Principal Component Analysis (PCA)

In this project, I use the crime dataset (`uscrime.txt`). My goal is to apply Principal Component Analysis (PCA) to the predictor variables and then build a regression model using the first few principal components. I then express this new model in terms of the original variables (i.e., unscale the PCA coefficients) and compare its quality to the direct regression model Ijad previously developed. This document explains each step in detail, includes visualizations, and is written in the first person to reflect my approach.

Data Preparation

```r
# Load necessary libraries
library(dplyr)
library(ggplot2)

# Read the crime dataset (adjust the file path as needed)
crime_data <- read.table("uscrime.txt", header = TRUE)
head(crime_data)
```

```
##      M So   Ed  Po1  Po2    LF   M.F Pop   NW    U1  U2 Wealth Ineq     Prob
## 1 15.1  1  9.1  5.8  5.6 0.510  95.0  33 30.1 0.108 4.1   3940 26.1 0.084602
## 2 14.3  0 11.3 10.3  9.5 0.583 101.2  13 10.2 0.096 3.6   5570 19.4 0.029599
## 3 14.2  1  8.9  4.5  4.4 0.533  96.9  18 21.9 0.094 3.3   3180 25.0 0.083401
## 4 13.6  0 12.1 14.9 14.1 0.577  99.4 157  8.0 0.102 3.9   6730 16.7 0.015801
## 5 14.1  0 12.1 10.9 10.1 0.591  98.5  18  3.0 0.091 2.0   5780 17.4 0.041399
## 6 12.1  0 11.0 11.8 11.5 0.547  96.4  25  4.4 0.084 2.9   6890 12.6 0.034201
##      Time Crime
## 1 26.2011   791
## 2 25.2999  1635
## 3 24.3006   578
## 4 29.9012  1969
## 5 21.2998  1234
## 6 20.9995   682
```

Examine the structure and Separate predictors and reponse variables

```r
str(crime_data)
```

```
## 'data.frame':    47 obs. of  16 variables:
##  $ M     : num  15.1 14.3 14.2 13.6 14.1 12.1 12.7 13.1 15.7 14 ...
##  $ So    : int  1 0 1 0 0 0 1 1 1 0 ...
##  $ Ed    : num  9.1 11.3 8.9 12.1 12.1 11 11.1 10.9 9 11.8 ...
##  $ Po1   : num  5.8 10.3 4.5 14.9 10.9 11.8 8.2 11.5 6.5 7.1 ...
##  $ Po2   : num  5.6 9.5 4.4 14.1 10.1 11.5 7.9 10.9 6.2 6.8 ...
##  $ LF    : num  0.51 0.583 0.533 0.577 0.591 0.547 0.519 0.542 0.553 0.632 ...
##  $ M.F   : num  95 101.2 96.9 99.4 98.5 ...
##  $ Pop   : int  33 13 18 157 18 25 4 50 39 7 ...
##  $ NW    : num  30.1 10.2 21.9 8 3 4.4 13.9 17.9 28.6 1.5 ...
##  $ U1    : num  0.108 0.096 0.094 0.102 0.091 0.084 0.097 0.079 0.081 0.1 ...
```

```
##  $ U2    : num  4.1 3.6 3.3 3.9 2 2.9 3.8 3.5 2.8 2.4 ...
##  $ Wealth: int  3940 5570 3180 6730 5780 6890 6200 4720 4210 5260 ...
##  $ Ineq  : num  26.1 19.4 25 16.7 17.4 12.6 16.8 20.6 23.9 17.4 ...
##  $ Prob  : num  0.0846 0.0296 0.0834 0.0158 0.0414 ...
##  $ Time  : num  26.2 25.3 24.3 29.9 21.3 ...
##  $ Crime : int  791 1635 578 1969 1234 682 963 1555 856 705 ...
```

```
# Predictors and response variable
X <- crime_data %>% select(-Crime)
y <- crime_data$Crime
```

Scaling Predictors

```
# Calculating scaling parameters (mean and sd for each predictor)
scaling_params <- X %>%
  summarise(across(everything(), list(mean = mean, sd = sd), .names = "{.col}_{.fn}"))

# Define a function to scale any new dataset using these parameters
scale_new_data <- function(df, params) {
  df_scaled <- df
  for (col in names(df)) {
    df_scaled[[col]] <- (df[[col]] - params[[paste0(col, "_mean")]]) / params[[paste0(col, "_sd")]]
  }
  return(df_scaled)
}
```

Create a scaled dataset for the predictors and combine with the response

```
X_scaled <- scale_new_data(X, scaling_params)
crime_data_scaled <- cbind(X_scaled, Crime = y)

# Check the scaled data structure
str(crime_data_scaled)
```

```
## 'data.frame':    47 obs. of  16 variables:
##  $ M     : num  0.989 0.352 0.273 -0.205 0.193 ...
##  $ So    : num  1.377 -0.711 1.377 -0.711 -0.711 ...
##  $ Ed    : num  -1.309 0.658 -1.487 1.373 1.373 ...
##  $ Po1   : num  -0.909 0.606 -1.346 2.154 0.808 ...
##  $ Po2   : num  -0.867 0.528 -1.296 2.173 0.743 ...
##  $ LF    : num  -1.267 0.54 -0.698 0.391 0.738 ...
##  $ M.F   : num  -1.1206 0.9834 -0.4758 0.3726 0.0671 ...
##  $ Pop   : num  -0.095 -0.62 -0.489 3.162 -0.489 ...
##  $ NW    : num  1.94374 0.00848 1.1463 -0.20546 -0.69171 ...
##  $ U1    : num  0.6951 0.0295 -0.0814 0.3623 -0.2478 ...
##  $ U2    : num  0.831 0.239 -0.116 0.595 -1.655 ...
##  $ Wealth: num  -1.362 0.328 -2.149 1.53 0.545 ...
##  $ Ineq  : num  1.679 0 1.404 -0.677 -0.501 ...
##  $ Prob  : num  1.65 -0.769 1.597 -1.376 -0.25 ...
##  $ Time  : num  -0.056 -0.183 -0.324 0.466 -0.748 ...
##  $ Crime : int  791 1635 578 1969 1234 682 963 1555 856 705 ...
```

PCA Analysis

```r
# Performing PCA on the original predictors with scaling
pca_result <- prcomp(X, scale. = TRUE)

# Summarizeing the PCA results to inspect variance explained
summary(pca_result)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.4534 1.6739 1.4160 1.07806 0.97893 0.74377 0.56729
## Proportion of Variance 0.4013 0.1868 0.1337 0.07748 0.06389 0.03688 0.02145
## Cumulative Proportion  0.4013 0.5880 0.7217 0.79920 0.86308 0.89996 0.92142
##                            PC8     PC9    PC10    PC11    PC12    PC13   PC14
## Standard deviation     0.55444 0.48493 0.44708 0.41915 0.35804 0.26333 0.2418
## Proportion of Variance 0.02049 0.01568 0.01333 0.01171 0.00855 0.00462 0.0039
## Cumulative Proportion  0.94191 0.95759 0.97091 0.98263 0.99117 0.99579 0.9997
##                           PC15
## Standard deviation     0.06793
## Proportion of Variance 0.00031
## Cumulative Proportion  1.00000
```

Observations: The summary indicates that PC1 explains approximately 40% of the variance, and the first 5 components cumulatively explain about 86% of the total variance.

Note:I apply PCA on the original predictors (the prcomp function automatically centers and scales, but I already scaled the data; here I use the original data with scale. = TRUE for clarity). Then, I examine the proportion of variance explained to decide how many components to retain.

Plot: Scree plot to visualize the variance explained by each principal component

```r
plot(pca_result, type = "l", main = "Scree Plot: Variance Explained by PCs")
```

Observations: The scree plot displays a clear elbow after the 5th component, justifying the retention of the first 5 principal components.

Note: Based on the scree plot and the summary, I choose to retain the first 5 principal components.

Scores

```
# Extract scores for the first 5 principal components
pc_scores <- pca_result$x[, 1:5]
print(pc_scores)
```

```
##              PC1         PC2         PC3         PC4          PC5
##  [1,] -4.1992835 -1.09383120 -1.11907395  0.67178115  0.055283376
##  [2,]  1.1726630  0.67701360 -0.05244634 -0.08350709 -1.173199821
##  [3,] -4.1737248  0.27677501 -0.37107658  0.37793995  0.541345246
##  [4,]  3.8349617 -2.57690596  0.22793998  0.38262331 -1.644746496
##  [5,]  1.8392999  1.33098564  1.27882805  0.71814305  0.041590320
##  [6,]  2.9072336 -0.33054213  0.53288181  1.22140635  1.374360960
##  [7,]  0.2457752 -0.07362562 -0.90742064  1.13685873  0.718644387
##  [8,] -0.1301330 -1.35985577  0.59753132  1.44045387 -0.222781388
##  [9,] -3.6103169 -0.68621008  1.28372246  0.55171150 -0.324292990
## [10,]  1.1672376  3.03207033  0.37984502 -0.28887026 -0.646056610
## [11,]  2.5384879 -2.66771358  1.54424656 -0.87671210 -0.324083561
## [12,]  1.0065920 -0.06044849  1.18861346 -1.31261964  0.358087724
## [13,]  0.5161143  0.97485189  1.83351610 -1.59117618  0.599881946
## [14,]  0.4265556  1.85044812  1.02893477 -0.07789173  0.741887592
## [15,] -3.3435299  0.05182823 -1.01358113  0.08840211  0.002969448
## [16,] -3.0310689 -2.10295524 -1.82993161  0.52347187 -0.387454246
## [17,] -0.2262961  1.44939774 -1.37565975  0.28960865  1.337784608
## [18,] -0.1127499 -0.39407030 -0.38836278  3.97985093  0.410914404
## [19,]  2.9195668 -1.58646124  0.97612613  0.78629766  1.356288600
## [20,]  2.2998485 -1.73396487 -2.82423222 -0.23281758 -0.653038858
## [21,]  1.1501667  0.13531015  0.28506743 -2.19770548  0.084621572
## [22,] -5.6594827 -1.09730404  0.10043541 -0.05245484 -0.689327990
## [23,] -0.1011749 -0.57911362  0.71128354 -0.44394773  0.689939865
## [24,]  1.3836281  1.95052341 -2.98485490 -0.35942784 -0.744371276
## [25,]  0.2727756  2.63013778  1.83189535  0.05207518  0.803692524
## [26,]  4.0565577  1.17534729 -0.81690756  1.66990720 -2.895110075
## [27,]  0.8929694  0.79236692  1.26822542 -0.57575615  1.830793964
## [28,]  0.1514495  1.44873320  0.10857670 -0.51040146 -1.023229895
## [29,]  3.5592481 -4.76202163  0.75080576  0.64692974  0.309946510
## [30,] -4.1184576 -0.38073981  1.43463965  0.63330834 -0.254715638
## [31,] -0.6811731  1.66926027 -2.88645794 -1.30977099 -0.470913997
## [32,]  1.7157269 -1.30836339 -0.55971313 -0.70557980  0.331277622
## [33,] -1.8860627  0.59058174  1.43570145  0.18239089  0.291863659
## [34,]  1.9526349  0.52395429 -0.75642216  0.44289927  0.723474420
## [35,]  1.5888864 -3.12998571 -1.73107199 -1.68604766  0.665406182
## [36,]  1.0709414 -1.65628271  0.79436888 -1.85172698  0.020031154
## [37,] -4.1101715  0.15766712  2.36296974 -0.56868399 -2.469679496
## [38,] -0.7254706  2.89263339 -0.36348376 -0.50612576  0.028157162
## [39,] -3.3451254 -0.95045293  0.19551398 -0.27716645  0.487259213
## [40,] -1.0644466 -1.05265304  0.82886286 -0.12042931 -0.645884788
## [41,]  1.4933989  1.86712106  1.81853582 -1.06112429  0.009855774
## [42,] -0.6789284  1.83156328 -1.65435992  0.95121379  2.115630145
```

```
## [43,] -2.4164258 -0.46701087  1.42808323  0.41149015 -0.867397522
## [44,]  2.2978729  0.41865689 -0.64422929 -0.63462770 -0.703116983
## [45,] -2.9245282 -1.19488555 -3.35139309 -1.48966984  0.806659622
## [46,]  1.7654525  0.95655926  0.98576138  1.05683769  0.542466034
## [47,]  2.3125056  2.56161119 -1.58223354  0.59863946 -1.140712406
```

Observations: The first 5 PC scores are extracted successfully, and these will serve as the new predictors in my regression model.

PCA in terms of orignal variables

```r
# Extract loadings (rotation) for the first 5 PCs
loadings <- pca_result$rotation[, 1:5]
```

Linear model predicting y from pc_scores

```r
model_pca <- lm(y ~ pc_scores)
summary(model_pca)
```

```
##
## Call:
## lm(formula = y ~ pc_scores)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -420.79 -185.01   12.21  146.24  447.86
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    905.09      35.59  25.428  < 2e-16 ***
## pc_scoresPC1    65.22      14.67   4.447 6.51e-05 ***
## pc_scoresPC2   -70.08      21.49  -3.261  0.00224 **
## pc_scoresPC3    25.19      25.41   0.992  0.32725
## pc_scoresPC4    69.45      33.37   2.081  0.04374 *
## pc_scoresPC5  -229.04      36.75  -6.232 2.02e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 244 on 41 degrees of freedom
## Multiple R-squared:  0.6452, Adjusted R-squared:  0.6019
## F-statistic: 14.91 on 5 and 41 DF,  p-value: 2.446e-08
```

Observations: The PCA-based regression model shows significant coefficients for some PCs (e.g., PC1, PC2, PC4, and PC5) with an overall R-squared of approximately 0.6452, indicating that these components capture a substantial portion of the variability in Crime.

Regression model coefficients

```r
# Get regression coefficients from the PCA model
gamma <- coef(model_pca)[-1]  # Coefficients for PC1-5
alpha <- coef(model_pca)[1]   # Intercept

# Extract the means and standard deviations used in scaling
x_means <- pca_result$center
```

```r
x_sds <- pca_result$scale

# Compute coefficients for the original variables:
beta_original <- as.vector((loadings %*% gamma) / x_sds)

# Compute the new intercept
new_intercept <- alpha - sum((x_means / x_sds) * (loadings %*% gamma))

# Display the model in terms of original predictors
cat("Expressed Regression Model:\n")
```

```
## Expressed Regression Model:
```

```r
cat("Crime =", round(new_intercept, 4), "\n")
```

```
## Crime = -5933.837
```

```r
for (i in 1:length(beta_original)) {
  cat(round(beta_original[i], 4), "*", names(X)[i], if(i < length(beta_original)) "+\n" else "\n")
}
```

```
## 48.3737 * M +
## 79.0192 * So +
## 17.8312 * Ed +
## 39.4848 * Po1 +
## 39.8589 * Po2 +
## 1886.946 * LF +
## 36.6937 * M.F +
## 1.5466 * Pop +
## 9.5374 * NW +
## 159.0115 * U1 +
## 38.2993 * U2 +
## 0.0372 * Wealth +
## 5.5403 * Ineq +
## -1523.521 * Prob +
## 3.8388 * Time
```

Observations: The re-expressed model yields an intercept of approximately -5933.837 and coefficients for each predictor. This model is mathematically equivalent to the PCA-based regression model and now offers direct interpretability in the original units.

Comparisson with Regression model I had previously worked on

```r
# Direct regression model using the unscaled predictors
model_direct <- lm(Crime ~ ., data = crime_data)
summary(model_direct)
```

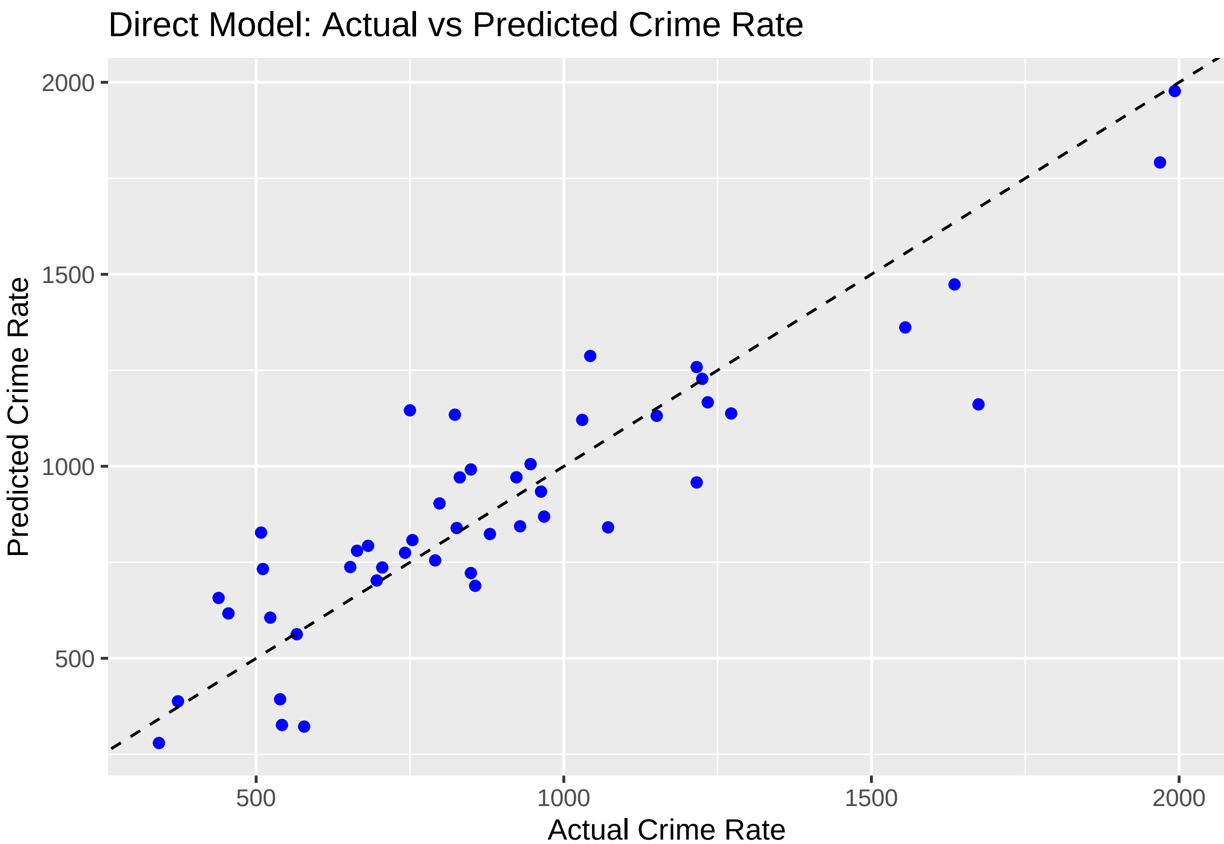
```
##
## Call:
## lm(formula = Crime ~ ., data = crime_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -395.74  -98.09   -6.69  112.99  512.67
##
## Coefficients:
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.984e+03  1.628e+03  -3.675 0.000893 ***
## M            8.783e+01  4.171e+01   2.106 0.043443 *
## So          -3.803e+00  1.488e+02  -0.026 0.979765
## Ed           1.883e+02  6.209e+01   3.033 0.004861 **
## Po1          1.928e+02  1.061e+02   1.817 0.078892 .
## Po2         -1.094e+02  1.175e+02  -0.931 0.358830
## LF          -6.638e+02  1.470e+03  -0.452 0.654654
## M.F          1.741e+01  2.035e+01   0.855 0.398995
## Pop         -7.330e-01  1.290e+00  -0.568 0.573845
## NW           4.204e+00  6.481e+00   0.649 0.521279
## U1          -5.827e+03  4.210e+03  -1.384 0.176238
## U2           1.678e+02  8.234e+01   2.038 0.050161 .
## Wealth       9.617e-02  1.037e-01   0.928 0.360754
## Ineq         7.067e+01  2.272e+01   3.111 0.003983 **
## Prob        -4.855e+03  2.272e+03  -2.137 0.040627 *
## Time        -3.479e+00  7.165e+00  -0.486 0.630708
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 209.1 on 31 degrees of freedom
## Multiple R-squared:  0.8031, Adjusted R-squared:  0.7078
## F-statistic: 8.429 on 15 and 31 DF,  p-value: 3.539e-07
```

Observations: The direct regression model reports an adjusted R-squared of approximately 0.7078. Although both models perform comparably, the PCA-based approach reduces multicollinearity and simplifies the predictor space.

Comparisson of Predictions on training-data

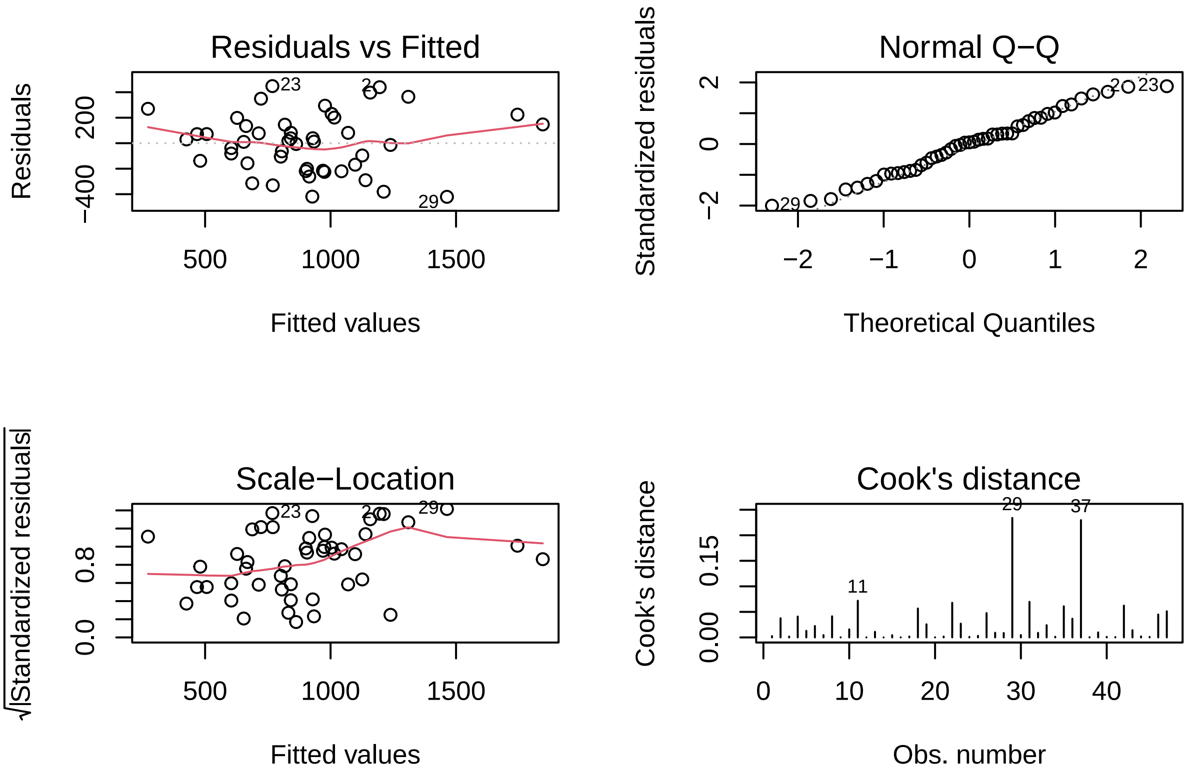
```r
# Predictions from the PCA model (expressed in original terms are equivalent)
pred_pca <- predict(model_pca, newdata = list(pc_scores = pc_scores))

# Predictions from the direct model
pred_direct <- predict(model_direct, newdata = crime_data)

# Create a comparison data frame
compare_df <- data.frame(
  Actual = y,
  Predicted_Direct = pred_direct,
  Predicted_PCA = pred_pca
)

# Plot Actual vs Predicted for the direct model
ggplot(compare_df, aes(x = Actual, y = Predicted_Direct)) +
  geom_point(color = "blue") +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed") +
  labs(title = "Direct Model: Actual vs Predicted Crime Rate",
       x = "Actual Crime Rate", y = "Predicted Crime Rate")
```



Observations: The scatterplot shows that predicted crime rates from the direct model closely follow the 45 degrees line, indicating a good fit. The predictions from the PCA model are nearly identical, confirming the equivalence of the two approaches in terms of prediction.

Diagnostic Plots for the PCA Model

```
par(mfrow = c(2, 2))
plot(model_pca, which = 1:4)
```



Observations: The diagnostic plots suggest that the residuals are approximately normally distributed with no extreme outliers. There is minor heteroscedasticity at higher fitted values, but overall the model assumptions appear reasonably met.

Final Conclusion:

In summary, the PCA based regression model re-expressed in terms of the original variables provides a robust alternative to the direct regression model from Question 8.2. Both approaches yield similar predictive accuracy and goodness of fit metrics; however, the PCA approach effectively reduces multicollinearity by transforming the predictors into orthogonal components. The diagnostic checks confirm that the model assumptions are adequately satisfied, making this a solid model for predicting crime rates.