# Polynomial and Multiple Regression using Physical Measures

White team

2023-10-10

## Packages

```
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(GGally)
```

```
## Loading required package: ggplot2
```

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

```
library(car)
```

```
## Loading required package: carData
```

```
library(readxl)
library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:car':
##
##     recode
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths
```

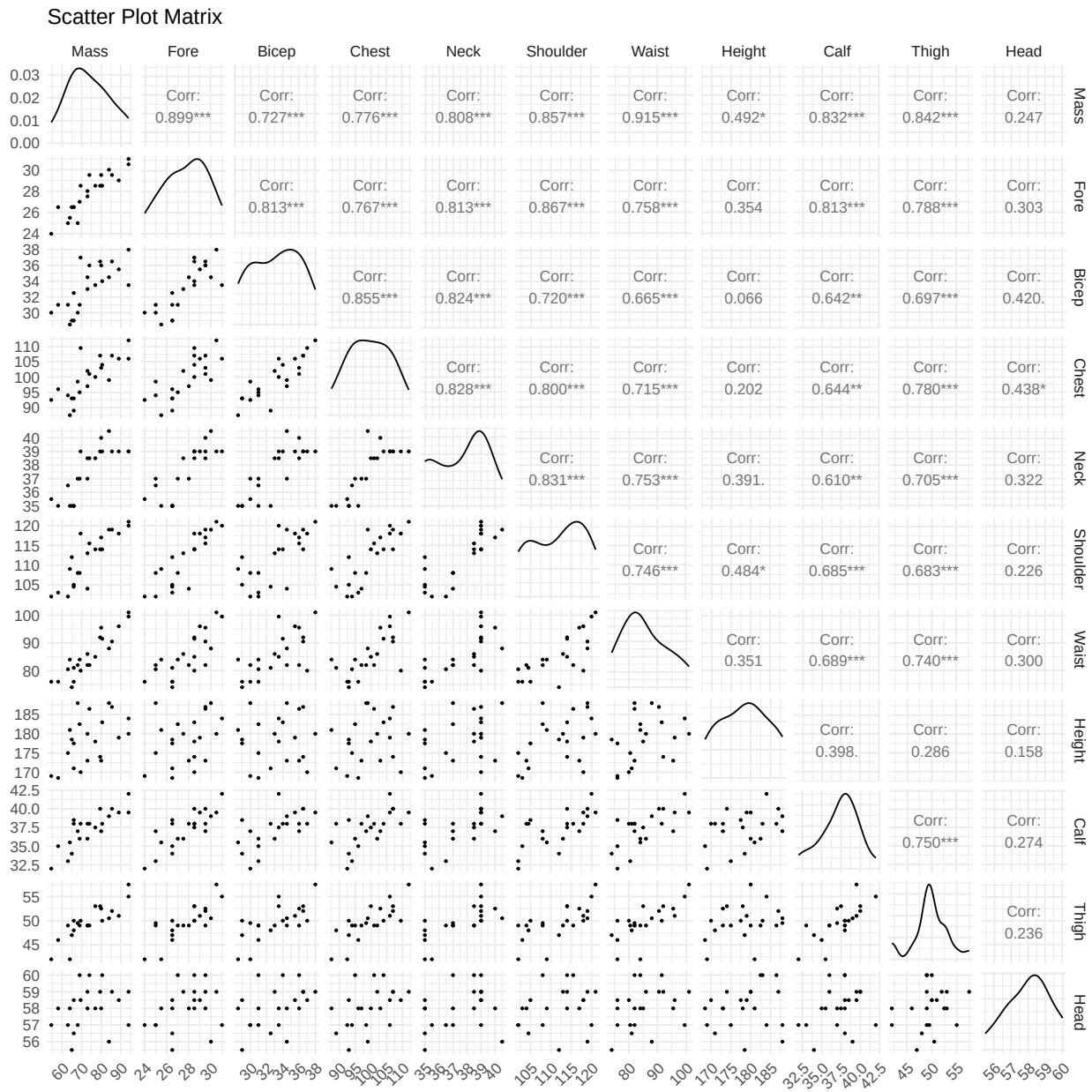
## load data

```
pm_dat <- read_excel("C:/Users/Chang/Downloads/physicalmeasures.xlsx")
```

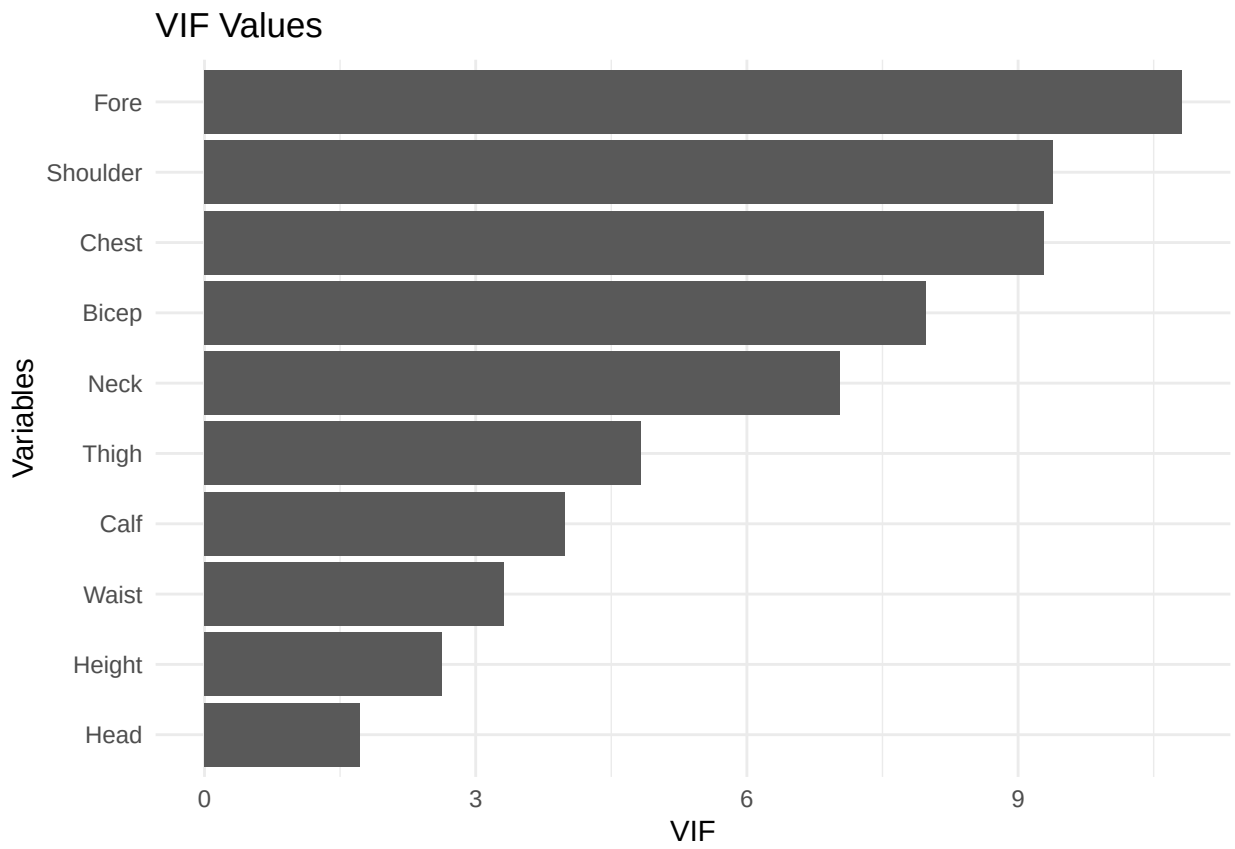
Before building a multiple linear regression model, it's essential to examine the correlations among the variables to detect any potential multicollinearity issues.

## Scatter Plot Matrix

```
ggpairs(pm_dat ,
        title="Scatter Plot Matrix",
        lower = list(continuous = wrap("points", size = 0.5)),
        upper = list(continuous = wrap("cor", size = 3.5))) +
  theme_minimal() +
  theme(text = element_text(size=12),
        axis.text.x = element_text(size=10, angle=45, hjust=1),
        axis.text.y = element_text(size=10))
```



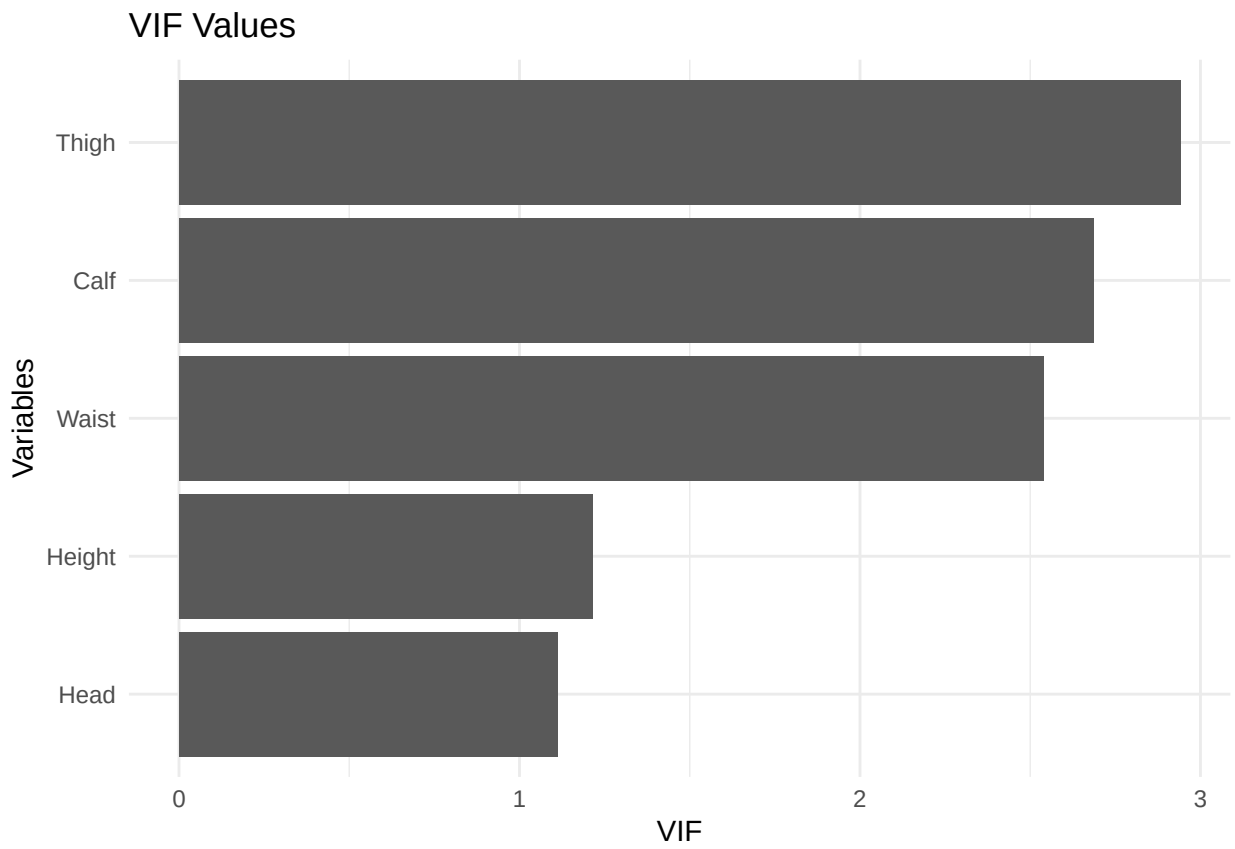

```r
model <- lm(Mass ~ ., data=pm_dat)
vif_values <- vif(model)
vif_df <- data.frame(Variable = names(vif_values), VIF = vif_values)
ggplot(vif_df, aes(x = reorder(Variable, VIF), y = VIF)) +
  geom_bar(stat = "identity") +
  coord_flip() +  # Makes it a horizontal bar plot
  labs(title = "VIF Values", x = "Variables", y = "VIF") +
  theme_minimal()
```




### multiple regression mode(AI choices based on the above information )

The variables 'Fore', 'Bicep', 'Chest', 'Neck', and 'Shoulder' have VIF values greater than 5, indicating potential multicollinearity. Given that 'Waist', 'Fore', 'Shoulder', 'Thigh', and 'Calf' have the highest absolute correlation with 'Mass', they might be the most informative predictors. However, due to multicollinearity concerns (as evidenced by high VIF), we might need to exclude some of them.(remove some variable with high VIF and retain high correlation predictors)

```
mt_model <- lm(Mass ~ Waist + Thigh + Calf + Height + Head, data = pm_dat)

vif_mtdf <- data.frame(Variable = names(vif(mt_model)), VIF = vif(mt_model))
ggplot(vif_mtdf, aes(x = reorder(Variable, VIF), y = VIF)) +
  geom_bar(stat = "identity") +
  coord_flip() +  # Makes it a horizontal bar plot
  labs(title = "VIF Values", x = "Variables", y = "VIF") +
  theme_minimal()
```




```
summary(mt_model)
```

```
##
## Call:
## lm(formula = Mass ~ Waist + Thigh + Calf + Height + Head, data = pm_dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7283 -1.6711  0.6758  1.4648  4.1414
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -83.5633    31.9510  -2.615   0.0187 *
## Waist         0.7650     0.1241   6.166 1.36e-05 ***
## Thigh         0.7004     0.2895   2.419   0.0278 *
## Calf          1.1001     0.4115   2.673   0.0167 *
## Height        0.2622     0.1079   2.431   0.0272 *
## Head         -0.5280     0.5093  -1.037   0.3152
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.781 on 16 degrees of freedom
## Multiple R-squared:  0.951,  Adjusted R-squared:  0.9357
## F-statistic: 62.08 on 5 and 16 DF,  p-value: 6.591e-10
```

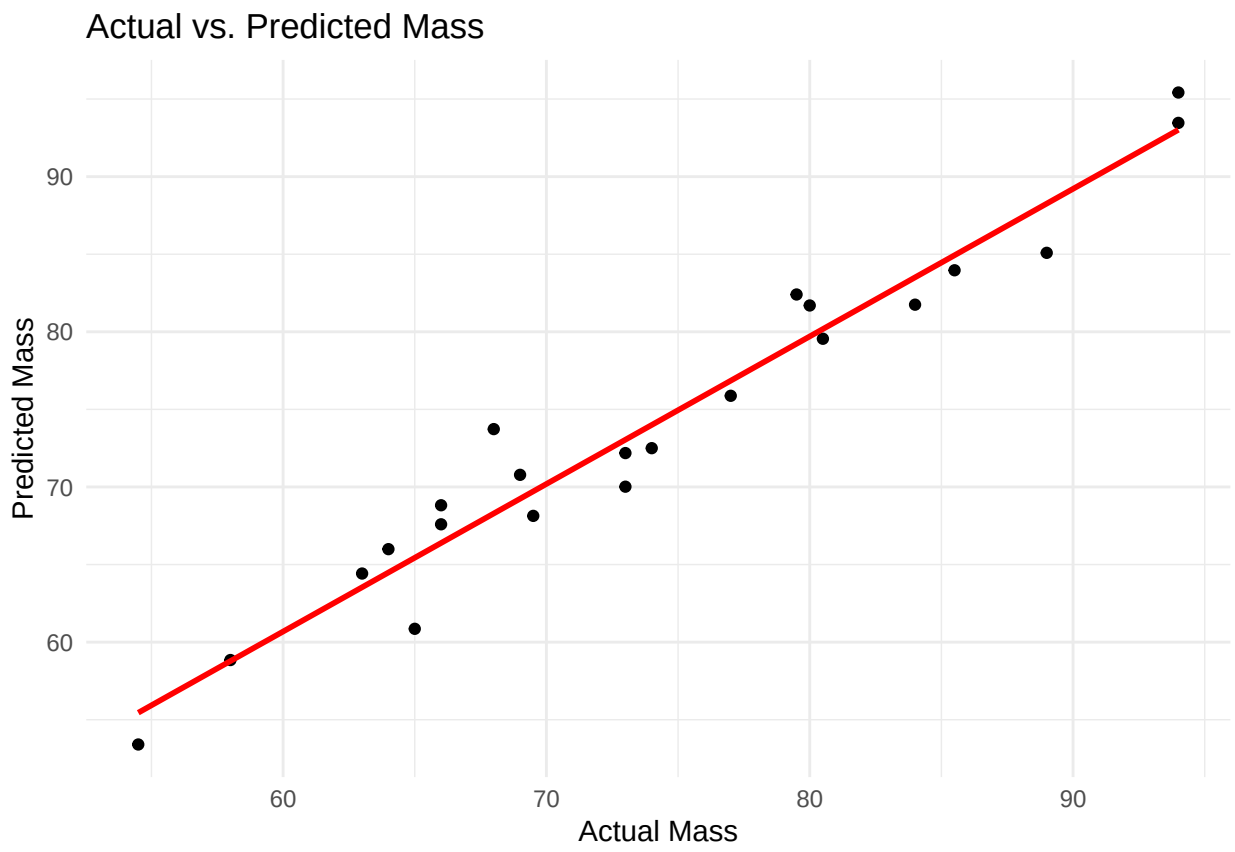
We note that all the VIF values are below 5, indicating that the model is unlikely to experience significant

multicollinearity. Additionally, the R-squared value implies a good fit for the model.

### The plot for this model

```
# Compute the predicted values
pm_dat$predicted_mass <- predict(mt_model, pm_dat)

# Plotting
ggplot(data = pm_dat, aes(x = Mass, y = predicted_mass)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "red") +  # Adds a regression line
  theme_minimal() +
  labs(title = "Actual vs. Predicted Mass",
       x = "Actual Mass",
       y = "Predicted Mass")
```

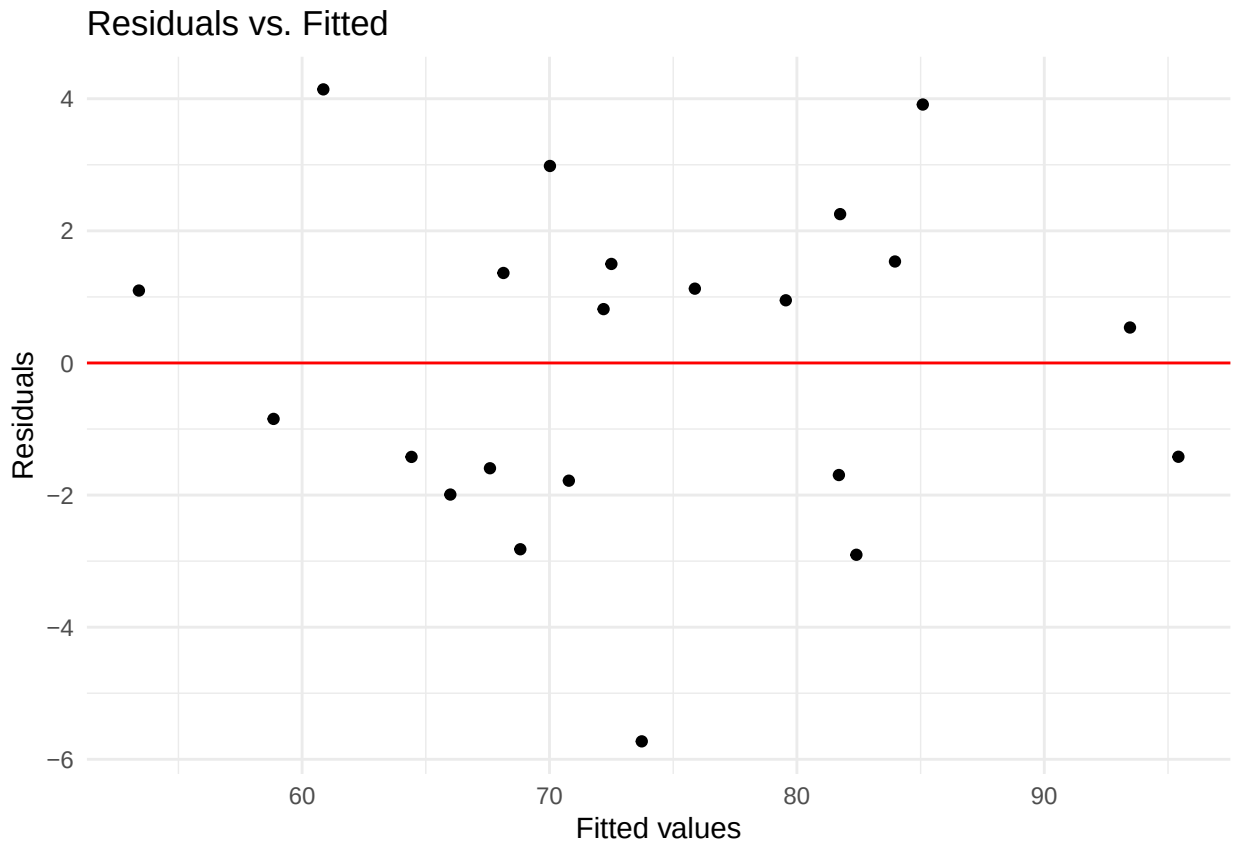
```
## `geom_smooth()` using formula = 'y ~ x'
```




### diagnostic plots

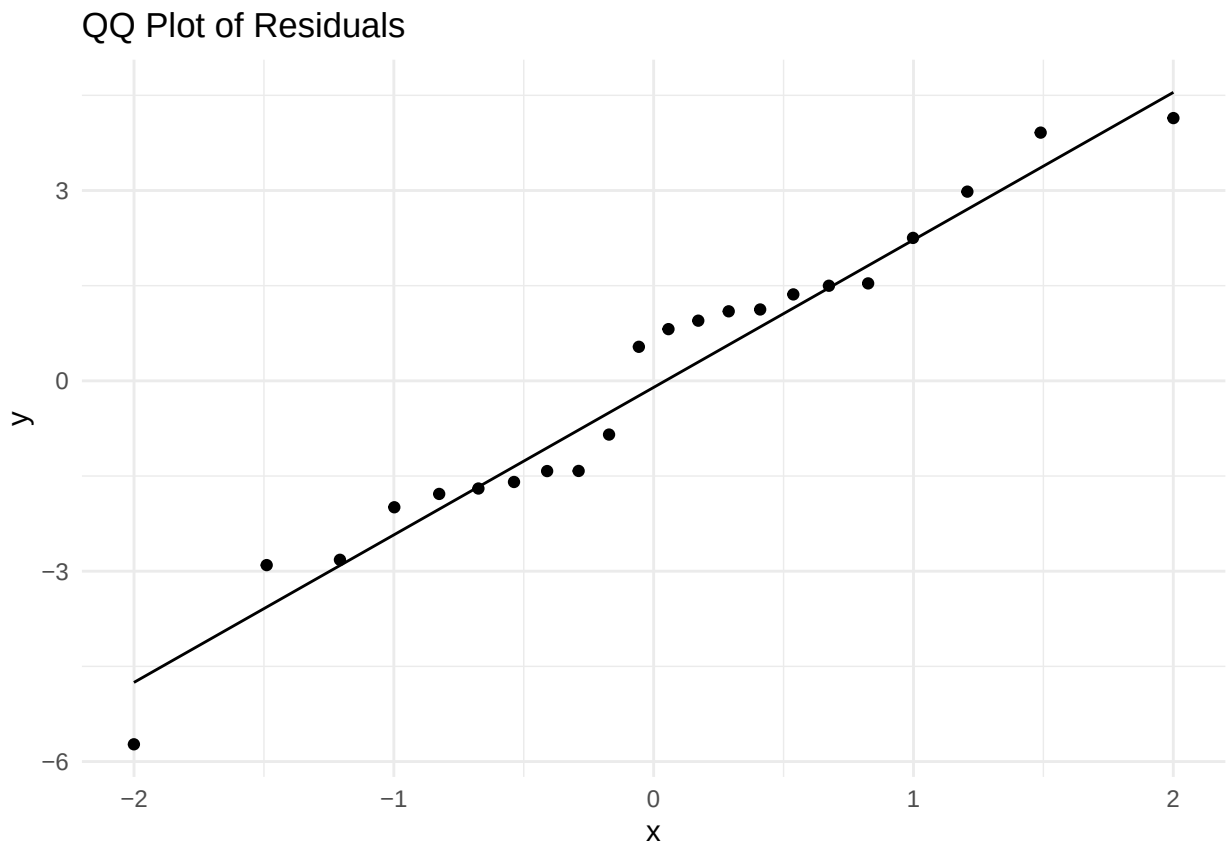
Residuals vs. Fitted plot:(homoscedasticity) of residuals.

```r
ggplot(data = pm_dat, aes(x = fitted(mt_model), y = residuals(mt_model))) +
  geom_point() +
  geom_hline(yintercept = 0, color = "red") +
  theme_minimal() +
  labs(title = "Residuals vs. Fitted",
       x = "Fitted values",
       y = "Residuals")
```



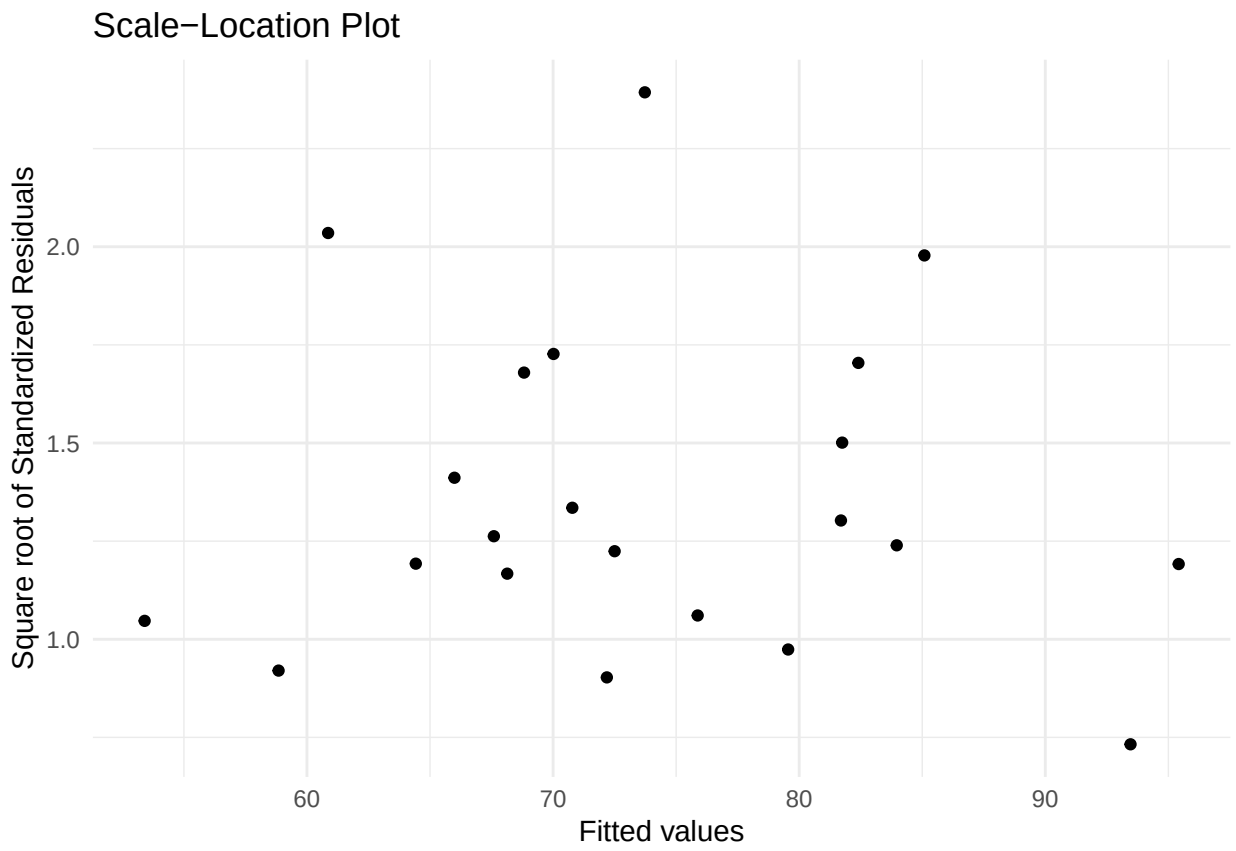

QQ plot: Checks the normality of residuals.

```r
# QQ plot
ggplot(data = pm_dat, aes(sample = residuals(mt_model))) +
  geom_qq() +
  geom_qq_line() +
  theme_minimal() +
  labs(title = "QQ Plot of Residuals")
```



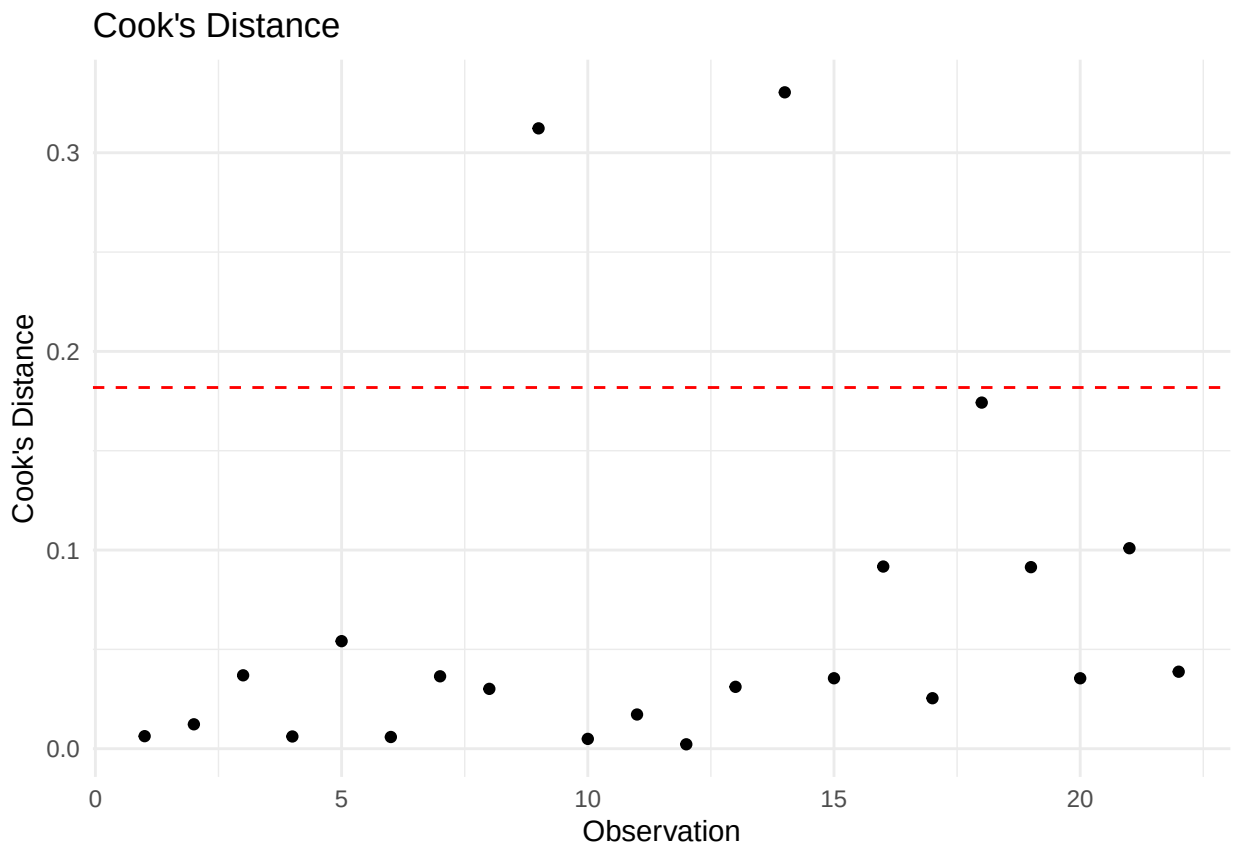

Scale-Location plot: Another way to check homoscedasticity.

```
# Scale-Location plot
ggplot(data = pm_dat, aes(x = fitted(mt_model), y = sqrt(abs(residuals(mt_model))))) +
  geom_point() +
  theme_minimal() +
  labs(title = "Scale-Location Plot",
       x = "Fitted values",
       y = "Square root of Standardized Residuals")
```



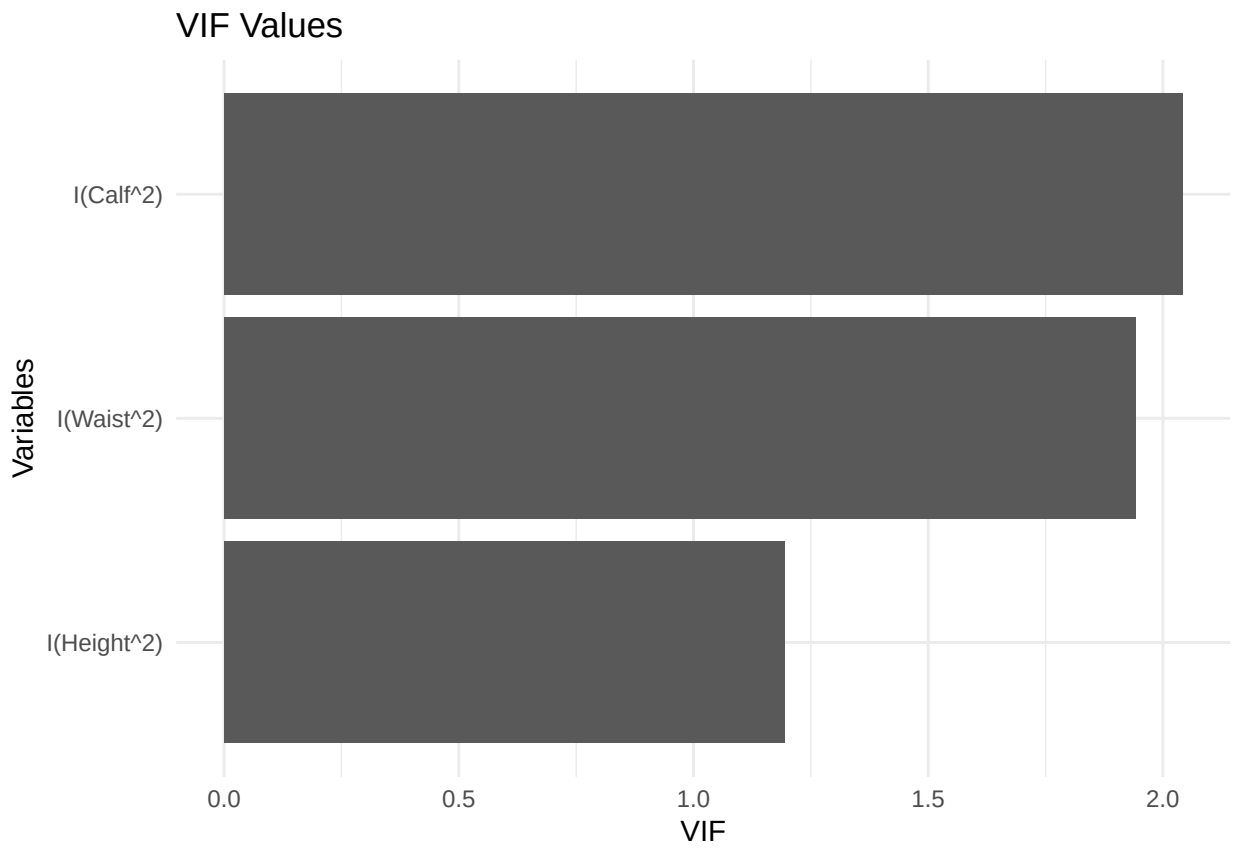

Cook's Distance: To detect influential observations.

```r
# Compute Cook's Distance
cooksd <- cooks.distance(mt_model)

# Cook's Distance plot
ggplot(data = as.data.frame(cooksd), aes(x = seq_along(cooksd), y = cooksd)) +
  geom_point() +
  geom_hline(yintercept = 4/nrow(pm_dat), linetype = "dashed", color = "red") +  # Threshold line
  theme_minimal() +
  labs(title = "Cook's Distance",
       x = "Observation",
       y = "Cook's Distance")
```




**polynomial regression model(AI choices based on the above information)**

```
poly_model<- lm(Mass ~ I(Waist^2) + I(Calf^2) + I(Height^2), data = pm_dat)


vif_pmdf <- data.frame(Variable = names(vif(poly_model)), VIF = vif(poly_model))
ggplot(vif_pmdf, aes(x = reorder(Variable, VIF), y = VIF)) +
  geom_bar(stat = "identity") +
  coord_flip() +  # Makes it a horizontal bar plot
  labs(title = "VIF Values", x = "Variables", y = "VIF") +
  theme_minimal()
```



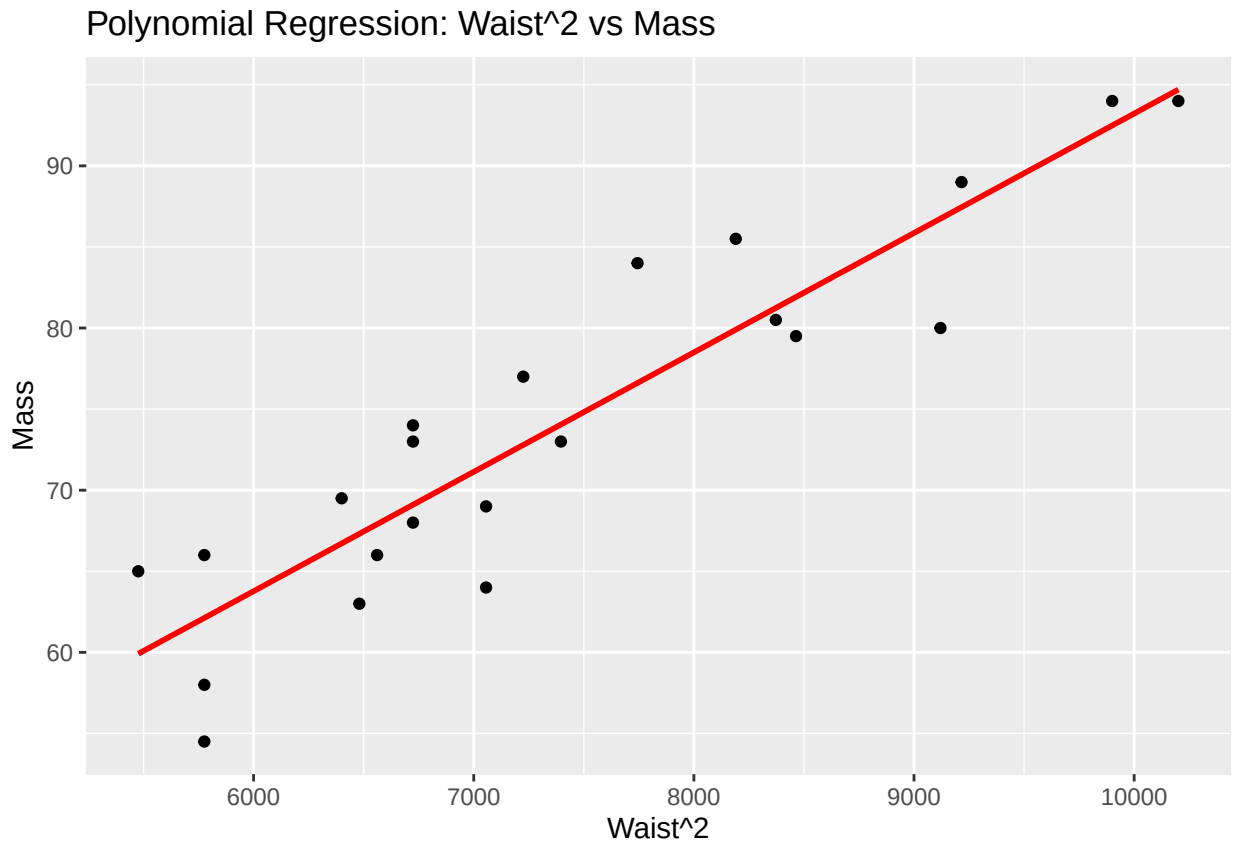

```
summary(poly_model)
```

```
##
## Call:
## lm(formula = Mass ~ I(Waist^2) + I(Calf^2) + I(Height^2), data = pm_dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9369 -1.7579 -0.0528  1.9572  5.8969
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.563e+01  9.823e+00  -1.591 0.128941
## I(Waist^2)   5.051e-03  6.841e-04   7.384 7.52e-07 ***
## I(Calf^2)    2.128e-02  5.365e-03   3.967 0.000904 ***
## I(Height^2)  7.053e-04  3.302e-04   2.136 0.046689 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.065 on 18 degrees of freedom
## Multiple R-squared:  0.933,  Adjusted R-squared:  0.9219
## F-statistic: 83.58 on 3 and 18 DF,  p-value: 9.268e-11
```

After careful examination of the Variance Inflation Factors (VIFs), it was determined that multicollinearity was not a concern for our model. Further assessment of the model's coefficients reinforced its robustness and validity. The findings from both these evaluations are promising and indicative of a well-fitted model

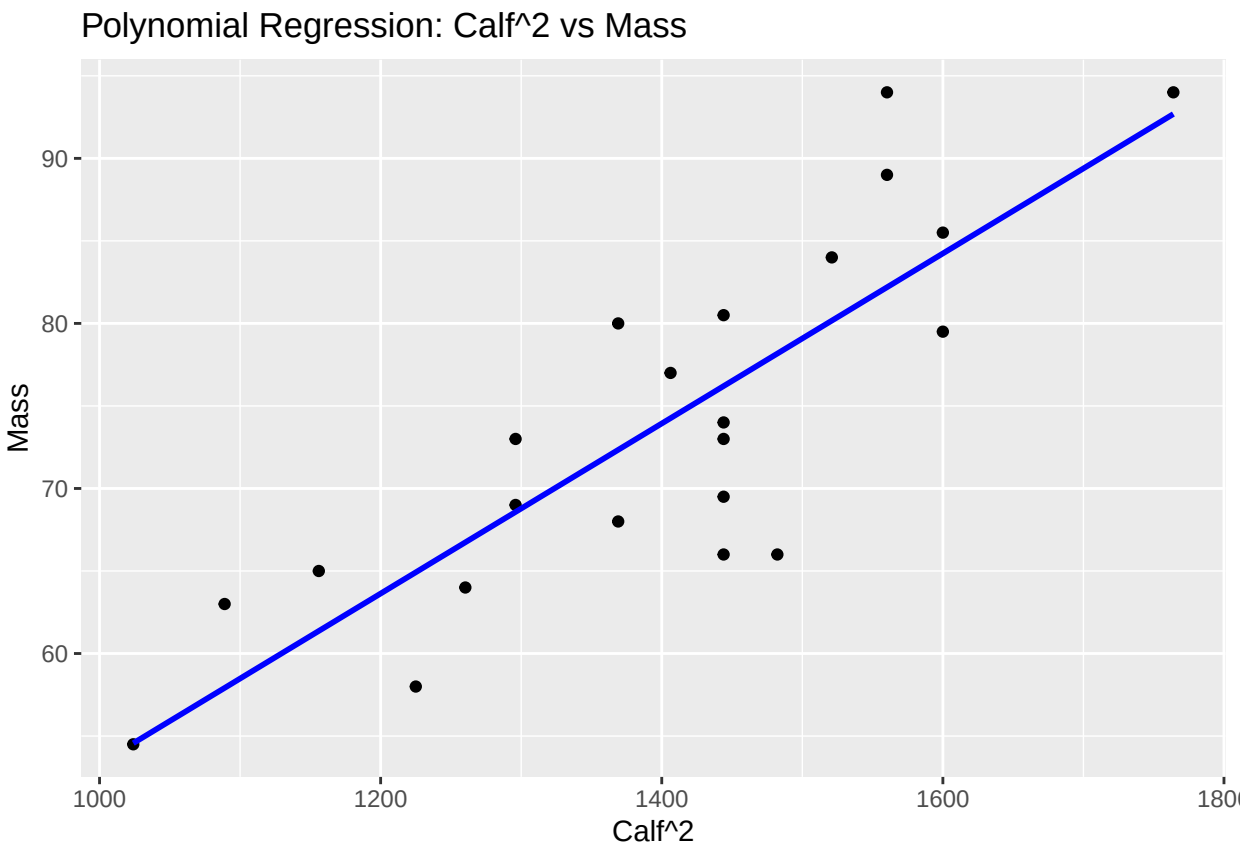
```
# 1. Plot for Waist^2 vs Mass
ggplot(pm_dat, aes(x=I(Waist^2), y=Mass)) +
  geom_point() +
  geom_smooth(method="lm", se=FALSE, color="red") +
  labs(title="Polynomial Regression: Waist^2 vs Mass",
       x="Waist^2", y="Mass")
```

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




```
# 2. Plot for Calf^2 vs Mass
ggplot(pm_dat, aes(x=I(Calf^2), y=Mass)) +
  geom_point() +
  geom_smooth(method="lm", se=FALSE, color="blue") +
  labs(title="Polynomial Regression: Calf^2 vs Mass",
       x="Calf^2", y="Mass")
```

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

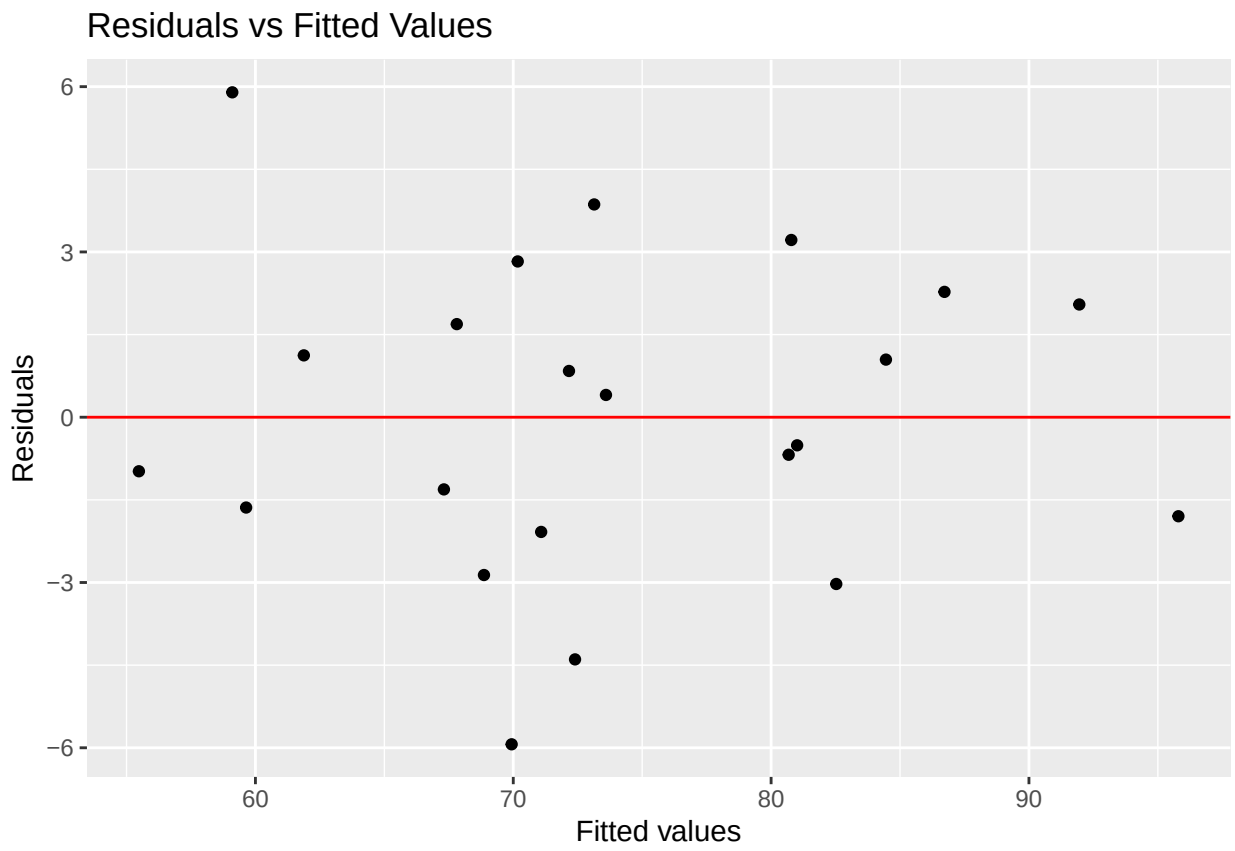



```
# 3. Plot for Height^2 vs Mass
ggplot(pm_dat, aes(x=I(Height^2), y=Mass)) +
  geom_point() +
  geom_smooth(method="lm", se=FALSE, color="green") +
  labs(title="Polynomial Regression: Height^2 vs Mass",
       x="Height^2", y="Mass")
```

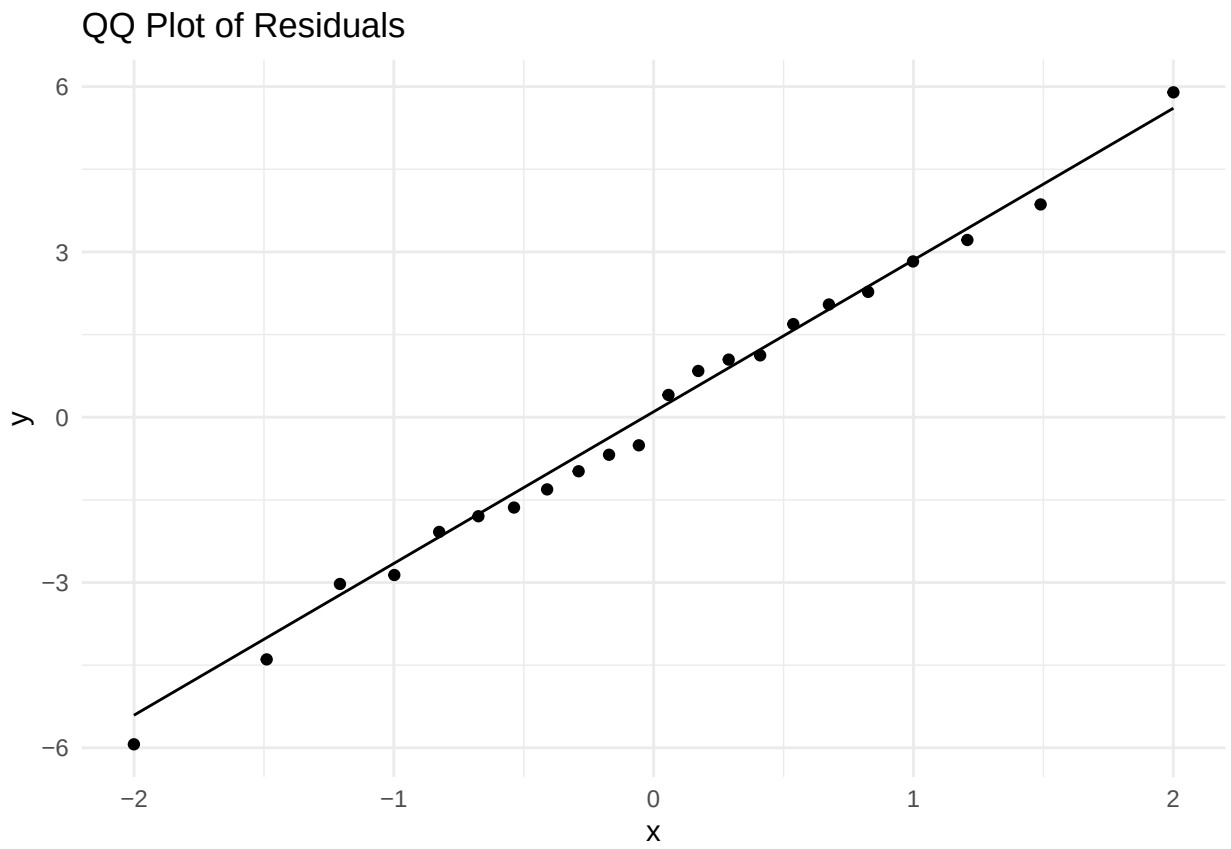
```
## `geom_smooth()` using formula = 'y ~ x'
```


### diagnostic plots

```
# Residuals vs Fitted values (for linearity)
df_plot <- data.frame(Fitted = poly_model$fitted.values, Residuals = poly_model$residuals)

# Use ggplot2 to plot
ggplot(df_plot, aes(x = Fitted, y = Residuals)) +
  geom_point() +                                    # Scatterplot points
  geom_hline(yintercept = 0, color = "red") +       # Horizontal line at y = 0
  labs(title = "Residuals vs Fitted Values",
       x = "Fitted values",
       y = "Residuals")
```




```
# Q-Q Plot (for normality)
ggplot(data = pm_dat, aes(sample = residuals(poly_model))) +
  geom_qq() +
  geom_qq_line() +
  theme_minimal() +
  labs(title = "QQ Plot of Residuals")
```




Check autocorrelation using Durbin-Watson Test

```
dwtest(poly_model)
```

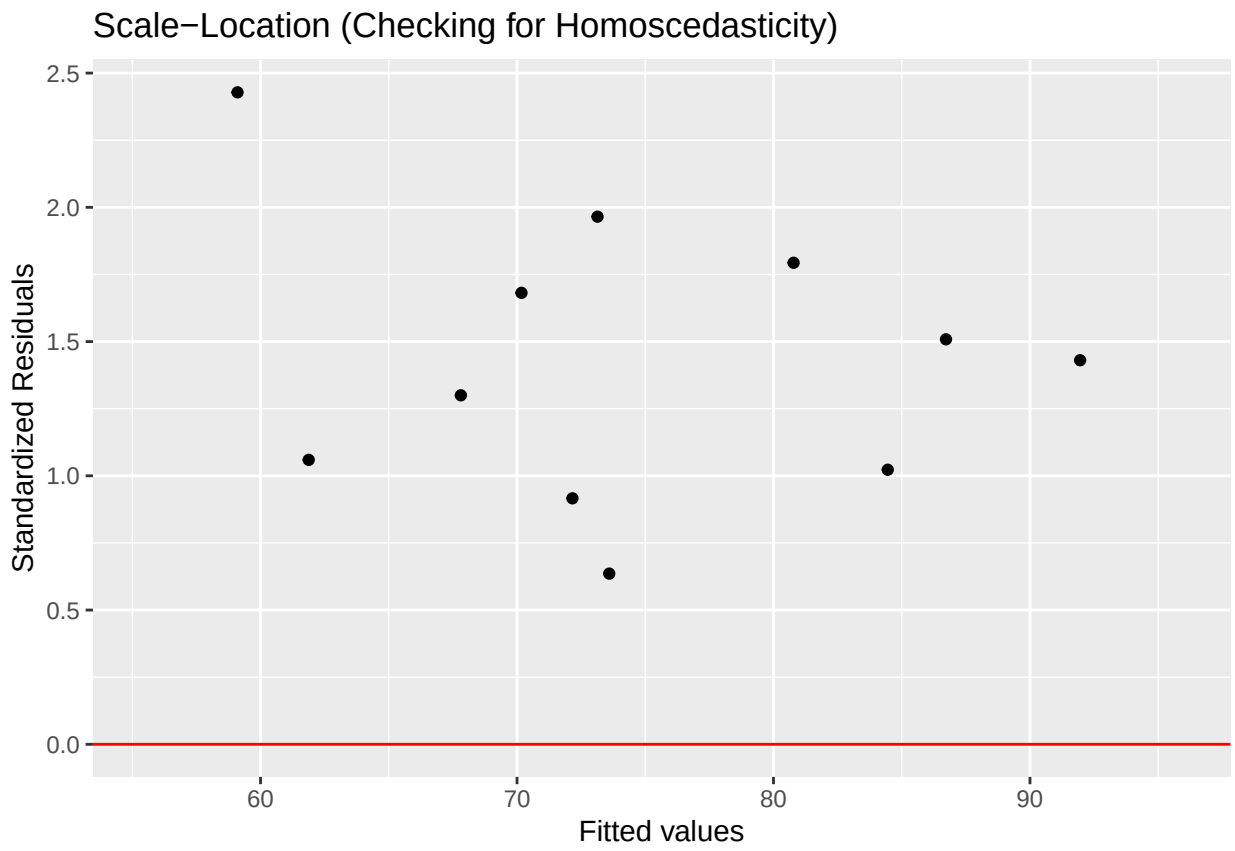
```
##
##  Durbin-Watson test
##
## data:  poly_model
## DW = 2.1148, p-value = 0.5935
## alternative hypothesis: true autocorrelation is greater than 0
```

```
df_homoscedasticity <- data.frame(Fitted = poly_model$fitted.values,
                                  StandardizedResiduals = abs(sqrt(poly_model$residuals)))
```

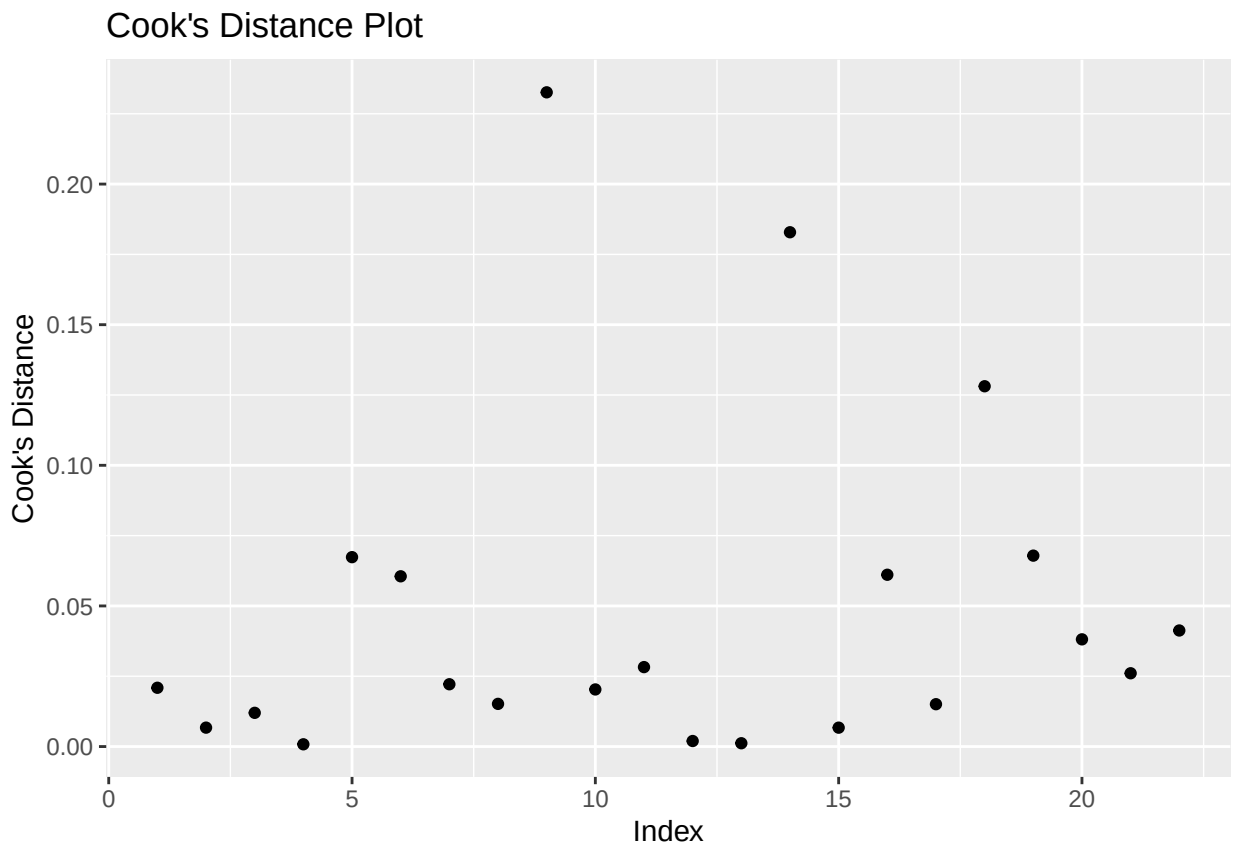
```
## Warning in sqrt(poly_model$residuals): NaNs produced
```

```
ggplot(df_homoscedasticity, aes(x = Fitted, y = StandardizedResiduals)) +
  geom_point() +
  labs(title = "Scale-Location (Checking for Homoscedasticity)",
       x = "Fitted values",
       y = "Standardized Residuals") +
  geom_hline(yintercept = 0, color = "red")
```

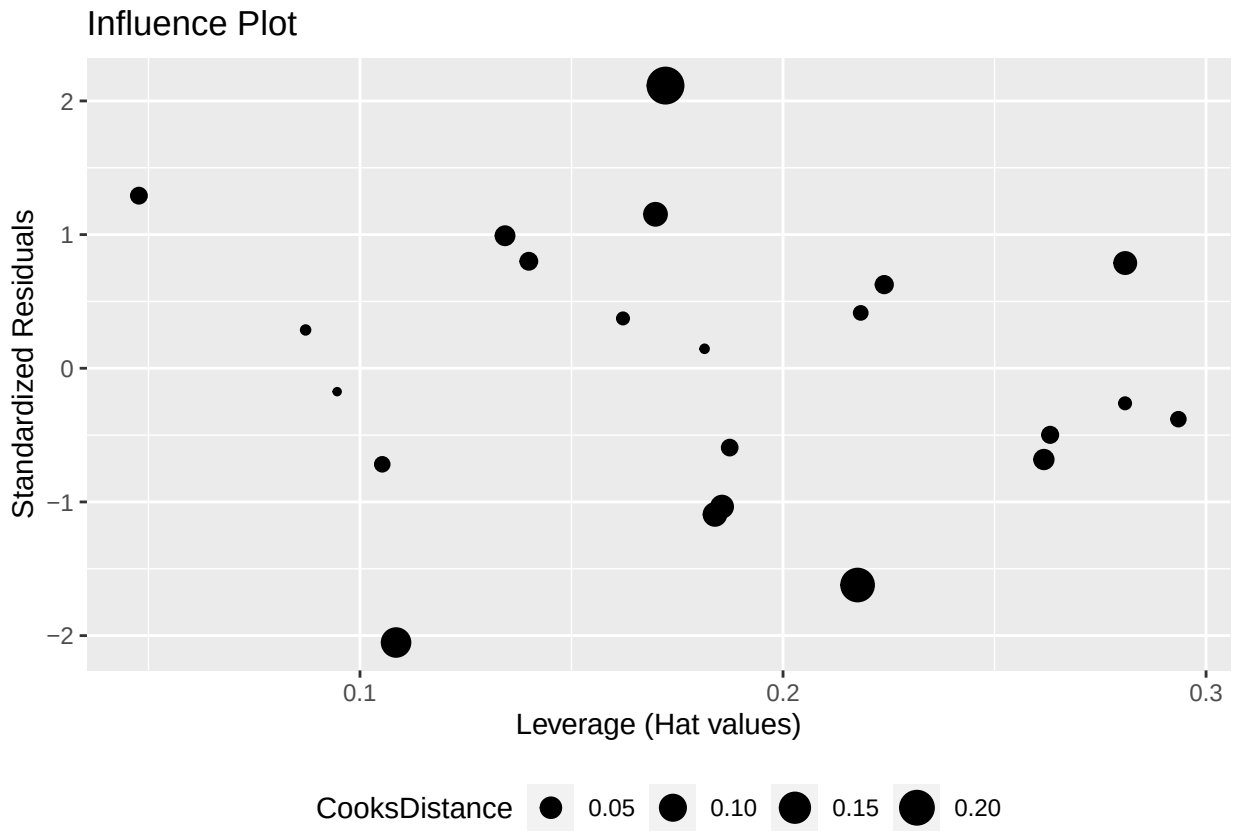
```
## Warning: Removed 11 rows containing missing values (‘geom_point()‘).
```




```r
cooks_d <- cooks.distance(poly_model)

df_cooks <- data.frame(index = 1:length(cooks_d),
                       CooksDistance = cooks_d)

ggplot(df_cooks, aes(x = index, y = CooksDistance)) +
  geom_point() +
  labs(title = "Cook's Distance Plot",
       x = "Index",
       y = "Cook's Distance")
```




```r
# Compute hat values (leverages) and standardized residuals
hat_values <- hatvalues(poly_model)
std_residuals <- rstandard(poly_model)

# Compute Cook's distance
cooks_d <- cooks.distance(poly_model)

df_influence <- data.frame(
  StandardizedResiduals = std_residuals,
  Hat = hat_values,
  CooksDistance = cooks_d
)

ggplot(df_influence, aes(x = Hat, y = StandardizedResiduals)) +
  geom_point(aes(size = CooksDistance)) +
  labs(
    title = "Influence Plot",
    x = "Leverage (Hat values)",
    y = "Standardized Residuals"
  ) +
  theme(legend.position = "bottom")
```

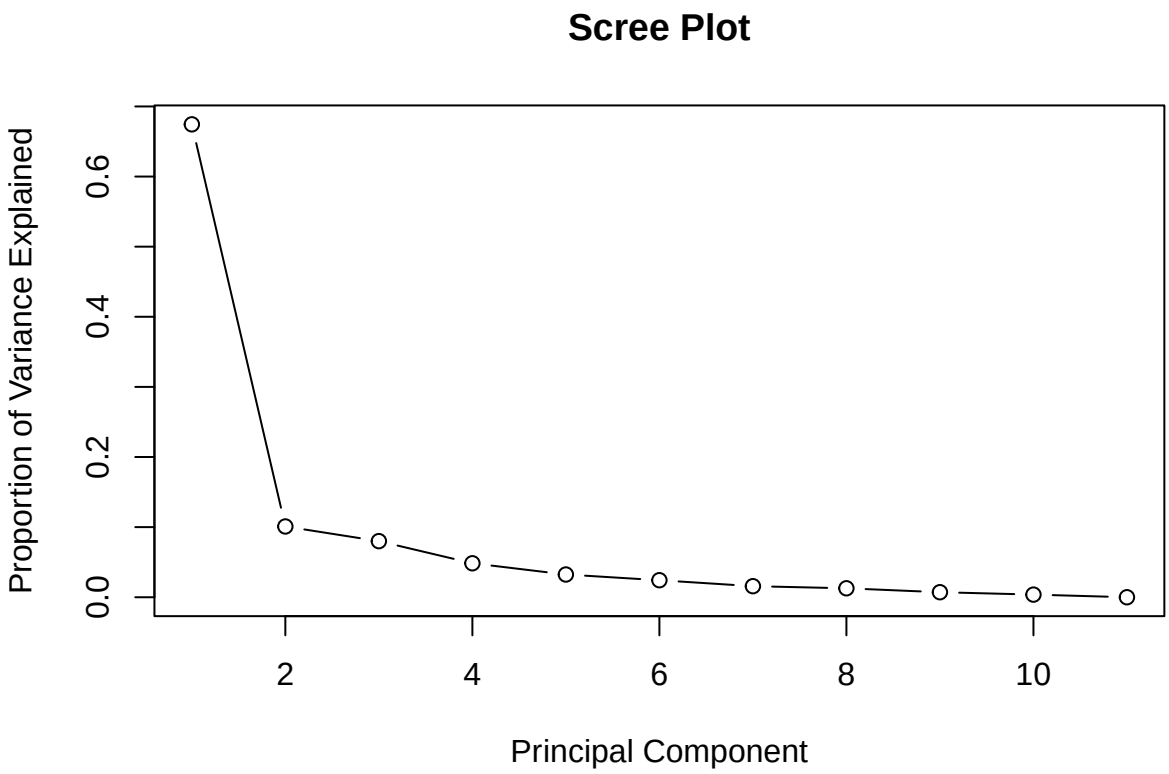



## PCA

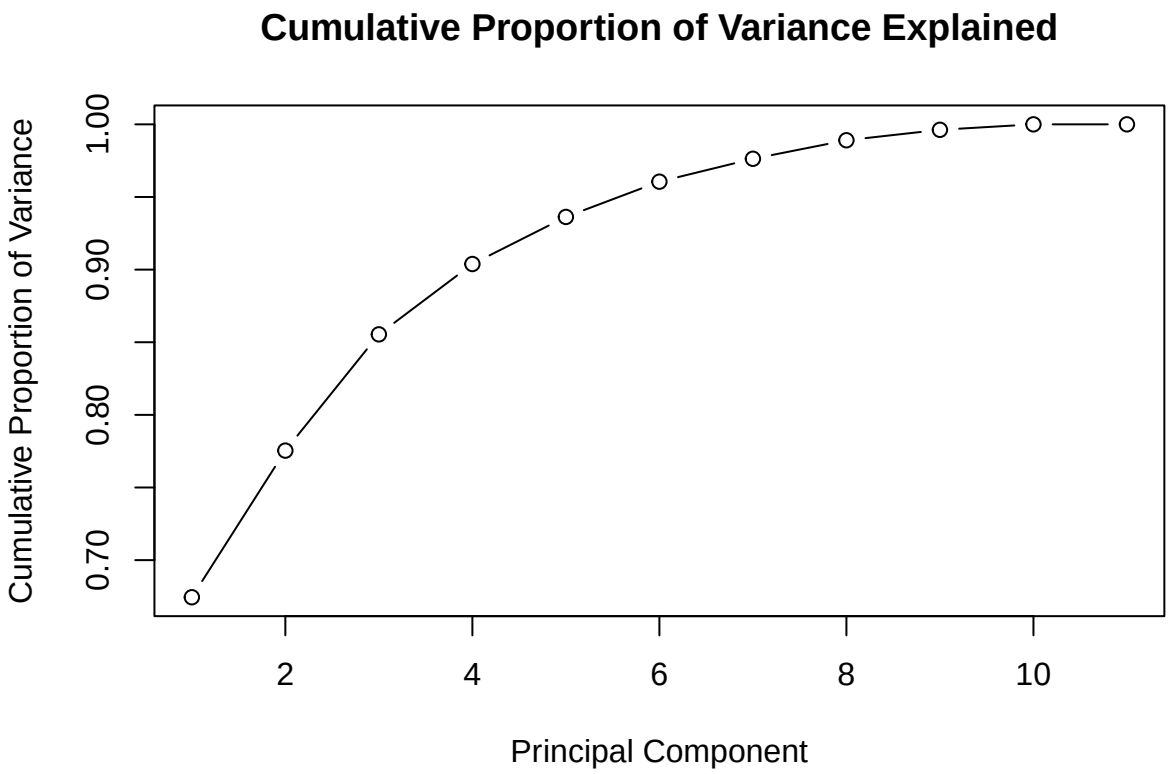
```
predictors <- scale(pm_dat[, -which(names(pm_dat) == "Mass")])
pca_result <- prcomp(predictors, center = TRUE, scale. = TRUE)
summary(pca_result)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.7238 1.0537 0.93819 0.72996 0.59709 0.51672 0.41643
## Proportion of Variance 0.6744 0.1009 0.08002 0.04844 0.03241 0.02427 0.01576
## Cumulative Proportion  0.6744 0.7754 0.85541 0.90385 0.93626 0.96053 0.97630
##                           PC8     PC9    PC10      PC11
## Standard deviation     0.37460 0.28104 0.20354 3.992e-16
## Proportion of Variance 0.01276 0.00718 0.00377 0.000e+00
## Cumulative Proportion  0.98905 0.99623 1.00000 1.000e+00
```

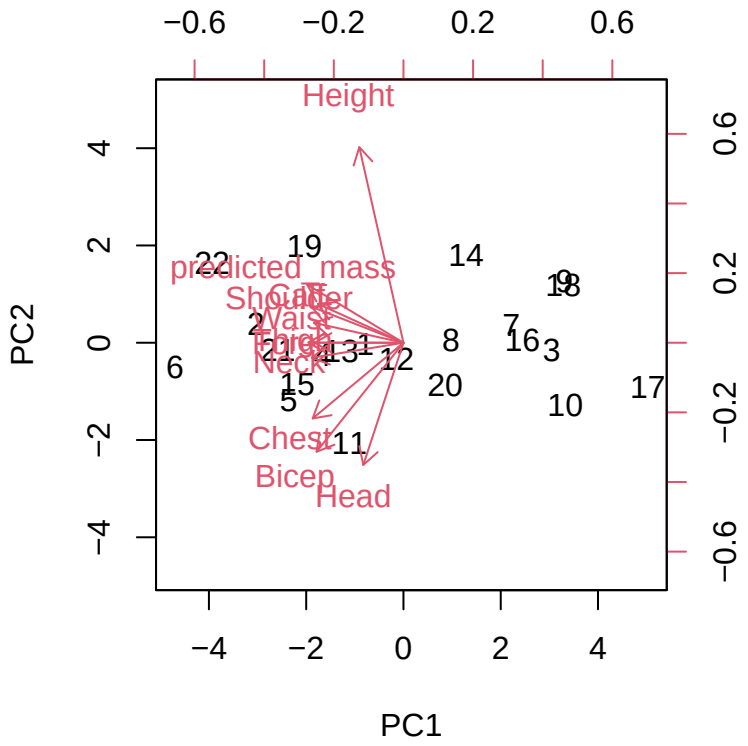
```
# Scree Plot
eigenvalues <- pca_result$sdev^2
proportion_variance <- eigenvalues / sum(eigenvalues)

plot(proportion_variance, type = "b", main="Scree Plot", xlab="Principal Component", ylab="Proportion o
```




```
cumulative_variance <- cumsum(proportion_variance)

plot(cumulative_variance, type = "b", main="Cumulative Proportion of Variance Explained", xlab="Principa
```




```
biplot(pca_result, scale=0)
```




Based on the plots, we think 4 components may be a good choice.

```
pc_data <- data.frame(Mass = pm_dat$Mass, PCA1 = pca_result$x[,1], PCA2 = pca_result$x[,2], PCA3 = pca_

model_with_pca <- lm(Mass ~ ., data = pc_data)
summary(model_with_pca)
```

```
##
## Call:
## lm(formula = Mass ~ ., data = pc_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.0668 -1.3381 -0.1453  1.4658  5.2052
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  73.9318     0.5251 140.801  < 2e-16 ***
## PCA1         -3.8443     0.1973 -19.483 4.59e-13 ***
## PCA2          1.9129     0.5100   3.751  0.00159 **
## PCA3          0.8925     0.5728   1.558  0.13764
## PCA4         -1.2869     0.7363  -1.748  0.09852 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.463 on 17 degrees of freedom
```

```
## Multiple R-squared:  0.9591, Adjusted R-squared:  0.9495
## F-statistic: 99.79 on 4 and 17 DF,  p-value: 1.435e-11
```

I found the pca3 and pca4 is not significant. Maybe we should choose two components.

```
pc_data_new <- data.frame(Mass = pm_dat$Mass, PCA1 = pca_result$x[,1], PCA2 = pca_result$x[,2])

model_with_pca_new <- lm(Mass ~ ., data = pc_data_new)
summary(model_with_pca_new)
```

```
##
## Call:
## lm(formula = Mass ~ ., data = pc_data_new)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7464 -1.1345 -0.0024  1.3932  5.3977
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  73.9318     0.5712 129.438  < 2e-16 ***
## PCA1         -3.8443     0.2146 -17.911 2.34e-13 ***
## PCA2          1.9129     0.5548   3.448   0.0027 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.679 on 19 degrees of freedom
## Multiple R-squared:  0.946,  Adjusted R-squared:  0.9403
## F-statistic: 166.3 on 2 and 19 DF,  p-value: 9.115e-13
```

Due to the limitations in the size of the dataset, the principal component analysis model we built is highly susceptible to overfitting. Moreover, because of the small sample size of the dataset, it's challenging for us to employ cross-validation to verify the model's fit.