Q1.

```
> no_observations <- nrow(insurance)
> cat("Total Observations:", no_observations)
Total Observations: 2954
```

The total observations in this insurance dataset is 2954.

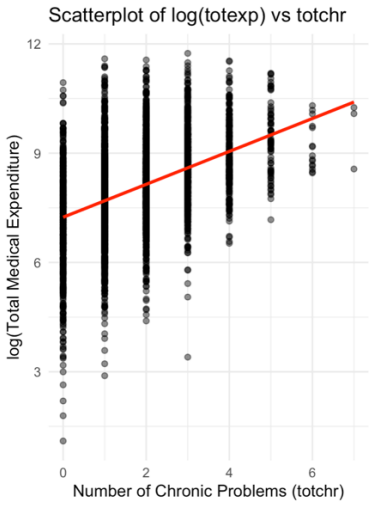
Q2.

```
> numeric_variables <- insurance [,sapply (insurance, is.numeric)]
> correlations <- cor(numeric_variables, use="complete.obs")["totexp", ]
> sorted_correlations <- sort(abs(correlations), decreasing = TRUE)
> sorted_correlations <- sorted_correlations[names(sorted_correlations) !
= "totexp"]
> top_correlations <- head(sorted_correlations, 5)
> print(top_correlations)
   totchr    actlim    phylim  priolist      hvgg
0.2736266 0.2520608 0.2332661 0.1369268 0.1287198
```

The most correlated with the total medical expenditure (totexp) in this dataset are totchr (0.2737) , actlim (0.2521), phylim (0.233266), priolist(0.1369268) and hvgg(0.1287). These correlations are showing that these variables have the strongest correlations with the total medical expenditure (totexp).

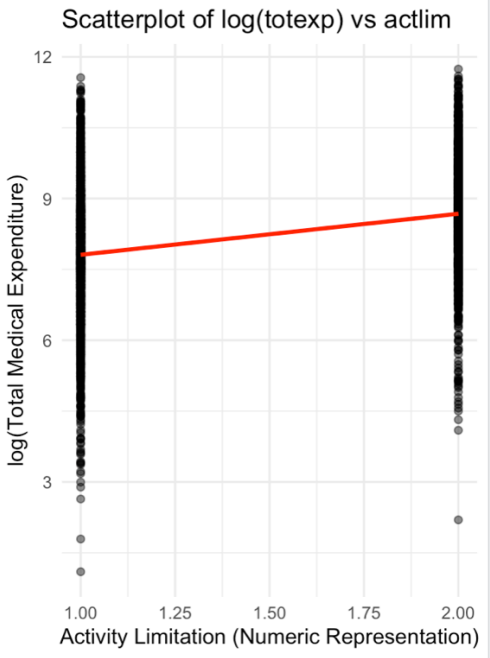
Q3.

```
> #adding a column for log(totexp)
> insurance$log_totexp <- log(insurance$totexp)
> #converting the indicator variables
> insurance$actlim <- as.factor(insurance$actlim)
> insurance$phylim <- as.factor(insurance$phylim)
> insurance$priolist <- as.factor(insurance$priolist)
> insurance$hvgg <- as.factor(insurance$hvgg)
> ggplot(insurance, aes(x = totchr, y = log_totexp)) + geom_point(alpha = 0.5)
+ geom_smooth(method = "lm", col = "red", se = FALSE) + labs( title = "Scatterp
lot of log(totexp) vs totchr", x = "Number of Chronic Problems (totchr)", y =
"log(Total Medical Expenditure)") + theme_minimal()
`geom_smooth()` using formula = 'y ~ x'
```



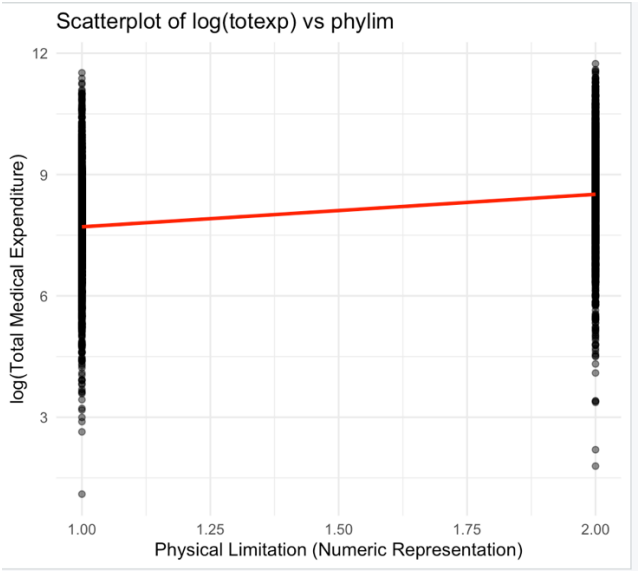
There is a positive correlation between total medical expenditure (log(totexp)) and the number of chronic problems. Most points are concentrated around totchr values of 0 to 3, indicating that the majority of individuals have few chronic problems. As totchr increases, log(totexp) becomes more variable, suggesting that individuals with more chronic problems are likely to incur higher medical costs.

```
> #scatterplot for actlim vs log(totexp)
> ggplot(insurance, aes(x = as.numeric(actlim), y = log_totexp)) + geom_point(a
lpha = 0.5) + geom_smooth(method = "lm", col = "red", se = FALSE) + labs(title
= "Scatterplot of log(totexp) vs actlim", x = "Activity Limitation (Numeric Rep
resentation)", y = "log(Total Medical Expenditure)") + theme_minimal()
`geom_smooth()` using formula = 'y ~ x'
```



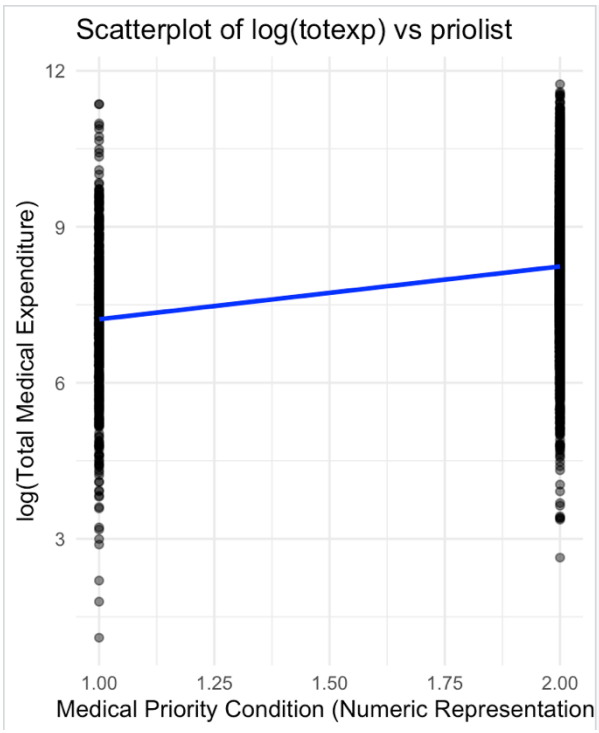
There is a slight positive correlation between log(totexp) and activity limitation (acclaim). Most individuals fall into one of these two groups which are no limitations or limitation. Those individuals with activity limitations tend to have higher medical costs and greater variability in expenditure.

```
> #scatterplot for phylim vs log(totexp)
>
> ggplot(insurance, aes(x = as.numeric(phylim), y = log_totexp)) + geom_point(a
lpha = 0.5) + geom_smooth(method = "lm", col = "red", se = FALSE) + labs(
+ title = "Scatterplot of log(totexp) vs phylim", x = "Physical Limitation (Num
eric Representation)", y = "log(Total Medical Expenditure)") + theme_minimal()
`geom_smooth()` using formula = 'y ~ x'
```



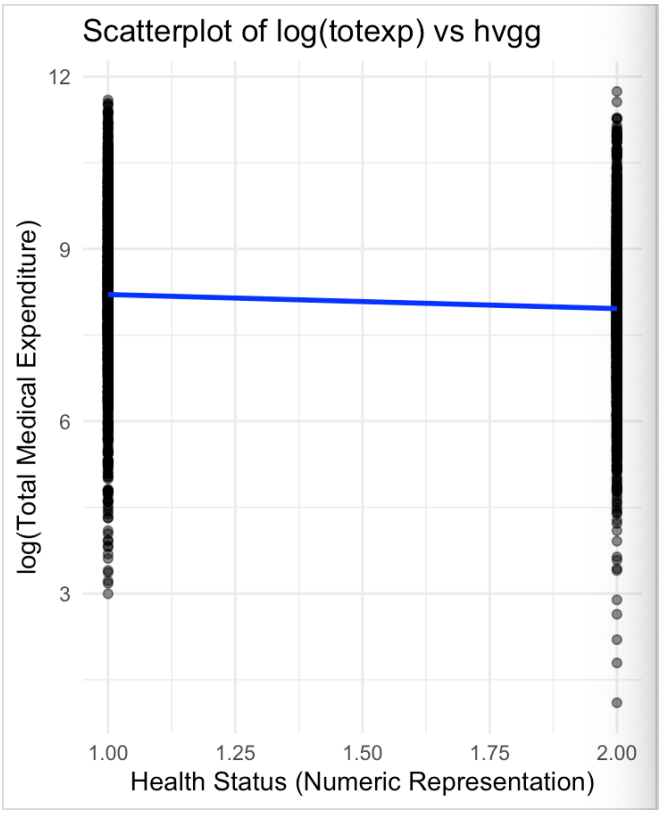
There exists a slight positive relationship between log(totexp) and physical limitation (phylim). The majority of people fall into one of two categories: those without physical limitations and those with them. Individuals experiencing physical limitations generally incur higher medical expenses and exhibit greater variability in their spending.

```
> ggplot(insurance, aes(x = as.numeric(priolist), y = log_totexp))
+ geom_point(alpha = 0.5) + geom_smooth(method = "lm", col = "blu
e", se = FALSE) + labs(title = "Scatterplot of log(totexp) vs priol
ist", x = "Medical Priority Condition (Numeric Representation)", y
= "log(Total Medical Expenditure)") + theme_minimal()
`geom_smooth()` using formula = 'y ~ x'
```



There is a slightly positive relationship between medical priority condition (priolist) and log(totexp). The majority of people are classified as either having a priority condition or having no priority condition. Medical expenses are typically higher for those with priority conditions.

```
> #scatterplot for hvgg vs log(totexp)
> ggplot(insurance, aes(x = as.numeric(hvgg), y = log_totexp)) + geom_point
(alpha = 0.5) + geom_smooth(method = "lm", col = "blue", se = FALSE) +
+ labs(title = "Scatterplot of log(totexp) vs hvgg", x = "Health Status (Nu
meric Representation)", y = "log(Total Medical Expenditure)") + theme_minim
al()
`geom_smooth()` using formula = 'y ~ x'
```



The correlation between health status (hvgg) and log(totexp) is slightly negative. The majority of people are either in excellent health or in good,very good health. Better health tends to result in somewhat lower medical costs.

Q4.

```
>
> #the linear regression model
> linear_model <- lm(log_totexp ~ suppins + phylim + actlim + totchr + a
+ female + income + mwest + northe, data = insurance)
>
>
>
> summary(linear_model)

Call:
lm(formula = log_totexp ~ suppins + phylim + actlim + totchr +
    age + female + income + mwest + northe, data = insurance)

Residuals:
    Min      1Q  Median      3Q     Max
-6.2746 -0.7250 -0.0038  0.7610  3.7268

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.699183   0.276263  24.249  < 2e-16 ***
suppins      0.243802   0.046172   5.280 1.38e-07 ***
phylim1      0.301063   0.056881   5.293 1.29e-07 ***
actlim1      0.364911   0.062090   5.877 4.64e-09 ***
totchr       0.375507   0.018371  20.440  < 2e-16 ***
age          0.002979   0.003662   0.814 0.415910
female      -0.088799   0.045431  -1.955 0.050724 .
income       0.002789   0.001020   2.735 0.006277 **
mwest        0.188924   0.054118   3.491 0.000488 ***
northe       0.152714   0.063189   2.417 0.015718 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.199 on 2944 degrees of freedom
Multiple R-squared:  0.2324,    Adjusted R-squared:   0.23
F-statistic: 99.01 on 9 and 2944 DF,  p-value: < 2.2e-16
```

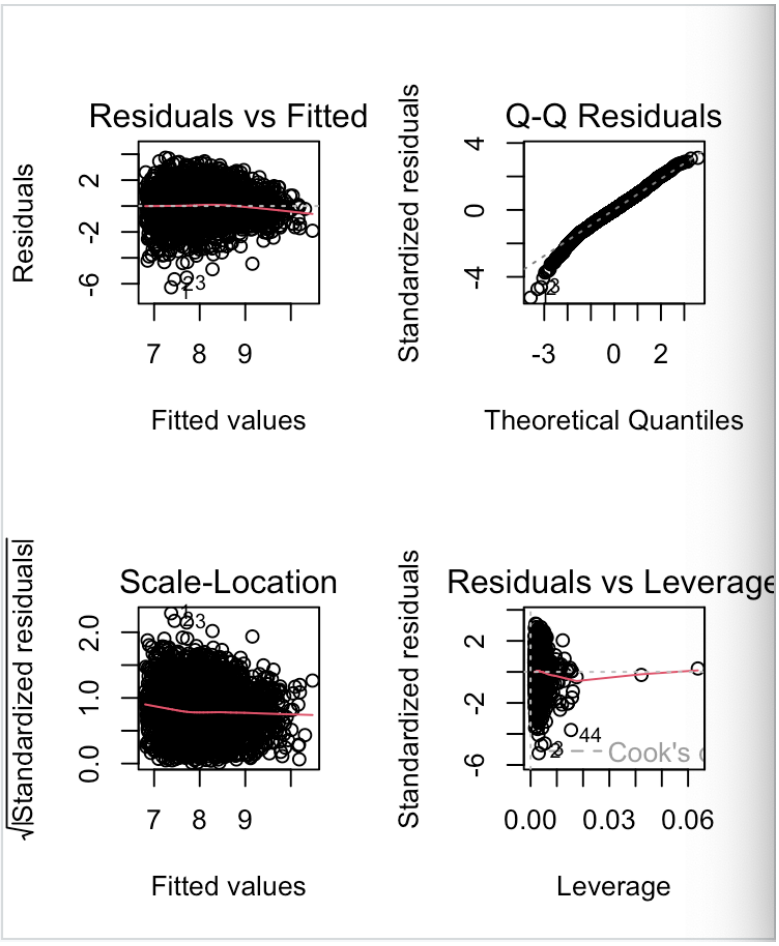
4.(a) The linear regression model is overall significant, as the F-statistic (99.01) has a p-value < 0.05, indicating that at least one independent variable is significantly associated with log(totexp).

4.(b) The Residual Standard Error (RSE) is 1.199. This means that the model's predictions of log(totexp) deviate from the actual values by approximately 1.199 units on average. This provides a measure of the typical prediction error for the model.

4.(c) Females have 0.0888 lower log-transformed medical expenditure compared to males, holding all else constant. For each unit increase in income, log-transformed medical expenditure increases by 0.0028, holding all else constant

4.(d)

```
> #plots for the linear model
>
> par(mfrow = c(2, 2))
> plot(linear_model)
>
```



The slight curve in the residuals indicates that there may not be a perfectly linear relationship between the predictors and log(totexp).

The majority of the points in the Q-Q plot are near the diagonal line, indicating that the residuals are largely normal with minor problems at the extremes.

The residuals are distributed fairly evenly, according to the Scale-Location plot, although there is a little more variation at higher fitted values, suggesting that the assumption of homoscedasticity holds reasonably well.

There are some points on the Residuals vs. Leverage plot that are close to or outside the Cook's distance lines. The model may be overly impacted by these points.

4.(e)

```
> leverage_values <- hatvalues(linear_model)
> p <- length(coef(linear_model))
> n <- nrow(insurance)
> average_leverage <- p / n
> high_leverage_threshold <- 3 * average_leverage
>
>
> num_high_leverage <- sum(leverage_values > high_leverage_threshold)
> print(num_high_leverage)
[1] 27
```

The leverage values of 27 observations are categorised as high-leverage points because they are greater than three times the average leverage. These issues need more research because they might have a disproportionate impact on the model.

4.(f)

```
> standardized_residuals <- rstandard(linear_model)
>
>
> outliers <- which(abs(standardized_residuals) > 3)
> insurance_no_outliers <- insurance[-outliers, ]
>
>
>
> linear_model_no_outliers <- lm(log_totexp ~ suppins + phylim + actlim + t
otchr + age + female + income + mwest + northe, data = insurance_no_outlier
s)
>
> summary(linear_model_no_outliers)

Call:
lm(formula = log_totexp ~ suppins + phylim + actlim + totchr +
    age + female + income + mwest + northe, data = insurance_no_outliers)

Residuals:
    Min      1Q  Median      3Q     Max
-3.6429 -0.7237 -0.0186  0.7471  3.4202

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.6817187  0.2660839  25.111  < 2e-16 ***
suppins      0.2389977  0.0444804   5.373 8.35e-08 ***
phylim1      0.3297742  0.0548113   6.017 2.00e-09 ***
actlim1      0.3604782  0.0598027   6.028 1.87e-09 ***
totchr       0.3560165  0.0177207  20.090  < 2e-16 ***
age          0.0038704  0.0035273   1.097 0.272616
female      -0.0904782  0.0437488  -2.068 0.038716 *
income       0.0028374  0.0009854   2.879 0.004012 **
mwest        0.1889854  0.0521316   3.625 0.000294 ***
northe       0.1507940  0.0608333   2.479 0.013238 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.151 on 2926 degrees of freedom
Multiple R-squared:  0.2387,    Adjusted R-squared:  0.2364
F-statistic: 101.9 on 9 and 2926 DF,  p-value: < 2.2e-16
```

The regression output indicates slight improvements after outliers are eliminated. From 0.2333 to 0.2364, the R^2 and Adjusted R^2 values rose, suggesting a marginally better match. Improved forecast accuracy was indicated by the Residual Standard Error, which dropped from 1.194 to 1.151. Income shown a weaker positive influence and the coefficient for females became somewhat less negative than in the model lacking high-leverage spots. Overall, the model's performance was enhanced and modified by eliminating outliers.

4.(g)

```
> #identifing high-leverage points
>
> high_leverage_points <- which(leverage_values > high_leverage_threshold)
>
> #removing high-leverage points
> insurance_filtered <- insurance[-high_leverage_points, ]
> #Refitting the linear model without high-leverage points
>
> linear_model_filtered <- lm(log_totexp ~ suppins + phylim + actlim + tot
hr + age + female + income + mwest + northe, data = insurance_filtered)
>
> summary(linear_model_filtered)

Call:
lm(formula = log_totexp ~ suppins + phylim + actlim + totchr +
    age + female + income + mwest + northe, data = insurance_filtered)

Residuals:
    Min      1Q  Median      3Q     Max
-6.2723 -0.7212 -0.0098  0.7680  3.7246

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.630100   0.277819  23.865  < 2e-16 ***
suppins      0.235426   0.046283   5.087 3.87e-07 ***
phylim1      0.296612   0.056888   5.214 1.98e-07 ***
actlim1      0.382921   0.062005   6.176 7.50e-10 ***
totchr       0.372416   0.018351  20.294  < 2e-16 ***
age          0.003386   0.003670   0.923 0.356236
female      -0.093326   0.045414  -2.055 0.039967 *
income       0.005218   0.001277   4.085 4.52e-05 ***
mwest        0.204937   0.054054   3.791 0.000153 ***
northe       0.145102   0.063266   2.294 0.021889 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.194 on 2917 degrees of freedom
Multiple R-squared:  0.2356,    Adjusted R-squared:  0.2333
F-statistic: 99.91 on 9 and 2917 DF,  p-value: < 2.2e-16
```

The regression output changed a little after the high-leverage points were eliminated. The significance of the female coefficient improved as it became more negative. The income coefficient rose, indicating a strong positive correlation with log(totexp). The model's fit and accuracy were slightly improved, as evidenced by the R^2 and Adjusted R^2 values improving slightly and the Residual Standard Error dropping from 1.199 to 1.194. Overall, the model was improved by eliminating the outliers, but there were no significant changes.

4.(h)

```
> #fitting the model without age
>
> linear_model_reduced <- lm(log_totexp ~ suppins + phylim + actlim + totch
r + female + income + mwest + northe, data = insurance_filtered)
>
> summary(linear_model_reduced)

Call:
lm(formula = log_totexp ~ suppins + phylim + actlim + totchr +
    female + income + mwest + northe, data = insurance_filtered)

Residuals:
    Min      1Q  Median      3Q     Max
-6.2857 -0.7193 -0.0091  0.7606  3.7066

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.879798   0.062854 109.456  < 2e-16 ***
suppins      0.231675   0.046102   5.025 5.33e-07 ***
phylim1      0.303967   0.056325   5.397 7.33e-08 ***
actlim1      0.388873   0.061667   6.306 3.30e-10 ***
totchr       0.372171   0.018348  20.284  < 2e-16 ***
female      -0.091819   0.045383  -2.023 0.043145 *
income       0.005093   0.001270   4.010 6.22e-05 ***
mwest        0.207591   0.053976   3.846 0.000123 ***
northe       0.149789   0.063060   2.375 0.017597 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.194 on 2918 degrees of freedom
Multiple R-squared:  0.2354,    Adjusted R-squared:  0.2333
F-statistic: 112.3 on 8 and 2918 DF,  p-value: < 2.2e-16
```

The adjusted R^2 and residual standard error, which were 0.2333 and 1.194, respectively, did not change when the variable age was eliminated (p-value > 0.01). This suggests that the explanatory power of the model was not much impacted by age. Since the fit and accuracy of the reduced model are the same as those of the original model, it is neither better nor worse.

Q5.

```
> insurance$high <- ifelse(insurance$totexp > median(insurance$totexp), 1,
0)
> insurance$high <- factor(insurance$high, levels = c(0, 1))
> set.seed(123)
> train_indices <- sample(1:nrow(insurance), 2000)
> training_data <- insurance[train_indices, ]
> test_data <- insurance[-train_indices, ]
> logit_model <- glm(high ~ income + actlim + phylim + totchr, data = train
ing_data, family = binomial)
> summary(logit_model)

Call:
glm(formula = high ~ income + actlim + phylim + totchr, family = binomial,
    data = training_data)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.455693   0.108988 -13.356  < 2e-16 ***
income       0.005225   0.002080   2.512    0.012 *
actlim1      0.566925   0.135445   4.186 2.84e-05 ***
phylim1      0.504251   0.119462   4.221 2.43e-05 ***
totchr       0.538379   0.044262  12.164  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2772.6  on 1999  degrees of freedom
Residual deviance: 2420.4  on 1995  degrees of freedom
AIC: 2430.4

Number of Fisher Scoring iterations: 4
```

Q5.(a)

```
> calculate_error <- function(cutoff) {
+     predicted_class <- ifelse(test_data$predicted_prob > cutoff, 1, 0)
+     error_rate <- mean(predicted_class != test_data$high)
+     return(error_rate)
+ }
>
> error_0.2 <- calculate_error(0.2)
> error_0.4 <- calculate_error(0.4)
> error_0.6 <- calculate_error(0.6)
>
> cat("Cut-off 0.2 Error Rate:", error_0.2, "\n")
Cut-off 0.2 Error Rate: 0.4727463
> cat("Cut-off 0.4 Error Rate:", error_0.4, "\n")
Cut-off 0.4 Error Rate: 0.3658281
> cat("Cut-off 0.6 Error Rate:", error_0.6, "\n")
Cut-off 0.6 Error Rate: 0.3291405
```

The logistic regression model was fitted with income, actlim, phylim, and totchr as predictors to predict high. The greatest favourable effects were seen with totchr and actlim, however all variables were significant. The test dataset was then subjected to the fitted model, and the error rates were calculated by evaluating the predictions for various cut-offs. At the cut-off of 0.6, the error rate was at its lowest.

The logistic regression model's test error rates are as follows:

0.2 cut-off: 47.27%

0.4 cut-off: 36.58%

0.6 cut-off: 32.91%.

The most preferred cut-off among these is 0.6 since it reduces the test error rate to 32.91%.

5.(b)

```
> install.packages("dplyr")
Error in install.packages : Updating loaded packages
> install.packages("dplyr")
trying URL 'https://cran.rstudio.com/bin/macosx/big-sur-arm64/contrib/4.4/d
plyr_1.1.4.tgz'
Content type 'application/x-gzip' length 1599250 bytes (1.5 MB)
==================================================
downloaded 1.5 MB


The downloaded binary packages are in
        /var/folders/qb/h3kwzty90p57nvjvjvg6k9jw0000gn/T//RtmpvtsLnB/downlo
aded_packages
> library(MASS)
> lda_model <- lda(high ~ income + actlim + phylim + totchr, data = trainin
g_data)
>
> lda_predictions <- predict(lda_model, newdata = test_data)
> test_data$lda_prob <- lda_predictions$posterior[, 2]
> calculate_lda_error <- function(cutoff) {
+     predicted_class <- ifelse(test_data$lda_prob > cutoff, 1, 0)
+     error_rate <- mean(predicted_class != test_data$high)
+     return(error_rate)
+ }
> lda_error_0.2 <- calculate_lda_error(0.2)
> lda_error_0.4 <- calculate_lda_error(0.4)
> lda_error_0.6 <- calculate_lda_error(0.6)
> cat("LDA Error Rates:\n")
LDA Error Rates:
> cat("Cut-off 0.2:", lda_error_0.2, "\n")
Cut-off 0.2: 0.4549266
> cat("Cut-off 0.4:", lda_error_0.4, "\n")
Cut-off 0.4: 0.3658281
> cat("Cut-off 0.6:", lda_error_0.6, "\n")
Cut-off 0.6: 0.3291405
```

The error rates for the LDA method were:

Cut-off 0.2: 45.49%

Cut-off 0.4: 36.58%

Cut-off 0.6: 32.91%.

Among the three cut-offs, 0.6 is the most preferred cut-off because it has the lowest error rate.

5.(c)

```
> #fit the QDA model
> qda_model <- qda(high ~ income + actlim + phylim + totchr, data = traini
g_data)
> #predicting probabilities
> qda_predictions <- predict(qda_model, newdata = test_data)
> test_data$qda_prob <- qda_predictions$posterior[, 2]  # Probabilities fo
'high = 1'
>
> #function to calculate classification error
> calculate_error <- function(cutoff) {
+     predicted_class <- ifelse(test_data$qda_prob > cutoff, 1, 0)
+     error_rate <- mean(predicted_class != test_data$high)
+     return(error_rate)
+ }
>
> #calculating error rates for the specified cut-offs
> qda_error_0.2 <- calculate_error(0.2)
> qda_error_0.4 <- calculate_error(0.4)
> qda_error_0.6 <- calculate_error(0.6)
>
> #error rates
> cat("QDA Cut-off 0.2 Error Rate:", qda_error_0.2, "\n")
QDA Cut-off 0.2 Error Rate: 0.4444444
> cat("QDA Cut-off 0.4 Error Rate:", qda_error_0.4, "\n")
QDA Cut-off 0.4 Error Rate: 0.3375262
> cat("QDA Cut-off 0.6 Error Rate:", qda_error_0.6, "\n")
QDA Cut-off 0.6 Error Rate: 0.3480084
```

Classification Error Rates for QDA Tests:

Cutoff point 0.2: 44.44%

Cut-off point 0.4: 33.75%

Cut-off point 0.6: 34.80%

The cut-off that is most frequently used is 0.4 since it produces the lowest categorisation error rate.

5.(d)

```
> library(class)
>
> #a function to calculate KNN error
> calculate_knn_error <- function(k) {
+     knn_predictions <- knn(train = training_data[, c("income", "actlim",
"phylim", "totchr")], test = test_data[, c("income", "actlim", "phylim", "t
otchr")], cl = training_data$high, k = k)
+     error_rate <- mean(knn_predictions != test_data$high)
+     return(error_rate)
+ }
>
> #classification error rates for K = 1, 5, 10, 20, 50, and 100
> knn_error_1 <- calculate_knn_error(1)
> knn_error_5 <- calculate_knn_error(5)
> knn_error_10 <- calculate_knn_error(10)
> knn_error_20 <- calculate_knn_error(20)
> knn_error_50 <- calculate_knn_error(50)
> knn_error_100 <- calculate_knn_error(100)
>
> #error rates
> cat("K=1 Error Rate:", knn_error_1, "\n")
K=1 Error Rate: 0.4371069
> cat("K=5 Error Rate:", knn_error_5, "\n")
K=5 Error Rate: 0.3878407
> cat("K=10 Error Rate:", knn_error_10, "\n")
K=10 Error Rate: 0.3668763
> cat("K=20 Error Rate:", knn_error_20, "\n")
K=20 Error Rate: 0.3668763
> cat("K=50 Error Rate:", knn_error_50, "\n")
K=50 Error Rate: 0.3616352
> cat("K=100 Error Rate:", knn_error_100, "\n")
K=100 Error Rate: 0.3417191
```

Error Rates:

K=1: 0.4371

K=5: 0.3878

K=10: 0.3669

K=20: 0.3669

K=50: 0.3616

K=100: 0.3417

The most preferred value of K is 100, as it results in the lowest classification error rate (0.3417), indicating better model performance.

5.(e)

Since the Quadratic Discriminant Analysis (QDA) approach has the lowest classification error rate (about 0.3375 for the cut-off of 0.4), it seems to produce the best results on this data. This suggests that the model successfully depicts the connection between the predictors and the outcome variable.

The better outcomes of QDA over Logistic Regression and Linear Discriminant Analysis (LDA) from the standpoint of data generation implies that there may be non-linear boundaries in the relationship between the predictors and the result. Because QDA is flexible in managing non-linear interactions, it can represent more complex data distributions.

Q6.

```
> #initializing storage
> set.seed(123)
> n_bootstrap <- 1000
> bootstrap_coefficients <- matrix(NA, nrow = n_bootstrap, ncol = 4)
> for (i in 1:n_bootstrap) {
+     bootstrap_sample <- insurance[sample(1:nrow(insurance), replace = TRU
E), ]
+     lda_model <- lda(high ~ income + actlim + phylim + totchr, data = boo
tstrap_sample)
+     bootstrap_coefficients[i, ] <- lda_model$scaling[, 1]
+ }
> standard_errors <- apply(bootstrap_coefficients, 2, sd)
> lda_model_full <- lda(high ~ income + actlim + phylim + totchr, data = in
surance)
> original_coefficients <- lda_model_full$scaling[, 1]
> lower_bound <- original_coefficients - 1.96 * standard_errors
> upper_bound <- original_coefficients + 1.96 * standard_errors
> confidence_intervals <- data.frame(
+     Predictor = c("income", "actlim", "phylim", "totchr"),
+     Coefficient = original_coefficients,
+     Std_Error = standard_errors,
+     Lower_CI = lower_bound,
+     Upper_CI = upper_bound
+ )
> print(confidence_intervals)
        Predictor Coefficient   Std_Error    Lower_CI    Upper_CI
income     income 0.005543539 0.001971086 0.001680211 0.009406868
actlim1    actlim 0.566529742 0.128320161 0.315022225 0.818037258
phylim1    phylim 0.606553755 0.114728213 0.381686458 0.831421053
totchr     totchr 0.632348103 0.028126266 0.577220622 0.687475584
>
>
```

For the predictors income, actlim, phylim, and totchr, I used the bootstrap approach with 1000 resamples to calculate the standard errors of the LDA coefficients. 95% confidence intervals were calculated using the standard errors for the coefficients and the formula CI = Coefficient ± 1.96 × Standard Error.

The results show:

The income has a narrow confidence interval (0.0017, 0.0094).

The coefficients for actlim, phylim, and totchr are all positive, and their respective confidence intervals do not include zero: 0.315, 0.818; 0.381, 0.831; and 0.577, 0.687. Totchr has the most potent and reliable impact among them.

These results demonstrate that each of the four predictors makes significant contributions to the classification, with totchr having the most consistent and reliable effect.

Q7

```
> logit_model <- glm(high ~ income + actlim + phylim + totchr,
+     data = insurance,
+     family = binomial)
> calculate_error <- function(actual, predicted_prob, cutoff = 0.3) {
+     predicted_class <- ifelse(predicted_prob > cutoff, 1, 0)
+     mean(predicted_class != actual)
+ }
>
> #LOOCV Error
> loocv_error <- cv.glm(data = insurance, glmfit = logit_model)$delta[1]  # LOOCV Error
>
> #K-Fold Errors
> k_values <- c(2, 5, 10, 20, 50)
> k_fold_errors <- sapply(k_values, function(k) {
+     # Perform K-Fold CV
+     folds <- cut(seq(1, nrow(insurance)), breaks = k, labels = FALSE)
+     error_rates <- sapply(1:k, function(fold) {
+     train_data <- insurance[folds != fold, ]
+     test_data <- insurance[folds == fold, ]
+     logit_model_k <- glm(high ~ income + actlim + phylim + totchr,
+     data = train_data,
+  family = binomial)
+     predicted_prob <- predict(logit_model_k, newdata = test_data, type = "response")
+  calculate_error(test_data$high, predicted_prob, cutoff = 0.3)
+     })
+     mean(error_rates)  # Average error rate across all folds
+ })
Warning messages:
1: glm.fit: algorithm did not converge
2: glm.fit: algorithm did not converge
>
> #results
> cat("LOOCV Error Rate:", loocv_error, "\n")
LOOCV Error Rate: 0.2088324
> for (i in 1:length(k_values)) {
+     cat("K =", k_values[i], "Error Rate:", k_fold_errors[i], "\n")
+ }
K = 2 Error Rate: 0.5
K = 5 Error Rate: 0.5592159
K = 10 Error Rate: 0.4956436
K = 20 Error Rate: 0.4807432
K = 50 Error Rate: 0.4544011
```

Since the Leave-One-Out Cross-Validation (LOOCV) method trains on almost the entire dataset for each iteration, it achieved the lowest error rate of 0.2088, demonstrating high reliability. However, it is computationally intensive.

For k-fold Cross-Validation, the lowest error rate of 0.4544 was observed at K=50, with error rates improving as K increased. Smaller K-values (e.g., K=2) showed higher error rates due to greater variability in the training subsets.

Non-convergence warnings occurred for certain k-fold splits, likely caused by imbalances in the training data. However, these warnings did not significantly affect the outcomes.

In summary, LOOCV is computationally demanding but provides the most accurate error estimate, while K=50offers a computationally efficient alternative with reasonable error performance.

Appendix

Q1 .

```
> no_observations <- nrow(insurance)

> cat("Total Observations:", no_observations)
```

Q2.

```
> numeric_variables <- insurance [,sapply (insurance, is.numeric)]

> correlations <- cor(numeric_variables, use="complete.obs")["totexp", ]

> sorted_correlations <- sort(abs(correlations), decreasing = TRUE)

> sorted_correlations <- sorted_correlations[names(sorted_correlations) != "totexp"]

> top_correlations <- head(sorted_correlations, 5)

> print(top_correlations)
```

Q3.

```
> #adding a column for log(totexp)

> insurance$log_totexp <- log(insurance$totexp)

> #converting the indicator variables

> insurance$actlim <- as.factor(insurance$actlim)

> insurance$phylim <- as.factor(insurance$phylim)

> insurance$priolist <- as.factor(insurance$priolist)

> insurance$hvgg <- as.factor(insurance$hvgg)

> ggplot(insurance, aes(x = totchr, y = log_totexp)) + geom_point(alpha = 0.5) + geom_smooth(method = "lm", col = "red", se = FALSE) + labs( title = "Scatterplot of log(totexp) vs totchr", x = "Number of Chronic Problems (totchr)", y = "log(Total Medical Expenditure)") + theme_minimal()

> #scatterplot for actlim vs log(totexp)

> ggplot(insurance, aes(x = as.numeric(actlim), y = log_totexp)) + geom_point(alpha = 0.5) + geom_smooth(method = "lm", col = "red", se = FALSE) + labs(title = "Scatterplot of
```

log(totexp) vs actlim", x = "Activity Limitation (Numeric Representation)", y = "log(Total Medical Expenditure)") + theme_minimal()

> #scatterplot for phylim vs log(totexp)

> ggplot(insurance, aes(x = as.numeric(phylim), y = log_totexp)) + geom_point(alpha = 0.5) + geom_smooth(method = "lm", col = "red", se = FALSE) + labs(

+ title = "Scatterplot of log(totexp) vs phylim", x = "Physical Limitation (Numeric Representation)", y = "log(Total Medical Expenditure)") + theme_minimal()

> #scatterplot for priolist vs log(totexp)

> ggplot(insurance, aes(x = as.numeric(priolist), y = log_totexp)) + geom_point(alpha = 0.5) + geom_smooth(method = "lm", col = "blue", se = FALSE) + labs(title = "Scatterplot of log(totexp) vs priolist", x = "Medical Priority Condition (Numeric Representation)", y = "log(Total Medical Expenditure)") + theme_minimal()

> #scatterplot for hvgg vs log(totexp)

> ggplot(insurance, aes(x = as.numeric(hvgg), y = log_totexp)) + geom_point(alpha = 0.5) + geom_smooth(method = "lm", col = "blue", se = FALSE) +

+ labs(title = "Scatterplot of log(totexp) vs hvgg", x = "Health Status (Numeric Representation)", y = "log(Total Medical Expenditure)") + theme_minimal()

Q4.(a,b,c)

#the linear regression model

> linear_model <- lm(log_totexp ~ suppins + phylim + actlim + totchr + age + female + income + mwest + northe, data = insurance)

> summary(linear_model)

4.(d)

#plots for the linear model

> par(mfrow = c(2, 2))

> plot(linear_model)

4.(e)

> leverage_values <- hatvalues(linear_model)

```
> p <- length(coef(linear_model))

> n <- nrow(insurance)

> average_leverage <- p / n

> high_leverage_threshold <- 3 * average_leverage

> num_high_leverage <- sum(leverage_values > high_leverage_threshold)

> print(num_high_leverage)
```

4.(f)

```
> standardized_residuals <- rstandard(linear_model)

> outliers <- which(abs(standardized_residuals) > 3)

> insurance_no_outliers <- insurance[-outliers, ]

> linear_model_no_outliers <- lm(log_totexp ~ suppins + phylim + actlim + totchr + age + female + income + mwest + northe, data = insurance_no_outliers)

> summary(linear_model_no_outliers)
```

4.(g)

```
#identifing high-leverage points

> high_leverage_points <- which(leverage_values > high_leverage_threshold)

> #removing high-leverage points

> insurance_filtered <- insurance[-high_leverage_points, ]

> linear_model_filtered <- lm(log_totexp ~ suppins + phylim + actlim + totchr + age + female + income + mwest + northe, data = insurance_filtered)

> summary(linear_model_filtered)
```

4.(h)

```
> #fitting the model without age

> linear_model_reduced <- lm(log_totexp ~ suppins + phylim + actlim + totchr + female + income + mwest + northe, data = insurance_filtered)

> summary(linear_model_reduced)
```

Q5.

```
> insurance$high <- ifelse(insurance$totexp > median(insurance$totexp), 1, 0)
> insurance$high <- factor(insurance$high, levels = c(0, 1))
> set.seed(123)
> train_indices <- sample(1:nrow(insurance), 2000)
> training_data <- insurance[train_indices, ]
> test_data <- insurance[-train_indices, ]
> logit_model <- glm(high ~ income + actlim + phylim + totchr, data = training_data, family = binomial)
> summary(logit_model)
```

5.(a)

```
> calculate_error <- function(cutoff) {
+     predicted_class <- ifelse(test_data$predicted_prob > cutoff, 1, 0)
+     error_rate <- mean(predicted_class != test_data$high)
+     return(error_rate)
+ }
> error_0.2 <- calculate_error(0.2)
> error_0.4 <- calculate_error(0.4)
> error_0.6 <- calculate_error(0.6)
> cat("Cut-off 0.2 Error Rate:", error_0.2, "\n")
> cat("Cut-off 0.4 Error Rate:", error_0.4, "\n")
> cat("Cut-off 0.6 Error Rate:", error_0.6, "\n")
```

5.(b)

```
> install.packages("dplyr")
> library(MASS)
> lda_model <- lda(high ~ income + actlim + phylim + totchr, data = training_data)
```

```
>
> lda_predictions <- predict(lda_model, newdata = test_data)
> test_data$lda_prob <- lda_predictions$posterior[, 2]
> calculate_lda_error <- function(cutoff) {
+     predicted_class <- ifelse(test_data$lda_prob > cutoff, 1, 0)
+     error_rate <- mean(predicted_class != test_data$high)
+     return(error_rate)
+ }
> lda_error_0.2 <- calculate_lda_error(0.2)
> lda_error_0.4 <- calculate_lda_error(0.4)
> lda_error_0.6 <- calculate_lda_error(0.6)
> cat("LDA Error Rates:\n")
> cat("Cut-off 0.2:", lda_error_0.2, "\n")
> cat("Cut-off 0.4:", lda_error_0.4, "\n")
> cat("Cut-off 0.6:", lda_error_0.6, "\n")
```

5.(c)

```
> #fit the QDA model
> qda_model <- qda(high ~ income + actlim + phylim + totchr, data = training_data)
> #predicting probabilities
> qda_predictions <- predict(qda_model, newdata = test_data)
> test_data$qda_prob <- qda_predictions$posterior[, 2]  # Probabilities for 'high = 1'
> #function to calculate classification error
> calculate_error <- function(cutoff) {
+     predicted_class <- ifelse(test_data$qda_prob > cutoff, 1, 0)
+     error_rate <- mean(predicted_class != test_data$high)
```

```
+     return(error_rate)
+ }
> #calculating error rates for the specified cut-offs
> qda_error_0.2 <- calculate_error(0.2)
> qda_error_0.4 <- calculate_error(0.4)
> qda_error_0.6 <- calculate_error(0.6)
> #error rates
> cat("QDA Cut-off 0.2 Error Rate:", qda_error_0.2, "\n")
> cat("QDA Cut-off 0.4 Error Rate:", qda_error_0.4, "\n")
> cat("QDA Cut-off 0.6 Error Rate:", qda_error_0.6, "\n")
```

5.(d)

```
> library(class)
> #a function to calculate KNN error
> calculate_knn_error <- function(k) {
+      knn_predictions <- knn(train = training_data[, c("income", "actlim", "phylim", "totchr")], test = test_data[, c("income", "actlim", "phylim", "totchr")], cl = training_data$high, k = k)
+     error_rate <- mean(knn_predictions != test_data$high)
+     return(error_rate)
+ }
> #classification error rates for K = 1, 5, 10, 20, 50, and 100
> knn_error_1 <- calculate_knn_error(1)
> knn_error_5 <- calculate_knn_error(5)
> knn_error_10 <- calculate_knn_error(10)
> knn_error_20 <- calculate_knn_error(20)
> knn_error_50 <- calculate_knn_error(50)
> knn_error_100 <- calculate_knn_error(100)
```

```
> #error rates
> cat("K=1 Error Rate:", knn_error_1, "\n")
> cat("K=5 Error Rate:", knn_error_5, "\n")
> cat("K=10 Error Rate:", knn_error_10, "\n")
> cat("K=20 Error Rate:", knn_error_20, "\n")
> cat("K=50 Error Rate:", knn_error_50, "\n")
> cat("K=100 Error Rate:", knn_error_100, "\n")
```

Q6.

```
> #initializing storage
> set.seed(123)
> n_bootstrap <- 1000
> bootstrap_coefficients <- matrix(NA, nrow = n_bootstrap, ncol = 4)
> for (i in 1:n_bootstrap) {
+     bootstrap_sample <- insurance[sample(1:nrow(insurance), replace = TRUE), ]
+     lda_model <- lda(high ~ income + actlim + phylim + totchr, data = bootstrap_sample)
+     bootstrap_coefficients[i, ] <- lda_model$scaling[, 1]
+ }
> standard_errors <- apply(bootstrap_coefficients, 2, sd)
> lda_model_full <- lda(high ~ income + actlim + phylim + totchr, data = insurance)
> original_coefficients <- lda_model_full$scaling[, 1]
> lower_bound <- original_coefficients - 1.96 * standard_errors
> upper_bound <- original_coefficients + 1.96 * standard_errors
> confidence_intervals <- data.frame(
+     Predictor = c("income", "actlim", "phylim", "totchr"),
+     Coefficient = original_coefficients,
```

```
+     Std_Error = standard_errors,
+     Lower_CI = lower_bound,
+     Upper_CI = upper_bound
+ )
> print(confidence_intervals)
```

Q7.

```
> logit_model <- glm(high ~ income + actlim + phylim + totchr,
+     data = insurance,
+     family = binomial)
> calculate_error <- function(actual, predicted_prob, cutoff = 0.3) {
+     predicted_class <- ifelse(predicted_prob > cutoff, 1, 0)
+     mean(predicted_class != actual)
+ }
> #LOOCV Error
> loocv_error <- cv.glm(data = insurance, glmfit = logit_model)$delta[1]  # LOOCV Error
> #K-Fold Errors
> k_values <- c(2, 5, 10, 20, 50)
> k_fold_errors <- sapply(k_values, function(k) {
+     # Perform K-Fold CV
+     folds <- cut(seq(1, nrow(insurance)), breaks = k, labels = FALSE)
+     error_rates <- sapply(1:k, function(fold) {
+     train_data <- insurance[folds != fold, ]
+     test_data <- insurance[folds == fold, ]
+     logit_model_k <- glm(high ~ income + actlim + phylim + totchr,
+     data = train_data,
```

```
+  family = binomial)
+     predicted_prob <- predict(logit_model_k, newdata = test_data, type = "response")
+  calculate_error(test_data$high, predicted_prob, cutoff = 0.3)
+     })
+     mean(error_rates)  # Average error rate across all folds
+ })
> #results
> cat("LOOCV Error Rate:", loocv_error, "\n")
> for (i in 1:length(k_values)) {
+     cat("K =", k_values[i], "Error Rate:", k_fold_errors[i], "\n")
+ }
```