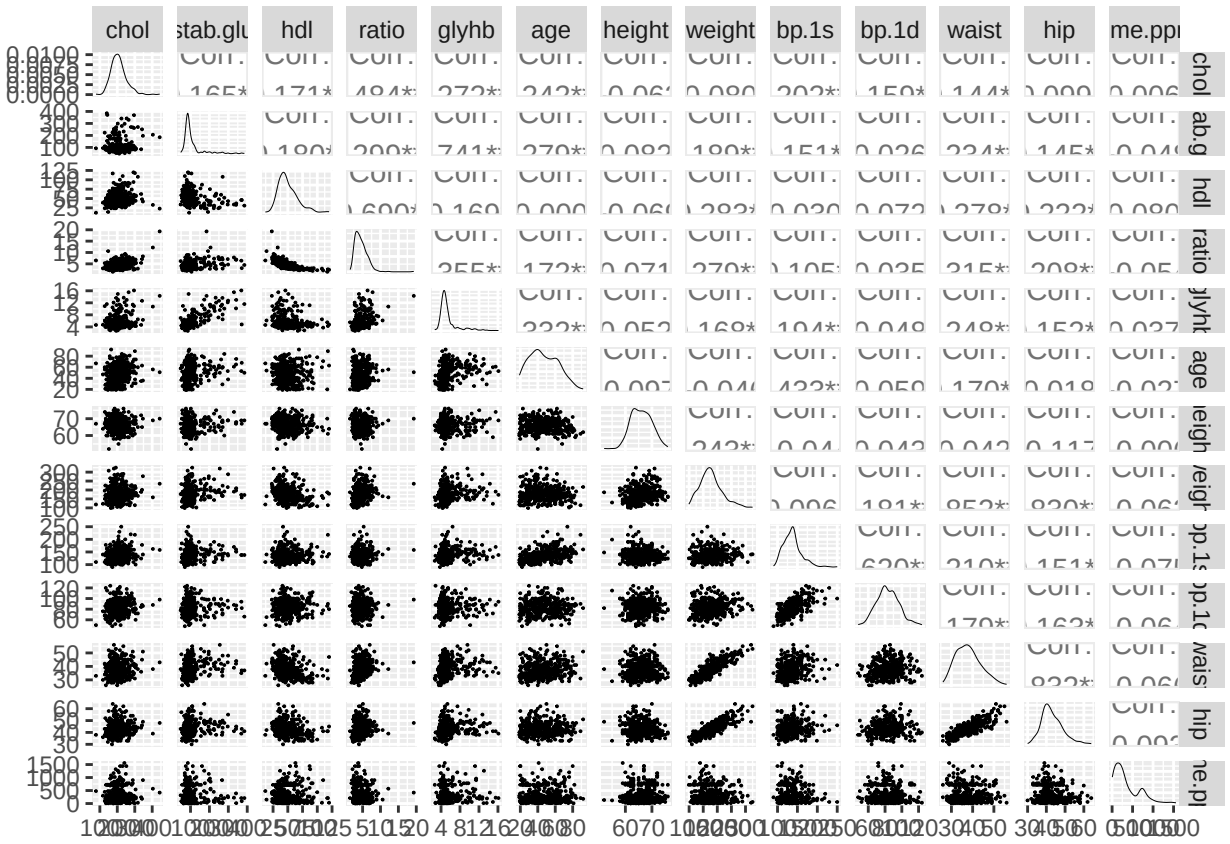


Location: The location data shows that 52.2% of the people are from Louisa and 47.8% from Buckingham. This means a little more than half of the people in the study are from Louisa. It's good that there is equal representation in the study.

Gender: There are more women than men. 58.5% of the participants are female, and 41.5% are male.

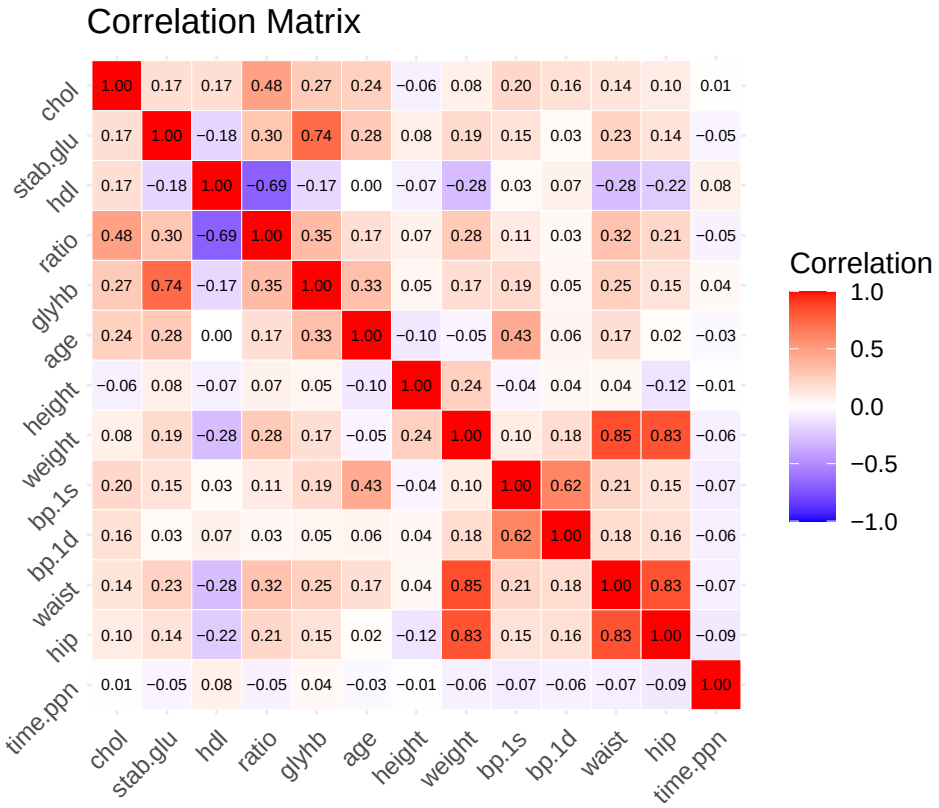
Frame: The body frame data shows that 47% of people have a large frame, while both small and medium frames are almost equally represented at around 26% each. This means almost half of the people in the study have a large body frame. This information is useful, especially when looking at how body size relates to diabetes.

```
library(GGally)
library(ggplot2)

# Select only quantitative variables for the scatterplot matrix
quantitative_vars <- diabetes_data[, c("chol", "stab.glu", "hdl", "ratio", "glyhb",
                                       "age", "height", "weight", "bp.1s", "bp.1d",
                                       "waist", "hip", "time.ppn")]

# Generating the scatterplot matrix
ggpairs(quantitative_vars,
        lower = list(continuous = wrap("points", size = .01)),
        diag = list(continuous = wrap("densityDiag", size = .1)))
```




```
library(ggplot2)
library(ggpubr)
library(magrittr)

# Select only quantitative variables for the correlation matrix
quantitative_vars <- diabetes_data[, c("chol", "stab.glu", "hdl", "ratio", "glyhb",
                                       "age", "height", "weight", "bp.1s", "bp.1d",
                                       "waist", "hip", "time.ppn")]

# Compute the correlation matrix
cor_matrix <- cor(quantitative_vars, use = "complete.obs")

# Create a dataframe from the correlation matrix
cor_df <- as.data.frame(as.table(cor_matrix))

# Reverse the order of variables on the y-axis
cor_df$Var2 <- factor(cor_df$Var2, levels = rev(unique(cor_df$Var2)))

# Plot using ggplot2
ggplot(cor_df, aes(Var1, Var2, fill = Freq)) +
    geom_tile(color = "white") +
    geom_text(aes(label = sprintf("%.2f", Freq)), size = 2, color = "black") +
    scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                         midpoint = 0, limit = c(-1,1), space = "Lab",
                         name="Correlation") +
    theme_minimal() +
    coord_fixed() +
```

```
    labs(x='', y='', title='Correlation Matrix') +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          axis.text.y = element_text(angle = 45, hjust = 1))
```



Strong Correlations: Some variables exhibit strong correlations with each other. For example, variables related to body measurements such as weight, waist, and hip sizes are strongly correlated. Similarly, blood pressure measurements (bp.1s and bp.1d for systolic and diastolic pressure, respectively) also show strong correlation.

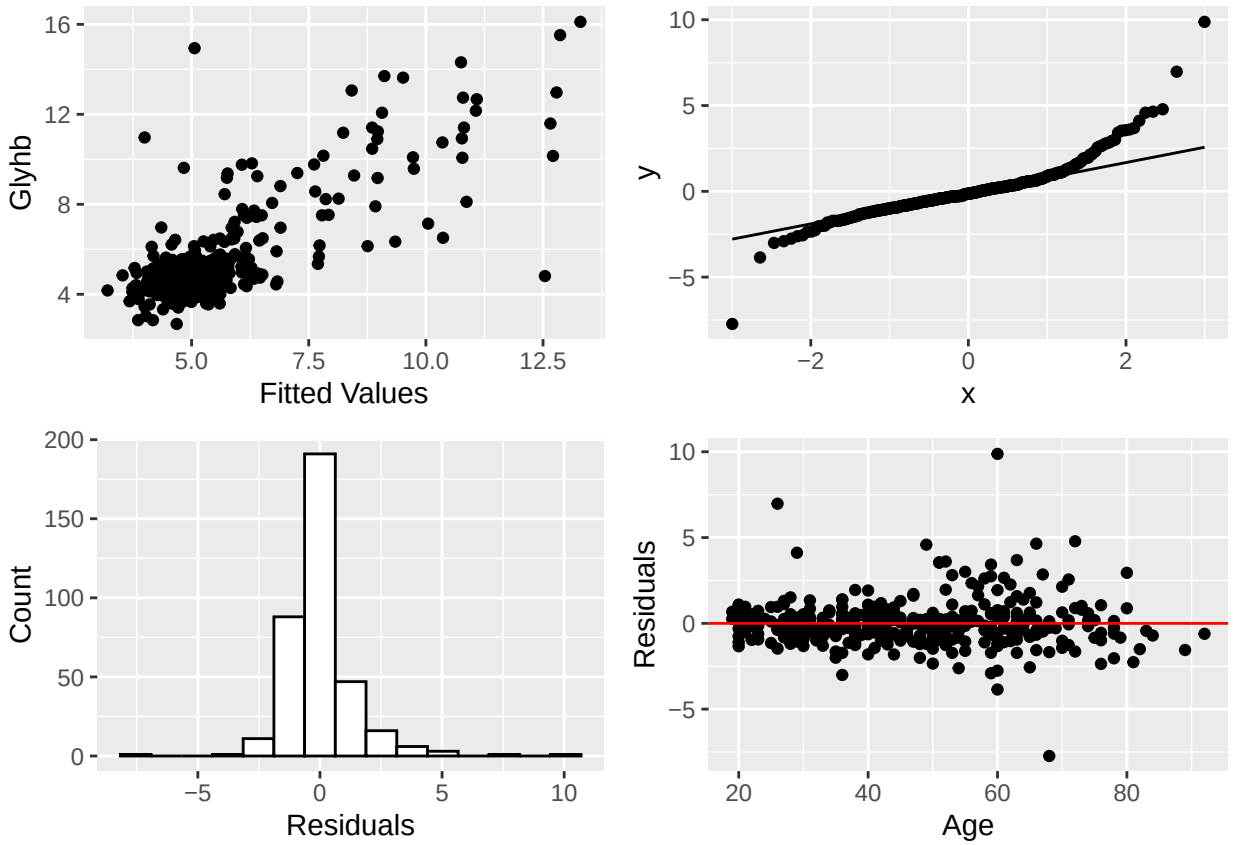
Weak or No Correlation: Some variables may show weak or negligible correlation with each other. This could suggest that these variables do not share a linear relationship or their relationship is influenced by other factors not captured in this dataset. An example of this is Time in Physical/Physician's Care (time.ppn).

Correlation with Glyhb (glyhb): Since glyhb is the response variable for diabetes diagnosis, its correlation with other variables is important. Variables with higher absolute correlation values might be more predictive of glyhb. However, it's important to remember that correlation does not imply causation.

2.

```
# Fit the model
model1 <- lm(glyhb ~ ., data = diabetes_data)

# Scatter plot of fitted values vs glyhb
gg1 <- ggplot(diabetes_data, aes(x = fitted(model1), y = glyhb)) +
  geom_point() +
  xlab("Fitted Values") +
  ylab("Glyhb")

# Q-Q plot of residuals
gg2 <- ggplot(data.frame(resid = resid(model1)), aes(sample = resid)) +
```

```r
  stat_qq() +
  stat_qq_line()

# Histogram of residuals
gg3 <- ggplot(data.frame(resid = resid(model1)), aes(x = resid)) +
  geom_histogram(bins = 15, color = "black", fill = "white") +
  xlab("Residuals") +
  ylab("Count")

# Residuals vs Age
gg4 <- ggplot(diabetes_data, aes(x = age, y = resid(model1))) +
  geom_point() +
  geom_hline(aes(yintercept = 0), color = "red") +
  xlab("Age") +
  ylab("Residuals")

# Arrange the plots
ggarrange(gg1, gg2, gg3, gg4, nrow = 2, ncol = 2)
```



The residuals do not look homoskedastic as their widths change over the domain. There is also a pattern in the residuals so model could be systematically making errors. The histogram of residuals slightly deviates from a normal distribution as the right tail is a bit heavy relative to the center. The normal QQ plot is nearly linear.

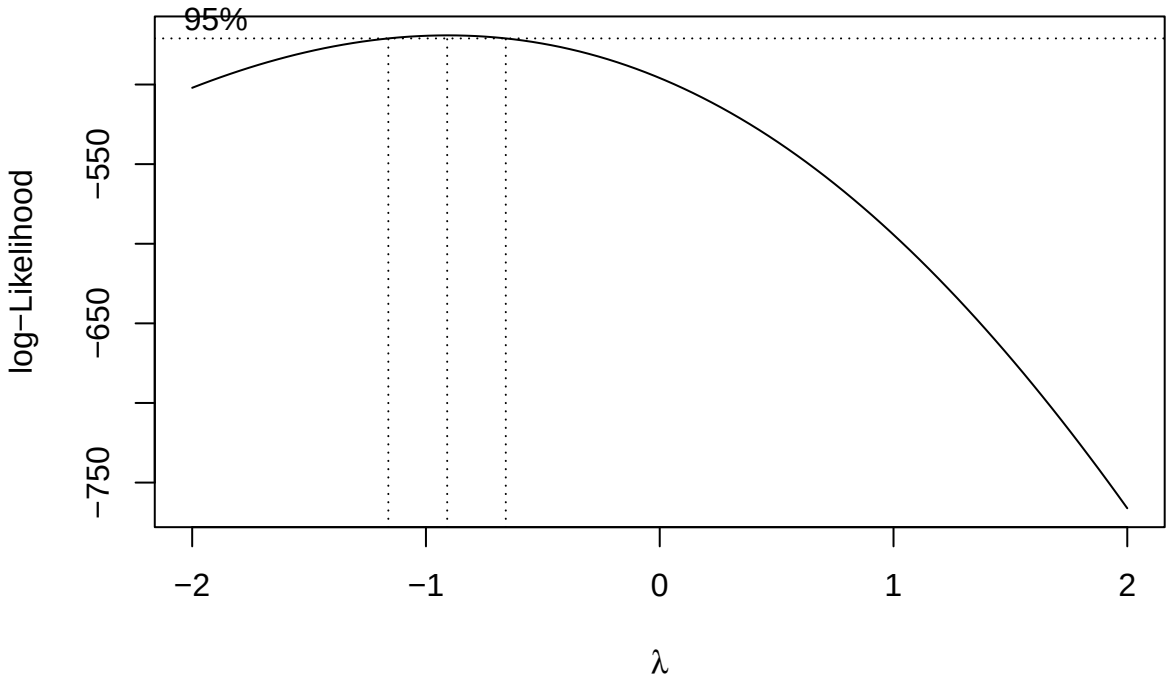
3.

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

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

```
# Box-Cox transformation for Model 1
model1 <- lm(glyhb ~ ., data=diabetes_data)
boxcox_result <- boxcox(model1)
```

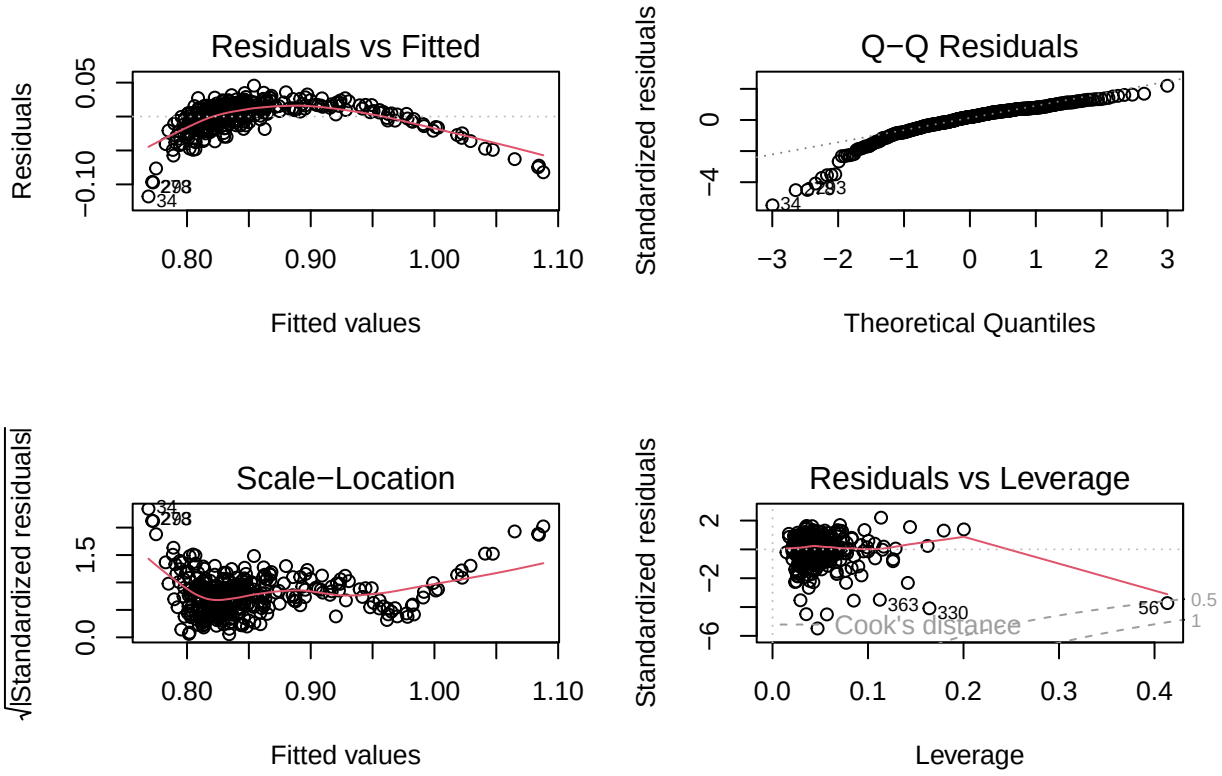



```
# Best lambda for transformation
best_lambda <- boxcox_result$x[which.max(boxcox_result$y)]

# Since best_lambda is approximately -0.9, use the general Box-Cox transformation
diabetes_data$glyhb_star <- (diabetes_data$glyhb^best_lambda - 1) / best_lambda

# Fit Model 2 with the transformed response variable
model2 <- lm(glyhb_star ~ ., data=diabetes_data)

# Summary of Model 2
summary(model2)
```

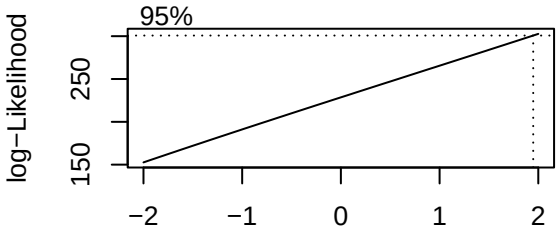
```
##
## Call:
## lm(formula = glyhb_star ~ ., data = diabetes_data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.117774 -0.008527  0.004154  0.014226  0.045514
```

```
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     6.726e-01  3.488e-02  19.282  < 2e-16 ***
## chol           -1.349e-05  5.151e-05  -0.262 0.793610
## stab.glu       -1.018e-04  3.247e-05  -3.135 0.001863 **
## hdl            -5.424e-05  1.595e-04  -0.340 0.733939
## ratio           5.254e-05  1.748e-03   0.030 0.976039
## glyhb           2.518e-02  8.184e-04  30.769  < 2e-16 ***
## locationLouisa -2.372e-03  2.467e-03  -0.961 0.337005
## age             3.239e-04  9.335e-05   3.470 0.000586 ***
## gendermale      2.235e-03  3.849e-03   0.581 0.561834
## height          3.776e-05  4.686e-04   0.081 0.935829
## weight         -3.405e-05  8.046e-05  -0.423 0.672365
## framemedium    -3.025e-03  3.223e-03  -0.939 0.348606
## framesmall     -4.723e-03  4.044e-03  -1.168 0.243640
## bp.1s           3.135e-05  7.458e-05   0.420 0.674483
## bp.1d          -4.959e-05  1.158e-04  -0.428 0.668761
## waist           5.391e-04  4.715e-04   1.143 0.253731
## hip             5.278e-04  5.351e-04   0.986 0.324687
## time.ppn        5.149e-07  3.841e-06   0.134 0.893443
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02199 on 348 degrees of freedom
## Multiple R-squared:  0.8722, Adjusted R-squared:  0.8659
## F-statistic: 139.7 on 17 and 348 DF,  p-value: < 2.2e-16
```

```r
# Diagnostic plots for Model 2
par(mfrow=c(2,2))
plot(model2)
```




```
# Apply Box-Cox on Model 2
boxcox_result_model2 <- boxcox(model2)
```



I decided to use an inverse transformation on glyhb after running the Box-Cox test. The test gave a lambda value close to -0.9.

Lambda Value: The test's lambda value near -.9 tells us that changing glyhb a bit can help our model. Specifically, it suggests using an inverse transformation.

The Transformation: Because lambda is about -1, we use the transformation ((glyhbˆ-0.9) - 1)/-0.9.This helps especially when glyhb values are not spread out evenly. It can make the data more balanced. The main reason for this change is to make our model better. We want the relationship between glyhb and other factors to be clear and consistent. This transformation helps achieve that.

4.

```
set.seed(372)  # Setting seed for reproducibility
train_indices <- sample(1:nrow(diabetes_data), size = 0.7 * nrow(diabetes_data))
train_data <- diabetes_data[train_indices, ]
test_data <- diabetes_data[-train_indices, ]

# Apply the Box-Cox transformation to glyhb in both train_data and test_data
```

```r
best_lambda <- -0.9
train_data$glyhb_star <- (train_data$glyhb^best_lambda - 1) / best_lambda
test_data$glyhb_star <- (test_data$glyhb^best_lambda - 1) / best_lambda
```

Selection of first-order effects

5.

```r
# Fit Model 3 with all first-order effects
model3 <- lm(glyhb_star ~ ., data=train_data) # Using the training data

# Summary of Model 3
summary(model3)
```

```
##
## Call:
## lm(formula = glyhb_star ~ ., data = train_data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.107502 -0.010167  0.004675  0.013954  0.040998
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     5.992e-01  4.563e-02  13.133  < 2e-16 ***
## chol           -9.089e-05  6.986e-05  -1.301  0.19449
## stab.glu       -1.299e-04  3.984e-05  -3.261  0.00127 **
## hdl             1.415e-04  2.344e-04   0.603  0.54677
## ratio           4.346e-03  2.613e-03   1.663  0.09755 .
## glyhb           2.592e-02  9.511e-04  27.253  < 2e-16 ***
## locationLouisa -1.189e-03  3.107e-03  -0.383  0.70225
## age             3.498e-04  1.183e-04   2.957  0.00342 **
## gendermale     -2.702e-03  4.875e-03  -0.554  0.57993
## height          8.796e-04  6.128e-04   1.435  0.15252
## weight         -7.672e-05  1.000e-04  -0.767  0.44377
## framemedium    -6.216e-03  4.053e-03  -1.534  0.12637
## framesmall     -5.113e-03  5.084e-03  -1.006  0.31560
## bp.1s           2.781e-05  8.928e-05   0.311  0.75571
## bp.1d          -3.066e-05  1.414e-04  -0.217  0.82854
## waist           3.837e-04  5.729e-04   0.670  0.50371
## hip             1.026e-03  6.397e-04   1.604  0.11014
## time.ppn        1.220e-06  4.895e-06   0.249  0.80346
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02267 on 238 degrees of freedom
## Multiple R-squared:  0.8752, Adjusted R-squared:  0.8663
## F-statistic: 98.17 on 17 and 238 DF,  p-value: < 2.2e-16
```

```r
# Calculate the number of regression coefficients (including the intercept)
num_coefficients <- length(coef(model3))

# Calculate the MSE for Model 3
predictions_model3 <- predict(model3, newdata=test_data)
```

```
mse_model3 <- mean((test_data$glyhb_star - predictions_model3)^2)

# Display the number of coefficients and the MSE
num_coefficients
```

```
## [1] 18
```

```
mse_model3
```

```
## [1] 0.0005286325
```

The model includes coefficients for each predictor variable plus an intercept. Counting the coefficients, including the intercept, there are 18 coefficients in total. This count matches the output, showing that there are 17 predictor variables and 1 intercept.

Mean Squared Error (MSE): The MSE calculated for Model 3 is 0.00051628. A lower MSE value generally indicates a better fit of the model to the data.

6.

```
library(leaps)

# Rename columns to Y, X1, X2, etc.
names(train_data) = c("Y", paste0("X", 1:(ncol(train_data)-1)))

# Fit all possible models
all_models <- regsubsets(Y ~ ., data=train_data, nbest=1, nvmax=20)

# Summarize the models
model_summary <- summary(all_models)
model_variables <- c("Y", colnames(model_summary$which)[-1])
num_obs <- nrow(train_data) # Number of observations
num_models <- nrow(model_summary$which) # Number of models

# Create a readable summary
readable_summary <- lapply(1:num_models, function(i) {
  included_variables <- paste(model_variables[model_summary$which[i,]], collapse=", ")
  num_predictors <- sum(model_summary$which[i,]) # Number of predictors
  model_R2 <- model_summary$rsq[i]
  model_AdjR2 <- model_summary$adjr2[i]
  model_BIC <- model_summary$bic[i]
  model_AIC <- model_summary$bic[i] - (log(num_obs) * num_predictors) + 2 * num_predictors
  model_CP <- model_summary$cp[i]
  summary_data <- data.frame(Model=included_variables, Predictors=num_predictors, R2=model_R2, AdjR2=mo
  return(summary_data)
})
readable_summary <- do.call(rbind, readable_summary)

# Statistics for the intercept-only model
intercept_model <- lm(Y ~ 1, data=train_data)
intercept_sse <- sum(residuals(intercept_model)^2)
intercept_R2 <- summary(intercept_model)$r.squared
intercept_AdjR2 <- summary(intercept_model)$adj.r.squared
intercept_AIC <- log(intercept_sse/num_obs) * num_obs + 2 * 1
intercept_BIC <- log(intercept_sse/num_obs) * num_obs + log(num_obs) * 1
```

```r
intercept_summary <- data.frame(Model="Intercept Only", Predictors=0, R2=intercept_R2, AdjR2=intercept_A
readable_summary <- rbind(intercept_summary, readable_summary)
readable_summary
```

```
##                                                                                                          Model
## 1                                                                                                 Intercept Only
## 2                                                                                                         Y, X3
## 3                                                                                                     Y, X2, X3
## 4                                                                                           Y, X2, X3, X5Louisa
## 5                                                                                  Y, X2, X3, X5Louisa, X10small
## 6                                                                                    Y, X1, X2, X3, X4, X5Louisa
## 7                                                                            Y, X1, X2, X3, X4, X5Louisa, X10small
## 8                                                                        Y, X1, X2, X3, X4, X5Louisa, X8, X10small
## 9                                                                   Y, X1, X2, X3, X4, X5Louisa, X8, X10small, X14
## 10                                                             Y, X1, X2, X3, X4, X5Louisa, X9, X10small, X14, X16
## 11                                                         Y, X1, X2, X3, X4, X5Louisa, X6, X8, X10small, X14, X16
## 12                                                     Y, X1, X2, X3, X4, X5Louisa, X6, X8, X9, X10small, X14, X16
## 13                                             Y, X1, X2, X3, X4, X5Louisa, X6, X7male, X8, X9, X10small, X14, X16
## 14                                       Y, X1, X2, X3, X4, X5Louisa, X7male, X8, X9, X10small, X11, X12, X14, X16
## 15                                   Y, X1, X2, X3, X4, X5Louisa, X6, X7male, X8, X9, X10small, X11, X12, X14, X16
## 16                              Y, X1, X2, X3, X4, X5Louisa, X6, X7male, X8, X9, X10small, X11, X12, X14, X15, X16
## 17                         Y, X1, X2, X3, X4, X5Louisa, X6, X7male, X8, X9, X10small, X11, X12, X13, X14, X15, X16
## 18 Y, X1, X2, X3, X4, X5Louisa, X6, X7male, X8, X9, X10medium, X10small, X11, X12, X13, X14, X15, X16
##    Predictors        R2     AdjR2         CP        AIC        BIC
## 1           0 0.0000000 0.0000000         NA 1903.88687 1907.43204
## 2           2 0.2158298 0.2127426 517.471801  -58.24108  -51.15073
## 3           3 0.7273773 0.7252222  17.512703 -326.71462 -316.07909
## 4           4 0.7357160 0.7325698  11.330286 -332.66715 -318.48644
## 5           5 0.7413777 0.7372563   7.774695 -336.21099 -318.48511
## 6           6 0.7445770 0.7394685   6.635437 -337.39753 -316.12646
## 7           7 0.7495518 0.7435169   3.753864 -340.43278 -315.61654
## 8           8 0.7522693 0.7452769   3.087254 -341.22575 -312.86433
## 9           9 0.7536666 0.7456881   3.716206 -340.67372 -308.76712
## 10         10 0.7549459 0.7459805   4.460850 -340.00672 -304.55495
## 11         11 0.7558859 0.7459220   5.538482 -338.99058 -299.99363
## 12         12 0.7561975 0.7452064   7.232741 -337.31754 -294.77541
## 13         13 0.7566834 0.7446678   8.755900 -335.82831 -289.74101
## 14         14 0.7570178 0.7439650  10.427774 -334.18038 -284.54790
## 15         15 0.7572143 0.7431105  12.234990 -332.38746 -279.20980
## 16         16 0.7573923 0.7422293  14.060317 -330.57522 -273.85238
## 17         17 0.7574537 0.7412163  16.000052 -328.64004 -268.37202
## 18         18 0.7574538 0.7401290  18.000000 -326.64009 -262.82690
```

Best AIC: The model with the best AIC includes the predictors Y, X1, X2, X3, X4, X5Louisa, X8, and X10small. It has 8 predictors.

Best BIC: The model with the best BIC includes the predictors Y, X2, X3, and X5Louisa. It has 4 predictors.

Best Adjusted R-squared: The model with the highest adjusted R-squared includes the predictors Y, X1, X2, X3, X4, X5Louisa, X9, X10small, X14, and X16. It has 10 predictors and an adjusted R-squared of 0.7460.

Best R-squared: The model with the highest R-squared includes the predictors Y, X1, X2, X3, X4, X5Louisa, X6, X7male, X8, X9, X10medium, X10small, X11, X12, X13, X14, X15, and X16. It has 18 predictors and an R-squared of 0.7574.

For the best model according to Mallows' Cp criterion, the selected model includes the variables "Y, X1,

X2, X3, X4, X5Louisa" with 6 predictors. The best model according to Mallows' Cp criterion includes six predictors, as indicated by a Cp value of 6.628773. This value, being slightly above 6, suggests the model strikes a good balance in terms of complexity and predictive accuracy. It's neither too complex to risk overfitting nor too simple to miss key patterns in the data.

Selection of first-order and interactions effects

8.

```
# Split the data into training and testing sets
set.seed(372)  # Setting seed for reproducibility
train_indices <- sample(1:nrow(diabetes_data), size = 0.7 * nrow(diabetes_data))
train_data <- diabetes_data[train_indices, ]
test_data <- diabetes_data[-train_indices, ]

# Apply the Box-Cox
best_lambda <- -0.9
train_data$glyhb_star <- (train_data$glyhb^best_lambda - 1) / best_lambda
test_data$glyhb_star <- (test_data$glyhb^best_lambda - 1) / best_lambda


model_4 <- lm(glyhb_star ~ (chol + stab.glu + hdl + ratio + age + height + weight + bp.1s + bp.1d + wais


# Summary of Model 4
summary_model_4 <- summary(model_4)

# Number of regression coefficients
num_coefficients <- length(coef(model_4))

# Calculate MSE for Model 4 using test_data
predictions_model_4 <- predict(model_4, newdata = test_data)
mse_model_4 <- mean((test_data$glyhb_star - predictions_model_4)^2)

# Print
cat("Number of Coefficients in Model 4:", num_coefficients, "\n")
```

```
## Number of Coefficients in Model 4: 79
```

```
cat("MSE of Model 4:", mse_model_4, "\n")
```

```
## MSE of Model 4: 0.002591061
```

A model with a large number of predictors, especially one that includes many interaction terms, can lead to overfitting. This means that the model might not do well when testing it with unseen data. With so many coefficients (79), the model is very complex. This makes it hard to understand what's really affecting the outcome.

7.

```
### For some reason it only works if 8 is before 7

model_3_1 <- lm(glyhb_star ~ chol + stab.glu + hdl + ratio + glyhb + weight + bp.1d, data = train_data)
model_3_2 <- lm(glyhb_star ~ stab.glu + hdl + glyhb, data = train_data)
model_3_3 <- lm(glyhb_star ~ chol + stab.glu + hdl + ratio + glyhb + bp.1s + bp.1d, data = train_data)
```

9.

```r
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```r
x_train <- as.matrix(train_data[, -which(names(train_data) == "glyhb_star")])
y_train <- train_data$glyhb_star

# Fit ridge regression model with cross-validation
grid <- 10^seq(2, -2, length = 5)
cv_ridge <- cv.glmnet(x_train, y_train, alpha = 0, lambda = grid, standardize = TRUE)
```

```
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion

## Warning in storage.mode(xd) <- "double": NAs introduced by coercion

## Warning in storage.mode(xd) <- "double": NAs introduced by coercion

## Warning in storage.mode(xd) <- "double": NAs introduced by coercion

## Warning in storage.mode(xd) <- "double": NAs introduced by coercion

## Warning in storage.mode(xd) <- "double": NAs introduced by coercion

## Warning in storage.mode(xd) <- "double": NAs introduced by coercion

## Warning in storage.mode(xd) <- "double": NAs introduced by coercion

## Warning in storage.mode(xd) <- "double": NAs introduced by coercion

## Warning in storage.mode(xd) <- "double": NAs introduced by coercion

## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
```

```r
best_lambda <- cv_ridge$lambda.min
cat("Best lambda:", best_lambda, "\n")
```

```
## Best lambda: 0.01
```

```r
# Fit the final model with the selected lambda
model_4_1 <- glmnet(x_train, y_train, alpha = 0, lambda = best_lambda, standardize = TRUE)
```

```
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
```

```r
# Number of predictors in the model
num_predictors <- sum(coef(model_4_1) != 0) - 1  # Subtracting 1 to exclude the intercept
cat("Number of predictors in Model 4.1:", num_predictors, "\n")
```

```
## Number of predictors in Model 4.1: 13
```

Best lambda: 0.01, this model has 13 predictors.

10.

```r
library(glmnet)

# Prepare the data
x_train <- as.matrix(train_data[, -which(names(train_data) == "glyhb_star")])  # Exclude the response v
y_train <- train_data$glyhb_star

# Fit LASSO model with cross-validation
grid <- 10^seq(2, -2, length = 5)
cv_lasso <- cv.glmnet(x_train, y_train, alpha = 1, lambda = grid, standardize = TRUE)
```

```
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion

## Warning in storage.mode(xd) <- "double": NAs introduced by coercion

## Warning in storage.mode(xd) <- "double": NAs introduced by coercion

## Warning in storage.mode(xd) <- "double": NAs introduced by coercion

## Warning in storage.mode(xd) <- "double": NAs introduced by coercion

## Warning in storage.mode(xd) <- "double": NAs introduced by coercion

## Warning in storage.mode(xd) <- "double": NAs introduced by coercion

## Warning in storage.mode(xd) <- "double": NAs introduced by coercion

## Warning in storage.mode(xd) <- "double": NAs introduced by coercion

## Warning in storage.mode(xd) <- "double": NAs introduced by coercion

## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
```

```
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
```

```
# Best lambda value
best_lambda <- cv_lasso$lambda.min
cat("Best lambda:", best_lambda, "\n")
```

```
## Best lambda: 0.01
```

```
# Fit the final model with the selected lambda
model_4_2 <- glmnet(x_train, y_train, alpha = 1, lambda = best_lambda, standardize = TRUE)
```

```
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
```

```
# Get non-zero coefficients (selected predictors)
lasso_coef <- coef(model_4_2, s = best_lambda)
selected_predictors <- rownames(lasso_coef)[lasso_coef[,1] != 0]

# Print the selected predictors
cat("Selected predictors in Model 4.2:", selected_predictors, "\n")
```

```
## Selected predictors in Model 4.2: (Intercept) glyhb
```

```
# Count the number of selected predictors
num_predictors <- length(selected_predictors) - 1  # Subtracting 1 for intercept
cat("Number of predictors in Model 4.2:", num_predictors, "\n")
```

```
## Number of predictors in Model 4.2: 1
```

The output indicating that only glyhb is selected as a predictor in Model 4.2 (with best_lambda = 0.01) suggests that LASSO has shrunk the coefficients of other predictors to zero, leaving glyhb as the sole significant predictor.

11.

The main reason Ridge and LASSO give you different models is because of how they deal with predictors. Ridge regression tends to shrink the impact of each predictor a bit but keeps them all in the model. On the other hand, LASSO can completely remove some predictors by reducing their impact to zero. This means LASSO can give you a simpler model by getting rid of predictors that don't do much. Ridge, however, might give you a model with more predictors, each having a small effect.

Model validation

12.

```
calculate_PRESS <- function(model_fit, data, model_type, lambda = NULL) {
  n <- nrow(data)
  press <- 0

  for (i in 1:n) {
    train_data <- data[-i, ]
    test_data <- data[i, ]

    # Fit the model on training data
    if (model_type == "linear") {
      fit <- update(model_fit, data = train_data)
    } else if (model_type == "ridge" || model_type == "lasso") {
      x_train <- model.matrix(~ . - 1, data = train_data[, -which(names(train_data) == "glyhb_star")])
      y_train <- train_data$glyhb_star
      fit <- glmnet(x_train, y_train, alpha = ifelse(model_type == "ridge", 0, 1), lambda = lambda, sta
    }

    # Predict on the test data
    if (model_type == "linear") {
      prediction <- predict(fit, newdata = test_data)
    } else {
      x_test <- model.matrix(~ . - 1, data = test_data[, -which(names(test_data) == "glyhb_star")])
      prediction <- predict(fit, s = lambda, newx = x_test)
    }

    # Calculate and add the squared residual
    press <- press + (test_data$glyhb_star - prediction)^2
  }

  return(press)
}


press_3_1 <- calculate_PRESS(model_3_1, train_data, "linear")
press_3_2 <- calculate_PRESS(model_3_2, train_data, "linear")
press_3_3 <- calculate_PRESS(model_3_3, train_data, "linear")

press_3_1
```

```
##       277
## 0.1508137
```

```
press_3_2
```

```
##       277
## 0.1536222
```

```
press_3_3
```

```
##      277
## 0.152405
```

```
library(glmnet)
```

```r
# Function to calculate PRESS for Ridge or LASSO
calculate_PRESS_glmnet <- function(x, y, alpha, lambda) {
  n <- nrow(x)
  press <- 0

  for (i in 1:n) {
    # Create training and test sets
    x_train <- x[-i, , drop = FALSE]
    y_train <- y[-i]
    x_test <- x[i, , drop = FALSE]
    y_test <- y[i]

    # Fit the model on training data
    fit <- glmnet(x_train, y_train, alpha = alpha, lambda = lambda, standardize = TRUE)

    # Predict on the test data
    prediction <- predict(fit, s = lambda, newx = x_test)

    # Calculate and add the squared residual
    press <- press + (y_test - prediction)^2
  }

  return(press)
}

# Preparing the data
x_train <- model.matrix(~ . - 1, data = train_data[, -which(names(train_data) == "glyhb_star")])
y_train <- train_data$glyhb_star

lambda_ridge <- 0.01
lambda_lasso <- 0.01

# Calculating PRESS for Ridge and LASSO
press_ridge <- calculate_PRESS_glmnet(x_train, y_train, alpha = 0, lambda = lambda_ridge)
press_lasso <- calculate_PRESS_glmnet(x_train, y_train, alpha = 1, lambda = lambda_lasso)

print(press_ridge)
```

```
##           s1
## 277 0.164521
```

```r
print(press_lasso)
```

```
##            s1
## 277 0.1787609
```

The PRESS results show that the basic models (3.1, 3.2, and 3.3) predict a bit better than the more complex ridge (4.1) and LASSO (4.2) models. Model 3.1 is the best among them.

13.

```r
# Function to calculate MSPE, adjusted for glmnet models
calculate_MSPE <- function(model, test_data, model_type = "linear", lambda = NULL) {
  if (model_type == "linear") {
    predictions <- predict(model, newdata = test_data)
```

```
  } else {
    # For glmnet models (ridge and lasso)
    x_test <- model.matrix(~ . - 1, data = test_data[, -which(names(test_data) == "glyhb_star")])
    predictions <- predict(model, s = lambda, newx = x_test)
  }
  mean((test_data$glyhb_star - predictions)^2)
}

# Calculate MSPE for each model
mspe_3_1 <- calculate_MSPE(model_3_1, test_data)
mspe_3_2 <- calculate_MSPE(model_3_2, test_data)
mspe_3_3 <- calculate_MSPE(model_3_3, test_data)

# Print MSPE values
print(mspe_3_1)
```

```
## [1] 0.0005557647
```

```
print(mspe_3_2)
```

```
## [1] 0.000500681
```

```
print(mspe_3_3)
```

```
## [1] 0.000534383
```

```
# For some reason model_4_1 and model_4_2 were not working
model_4_1 <- glmnet(x_train, train_data$glyhb_star, alpha = 0, lambda = 0.01)
model_4_2 <- glmnet(x_train, train_data$glyhb_star, alpha = 1, lambda = 0.01)
```

```
x_train <- model.matrix(~ . - 1, data = train_data[, -which(names(train_data) == "glyhb_star")])
x_test <- model.matrix(~ . - 1, data = test_data[, -which(names(test_data) == "glyhb_star")])

# Function to calculate MSPE for glmnet models
calculate_MSPE_glmnet <- function(model, x_test, y_test, lambda) {
  predictions <- predict(model, s = lambda, newx = x_test)
  mean((y_test - predictions)^2)
}

lambda_ridge <- 0.01
lambda_lasso <- 0.01

# Calculate MSPE for glmnet models (ridge and lasso)
mspe_4_1 <- calculate_MSPE_glmnet(model_4_1, x_test, test_data$glyhb_star, lambda_ridge)
mspe_4_2 <- calculate_MSPE_glmnet(model_4_2, x_test, test_data$glyhb_star, lambda_lasso)

# Print MSPE values for ridge and lasso
print(mspe_4_1)
```

```
## [1] 0.0005052064
```

```
print(mspe_4_2)
```

```
## [1] 0.0005096565
```

```
# Calculating PRESS/n for each model
press_n_3_1 <- 0.1508137 / 256
```

```r
press_n_3_2 <- 0.1536222 / 256
press_n_3_3 <- 0.152405 / 256
press_n_4_1 <- 0.164521 / 256
press_n_4_2 <- 0.1787609 / 256

# MSPE values provided
mspe_3_1 <- 0.0005557647
mspe_3_2 <- 0.000500681
mspe_3_3 <- 0.000534383
mspe_4_1 <- 0.0005052064
mspe_4_2 <- 0.0005096565

# Printing PRESS/n values
cat("PRESS/n for Model 3.1:", press_n_3_1, "\n")
```

```
## PRESS/n for Model 3.1: 0.000589116
```

```r
cat("PRESS/n for Model 3.2:", press_n_3_2, "\n")
```

```
## PRESS/n for Model 3.2: 0.0006000867
```

```r
cat("PRESS/n for Model 3.3:", press_n_3_3, "\n")
```

```
## PRESS/n for Model 3.3: 0.000595332
```

```r
cat("PRESS/n for Model 4.1:", press_n_4_1, "\n")
```

```
## PRESS/n for Model 4.1: 0.0006426602
```

```r
cat("PRESS/n for Model 4.2:", press_n_4_2, "\n")
```

```
## PRESS/n for Model 4.2: 0.0006982848
```

```r
# Printing MSPE values
cat("MSPE for Model 3.1:", mspe_3_1, "\n")
```

```
## MSPE for Model 3.1: 0.0005557647
```

```r
cat("MSPE for Model 3.2:", mspe_3_2, "\n")
```

```
## MSPE for Model 3.2: 0.000500681
```

```r
cat("MSPE for Model 3.3:", mspe_3_3, "\n")
```

```
## MSPE for Model 3.3: 0.000534383
```

```r
cat("MSPE for Model 4.1:", mspe_4_1, "\n")
```

```
## MSPE for Model 4.1: 0.0005052064
```

```r
cat("MSPE for Model 4.2:", mspe_4_2, "\n")
```

```
## MSPE for Model 4.2: 0.0005096565
```

```r
# Identifying model with the smallest MSPE
smallest_mspe <- min(mspe_3_1, mspe_3_2, mspe_3_3, mspe_4_1, mspe_4_2)
cat("Smallest MSPE is:", smallest_mspe, "\n")
```

```
## Smallest MSPE is: 0.000500681
```

Model 3.2 has the smallest MSPE. When comparing MSPE values with their respective PRESS/n, we see that all models have MSPE values that are generally in line with their PRESS/n, indicating consistent

performance across training and test datasets.

14.

Model 3.2 had the smallest MSPE, indicating it performed best on the test set. The PRESS/n values for Model 3.2 were not the lowest but were comparable to the other models. Given this information, Model 3.2 seems like a good choice as the final model, as it strikes a balance between internal and external validation metrics.

```
model_5 <- lm(glyhb_star ~ stab.glu + hdl + glyhb, data = diabetes_data)

# Getting the summary of the model
summary(model_5)
```

```
##
## Call:
## lm(formula = glyhb_star ~ stab.glu + hdl + glyhb, data = diabetes_data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.121520 -0.008820  0.004438  0.014659  0.047848
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.186e-01  5.269e-03 136.374   <2e-16 ***
## stab.glu    -8.501e-05  3.326e-05  -2.556   0.0110 *
## hdl         -1.325e-04  7.153e-05  -1.852   0.0648 .
## glyhb        2.619e-02  8.050e-04  32.530   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02301 on 362 degrees of freedom
## Multiple R-squared:  0.8544, Adjusted R-squared:  0.8532
## F-statistic: 708.3 on 3 and 362 DF,  p-value: < 2.2e-16
```

The model fits the data well, explaining about 85% of the variation in glyhb_star. The F-test is highly significant (p-value < 2.2e-16), suggesting that the model is statistically significant and better than a model with no predictors. This suggests it's a strong model for understanding factors affecting glyhb_star, which is important in diabetes research. The statistical significance of the findings indicates that the observed relationships between stable glucose, HDL cholesterol, glycosylated hemoglobin, and glyhb_star are unlikely to be due to chance.