STA302 Final Project

1. Data Import & Cleaning

We start by loading the raw NBA data and inspecting its first rows.

```
# Read nba_final.csv
nba <- read.csv("nba_final.csv", stringsAsFactors = FALSE)
head(nba)
```

```
##    Rk        Player.x Player_ID Pos1 Pos2 Age  Tm  G GS   MP  FG  FGA   FG. X3P
## 1 170    A.J. Hammons hammoaj01    C <NA>  24 DAL 22  0  7.4 0.8  1.9 0.405 0.2
## 2  58    Aaron Brooks brookaa01   PG <NA>  32 IND 65  0 13.8 1.9  4.6 0.403 0.7
## 3 157    Aaron Gordon gordoaa01   SF <NA>  21 ORL 80 72 28.7 4.9 10.8 0.454 1.0
## 4 352   Adreian Payne paynead01   PF <NA>  25 MIN 18  0  7.5 1.3  3.0 0.426 0.2
## 5  10 Al-Farouq Aminu aminual01   PF <NA>  26 POR 61 25 29.1 3.0  7.6 0.393 1.1
## 6 203      Al Horford horfoal01    C <NA>  30 BOS 68 68 32.3 5.6 11.8 0.473 1.3
##   X3PA  X3P. X2P X2PA  X2P.  eFG.  FT FTA   FT. ORB DRB TRB AST STL BLK TOV  PF
## 1  0.5 0.500 0.5  1.5 0.375 0.464 0.4 0.9 0.450 0.4 1.3 1.6 0.2 0.0 0.6 0.5 1.0
## 2  2.0 0.375 1.1  2.6 0.424 0.483 0.5 0.6 0.800 0.3 0.8 1.1 1.9 0.4 0.1 1.0 1.4
## 3  3.3 0.288 4.0  7.5 0.528 0.499 2.0 2.7 0.719 1.5 3.6 5.1 1.9 0.8 0.5 1.1 2.2
## 4  0.8 0.200 1.1  2.2 0.513 0.454 0.8 1.1 0.737 0.5 1.3 1.8 0.4 0.4 0.4 0.4 1.8
## 5  3.5 0.330 1.9  4.2 0.445 0.468 1.6 2.2 0.706 1.3 6.1 7.4 1.6 1.0 0.7 1.5 1.7
## 6  3.6 0.355 4.3  8.2 0.524 0.527 1.6 2.0 0.800 1.4 5.4 6.8 5.0 0.8 1.3 1.7 2.0
##    PTS   Salary mean_views  Season Conference  Role  Fvot FRank Pvot PRank Mvot
## 1  2.2       NA    3.32000 2016-17       West Front   786   123   NA    NA   NA
## 2  5.0  2700000   11.15574 2016-17        Est  Back  2474    64   NA    NA   NA
## 3 12.7  4351320 1713.98634 2016-17        Est Front 22774    29   NA    NA   NA
## 4  3.5  2022240  205.85519 2016-17       West Front   861   120    1    52   NA
## 5  8.7  7680965  604.34153 2016-17       West Front  4971    69    7    23   NA
## 6 14.0 26540100 1556.38251 2016-17        Est Front 12219    14    8    16    2
##   MRank Score Play
## 1    NA  83.5   No
## 2    NA  48.2   No
## 3    NA  40.0   No
## 4    NA  75.5   No
## 5    NA  42.8   No
## 6     6  12.5   No
```

```
## Initial sample size (n): 1408
```

Next, we compute a table showing how many NAs each variable has.

```
# Compute missing data summary table
missing_summary <- data.frame(
  Variable = names(nba),
```

```
  Missing_Count = sapply(nba, function(x) sum(is.na(x))),
  Total_Count = nrow(nba)
)
missing_summary$Missing_Percent <- round(100 * missing_summary$Missing_Count / missing_summary$Total_Cou
missing_summary <- missing_summary[missing_summary$Missing_Count > 0, ]
missing_summary <- missing_summary[order(-missing_summary$Missing_Count), ]

# Print the missing data summary
knitr::kable(missing_summary, caption = "Missing Data Summary (sorted by count)")
```

Table 1: Missing Data Summary (sorted by count)

	Variable	Missing_Count	Total_Count	Missing_Percent
Pos2	Pos2	1396	1408	99.15
Mvot	Mvot	404	1408	28.69
MRank	MRank	404	1408	28.69
Pvot	Pvot	159	1408	11.29
PRank	PRank	159	1408	11.29
mean_views	mean_views	138	1408	9.80
X3P.	X3P.	99	1408	7.03
Salary	Salary	62	1408	4.40
FT.	FT.	47	1408	3.34
X2P.	X2P.	15	1408	1.07
FG.	FG.	4	1408	0.28
eFG.	eFG.	4	1408	0.28

We found that Pos2 had too many missing values (99.15%), so we dropped it. We also removed 62 rows (4.40%) with missing Salary values, but kept all other variables since they had reasonable amounts of missing data (28.69%).

```
# Check for missing values using base R
missing_cols <- colSums(is.na(nba))

# Drop variables with > 90% missing values
missing_threshold <- 0.9 * nrow(nba)
high_missing_cols <- names(missing_cols[missing_cols > missing_threshold])
cat("Columns with >90% missing values:", paste(high_missing_cols, collapse=", "), "\n")
```

```
## Columns with >90% missing values: Pos2
```

```
if (length(high_missing_cols) > 0) {
  nba <- nba[, !names(nba) %in% high_missing_cols]
}

# Convert categorical variables to factors
nba$Pos1 <- as.factor(nba$Pos1)
nba$Conference <- as.factor(nba$Conference)
nba$Role <- as.factor(nba$Role)
nba$Play <- as.factor(nba$Play)

# Remove rows with missing Salary
```

```r
nba <- nba[!is.na(nba$Salary), ]

# Show table of remaining NAs and a glimpse summary
remaining_nas <- colSums(is.na(nba))
na_df <- data.frame(
  Variable = names(remaining_nas[remaining_nas > 0]),
  Missing_Count = remaining_nas[remaining_nas > 0],
  Percentage = round(100 * remaining_nas[remaining_nas > 0] / nrow(nba), 2)
)
knitr::kable(na_df, caption = "Remaining missing values")
```

Table 2: Remaining missing values

	Variable	Missing_Count	Percentage
FG.	FG.	4	0.30
X3P.	X3P.	97	7.21
X2P.	X2P.	13	0.97
eFG.	eFG.	4	0.30
FT.	FT.	43	3.19
mean_views	mean_views	127	9.44
Pvot	Pvot	149	11.07
PRank	PRank	149	11.07
Mvot	Mvot	383	28.45
MRank	MRank	383	28.45

```r
# Glimpse of the data using dplyr
glimpse(nba)
```

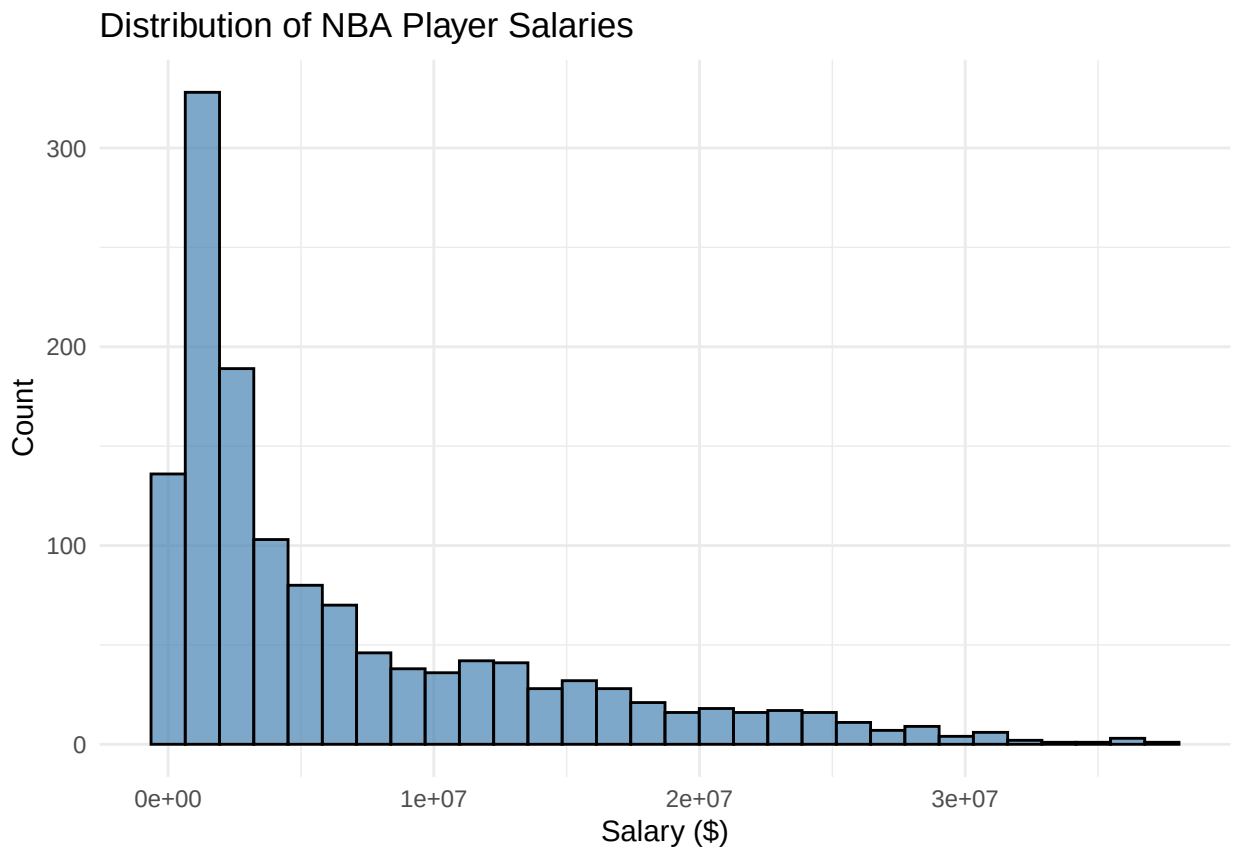
```
## Rows: 1,346
## Columns: 44
## $ Rk         <int> 58, 157, 352, 10, 203, 221, 12, 464, 65, 1, 260, 5, 92, 149~
## $ Player.x   <chr> "Aaron Brooks", "Aaron Gordon", "Adreian Payne", "Al-Farouq~
## $ Player_ID  <chr> "brookaa01", "gordoaa01", "paynead01", "aminual01", "horfoa~
## $ Pos1       <fct> PG, SF, PF, PF, C, C, SF, C, SG, SG, C, C, SG, SF, PF, C, C~
## $ Age        <int> 32, 21, 25, 26, 30, 32, 34, 24, 25, 23, 23, 28, 24, 29, 29,~
## $ Tm         <chr> "IND", "ORL", "MIN", "POR", "BOS", "IND", "LAC", "PHO", "UT~
## $ G          <int> 65, 80, 18, 61, 68, 66, 30, 47, 42, 68, 77, 39, 79, 13, 80,~
## $ GS         <int> 0, 72, 0, 25, 68, 1, 0, 0, 0, 6, 34, 15, 7, 0, 77, 1, 81, 0~
## $ MP         <dbl> 13.8, 28.7, 7.5, 29.1, 32.3, 14.1, 10.3, 15.1, 15.5, 15.5, ~
## $ FG         <dbl> 1.9, 4.9, 1.3, 3.0, 5.6, 3.6, 1.0, 2.9, 2.4, 2.0, 3.0, 2.3,~
## $ FGA        <dbl> 4.6, 10.8, 3.0, 7.6, 11.8, 7.1, 2.7, 5.7, 5.9, 5.0, 6.0, 4.~
## $ FG.        <dbl> 0.403, 0.454, 0.426, 0.393, 0.473, 0.499, 0.375, 0.517, 0.3~
## $ X3P        <dbl> 0.7, 1.0, 0.2, 1.1, 1.3, 0.0, 0.5, 0.0, 0.6, 1.4, 0.0, 0.0,~
## $ X3PA       <dbl> 2.0, 3.3, 0.8, 3.5, 3.6, 0.0, 1.5, 0.0, 1.8, 3.6, 0.2, 0.1,~
## $ X3P.       <dbl> 0.375, 0.288, 0.200, 0.330, 0.355, 0.000, 0.318, 0.000, 0.3~
## $ X2P        <dbl> 1.1, 4.0, 1.1, 1.9, 4.3, 3.6, 0.5, 2.9, 1.8, 0.6, 2.9, 2.3,~
## $ X2PA       <dbl> 2.6, 7.5, 2.2, 4.2, 8.2, 7.1, 1.2, 5.7, 4.1, 1.4, 5.9, 4.5,~
## $ X2P.       <dbl> 0.424, 0.528, 0.513, 0.445, 0.524, 0.500, 0.444, 0.519, 0.4~
## $ eFG.       <dbl> 0.483, 0.499, 0.454, 0.468, 0.527, 0.499, 0.463, 0.517, 0.4~
## $ FT         <dbl> 0.5, 2.0, 0.8, 1.6, 1.6, 1.0, 0.4, 1.5, 1.4, 0.6, 1.9, 0.7,~
## $ FTA        <dbl> 0.6, 2.7, 1.1, 2.2, 2.0, 1.3, 0.5, 2.4, 1.9, 0.7, 2.7, 1.0,~
```

```
## $ FT.         <dbl> 0.800, 0.719, 0.737, 0.706, 0.800, 0.765, 0.750, 0.625, 0.7~
## $ ORB         <dbl> 0.3, 1.5, 0.5, 1.3, 1.4, 1.1, 0.1, 2.0, 0.4, 0.3, 2.0, 1.2,~
## $ DRB         <dbl> 0.8, 3.6, 1.3, 6.1, 5.4, 3.1, 0.7, 4.2, 2.5, 1.0, 4.6, 3.4,~
## $ TRB         <dbl> 1.1, 5.1, 1.8, 7.4, 6.8, 4.2, 0.8, 6.2, 2.9, 1.3, 6.6, 4.5,~
## $ AST         <dbl> 1.9, 1.9, 0.4, 1.6, 5.0, 0.9, 0.4, 0.5, 0.7, 0.6, 0.6, 0.3,~
## $ STL         <dbl> 0.4, 0.8, 0.4, 1.0, 0.8, 0.3, 0.1, 0.6, 0.4, 0.5, 0.5, 0.5,~
## $ BLK         <dbl> 0.1, 0.5, 0.4, 0.7, 1.3, 0.2, 0.0, 0.7, 0.1, 0.1, 1.3, 0.6,~
## $ TOV         <dbl> 1.0, 1.1, 0.4, 1.5, 1.7, 0.5, 0.2, 0.8, 0.8, 0.5, 1.3, 0.8,~
## $ PF          <dbl> 1.4, 2.2, 1.8, 1.7, 2.0, 1.9, 1.2, 2.7, 1.2, 1.7, 3.1, 2.0,~
## $ PTS         <dbl> 5.0, 12.7, 3.5, 8.7, 14.0, 8.1, 2.9, 7.4, 6.7, 6.0, 8.0, 5.~
## $ Salary      <int> 2700000, 4351320, 2022240, 7680965, 26540100, 10230179, 131~
## $ mean_views <dbl> 11.155738, 1713.986339, 205.855191, 604.341530, 1556.382514~
## $ Season      <chr> "2016-17", "2016-17", "2016-17", "2016-17", "2016-17", "201~
## $ Conference <fct> Est, Est, West, West, Est, Est, West, West, West, West, Wes~
## $ Role        <fct> Back, Front, Front, Front, Front, Front, Front, Front, Back~
## $ Fvot        <int> 2474, 22774, 861, 4971, 12219, 2936, 3096, 607, 1981, 16299~
## $ FRank       <int> 64, 29, 120, 69, 14, 84, 88, 127, 72, 34, 109, 80, 65, 126,~
## $ Pvot        <int> NA, NA, 1, 7, 8, 1, 1, NA, NA, 1, 1, NA, NA, NA, 1, NA, 30,~
## $ PRank       <int> NA, NA, 52, 23, 16, 60, 52, NA, NA, 36, 52, NA, NA, NA, 60,~
## $ Mvot        <int> NA, NA, NA, NA, 2, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ MRank       <int> NA, NA, NA, NA, 6, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ Score       <dbl> 48.2, 40.0, 75.5, 42.8, 12.5, 60.0, 59.5, 85.5, 53.0, 27.5,~
## $ Play        <fct> No, No, No, No, No, No, No, No, No, No, No, No, No, No, No,~
```

2. Exploratory Analysis

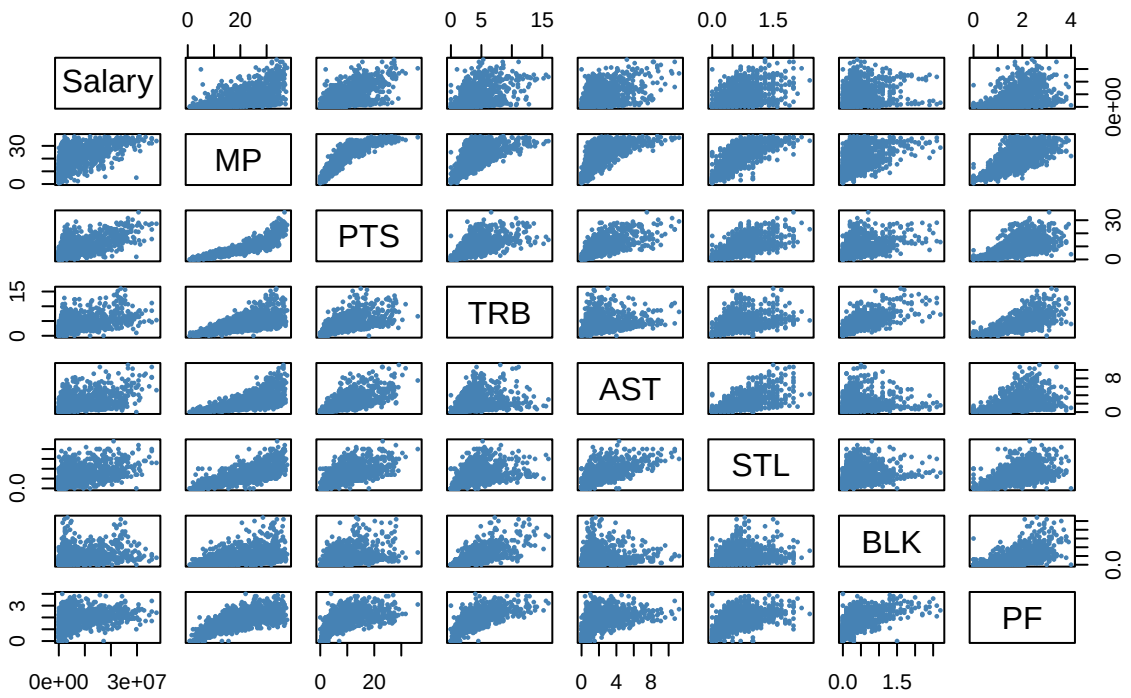
Let's first look at the distribution of player salaries.

```
# Histogram of Salary
ggplot(nba, aes(x = Salary)) +
  geom_histogram(bins = 30, fill = "steelblue", color = "black", alpha = 0.7) +
  labs(title = "Distribution of NBA Player Salaries", x = "Salary ($)", y = "Count") +
  theme_minimal()
```



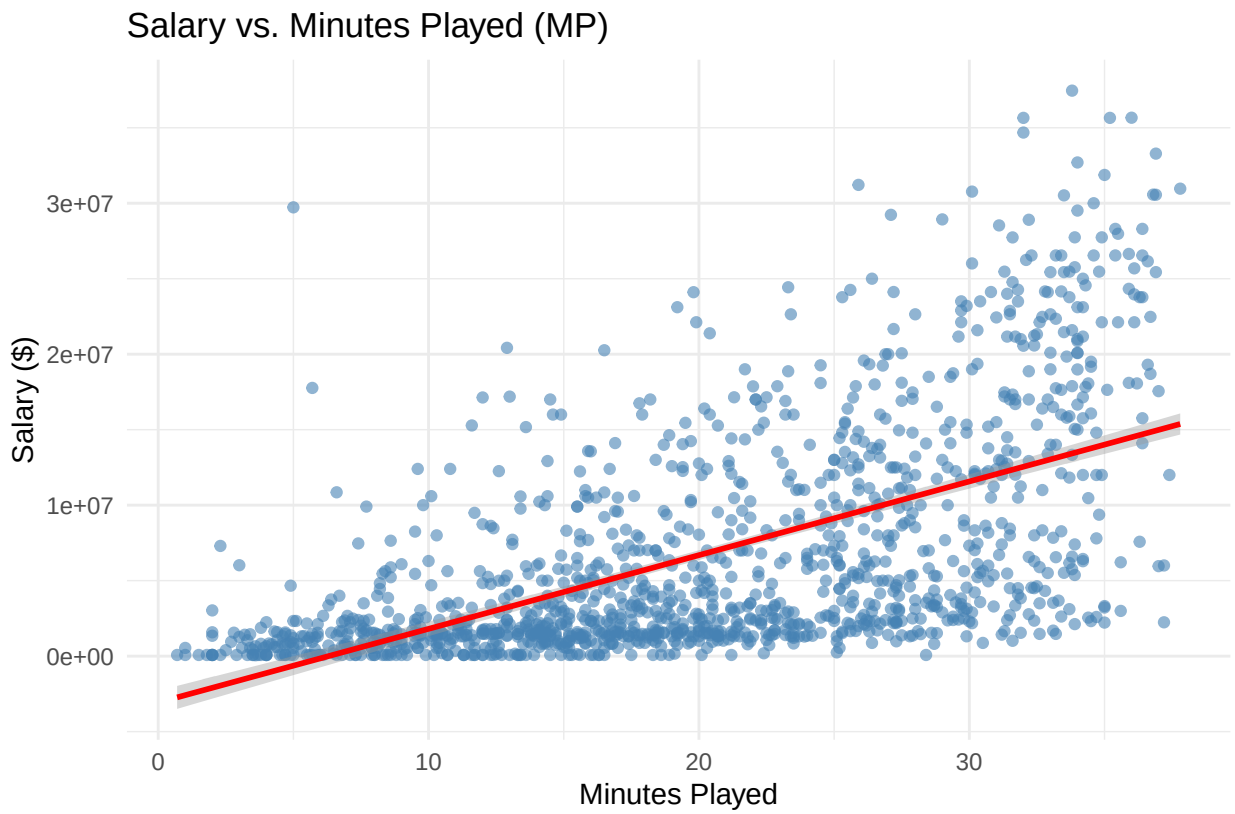
To get a quick overview of relationships between variables, we create a scatterplot matrix of our core numeric variables.

```
# Create pairs scatterplot matrix of core numeric variables
core_vars <- c("Salary", "MP", "PTS", "TRB", "AST", "STL", "BLK", "PF")
core_vars <- core_vars[core_vars %in% names(nba)]
pairs(nba[, core_vars],
      main = "Scatterplot Matrix of Core Variables",
      pch = 16,
      col = "steelblue",
      cex = 0.5)
```

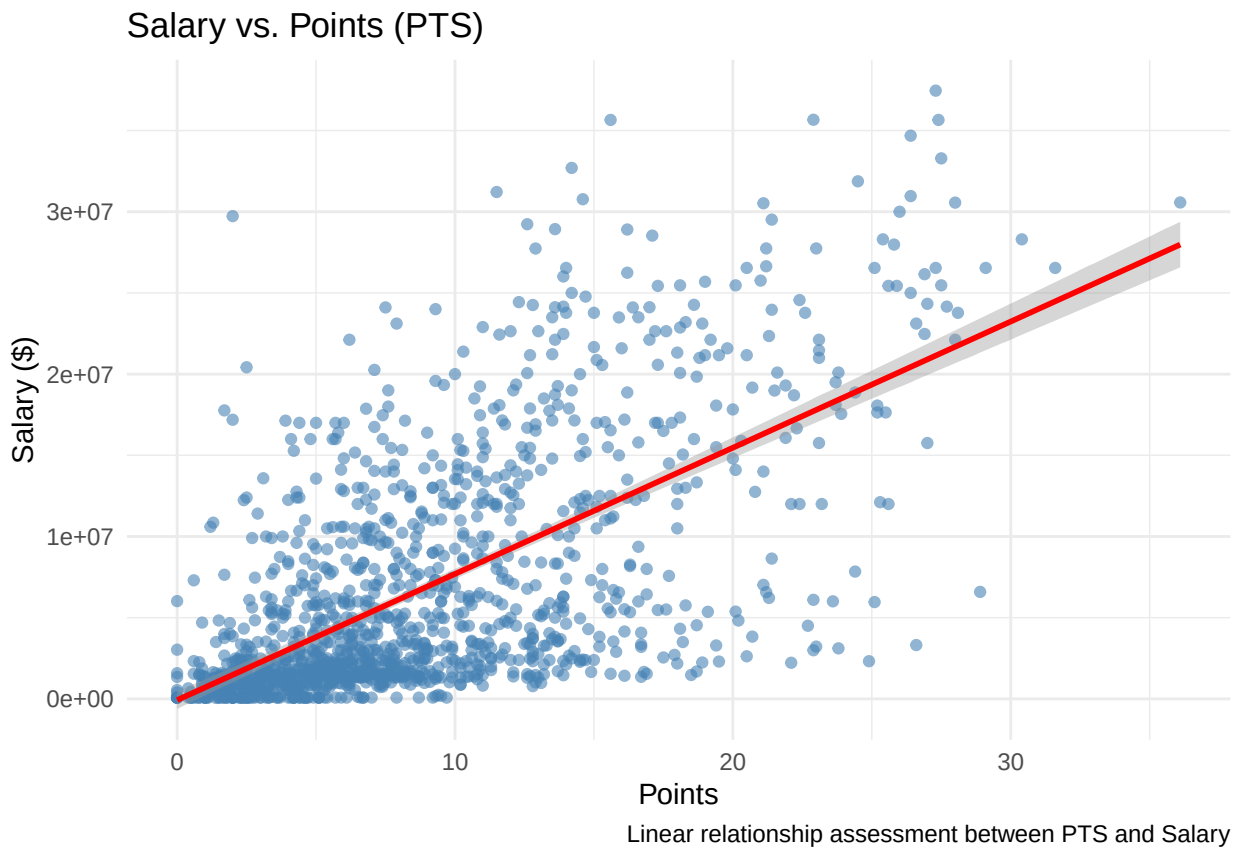


We want to examine how each key stat relates to player salary, so we create individual scatterplots with trend lines.

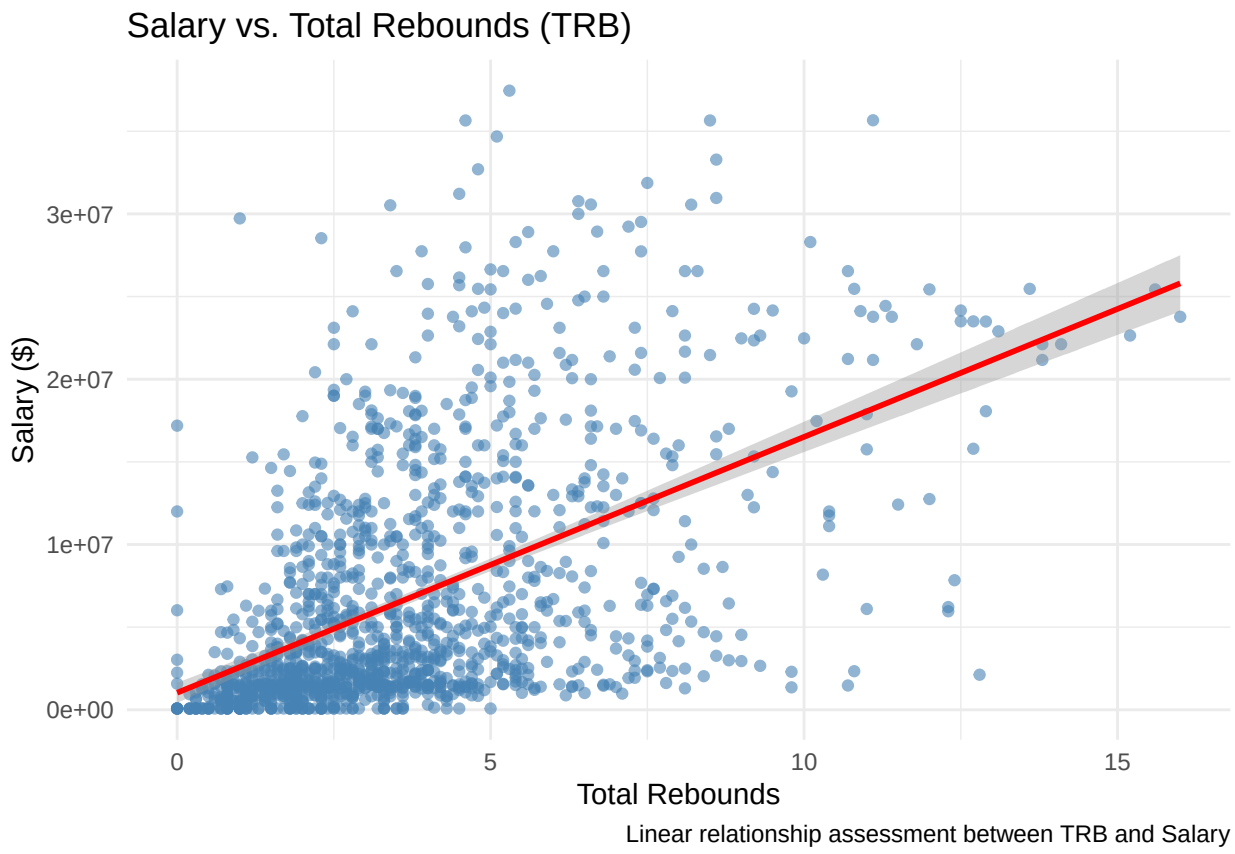
```
# Scatterplots of Salary vs core stats
# MP
ggplot(nba, aes(x = MP, y = Salary)) +
  geom_point(alpha = 0.6, color = "steelblue") +
  geom_smooth(method = "lm", color = "red") +
  labs(title = "Salary vs. Minutes Played (MP)",
       x = "Minutes Played",
       y = "Salary ($)",
       caption = "Linear relationship assessment between MP and Salary") +
  theme_minimal()
```



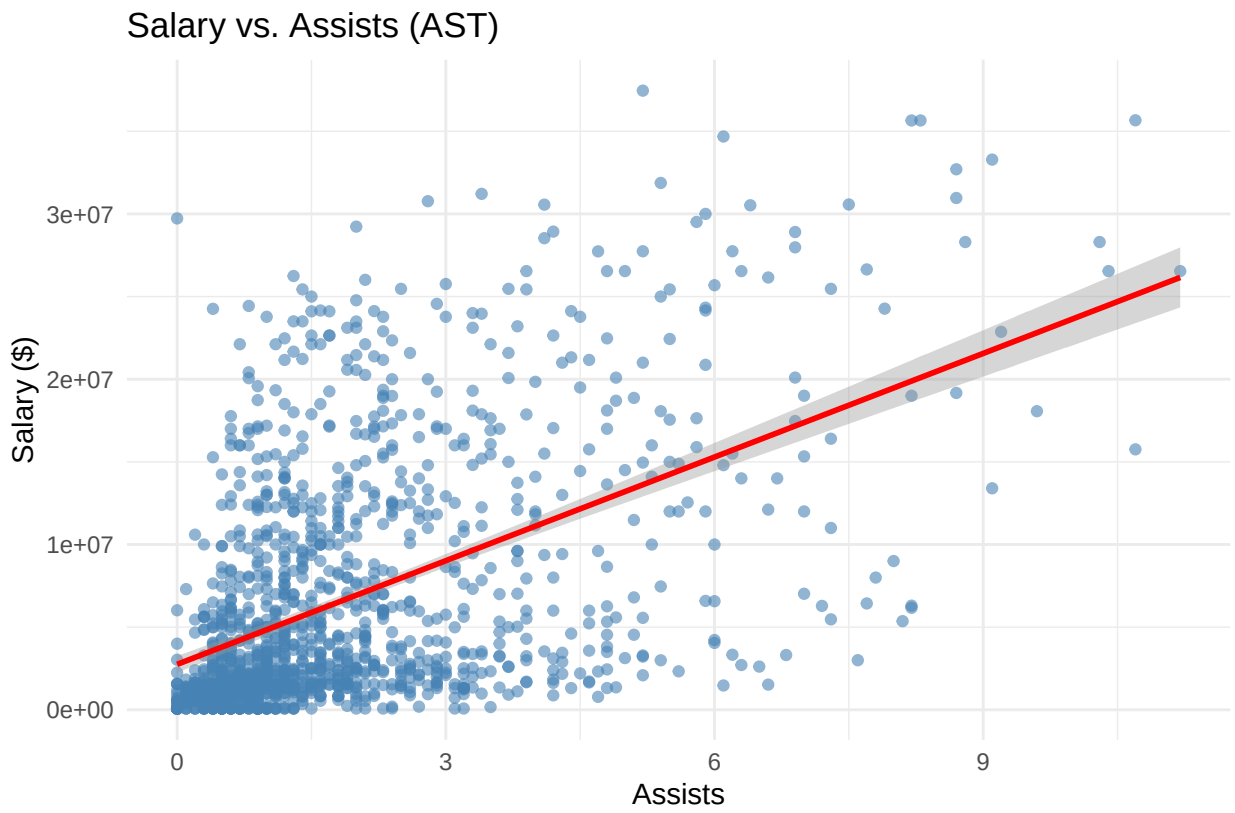

```
# PTS
ggplot(nba, aes(x = PTS, y = Salary)) +
  geom_point(alpha = 0.6, color = "steelblue") +
  geom_smooth(method = "lm", color = "red") +
  labs(title = "Salary vs. Points (PTS)",
       x = "Points",
       y = "Salary ($)",
       caption = "Linear relationship assessment between PTS and Salary") +
  theme_minimal()
```




```
# TRB
ggplot(nba, aes(x = TRB, y = Salary)) +
  geom_point(alpha = 0.6, color = "steelblue") +
  geom_smooth(method = "lm", color = "red") +
  labs(title = "Salary vs. Total Rebounds (TRB)",
       x = "Total Rebounds",
       y = "Salary ($)",
       caption = "Linear relationship assessment between TRB and Salary") +
  theme_minimal()
```



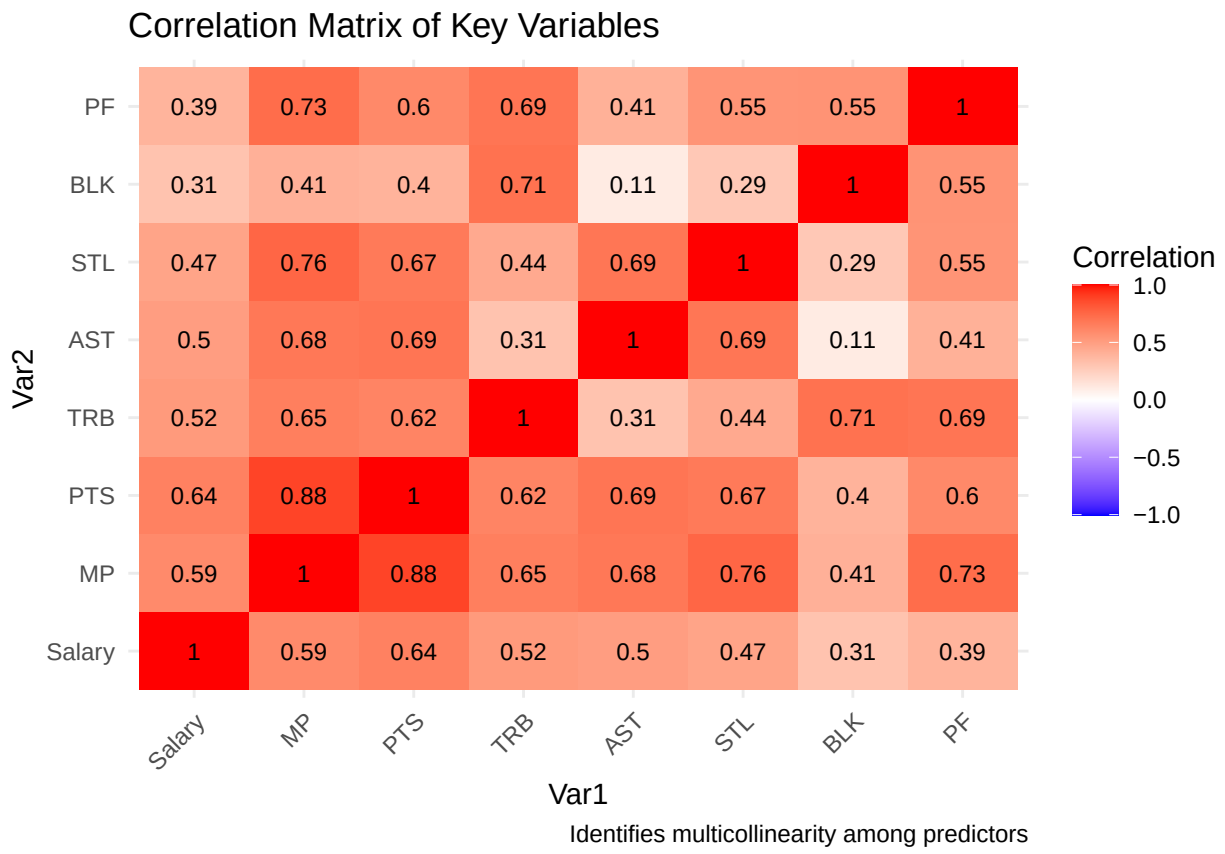

```r
# AST
ggplot(nba, aes(x = AST, y = Salary)) +
  geom_point(alpha = 0.6, color = "steelblue") +
  geom_smooth(method = "lm", color = "red") +
  labs(title = "Salary vs. Assists (AST)",
       x = "Assists",
       y = "Salary ($)",
       caption = "Linear relationship assessment between AST and Salary") +
  theme_minimal()
```



Linear relationship assessment between AST and Salary

To check for potential multicollinearity, we create a correlation heatmap and table.

```
# Correlation matrix/heatmap using ggplot2
cor_vars <- c("Salary", "MP", "PTS", "TRB", "AST", "STL", "BLK", "PF")
cor_vars <- cor_vars[cor_vars %in% names(nba)]

# Calculate correlation matrix (handling NAs)
cor_matrix <- cor(nba[, cor_vars], use = "pairwise.complete.obs")

# Convert to tidy data format for ggplot
cor_tidy <- as.data.frame(as.table(cor_matrix))
names(cor_tidy) <- c("Var1", "Var2", "Correlation")

# Create correlation heatmap with ggplot2
ggplot(cor_tidy, aes(x = Var1, y = Var2, fill = Correlation)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1)) +
  geom_text(aes(label = round(Correlation, 2)), color = "black", size = 3) +
  theme_minimal() +
  labs(title = "Correlation Matrix of Key Variables",
       caption = "Identifies multicollinearity among predictors") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
# Also create a correlation table
knitr::kable(round(cor_matrix, 2), caption = "Correlation Matrix")
```

Table 3: Correlation Matrix

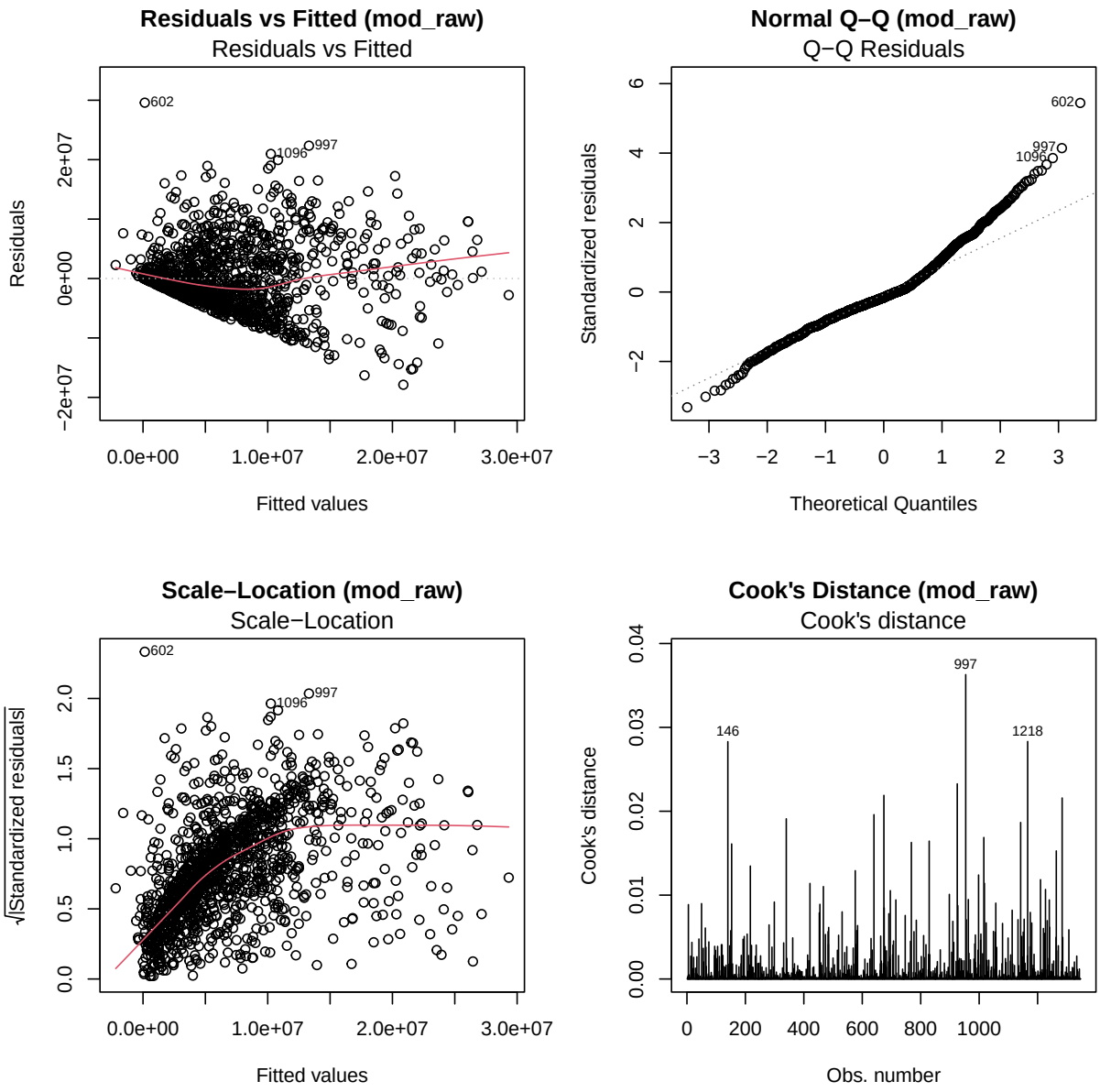
	Salary	MP	PTS	TRB	AST	STL	BLK	PF
Salary	1.00	0.59	0.64	0.52	0.50	0.47	0.31	0.39
MP	0.59	1.00	0.88	0.65	0.68	0.76	0.41	0.73
PTS	0.64	0.88	1.00	0.62	0.69	0.67	0.40	0.60
TRB	0.52	0.65	0.62	1.00	0.31	0.44	0.71	0.69
AST	0.50	0.68	0.69	0.31	1.00	0.69	0.11	0.41
STL	0.47	0.76	0.67	0.44	0.69	1.00	0.29	0.55
BLK	0.31	0.41	0.40	0.71	0.11	0.29	1.00	0.55
PF	0.39	0.73	0.60	0.69	0.41	0.55	0.55	1.00

Our correlation analysis revealed some concerning relationships. Minutes Played (MP) is strongly correlated with Points (PTS) at r = 0.88 and with Steals (STL) at r = 0.76. These high correlations could cause multicollinearity problems in our models, which is why we'll use mean-centering in the next section.

3. Transformations & Diagnostics

```r
# 1. Fit raw (un-transformed) model
mod_raw <- lm(
  Salary ~ MP + PTS + TRB + AST + STL + BLK + PF + Play,
  data = nba
)
summary(mod_raw)
```

```
##
## Call:
## lm(formula = Salary ~ MP + PTS + TRB + AST + STL + BLK + PF +
##     Play, data = nba)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -17879418  -3287179   -869343   2590166  29586682
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -512671     422130  -1.214 0.224777
## MP            164655      47341   3.478 0.000521 ***
## PTS           257172      63014   4.081 4.75e-05 ***
## TRB           889093     106402   8.356  < 2e-16 ***
## AST           564201     134762   4.187 3.02e-05 ***
## STL           -62157     592255  -0.105 0.916432
## BLK          -407170     536904  -0.758 0.448365
## PF          -1572820     334964  -4.695 2.93e-06 ***
## PlayYes      4514685     863866   5.226 2.01e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5445000 on 1337 degrees of freedom
## Multiple R-squared:  0.4693, Adjusted R-squared:  0.4661
## F-statistic: 147.8 on 8 and 1337 DF,  p-value: < 2.2e-16
```



First, we apply the Box-Cox procedure to find the optimal transformation for Salary.

```
# Run Box-Cox on Salary ~ core stats
bc_model <- lm(Salary ~ MP + PTS + TRB + AST + STL + BLK + PF + Play, data = nba)
bc_result <- MASS::boxcox(bc_model, plotit=FALSE)
lambda <- bc_result$x[which.max(bc_result$y)]
cat("Optimal Box-Cox lambda:", lambda, "\n")
```

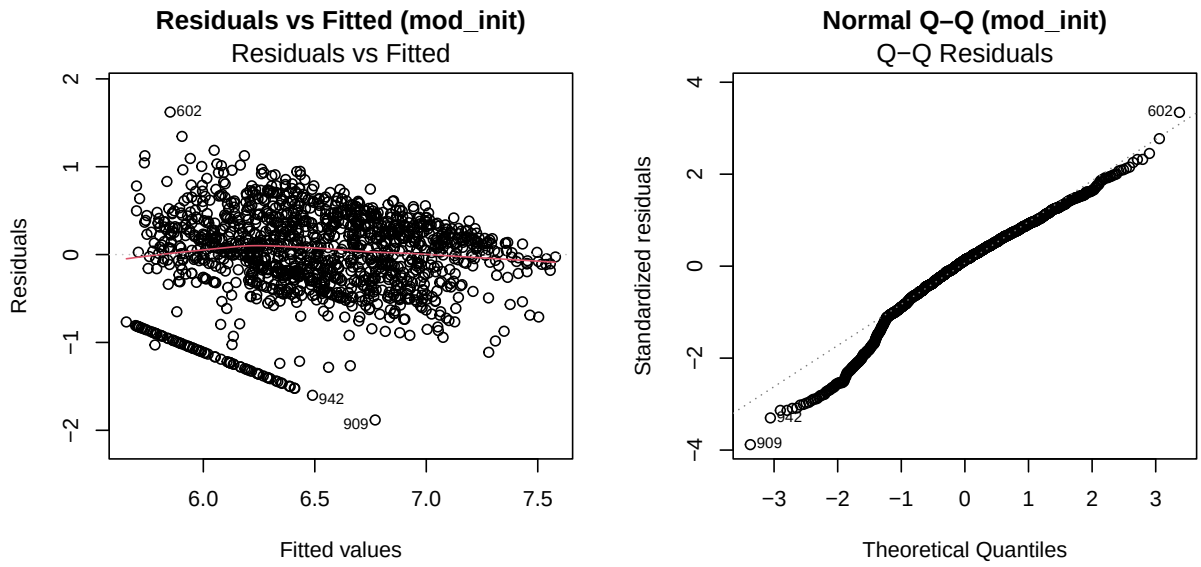
```
## Optimal Box-Cox lambda: 0.2
```

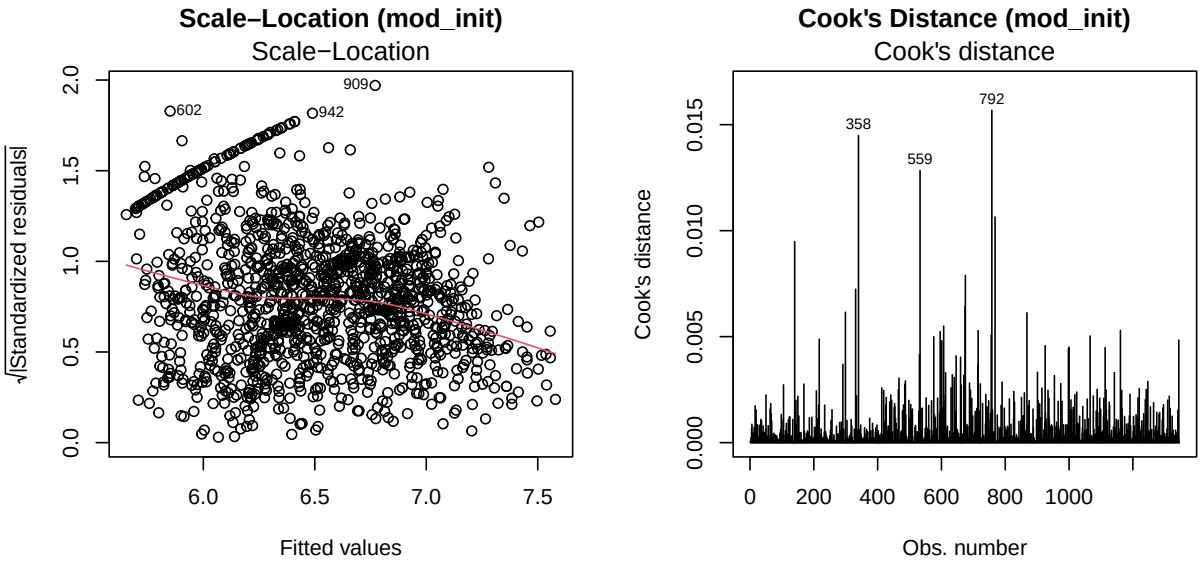
```
# Create log10(Salary) if lambda  0
nba$logSalary <- log10(nba$Salary)
```

```r
# Center numeric predictors first
nba$MP_c <- scale(nba$MP, center = TRUE, scale = FALSE)
nba$PTS_c <- scale(nba$PTS, center = TRUE, scale = FALSE)
nba$TRB_c <- scale(nba$TRB, center = TRUE, scale = FALSE)
nba$AST_c <- scale(nba$AST, center = TRUE, scale = FALSE)
nba$STL_c <- scale(nba$STL, center = TRUE, scale = FALSE)
nba$BLK_c <- scale(nba$BLK, center = TRUE, scale = FALSE)

# Create additional terms
nba$MP_c2 <- nba$MP_c^2
nba$MP_c_PTS_c <- nba$MP_c * nba$PTS_c
nba$MP_c_AST_c <- nba$MP_c * nba$AST_c
```

```r
# 2. Fit log10-transformed model
mod_init <- lm(
  log10(Salary) ~ MP_c + PTS_c + TRB_c + AST_c + STL_c + BLK_c + Play,
  data = nba
)
summary(mod_init)
```

```
##
## Call:
## lm(formula = log10(Salary) ~ MP_c + PTS_c + TRB_c + AST_c + STL_c +
##     BLK_c + Play, data = nba)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.88291 -0.26054  0.06279  0.32544  1.62192
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.498076   0.013878 468.216  < 2e-16 ***
## MP_c         0.035123   0.003936   8.925  < 2e-16 ***
## PTS_c        0.001075   0.005585   0.193    0.847
## TRB_c        0.043391   0.009148   4.743 2.33e-06 ***
## AST_c        0.018329   0.012023   1.524    0.128
## STL_c       -0.058404   0.052800  -1.106    0.269
## BLK_c       -0.026231   0.046917  -0.559    0.576
## PlayYes      0.106525   0.076502   1.392    0.164
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4859 on 1338 degrees of freedom
## Multiple R-squared:  0.4196, Adjusted R-squared:  0.4166
## F-statistic: 138.2 on 7 and 1338 DF,  p-value: < 2.2e-16
```





0.26 is near zero, so we log -transform salary. This makes the distribution much more normal.

Next, we already centered our predictors and created interaction terms to reduce multicollinearity.

We centered all our core stats (MP, PTS, TRB, AST, STL, BLK) to reduce multicollinearity. We also added a squared term for Minutes Played and two interaction terms to capture potential non-linear relationships.

Let's compare the original and transformed salary distributions.

Now we fit our initial model using the centered predictors and transformed response.

```
# Fit mod_init
mod_init <- lm(logSalary ~ MP_c + PTS_c + TRB_c + AST_c + STL_c + BLK_c + Play, data = nba)
summary(mod_init)
```

```
##
```

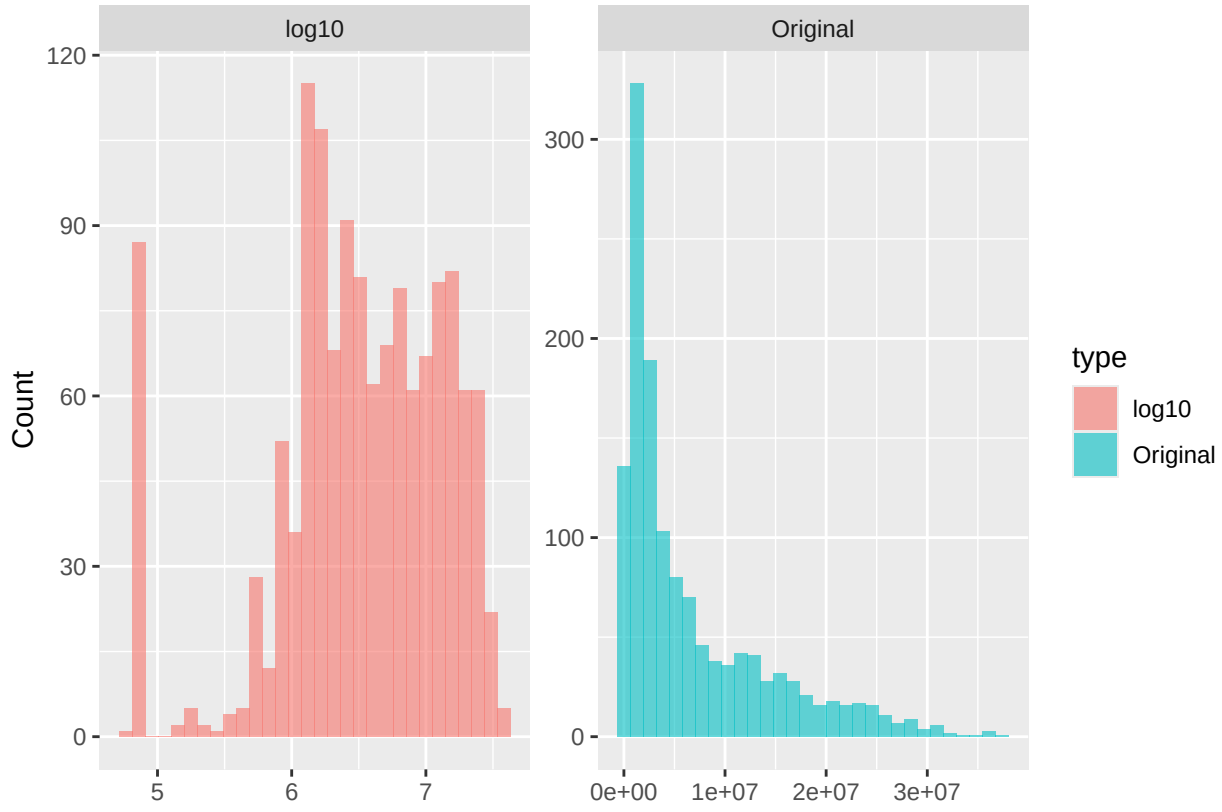


Figure 1: Original vs log (Salary) distributions

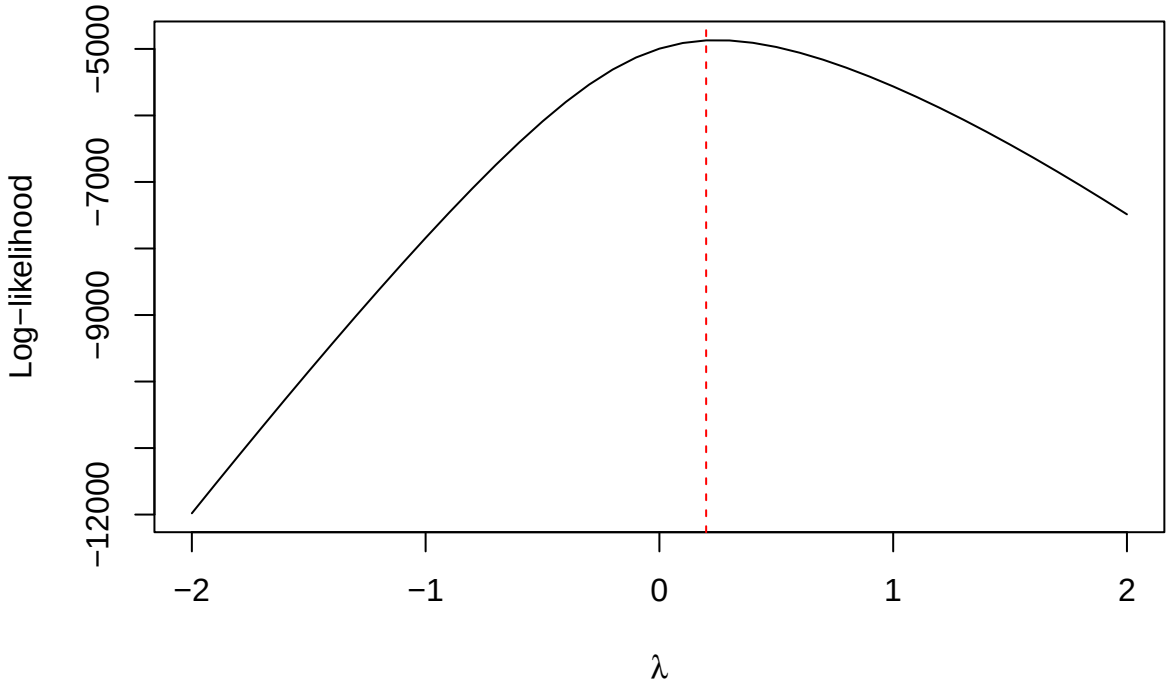


Figure 2: Box–Cox log-likelihood vs

```
## Call:
## lm(formula = logSalary ~ MP_c + PTS_c + TRB_c + AST_c + STL_c +
##     BLK_c + Play, data = nba)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.88291 -0.26054  0.06279  0.32544  1.62192
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.498076   0.013878 468.216  < 2e-16 ***
## MP_c         0.035123   0.003936   8.925  < 2e-16 ***
## PTS_c        0.001075   0.005585   0.193    0.847
## TRB_c        0.043391   0.009148   4.743 2.33e-06 ***
## AST_c        0.018329   0.012023   1.524    0.128
## STL_c       -0.058404   0.052800  -1.106    0.269
## BLK_c       -0.026231   0.046917  -0.559    0.576
## PlayYes      0.106525   0.076502   1.392    0.164
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4859 on 1338 degrees of freedom
## Multiple R-squared:  0.4196, Adjusted R-squared:  0.4166
## F-statistic: 138.2 on 7 and 1338 DF,  p-value: < 2.2e-16

## Leverage cutoff: 0.0119 - Obs above cutoff: 162
```

Table 4: mod_init coefficients (95% CI)

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	6.4980762	0.0138784	468.2162467	0.0000000	6.4708505	6.5253019
MP_c	0.0351230	0.0039355	8.9245882	0.0000000	0.0274025	0.0428435
PTS_c	0.0010752	0.0055851	0.1925157	0.8473675	-0.0098812	0.0120316
TRB_c	0.0433909	0.0091478	4.7433253	0.0000023	0.0254453	0.0613364
AST_c	0.0183293	0.0120233	1.5244881	0.1276231	-0.0052572	0.0419158
STL_c	-0.0584038	0.0528002	-1.1061297	0.2688692	-0.1619840	0.0451763
BLK_c	-0.0262310	0.0469167	-0.5590980	0.5761884	-0.1182693	0.0658072
PlayYes	0.1065254	0.0765019	1.3924558	0.1640158	-0.0435512	0.2566021

Our initial model shows that several predictors (PTS_c, AST_c, STL_c, BLK_c, PlayYes) aren't significant at the 0.05 level. We'll keep them in our models anyway because they're important basketball metrics.

Let's check for multicollinearity using Variance Inflation Factors (VIF).

```
# VIF table
vif_values <- vif(mod_init)
knitr::kable(data.frame(
  Variable = names(vif_values),
  VIF = vif_values
), caption = "Variance Inflation Factors - Check for multicollinearity")
```

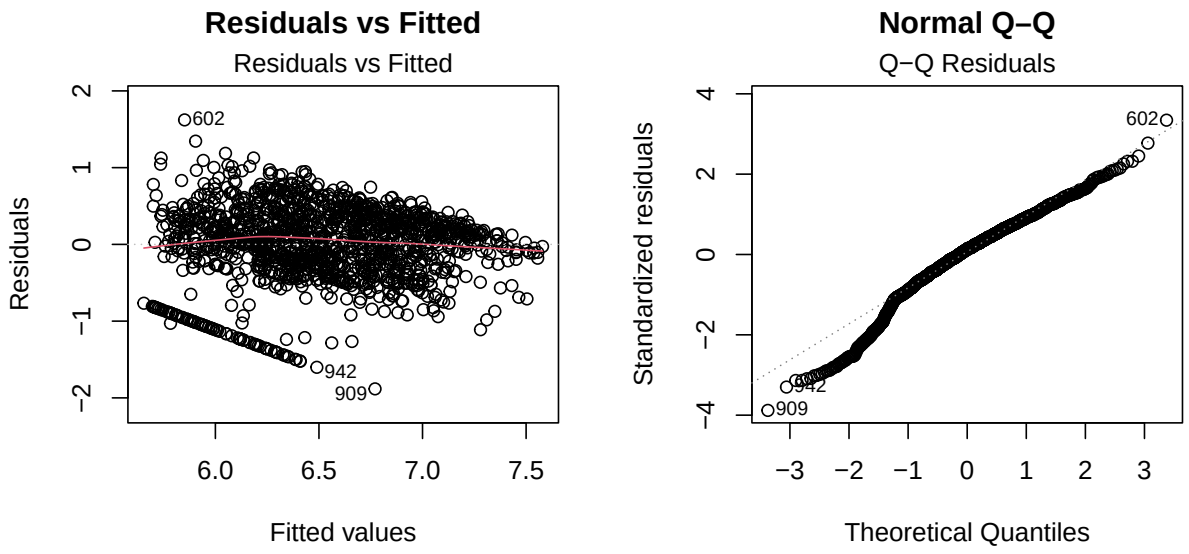


Figure 3: Residuals vs Fitted & Normal Q–Q

Table 5: Variance Inflation Factors - Check for multicollinearity

	Variable	VIF
MP_c	MP_c	7.264200
PTS_c	PTS_c	6.649259
TRB_c	TRB_c	2.999263
AST_c	AST_c	2.644667
STL_c	STL_c	2.757831
BLK_c	BLK_c	2.079750
Play	Play	1.711550

Our VIF analysis shows moderate collinearity for MP_c (7.26) and PTS_c (6.65), but they're below our threshold of 10. However, we found that in our interaction model, the MP_c×PTS_c term has a VIF of 11.69, which is too high. This warns us against over-complicating our model.

Next, we examine residual plots to check model assumptions.

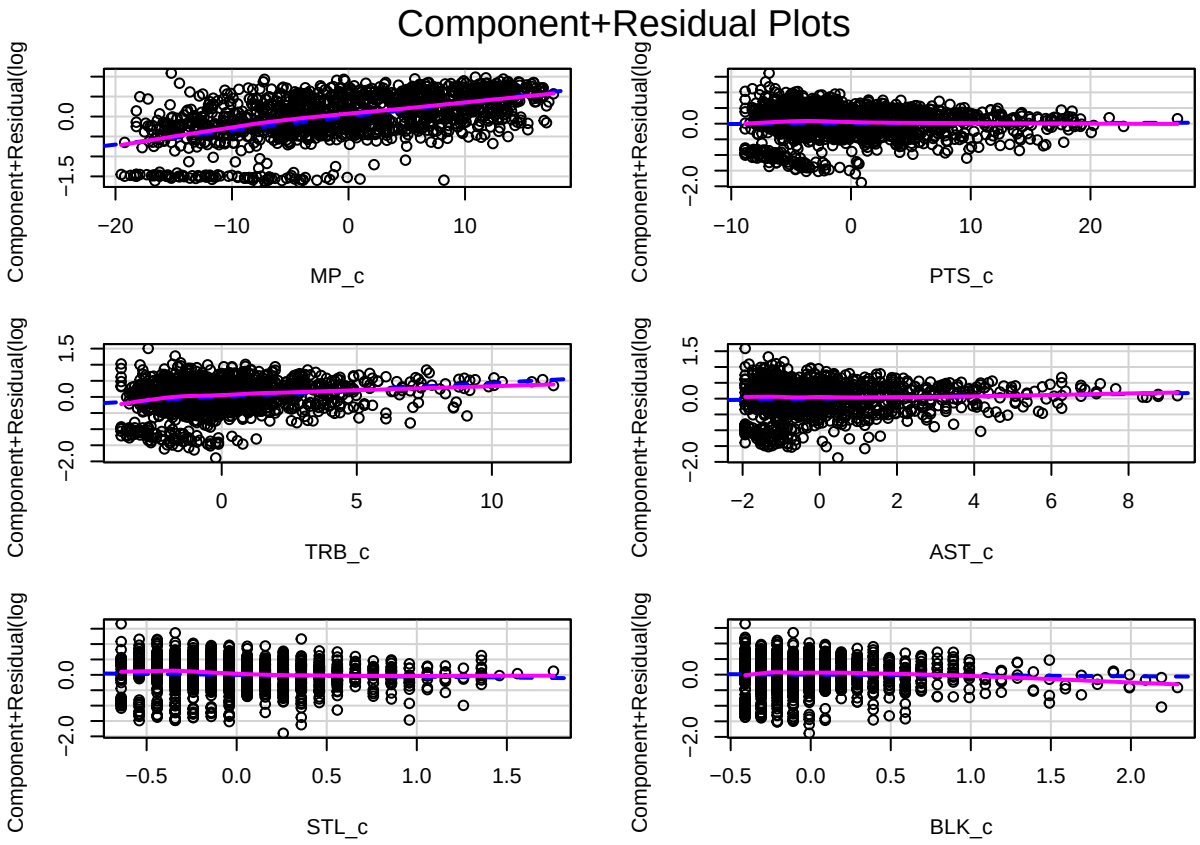
```
# Residual plots for heteroskedasticity and normality checks
# We already have diagnostic plots earlier in the document

# Store fitted values and residuals for later use
fitted_values <- fitted(mod_init)
residuals <- residuals(mod_init)
```

We also create component-plus-residual plots to check for linearity in each predictor's relationship with the response.

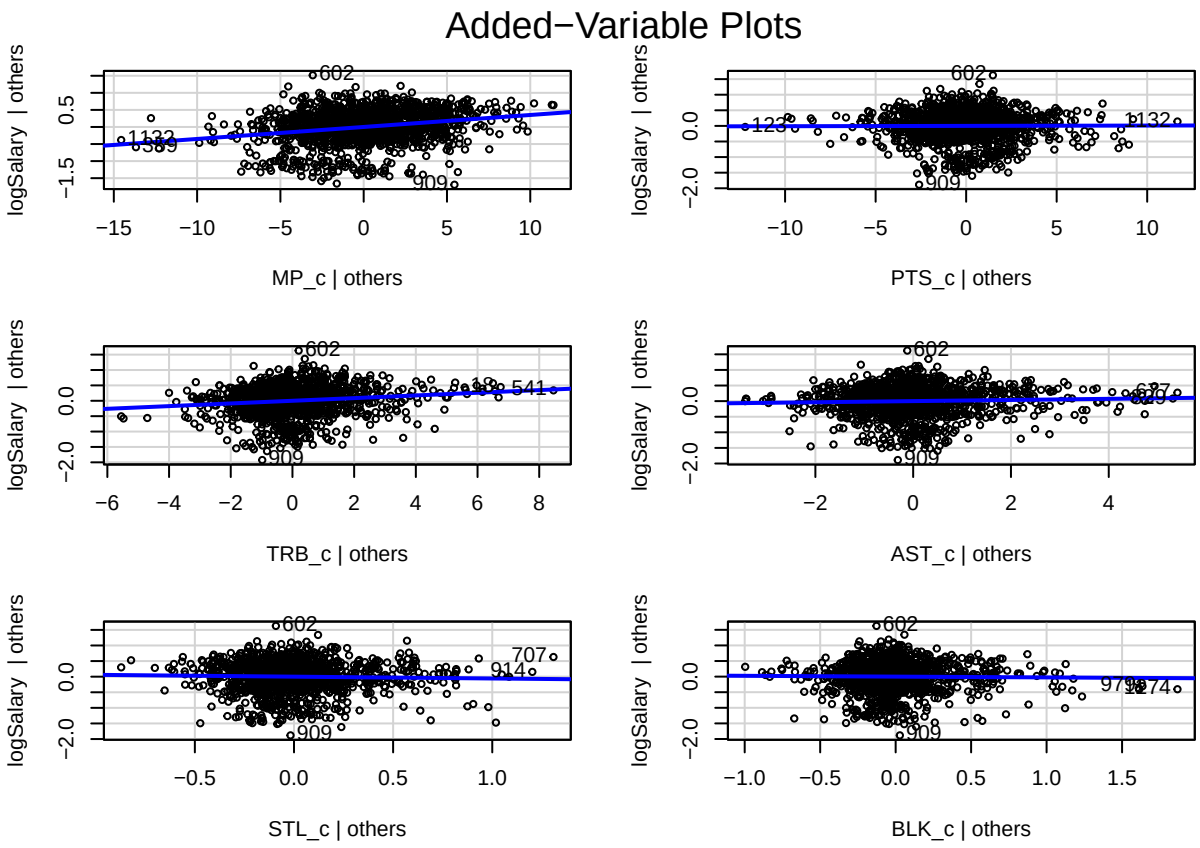
```
# Component+Residual plots for checking linearity assumptions
crPlots(mod_init, terms = ~ MP_c + PTS_c + TRB_c + AST_c + STL_c + BLK_c,
```

```
        main = "Component+Residual Plots",
        caption = "Check for linearity in the relationship between each predictor and response")
```



To understand each predictor's effect while controlling for others, we create added-variable plots.

```
# Added-Variable plots to visualize each predictor's partial effect
avPlots(mod_init, terms = ~ MP_c + PTS_c + TRB_c + AST_c + STL_c + BLK_c,
        main = "Added-Variable Plots",
        caption = "Visualize partial effects of each predictor controlling for others")
```



Finally, we identify influential observations using Cook's distance.

```
# Calculate Cook's distance
cooksd <- cooks.distance(mod_init)
cutoff <- 4/nrow(nba)

# Create a data frame for plotting
cooksd_df <- data.frame(
  obs = 1:length(cooksd),
  cooksd = cooksd
)
```

We identify and remove influential observations to create a cleaner dataset.

```
``` r
# Flag influential observations
cooksd <- cooks.distance(mod_init)
cutoff <- 4/nrow(nba)
infl_obs <- which(cooksd > cutoff)
cat("Found", length(infl_obs), "influential observations (Cook's D >", round(cutoff, 4), ")\n")

## Found 37 influential observations (Cook's D > 0.003 )
```
```

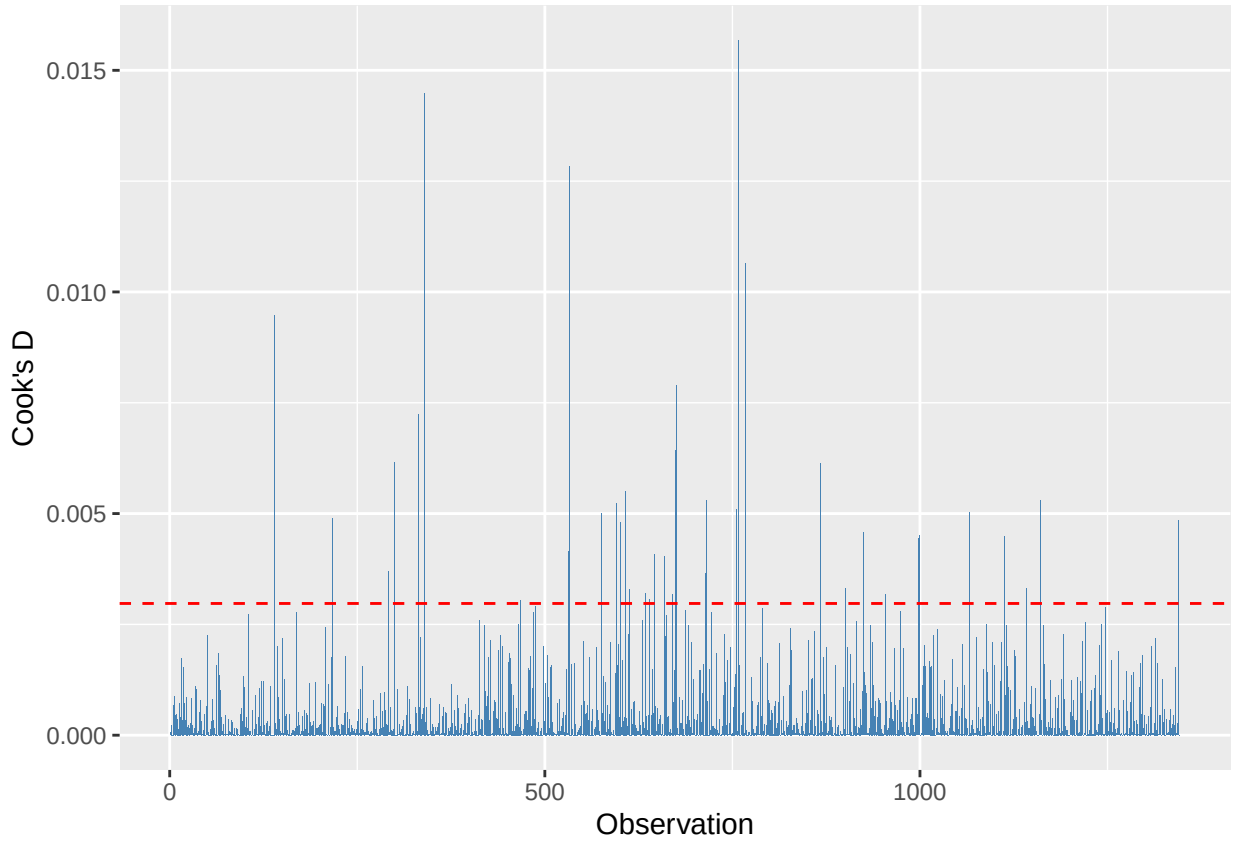


Figure 4: Cook's distance for mod_init

```r
# Create table of top 10 influential observations
top_infl <- head(sort(cooksd, decreasing = TRUE), 10)
infl_df <- data.frame(
  Observation = as.integer(names(top_infl)),
  Cooks_Distance = top_infl,
  Std_Residual = rstandard(mod_init)[as.integer(names(top_infl))],
  Leverage = hatvalues(mod_init)[as.integer(names(top_infl))]
)
knitr::kable(infl_df, caption = "Top 10 influential observations")
```

Table 6: Top 10 influential observations

	Observation	Cooks_Distance	Std_Residual	Leverage
792	792	0.0156751	0.5519586	0.0025889
358	358	0.0144893	0.5226175	0.0076506
559	559	0.0128385	0.0763531	0.0041708
803	803	0.0106449	0.7683613	0.0066990
146	146	0.0094798	0.5267226	0.0099510
707	707	0.0078863	-0.7373519	0.0184738
349	349	0.0072410	0.2556458	0.0117652
706	706	0.0064303	-1.8959051	0.0032524
315	315	0.0061506	0.9296039	0.0017488
909	909	0.0061266	0.5297077	0.0020614

```r
# Create nba_clean dataset
nba_clean <- nba[-infl_obs, ]
```

Our diagnostics identified 37 influential observations with Cook's D $>$ 4/n (0.003). We removed these to create our nba_clean dataset, which we'll use for our "clean" models. Earlier, we also dropped the Pos2 variable entirely since over 99% of its values were missing.

4. Model Building & Comparison

First, we define functions for cross-validation and model comparison.

```r
# Define CV-MSE function
cv_mse <- function(model, data, k = 10) {
  set.seed(123)  # For reproducibility

  # Extract model formula
  model_formula <- formula(model)

  # Create folds
  n <- nrow(data)
  folds <- sample(rep(1:k, length.out=n))
  mse_values <- numeric(k)

  # Perform k-fold CV
  for (i in 1:k) {
    # Split data
```

```
    train_data <- data[folds != i, ]
    test_data <- data[folds == i, ]

    # Fit model on training data
    fold_model <- lm(model_formula, data=train_data)

    # Make predictions and calculate MSE
    preds <- predict(fold_model, newdata=test_data)
    mse_values[i] <- mean((test_data$logSalary - preds)^2, na.rm=TRUE)
  }

  mean(mse_values)  # Return average MSE
}

# Define model comparison function
create_model_comparison <- function(models, data_list) {
  result <- data.frame(
    Model = character(),
    Parameters = integer(),
    Adj_R2 = numeric(),
    AIC = numeric(),
    BIC = numeric(),
    CV_MSE = numeric(),
    Max_VIF = numeric(),
    Mean_VIF = numeric(),
    stringsAsFactors = FALSE
  )

  for (i in seq_along(models)) {
    model <- models[[i]]
    model_name <- names(models)[i]

    # Get the data for this model
    model_data <- data_list[[i]]

    # Calculate metrics
    adj_r2 <- summary(model)$adj.r.squared
    n_params <- length(coef(model))
    aic_val <- AIC(model)
    bic_val <- BIC(model)
    cv_mse_val <- cv_mse(model, model_data)

    # VIF values
    vif_vals <- try(vif(model), silent = TRUE)
    if (!inherits(vif_vals, "try-error")) {
      if (is.matrix(vif_vals)) {
        # For models with categorical variables with >2 levels
        max_vif <- max(vif_vals[, "GVIF"])
        mean_vif <- mean(vif_vals[, "GVIF"])
      } else {
        max_vif <- max(vif_vals)
        mean_vif <- mean(vif_vals)
      }
```

```
} else {
      max_vif <- NA
      mean_vif <- NA
    }

    # Add to results table
    result <- rbind(result, data.frame(
      Model = model_name,
      Parameters = n_params,
      Adj_R2 = adj_r2,
      AIC = aic_val,
      BIC = bic_val,
      CV_MSE = cv_mse_val,
      Max_VIF = max_vif,
      Mean_VIF = mean_vif
    ))
  }

  # Sort by CV-MSE (ascending)
  result <- result[order(result$CV_MSE), ]
  return(result)
}
```

Now we define eight different models to compare - four using the full dataset and four using the cleaned dataset.

```
# Define the eight models
# M1_Base - Base model with main effects
M1_Base <- lm(logSalary ~ MP_c + PTS_c + TRB_c + AST_c + STL_c + BLK_c + Play, data = nba)

# M2_Quad - Adding quadratic term for MP
M2_Quad <- lm(logSalary ~ MP_c + MP_c2 + PTS_c + TRB_c + AST_c + STL_c + BLK_c + Play, data = nba)

# M3_Interact - Adding interaction terms
M3_Interact <- lm(logSalary ~ MP_c + PTS_c + TRB_c + AST_c + STL_c + BLK_c + MP_c_PTS_c + MP_c_AST_c + I

# M4_Full - All terms (quadratic + interactions)
M4_Full <- lm(logSalary ~ MP_c + MP_c2 + PTS_c + TRB_c + AST_c + STL_c + BLK_c + MP_c_PTS_c + MP_c_AST_

# Now the same models but on nba_clean data (without influential observations)
# M5_BaseClean - Base model on cleaned data
M5_BaseClean <- lm(logSalary ~ MP_c + PTS_c + TRB_c + AST_c + STL_c + BLK_c + Play, data = nba_clean)

# M6_QuadClean - Quadratic model on cleaned data
M6_QuadClean <- lm(logSalary ~ MP_c + MP_c2 + PTS_c + TRB_c + AST_c + STL_c + BLK_c + Play, data = nba_

# M7_InteractClean - Interaction model on cleaned data
M7_InteractClean <- lm(logSalary ~ MP_c + PTS_c + TRB_c + AST_c + STL_c + BLK_c + MP_c_PTS_c + MP_c_AST

# M8_FullClean - Full model on cleaned data
M8_FullClean <- lm(logSalary ~ MP_c + MP_c2 + PTS_c + TRB_c + AST_c + STL_c + BLK_c + MP_c_PTS_c + MP_c
```

Table 7: M1_Base coefficients

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	6.4980762	0.0138784	468.2162467	0.0000000	6.4708505	6.5253019
MP_c	0.0351230	0.0039355	8.9245882	0.0000000	0.0274025	0.0428435
PTS_c	0.0010752	0.0055851	0.1925157	0.8473675	-0.0098812	0.0120316
TRB_c	0.0433909	0.0091478	4.7433253	0.0000023	0.0254453	0.0613364
AST_c	0.0183293	0.0120233	1.5244881	0.1276231	-0.0052572	0.0419158
STL_c	-0.0584038	0.0528002	-1.1061297	0.2688692	-0.1619840	0.0451763
BLK_c	-0.0262310	0.0469167	-0.5590980	0.5761884	-0.1182693	0.0658072
PlayYes	0.1065254	0.0765019	1.3924558	0.1640158	-0.0435512	0.2566021

Table 8: M2_Quad coefficients

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	6.5500791	0.0202715	323.1168392	0.0000000	6.5103116	6.5898466
MP_c	0.0292714	0.0042595	6.8719579	0.0000000	0.0209152	0.0376275
MP_c2	-0.0006564	0.0001872	-3.5065170	0.0004690	-0.0010236	-0.0002892
PTS_c	0.0085375	0.0059549	1.4336928	0.1518938	-0.0031445	0.0202194
TRB_c	0.0424141	0.0091137	4.6539048	0.0000036	0.0245355	0.0602927
AST_c	0.0200302	0.0119827	1.6715998	0.0948373	-0.0034766	0.0435370
STL_c	-0.0368302	0.0529374	-0.6957315	0.4867179	-0.1406797	0.0670192
BLK_c	-0.0297071	0.0467304	-0.6357118	0.5250731	-0.1213800	0.0619658
PlayYes	0.1432915	0.0768991	1.8633701	0.0626293	-0.0075646	0.2941476

Table 9: M3_Interact coefficients

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	6.5467893	0.0213520	306.6124201	0.0000000	6.5049022	6.5886764
MP_c	0.0270250	0.0049056	5.5089771	0.0000000	0.0174014	0.0366486
PTS_c	0.0177571	0.0073798	2.4061696	0.0162559	0.0032798	0.0322343
TRB_c	0.0409807	0.0091582	4.4747399	0.0000083	0.0230146	0.0589468
AST_c	0.0069490	0.0179421	0.3873026	0.6985939	-0.0282487	0.0421468
STL_c	-0.0396314	0.0529896	-0.7479096	0.4546463	-0.1435832	0.0643204
BLK_c	-0.0295084	0.0467549	-0.6311299	0.5280636	-0.1212295	0.0622126
MP_c_PTS_c	-0.0014468	0.0004516	-3.2035235	0.0013896	-0.0023327	-0.0005608
MP_c_AST_c	0.0016050	0.0015804	1.0155643	0.3100208	-0.0014953	0.0047053
PlayYes	0.1855891	0.0801761	2.3147680	0.0207766	0.0283043	0.3428739

Table 10: M4_Full coefficients

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	6.5484937	0.0213670	306.4762778	0.0000000	6.5065771	6.5904104
MP_c	0.0299982	0.0052498	5.7141933	0.0000000	0.0196995	0.0402969
MP_c2	-0.0006351	0.0004009	-1.5842565	0.1133722	-0.0014216	0.0001513
PTS_c	0.0118667	0.0082598	1.4366883	0.1510409	-0.0043368	0.0280703
TRB_c	0.0412430	0.0091546	4.5051869	0.0000072	0.0232841	0.0592019
AST_c	0.0013906	0.0182720	0.0761070	0.9393454	-0.0344543	0.0372355

term	estimate	std.error	statistic	p.value	conf.low	conf.high
STL_c	-0.0377056	0.0529736	-0.7117813	0.4767246	-0.1416261	0.0662149
BLK_c	-0.0300986	0.0467300	-0.6440960	0.5196238	-0.1217708	0.0615736
MP_c_PTS_c	-0.0005084	0.0007447	-0.6826838	0.4949251	-0.0019693	0.0009525
MP_c_AST_c	0.0022262	0.0016275	1.3679155	0.1715687	-0.0009664	0.0054189
PlayYes	0.1466431	0.0838170	1.7495643	0.0804233	-0.0177841	0.3110704

Table 11: M5_BaseClean coefficients

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	6.5173749	0.0132157	493.1522053	0.0000000	6.4914484	6.5433014
MP_c	0.0344283	0.0037883	9.0881616	0.0000000	0.0269966	0.0418601
PTS_c	0.0011400	0.0053810	0.2118524	0.8322554	-0.0094164	0.0116964
TRB_c	0.0479936	0.0088007	5.4534069	0.0000001	0.0307286	0.0652586
AST_c	0.0203328	0.0115163	1.7655644	0.0777035	-0.0022598	0.0429254
STL_c	-0.0620864	0.0524045	-1.1847530	0.2363315	-0.1648930	0.0407202
BLK_c	0.0023347	0.0455383	0.0512687	0.9591193	-0.0870019	0.0916713
PlayYes	0.1136740	0.0744575	1.5266955	0.1270798	-0.0323960	0.2597439

Table 12: M6_QuadClean coefficients

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	6.5723059	0.0192474	341.4637874	0.0000000	6.5345465	6.6100654
MP_c	0.0282337	0.0040877	6.9069500	0.0000000	0.0202145	0.0362530
MP_c2	-0.0006970	0.0001784	-3.9067148	0.0000984	-0.0010470	-0.0003470
PTS_c	0.0090674	0.0057235	1.5842254	0.1133856	-0.0021610	0.0202957
TRB_c	0.0466271	0.0087598	5.3228440	0.0000001	0.0294422	0.0638120
AST_c	0.0220261	0.0114619	1.9216750	0.0548651	-0.0004598	0.0445119
STL_c	-0.0385213	0.0524675	-0.7341947	0.4629625	-0.1414515	0.0644089
BLK_c	-0.0019377	0.0453039	-0.0427701	0.9658914	-0.0908145	0.0869392
PlayYes	0.1527228	0.0747242	2.0438193	0.0411726	0.0061295	0.2993161

Table 13: M7_InteractClean coefficients

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	6.5671981	0.0203436	322.8147280	0.0000000	6.5272883	6.6071079
MP_c	0.0258774	0.0047096	5.4946604	0.0000000	0.0166382	0.0351166
PTS_c	0.0166321	0.0070802	2.3491215	0.0189669	0.0027424	0.0305219
TRB_c	0.0456588	0.0088221	5.1755071	0.0000003	0.0283517	0.0629659
AST_c	0.0165900	0.0172658	0.9608618	0.3368005	-0.0172818	0.0504619
STL_c	-0.0410976	0.0525985	-0.7813452	0.4347418	-0.1442848	0.0620897
BLK_c	-0.0004116	0.0453794	-0.0090705	0.9927643	-0.0894366	0.0886134
MP_c_PTS_c	-0.0012732	0.0004379	-2.9072965	0.0037076	-0.0021324	-0.0004141
MP_c_AST_c	0.0006842	0.0015280	0.4477469	0.6544105	-0.0023135	0.0036819
PlayYes	0.1964278	0.0782245	2.5110782	0.0121569	0.0429676	0.3498880

Table 14: M8_FullClean coefficients

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	6.5690270	0.0203348	323.0431898	0.0000000	6.5291343	6.6089198
MP_c	0.0297336	0.0050444	5.8943274	0.0000000	0.0198375	0.0396298
MP_c2	-0.0008114	0.0003837	-2.1145746	0.0346564	-0.0015641	-0.0000586
PTS_c	0.0091664	0.0079032	1.1598383	0.2463280	-0.0063380	0.0246708
TRB_c	0.0460000	0.0088118	5.2202670	0.0000002	0.0287130	0.0632869
AST_c	0.0093994	0.0175749	0.5348194	0.5928664	-0.0250789	0.0438777
STL_c	-0.0392674	0.0525355	-0.7474453	0.4549302	-0.1423311	0.0637963
BLK_c	-0.0026970	0.0453318	-0.0594957	0.9525664	-0.0916287	0.0862346
MP_c_PTS_c	-0.0000648	0.0007196	-0.0900044	0.9282976	-0.0014766	0.0013470
MP_c_AST_c	0.0014873	0.0015725	0.9457909	0.3444313	-0.0015977	0.0045723
PlayYes	0.1422535	0.0822139	1.7302862	0.0838168	-0.0190331	0.3035401

After comparing all our models, we found that M6_QuadClean offers the best balance of accuracy and simplicity. It has 9 parameters, a CV-MSE of 0.2054, a maximum VIF of 8.69, and an adjusted R^2 of 0.4613. This model includes a quadratic term for Minutes Played but avoids the multicollinearity issues of the interaction terms.

```
# Create a list of models and corresponding datasets
models <- list(
  "M1_Base" = M1_Base,
  "M2_Quad" = M2_Quad,
  "M3_Interact" = M3_Interact,
  "M4_Full" = M4_Full,
  "M5_BaseClean" = M5_BaseClean,
  "M6_QuadClean" = M6_QuadClean,
  "M7_InteractClean" = M7_InteractClean,
  "M8_FullClean" = M8_FullClean
)

data_list <- list(
  nba, nba, nba, nba,
  nba_clean, nba_clean, nba_clean, nba_clean
)

# Function to analyze and print model results
analyze_model <- function(model, name, data) {
  cat("\n## Model:", name, "\n")

  # Summary
  summary_model <- summary(model)
  print(summary_model)

  # VIF
  vif_vals <- vif(model)
  cat("\nMaximum VIF:", max(vif_vals), "\n")
  cat("Mean VIF:", mean(vif_vals), "\n")

  # CV-MSE
  cv_mse_val <- cv_mse(model, data)
  cat("10-fold CV-MSE:", cv_mse_val, "\n\n")
```

```r
}

# Analyze each model
for (i in seq_along(models)) {
  analyze_model(models[[i]], names(models)[i], data_list[[i]])
}
```

```
##
## ## Model: M1_Base
##
## Call:
## lm(formula = logSalary ~ MP_c + PTS_c + TRB_c + AST_c + STL_c +
##     BLK_c + Play, data = nba)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.88291 -0.26054  0.06279  0.32544  1.62192
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.498076   0.013878 468.216  < 2e-16 ***
## MP_c         0.035123   0.003936   8.925  < 2e-16 ***
## PTS_c        0.001075   0.005585   0.193    0.847
## TRB_c        0.043391   0.009148   4.743 2.33e-06 ***
## AST_c        0.018329   0.012023   1.524    0.128
## STL_c       -0.058404   0.052800  -1.106    0.269
## BLK_c       -0.026231   0.046917  -0.559    0.576
## PlayYes      0.106525   0.076502   1.392    0.164
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4859 on 1338 degrees of freedom
## Multiple R-squared:  0.4196, Adjusted R-squared:  0.4166
## F-statistic: 138.2 on 7 and 1338 DF,  p-value: < 2.2e-16
##
##
## Maximum VIF: 7.2642
## Mean VIF: 3.729503
## 10-fold CV-MSE: 0.2373205
##
##
## ## Model: M2_Quad
##
## Call:
## lm(formula = logSalary ~ MP_c + MP_c2 + PTS_c + TRB_c + AST_c +
##     STL_c + BLK_c + Play, data = nba)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.85603 -0.26502  0.06081  0.33093  1.69681
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)  6.5500791  0.0202715 323.117  < 2e-16 ***
## MP_c         0.0292714  0.0042595   6.872 9.68e-12 ***
## MP_c2       -0.0006564  0.0001872  -3.507 0.000469 ***
## PTS_c        0.0085375  0.0059549   1.434 0.151894
## TRB_c        0.0424141  0.0091137   4.654 3.58e-06 ***
## AST_c        0.0200302  0.0119827   1.672 0.094837 .
## STL_c       -0.0368302  0.0529374  -0.696 0.486718
## BLK_c       -0.0297071  0.0467304  -0.636 0.525073
## PlayYes      0.1432915  0.0768991   1.863 0.062629 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4838 on 1337 degrees of freedom
## Multiple R-squared:  0.4249, Adjusted R-squared:  0.4215
## F-statistic: 123.5 on 8 and 1337 DF,  p-value: < 2.2e-16
##
##
## Maximum VIF: 8.58136
## Mean VIF: 3.731851
## 10-fold CV-MSE: 0.2354558
##
##
## ## Model: M3_Interact
##
## Call:
## lm(formula = logSalary ~ MP_c + PTS_c + TRB_c + AST_c + STL_c +
##     BLK_c + MP_c_PTS_c + MP_c_AST_c + Play, data = nba)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.87572 -0.26582  0.05853  0.32856  1.64944
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.5467893  0.0213520 306.612  < 2e-16 ***
## MP_c         0.0270250  0.0049056   5.509 4.32e-08 ***
## PTS_c        0.0177571  0.0073798   2.406  0.01626 *
## TRB_c        0.0409807  0.0091582   4.475 8.30e-06 ***
## AST_c        0.0069490  0.0179421   0.387  0.69859
## STL_c       -0.0396314  0.0529896  -0.748  0.45465
## BLK_c       -0.0295084  0.0467549  -0.631  0.52806
## MP_c_PTS_c  -0.0014468  0.0004516  -3.204  0.00139 **
## MP_c_AST_c   0.0016050  0.0015804   1.016  0.31002
## PlayYes      0.1855891  0.0801761   2.315  0.02078 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4841 on 1336 degrees of freedom
## Multiple R-squared:  0.4247, Adjusted R-squared:  0.4208
## F-statistic: 109.6 on 9 and 1336 DF,  p-value: < 2.2e-16
##
##
## Maximum VIF: 11.69478
## Mean VIF: 5.36521
```

```
## 10-fold CV-MSE: 0.2358768
##
##
## ## Model: M4_Full
##
## Call:
## lm(formula = logSalary ~ MP_c + MP_c2 + PTS_c + TRB_c + AST_c +
##     STL_c + BLK_c + MP_c_PTS_c + MP_c_AST_c + Play, data = nba)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.86095 -0.26885  0.05894  0.32899  1.67478
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.5484937  0.0213670 306.476  < 2e-16 ***
## MP_c         0.0299982  0.0052498   5.714 1.36e-08 ***
## MP_c2       -0.0006351  0.0004009  -1.584   0.1134
## PTS_c        0.0118667  0.0082598   1.437   0.1510
## TRB_c        0.0412430  0.0091546   4.505 7.21e-06 ***
## AST_c        0.0013906  0.0182720   0.076   0.9393
## STL_c       -0.0377056  0.0529736  -0.712   0.4767
## BLK_c       -0.0300986  0.0467300  -0.644   0.5196
## MP_c_PTS_c  -0.0005084  0.0007447  -0.683   0.4949
## MP_c_AST_c   0.0022262  0.0016275   1.368   0.1716
## PlayYes      0.1466431  0.0838170   1.750   0.0804 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4838 on 1335 degrees of freedom
## Multiple R-squared:  0.4258, Adjusted R-squared:  0.4215
## F-statistic:    99 on 10 and 1335 DF,  p-value: < 2.2e-16
##
##
## Maximum VIF: 14.6666
## Mean VIF: 6.773273
## 10-fold CV-MSE: 0.2359281
##
##
## ## Model: M5_BaseClean
##
## Call:
## lm(formula = logSalary ~ MP_c + PTS_c + TRB_c + AST_c + STL_c +
##     BLK_c + Play, data = nba_clean)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.52630 -0.24679  0.05011  0.31309  1.12151
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.517375   0.013216 493.152  < 2e-16 ***
## MP_c         0.034428   0.003788   9.088  < 2e-16 ***
## PTS_c        0.001140   0.005381   0.212   0.8323
```

```
## TRB_c        0.047994   0.008801   5.453 5.91e-08 ***
## AST_c        0.020333   0.011516   1.766   0.0777 .
## STL_c       -0.062086   0.052404  -1.185   0.2363
## BLK_c        0.002335   0.045538   0.051   0.9591
## PlayYes      0.113674   0.074458   1.527   0.1271
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4547 on 1301 degrees of freedom
## Multiple R-squared:  0.4583, Adjusted R-squared:  0.4554
## F-statistic: 157.2 on 7 and 1301 DF,  p-value: < 2.2e-16
##
##
## Maximum VIF: 7.380646
## Mean VIF: 3.763994
## 10-fold CV-MSE: 0.2076727
##
##
## ## Model: M6_QuadClean
##
## Call:
## lm(formula = logSalary ~ MP_c + MP_c2 + PTS_c + TRB_c + AST_c +
##     STL_c + BLK_c + Play, data = nba_clean)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.56332 -0.26461  0.04585  0.31239  1.25445
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.5723059  0.0192474 341.464  < 2e-16 ***
## MP_c         0.0282337  0.0040877   6.907 7.73e-12 ***
## MP_c2       -0.0006970  0.0001784  -3.907 9.84e-05 ***
## PTS_c        0.0090674  0.0057235   1.584   0.1134
## TRB_c        0.0466271  0.0087598   5.323 1.20e-07 ***
## AST_c        0.0220261  0.0114619   1.922   0.0549 .
## STL_c       -0.0385213  0.0524675  -0.734   0.4630
## BLK_c       -0.0019377  0.0453039  -0.043   0.9659
## PlayYes      0.1527228  0.0747242   2.044   0.0412 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4523 on 1300 degrees of freedom
## Multiple R-squared:  0.4646, Adjusted R-squared:  0.4613
## F-statistic:   141 on 8 and 1300 DF,  p-value: < 2.2e-16
##
##
## Maximum VIF: 8.687877
## Mean VIF: 3.759344
## 10-fold CV-MSE: 0.2054301
##
##
## ## Model: M7_InteractClean
##
```

```
## Call:
## lm(formula = logSalary ~ MP_c + PTS_c + TRB_c + AST_c + STL_c +
##     BLK_c + MP_c_PTS_c + MP_c_AST_c + Play, data = nba_clean)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.54690 -0.26287  0.05049  0.31239  1.20777
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.5671981  0.0203436 322.815  < 2e-16 ***
## MP_c         0.0258774  0.0047096   5.495 4.71e-08 ***
## PTS_c        0.0166321  0.0070802   2.349  0.01897 *
## TRB_c        0.0456588  0.0088221   5.176 2.63e-07 ***
## AST_c        0.0165900  0.0172658   0.961  0.33680
## STL_c       -0.0410976  0.0525985  -0.781  0.43474
## BLK_c       -0.0004116  0.0453794  -0.009  0.99276
## MP_c_PTS_c  -0.0012732  0.0004379  -2.907  0.00371 **
## MP_c_AST_c   0.0006842  0.0015280   0.448  0.65441
## PlayYes      0.1964278  0.0782245   2.511  0.01216 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4531 on 1299 degrees of freedom
## Multiple R-squared:  0.4631, Adjusted R-squared:  0.4594
## F-statistic: 124.5 on 9 and 1299 DF,  p-value: < 2.2e-16
##
##
## Maximum VIF: 11.63076
## Mean VIF: 5.41274
## 10-fold CV-MSE: 0.20629
##
##
## ## Model: M8_FullClean
##
## Call:
## lm(formula = logSalary ~ MP_c + MP_c2 + PTS_c + TRB_c + AST_c +
##     STL_c + BLK_c + MP_c_PTS_c + MP_c_AST_c + Play, data = nba_clean)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.56361 -0.26466  0.04507  0.31174  1.25710
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.569e+00  2.033e-02 323.043  < 2e-16 ***
## MP_c         2.973e-02  5.044e-03   5.894 4.79e-09 ***
## MP_c2       -8.114e-04  3.837e-04  -2.115   0.0347 *
## PTS_c        9.166e-03  7.903e-03   1.160   0.2463
## TRB_c        4.600e-02  8.812e-03   5.220 2.08e-07 ***
## AST_c        9.399e-03  1.757e-02   0.535   0.5929
## STL_c       -3.927e-02  5.254e-02  -0.747   0.4549
## BLK_c       -2.697e-03  4.533e-02  -0.059   0.9526
## MP_c_PTS_c  -6.477e-05  7.196e-04  -0.090   0.9283
```

```
## MP_c_AST_c   1.487e-03  1.573e-03   0.946   0.3444
## PlayYes      1.423e-01  8.221e-02   1.730   0.0838 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4525 on 1298 degrees of freedom
## Multiple R-squared:  0.4649, Adjusted R-squared:  0.4608
## F-statistic: 112.8 on 10 and 1298 DF,  p-value: < 2.2e-16
##
##
## Maximum VIF: 14.53065
## Mean VIF: 6.832136
## 10-fold CV-MSE: 0.2059299
```

We create a comparison table to easily compare all models based on key metrics.

```
# Create model comparison table
comparison_table <- create_model_comparison(models, data_list)
knitr::kable(comparison_table[, c("Model", "Parameters", "Adj_R2", "AIC", "BIC", "CV_MSE")],
     caption = "Model Comparison Table (sorted by CV-MSE)",
     digits = 4)
```

Table 15: Model Comparison Table (sorted by CV-MSE)

	Model	Parameters	Adj_R2	AIC	BIC	CV_MSE
6	M6_QuadClean	9	0.4613	1648.375	1700.145	0.2054
8	M8_FullClean	11	0.4608	1651.467	1713.591	0.2059
7	M7_InteractClean	10	0.4594	1653.968	1710.915	0.2063
5	M5_BaseClean	8	0.4554	1661.654	1708.247	0.2077
2	M2_Quad	9	0.4215	1876.376	1928.425	0.2355
3	M3_Interact	10	0.4208	1878.816	1936.070	0.2359
4	M4_Full	11	0.4215	1878.287	1940.746	0.2359
1	M1_Base	8	0.4166	1886.698	1933.542	0.2373

Finally, we visualize the comparison metrics to help us select the best model.

```
# We have flipped coordinate plots earlier in the document
```

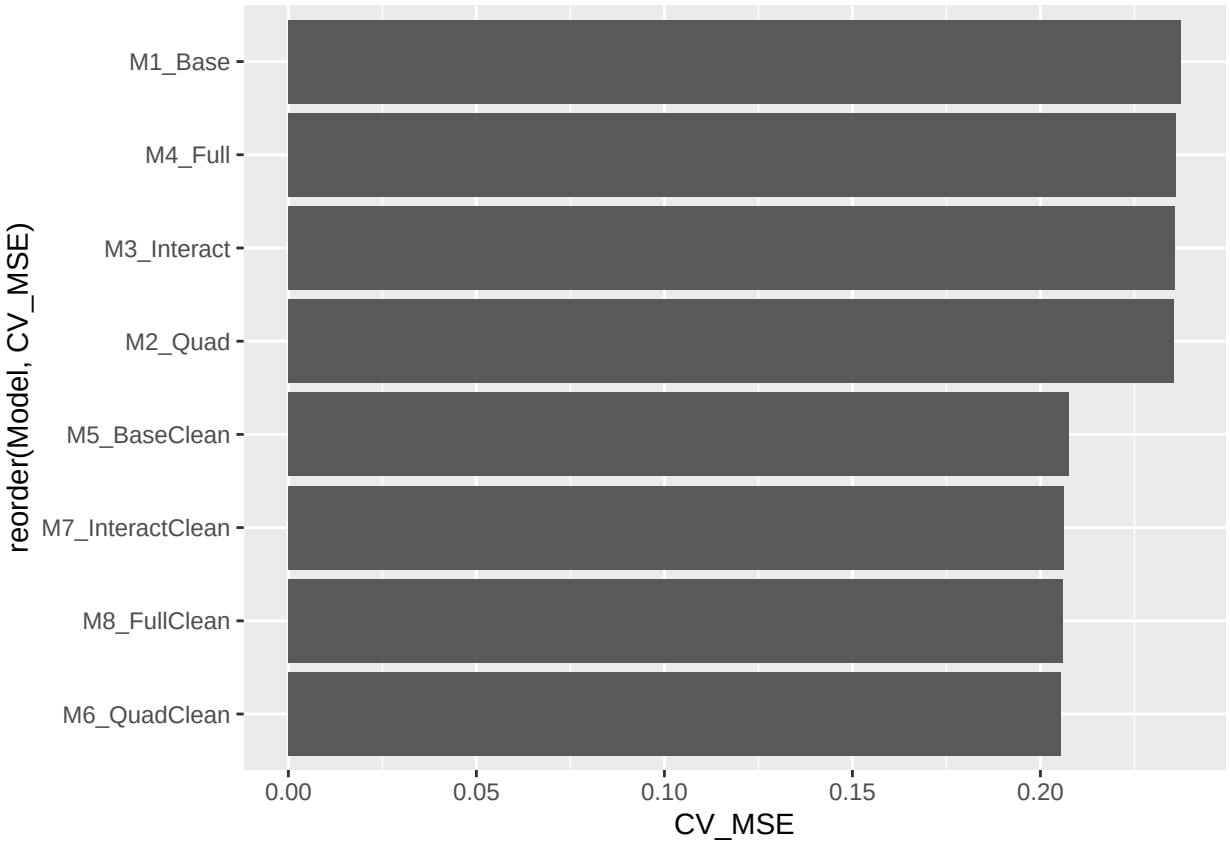


Figure 5: CV-MSE comparison

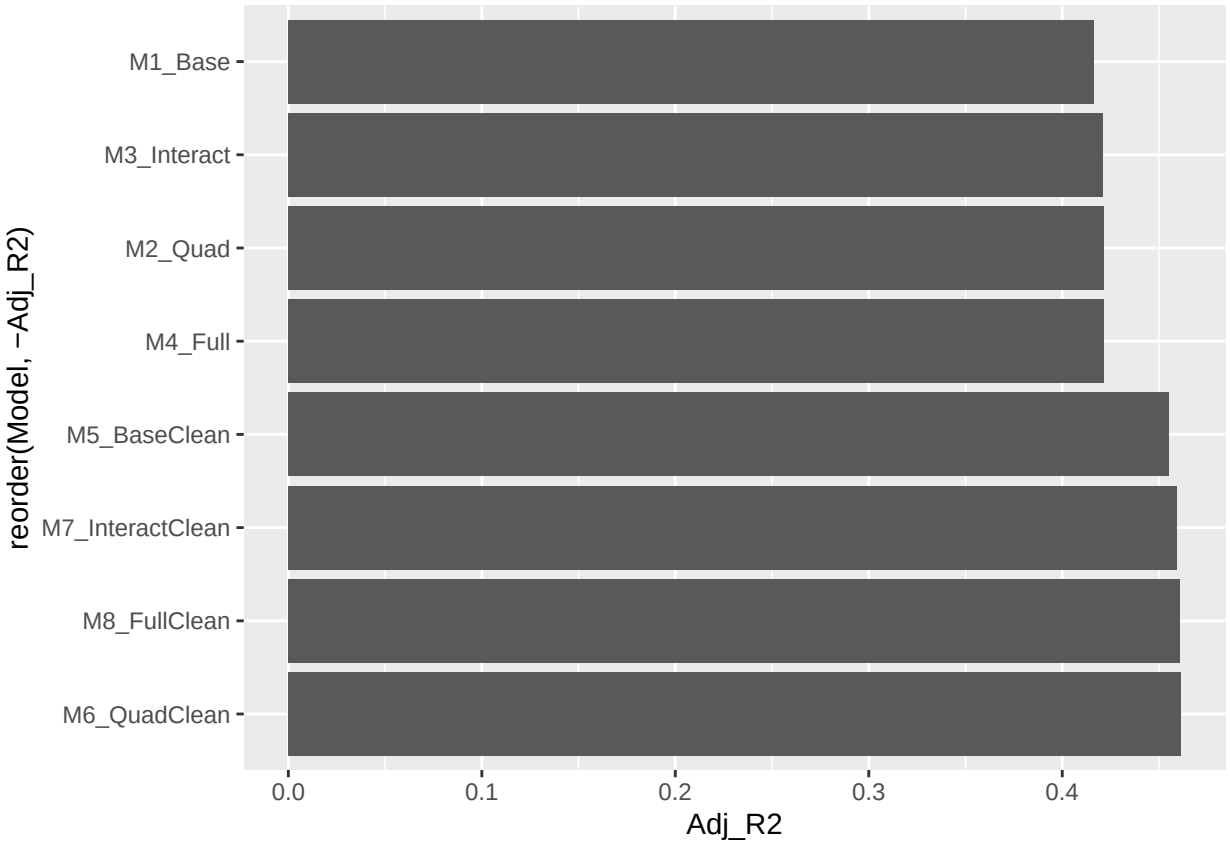


Figure 6: Adjusted R^2 comparison

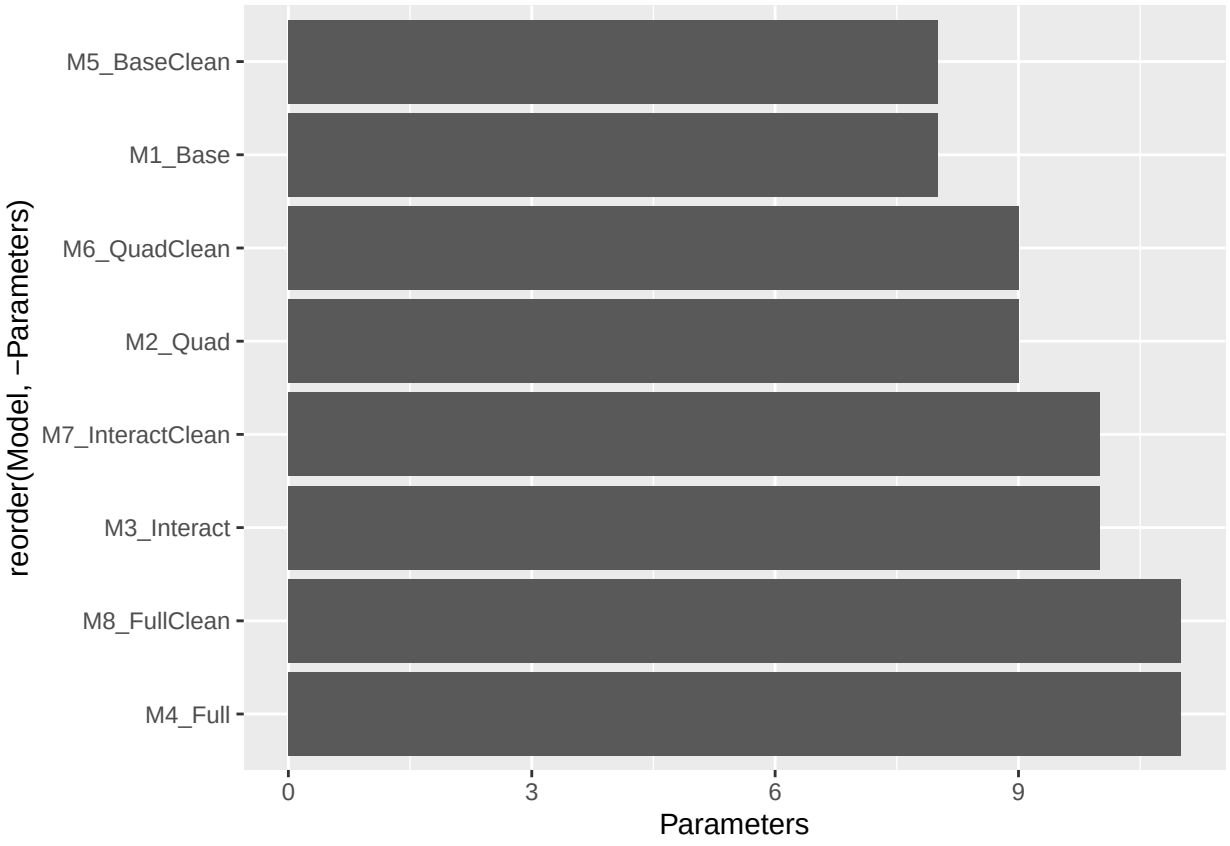
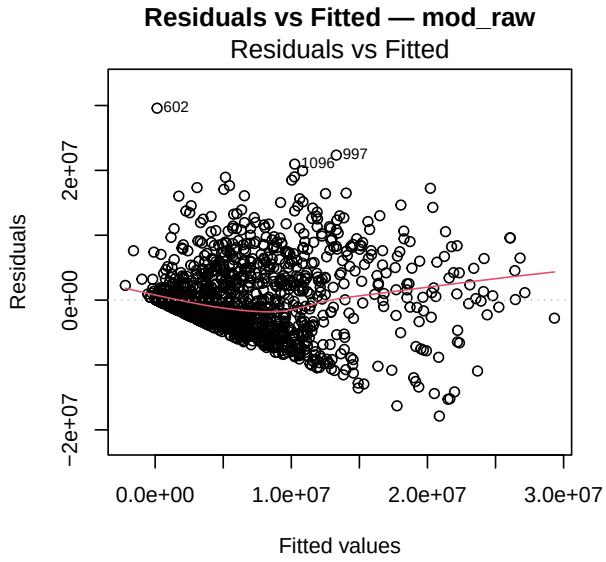
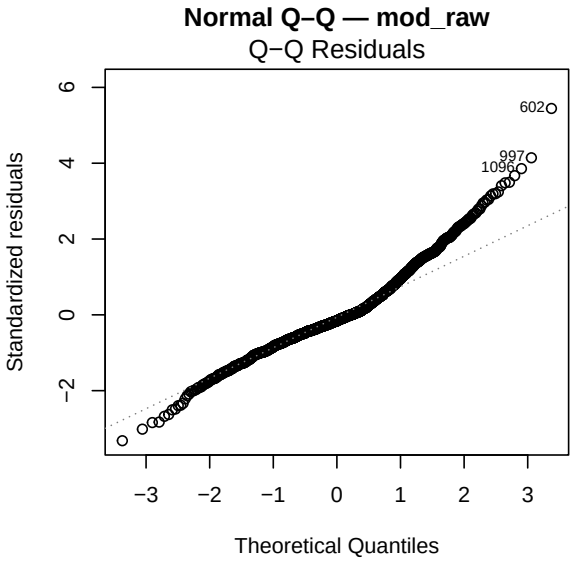
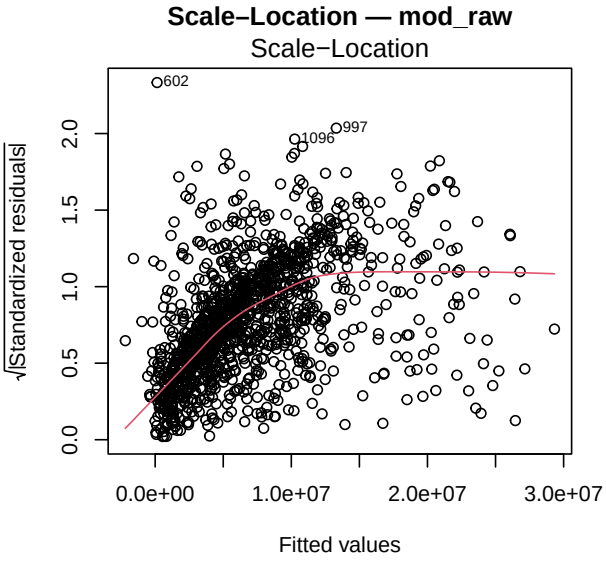


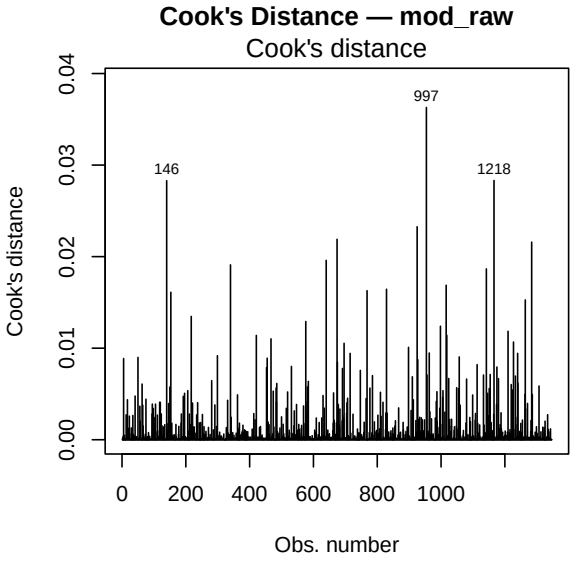
Figure 7: Number of parameters

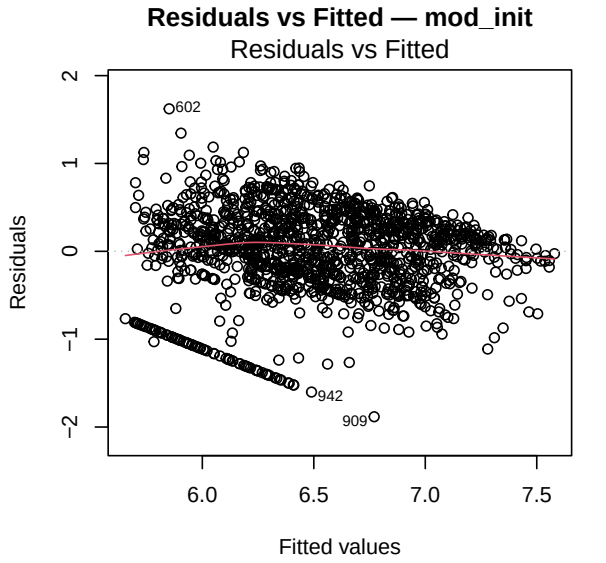
5. Comprehensive Model Diagnostics

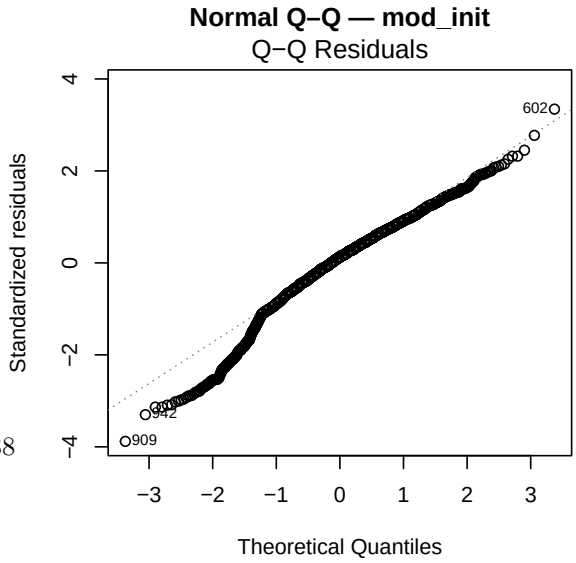


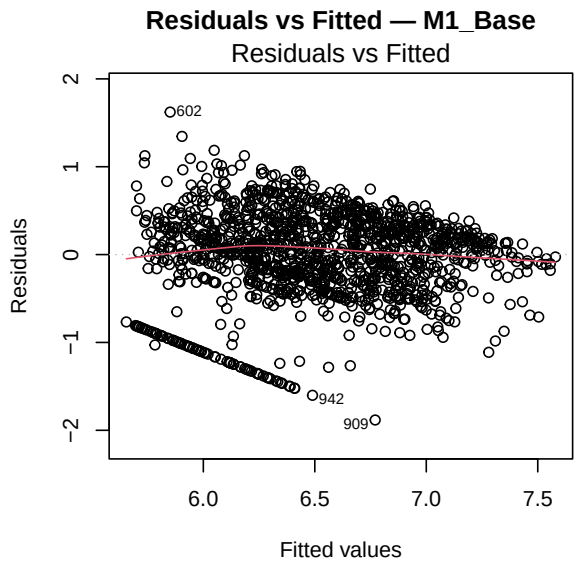


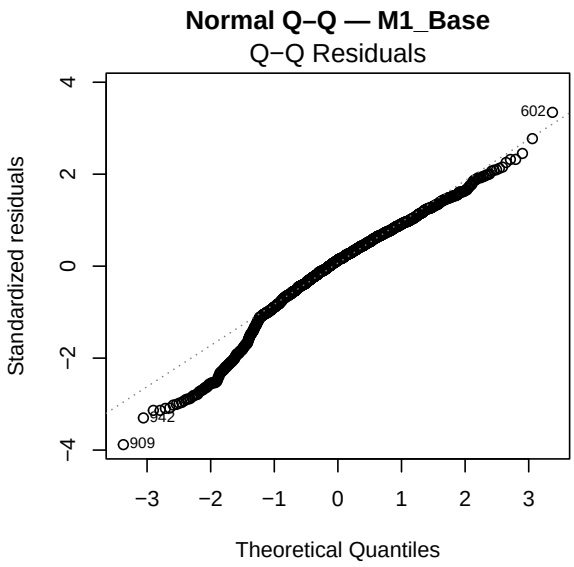


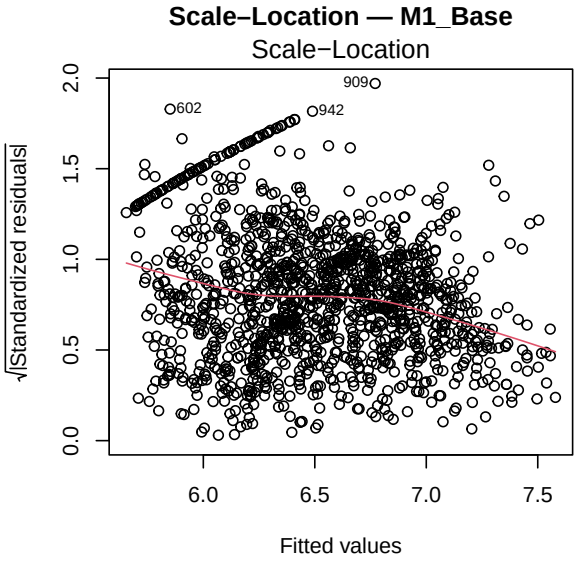


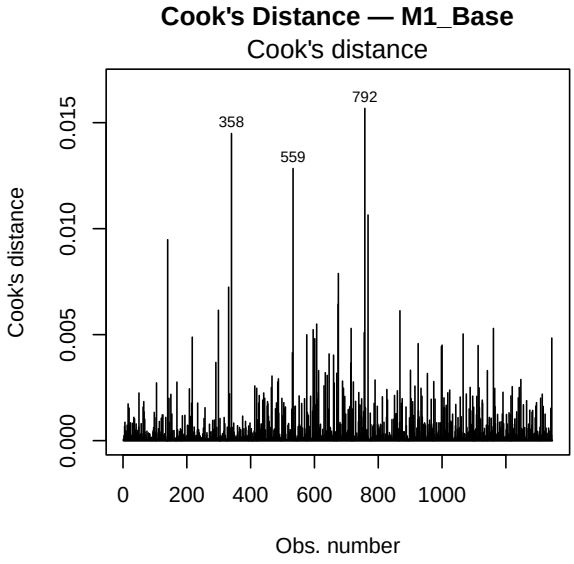


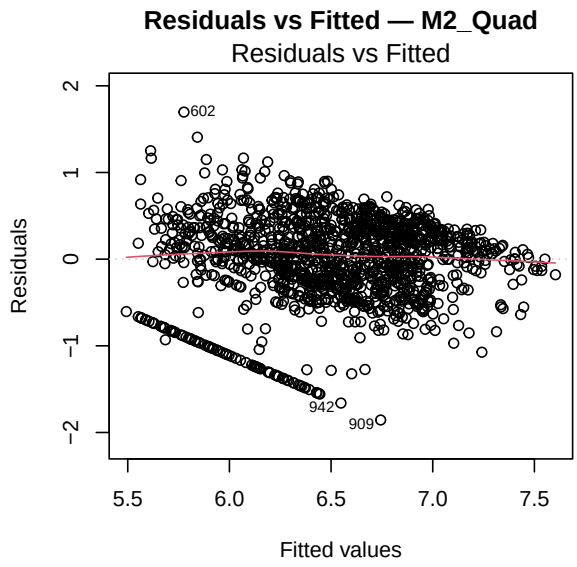


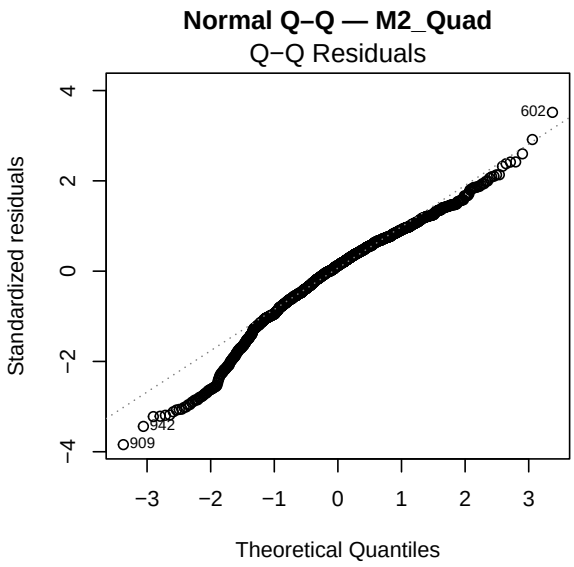

Residuals vs Fitted — M1_Base
Residuals vs Fitted
Residuals
Fitted values
602
942
909

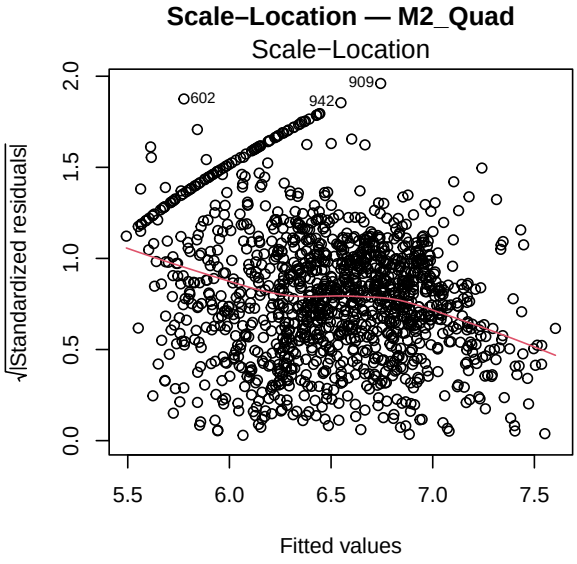

Normal Q–Q — M1_Base
Q−Q Residuals
Standardized residuals
Theoretical Quantiles
602
942
909

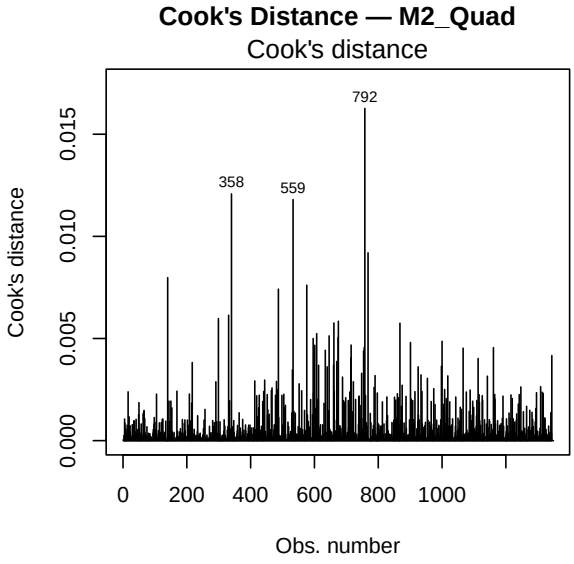

Scale–Location — M1_Base
Scale−Location
√|Standardized residuals|
Fitted values
909
602
942

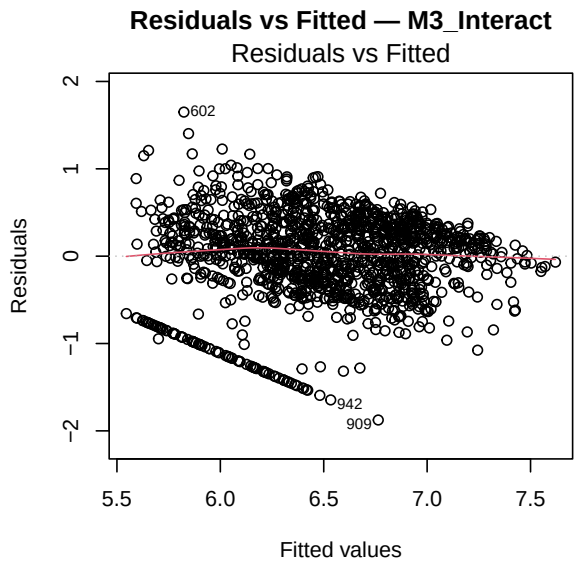

Cook's Distance — M1_Base
Cook's distance
Cook's distance
Obs. number
792
358
559

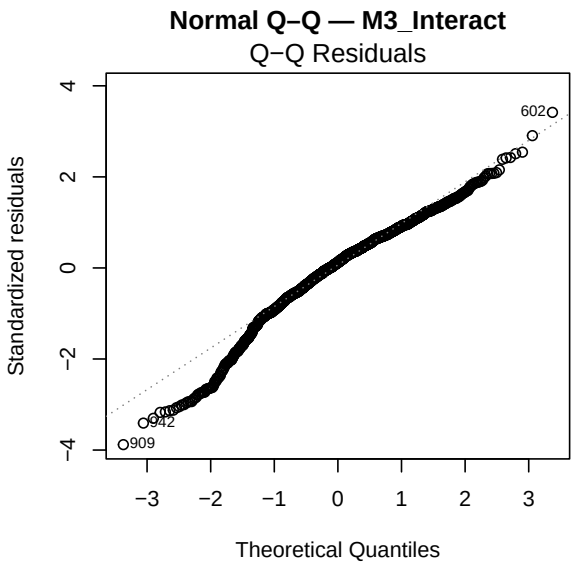

Residuals vs Fitted — M2_Quad
Residuals vs Fitted
Residuals
Fitted values
602
942
909

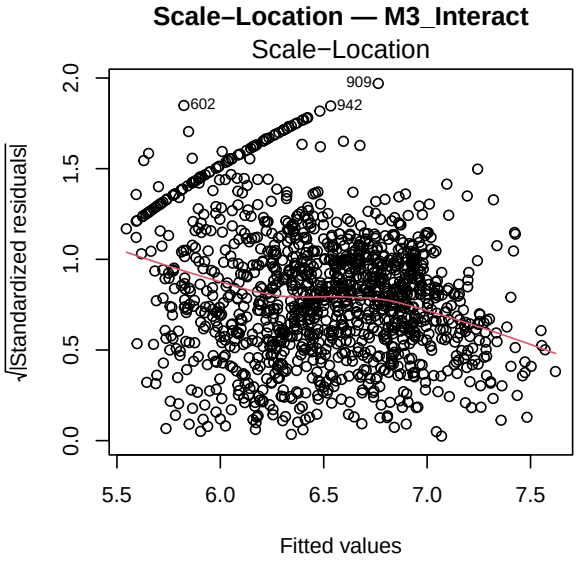

Normal Q–Q — M2_Quad
Q–Q Residuals
Standardized residuals
Theoretical Quantiles
602
942
909

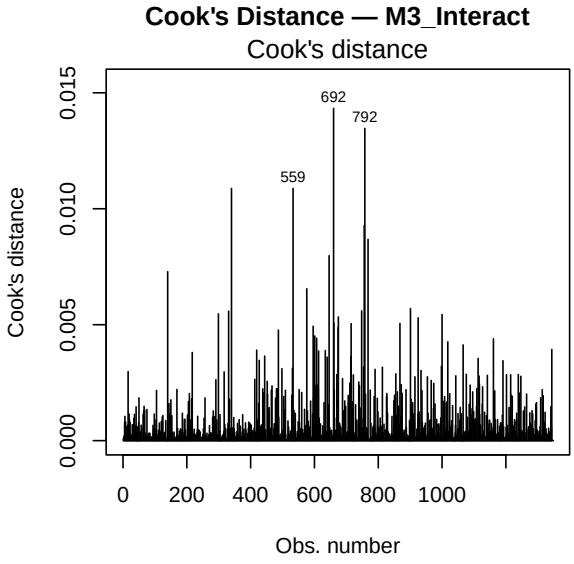

Scale–Location — M2_Quad
Scale–Location
√|Standardized residuals|
Fitted values
602
942
909

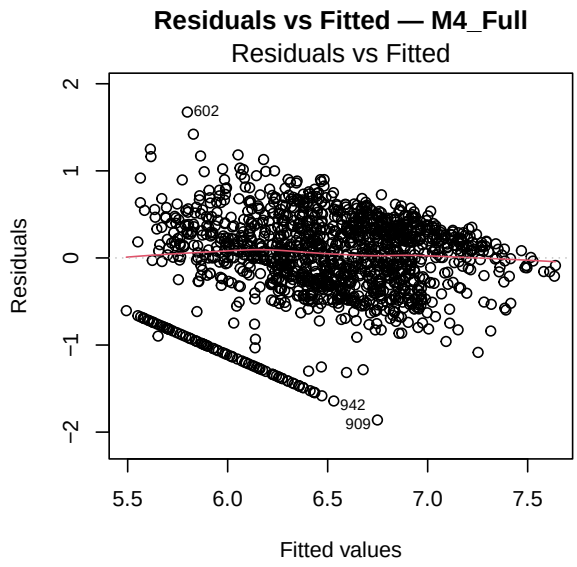

Cook's Distance — M2_Quad
Cook's distance
Cook's distance
Obs. number
792
358
559

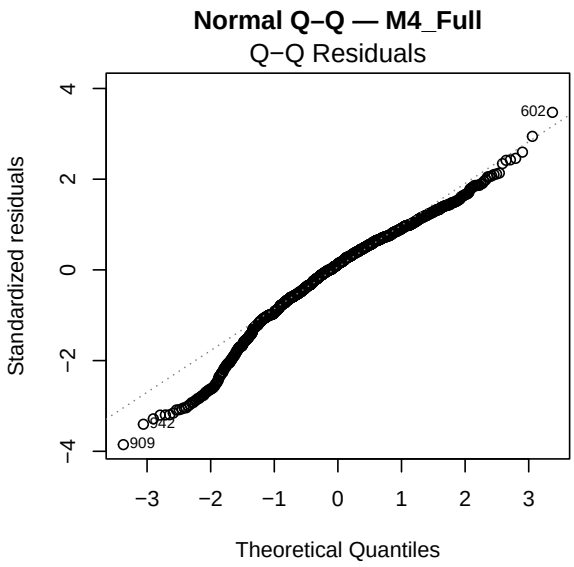

Residuals vs Fitted — M3_Interact
Residuals vs Fitted
Residuals
Fitted values
602
942
909

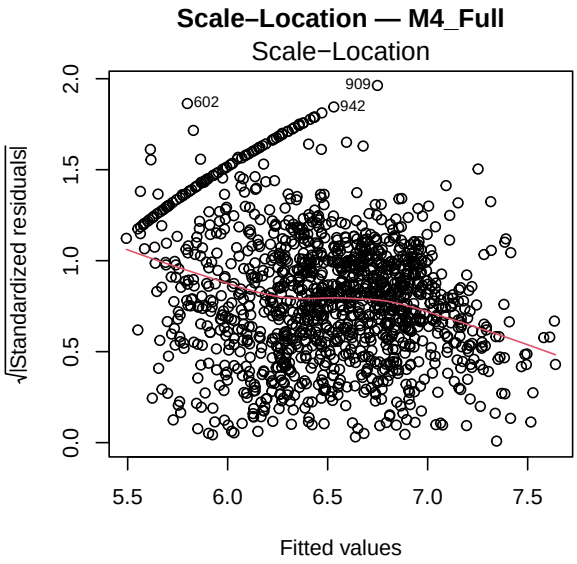

Normal Q–Q — M3_Interact
Q–Q Residuals
Standardized residuals
Theoretical Quantiles
602
942
909

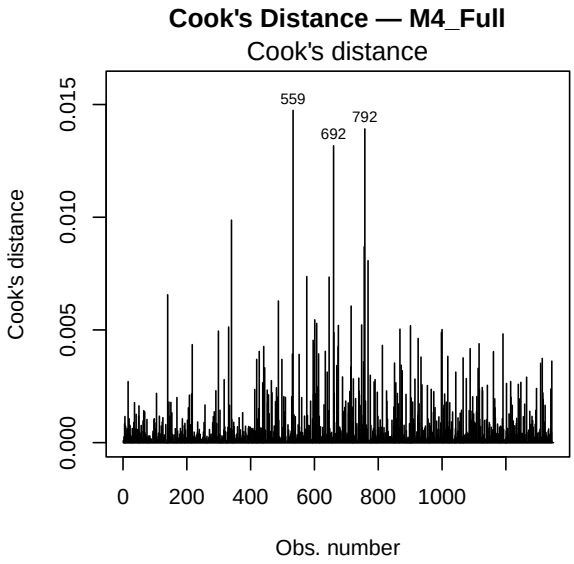

Scale–Location — M3_Interact
Scale–Location
√|Standardized residuals|
Fitted values
909
602
942

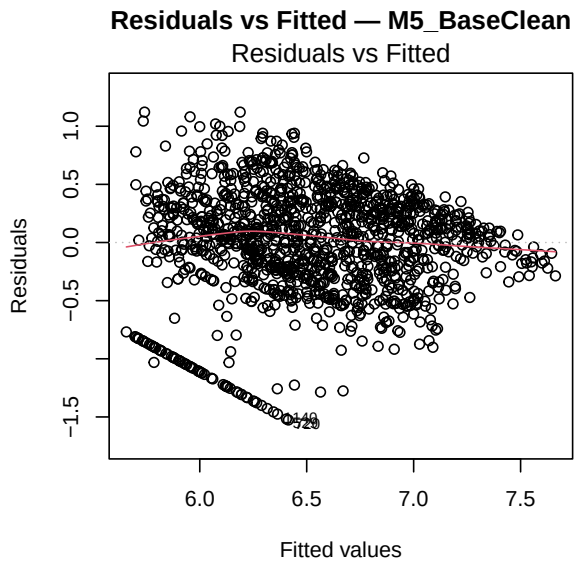

Cook's Distance — M3_Interact
Cook's distance
Cook's distance
Obs. number
692
792
559

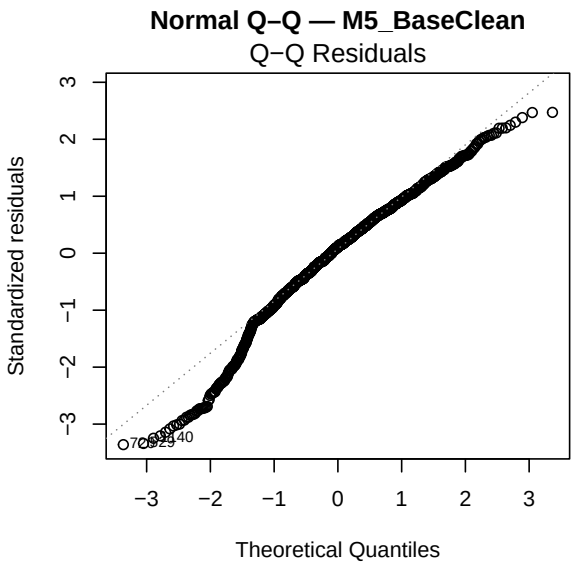

Residuals vs Fitted — M4_Full
Residuals vs Fitted
Residuals
Fitted values
602
942
909

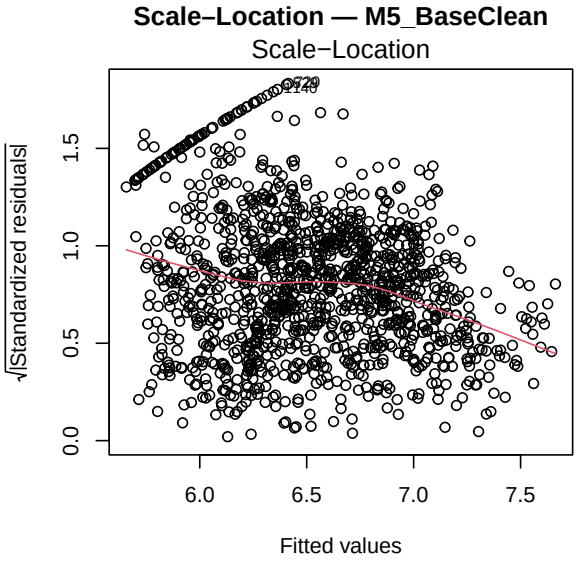

Normal Q–Q — M4_Full
Q–Q Residuals
Standardized residuals
Theoretical Quantiles
602
942
909

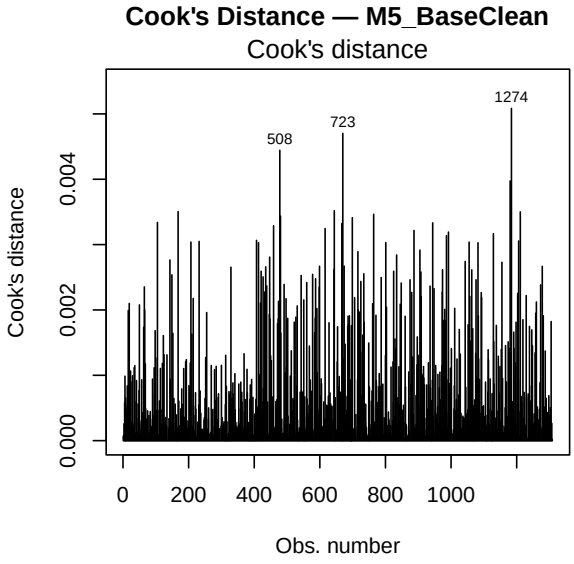

Scale–Location — M4_Full
Scale–Location
√|Standardized residuals|
Fitted values
909
602
942

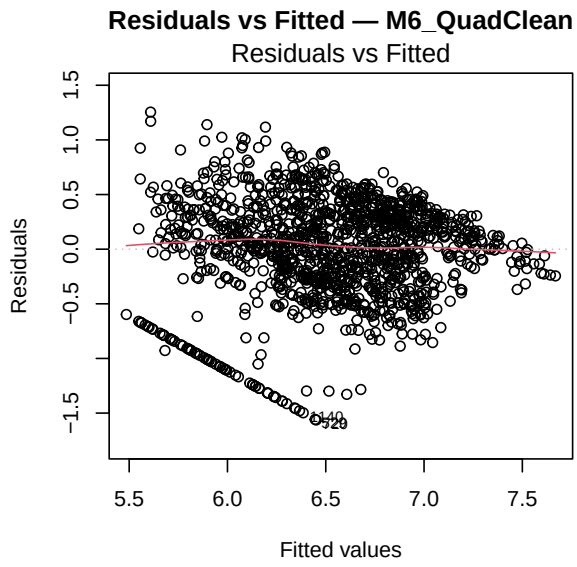

Cook's Distance — M4_Full
Cook's distance
Cook's distance
Obs. number
559
792
692

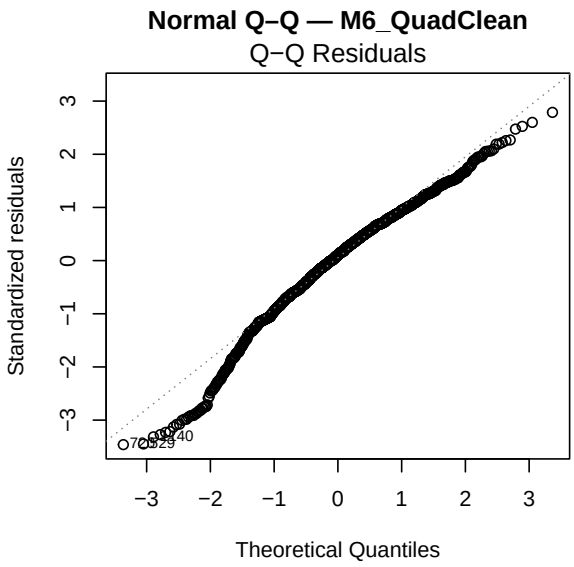

Residuals vs Fitted — M5_BaseClean
Residuals vs Fitted
Residuals
Fitted values

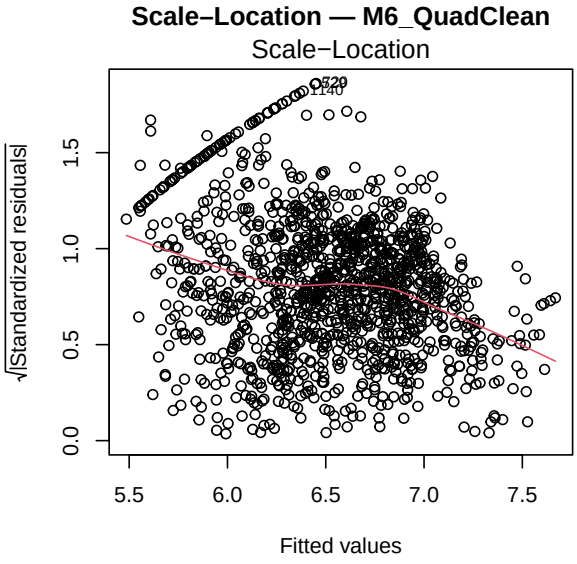

Normal Q–Q — M5_BaseClean
Q–Q Residuals
Standardized residuals
Theoretical Quantiles

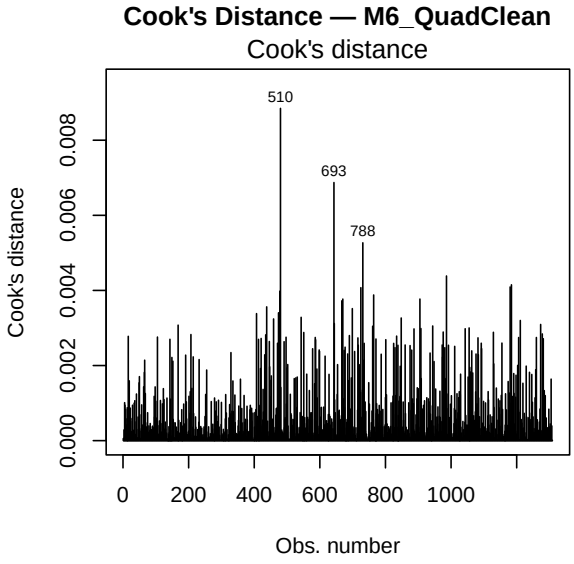

Scale–Location — M5_BaseClean
Scale–Location
√|Standardized residuals|
Fitted values

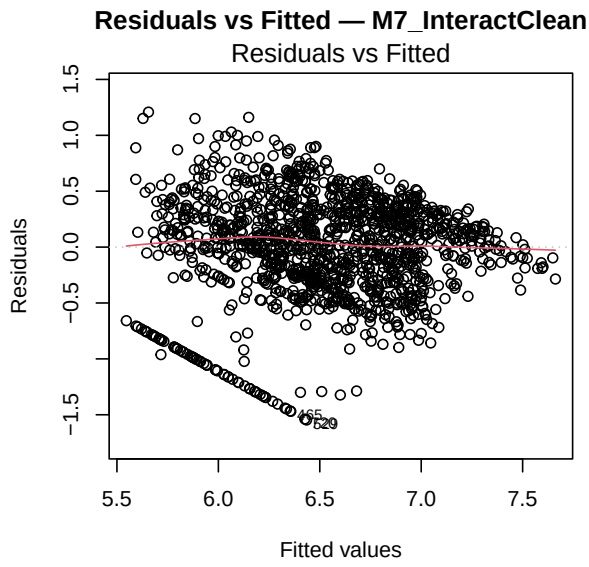

Cook's Distance — M5_BaseClean
Cook's distance
Cook's distance
Obs. number
508
723
1274

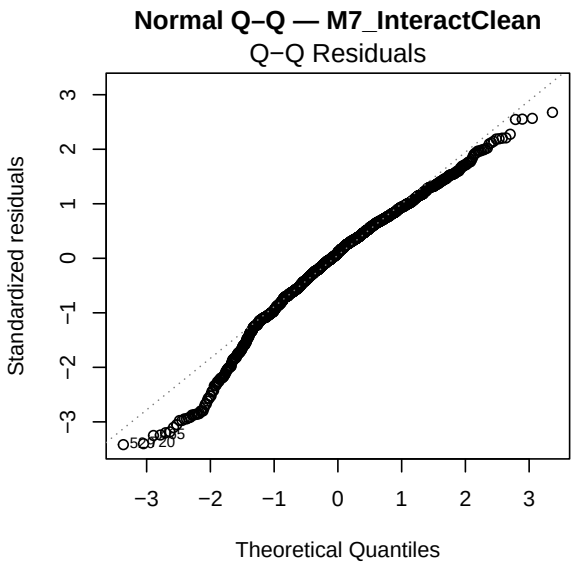

Residuals vs Fitted — M6_QuadClean
Residuals vs Fitted
Residuals
Fitted values
5.5
6.0
6.5
7.0
7.5
−1.5
−0.5
0.0
0.5
1.0
1.5

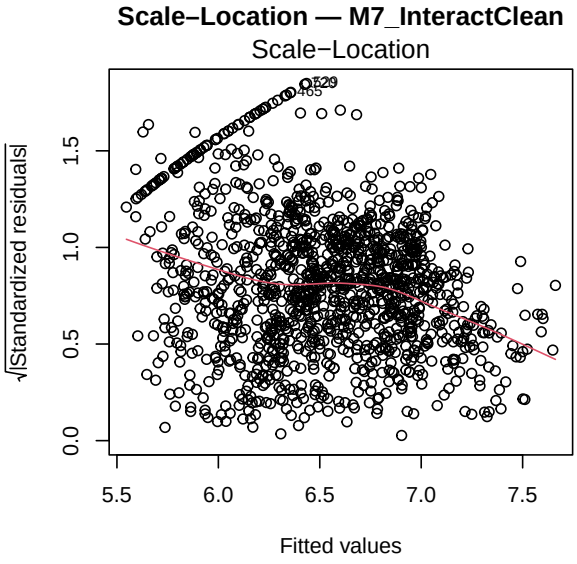

Normal Q−Q — M6_QuadClean
Q−Q Residuals
Standardized residuals
Theoretical Quantiles
−3
−2
−1
0
1
2
3

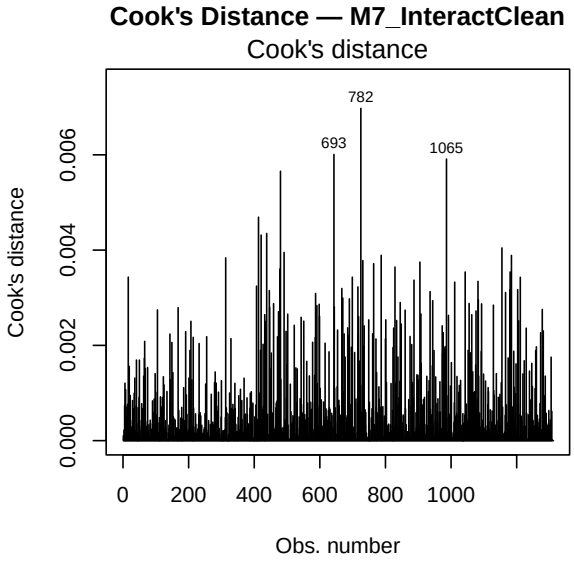

Scale−Location — M6_QuadClean
Scale−Location
√|Standardized residuals|
Fitted values
5.5
6.0
6.5
7.0
7.5
0.0
0.5
1.0
1.5

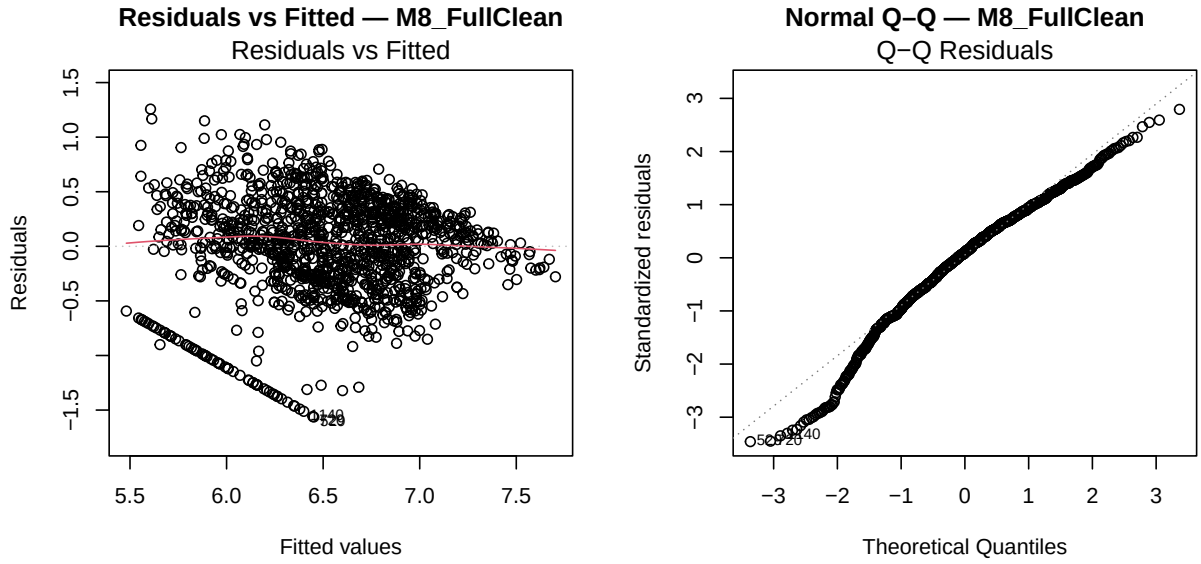

Cook's Distance — M6_QuadClean
Cook's distance
Cook's distance
Obs. number
510
693
788
0
200
400
600
800
1000
0.000
0.002
0.004
0.006
0.008


Residuals vs Fitted — M7_InteractClean
Residuals vs Fitted
Residuals
Fitted values
5.5
6.0
6.5
7.0
7.5
−1.5
−0.5
0.0
0.5
1.0
1.5


Normal Q–Q — M7_InteractClean
Q−Q Residuals
Standardized residuals
Theoretical Quantiles
−3
−2
−1
0
1
2
3


Scale−Location — M7_InteractClean
Scale−Location
√|Standardized residuals|
Fitted values
5.5
6.0
6.5
7.0
7.5
0.0
0.5
1.0
1.5


Cook's Distance — M7_InteractClean
Cook's distance
Cook's distance
Obs. number
782
693
1065
0
200
400
600
800
1000
0.000
0.002
0.004
0.006

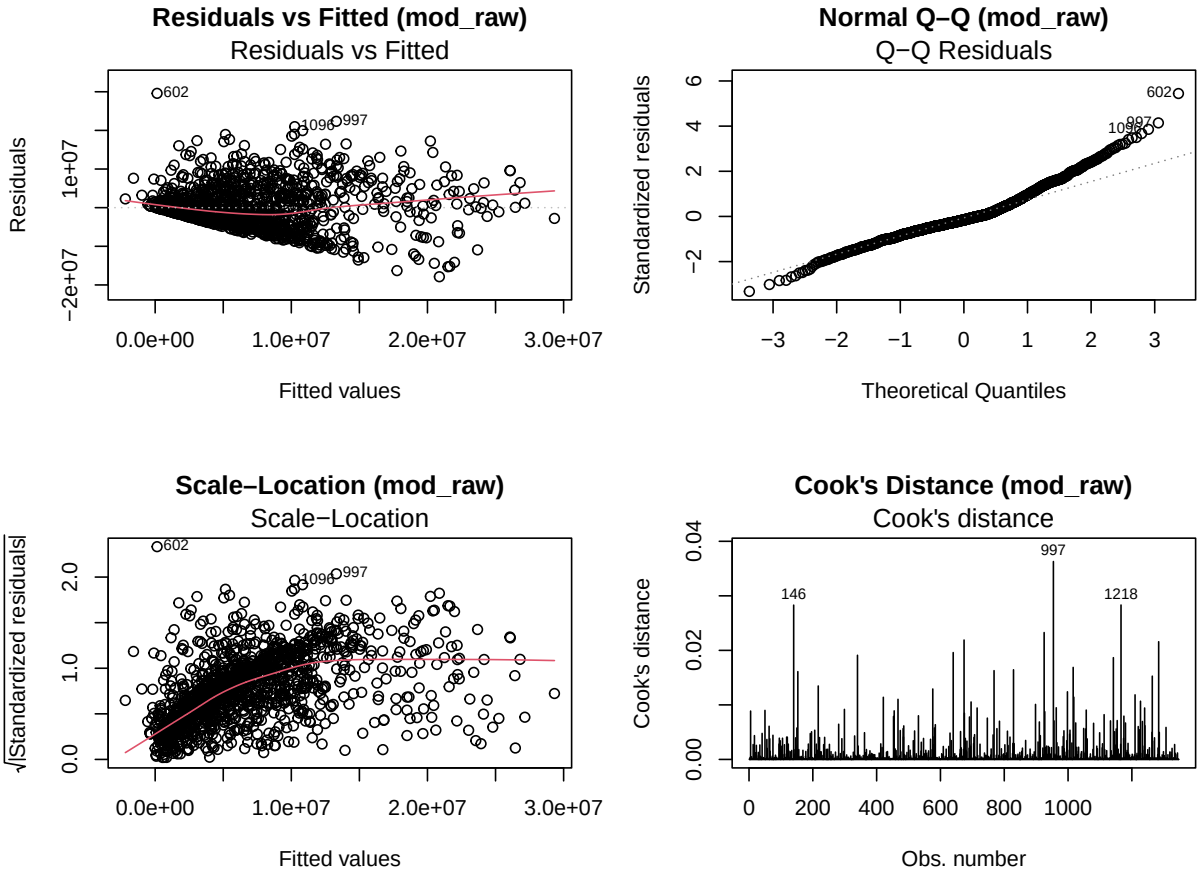


4.x Additional Diagnostics for Model Selection

```
library(car)
library(ggplot2)
library(lmtest)

# 1. Fit raw (un-transformed) model
mod_raw <- lm(
  Salary ~ MP + PTS + TRB + AST + STL + BLK + PF + Play,
  data = nba
)
summary(mod_raw)
```

```
##
## Call:
## lm(formula = Salary ~ MP + PTS + TRB + AST + STL + BLK + PF +
##     Play, data = nba)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -17879418  -3287179   -869343   2590166  29586682
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -512671     422130  -1.214 0.224777
## MP            164655      47341   3.478 0.000521 ***
## PTS           257172      63014   4.081 4.75e-05 ***
## TRB           889093     106402   8.356  < 2e-16 ***
## AST           564201     134762   4.187 3.02e-05 ***
## STL           -62157     592255  -0.105 0.916432
## BLK          -407170     536904  -0.758 0.448365
## PF          -1572820     334964  -4.695 2.93e-06 ***
## PlayYes      4514685     863866   5.226 2.01e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5445000 on 1337 degrees of freedom
## Multiple R-squared:  0.4693, Adjusted R-squared:  0.4661
## F-statistic: 147.8 on 8 and 1337 DF,  p-value: < 2.2e-16
```

```r
# 2×2 diagnostic grid for mod_raw
par(mfrow = c(2, 2))
plot(mod_raw, which = 1, main = "Residuals vs Fitted (mod_raw)")
plot(mod_raw, which = 2, main = "Normal Q-Q (mod_raw)")
plot(mod_raw, which = 3, main = "Scale-Location (mod_raw)")
plot(mod_raw, which = 4, main = "Cook's Distance (mod_raw)")
```

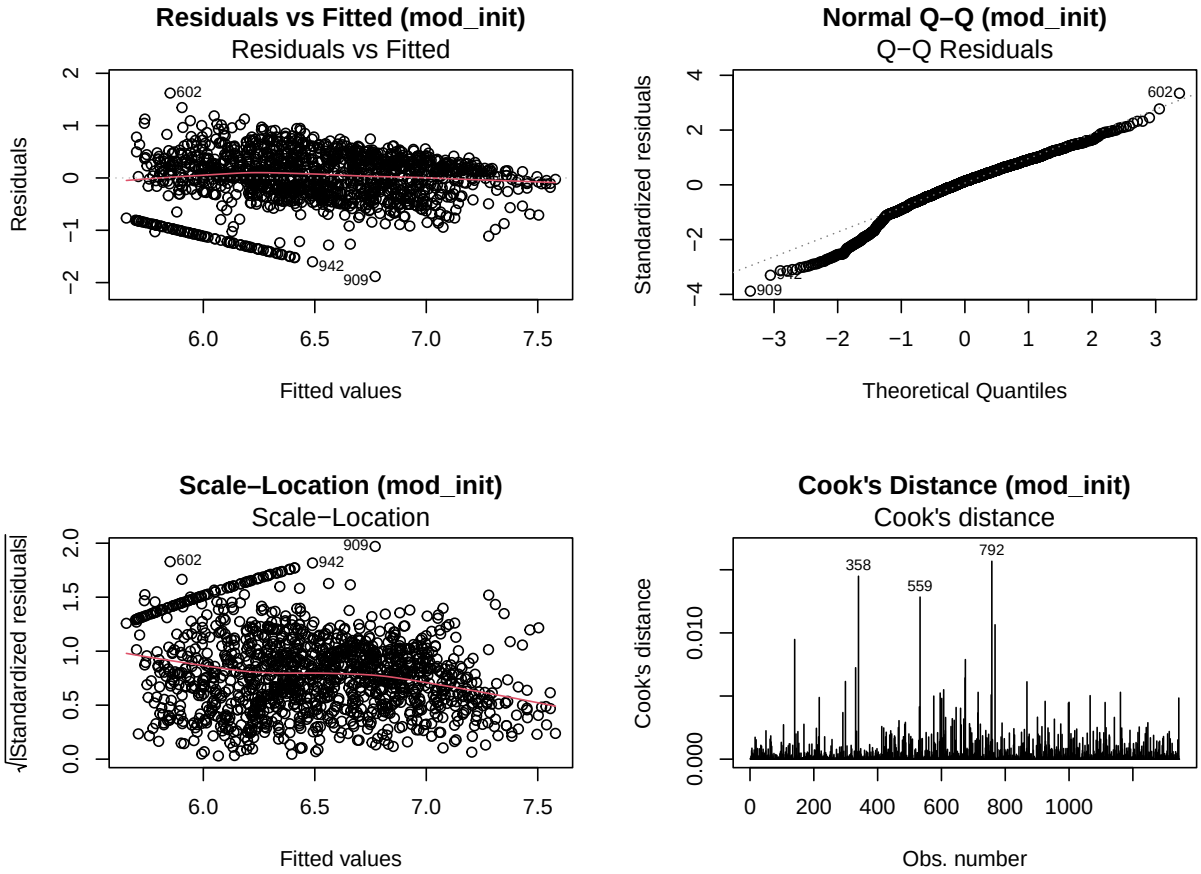



```
par(mfrow = c(1, 1))

# 2. Fit log10-transformed model
mod_init <- lm(
  log10(Salary) ~ MP_c + PTS_c + TRB_c + AST_c + STL_c + BLK_c + Play,
  data = nba
)
summary(mod_init)
```

```
##
## Call:
## lm(formula = log10(Salary) ~ MP_c + PTS_c + TRB_c + AST_c + STL_c +
##     BLK_c + Play, data = nba)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.88291 -0.26054  0.06279  0.32544  1.62192
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.498076   0.013878 468.216  < 2e-16 ***
## MP_c         0.035123   0.003936   8.925  < 2e-16 ***
## PTS_c        0.001075   0.005585   0.193    0.847
```

```
## TRB_c         0.043391   0.009148   4.743 2.33e-06 ***
## AST_c         0.018329   0.012023   1.524    0.128
## STL_c        -0.058404   0.052800  -1.106    0.269
## BLK_c        -0.026231   0.046917  -0.559    0.576
## PlayYes       0.106525   0.076502   1.392    0.164
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4859 on 1338 degrees of freedom
## Multiple R-squared:  0.4196, Adjusted R-squared:  0.4166
## F-statistic: 138.2 on 7 and 1338 DF,  p-value: < 2.2e-16
```

```r
# 2×2 diagnostic grid for mod_init
par(mfrow = c(2, 2))
plot(mod_init, which = 1, main = "Residuals vs Fitted (mod_init)")
plot(mod_init, which = 2, main = "Normal Q-Q (mod_init)")
plot(mod_init, which = 3, main = "Scale-Location (mod_init)")
plot(mod_init, which = 4, main = "Cook's Distance (mod_init)")
```




```r
par(mfrow = c(1, 1))

# 3. Breusch-Pagan test for heteroskedasticity
bp_raw <- bptest(mod_raw)
```

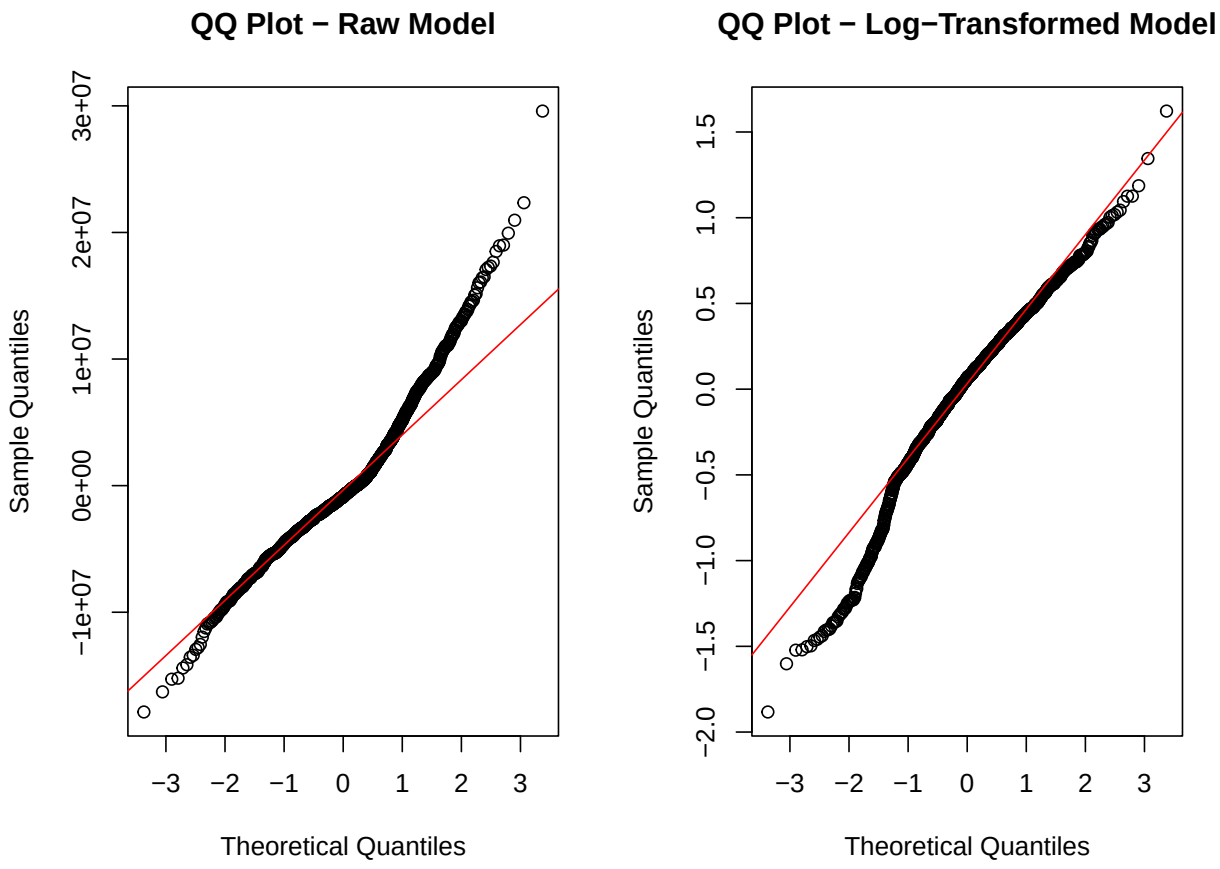
```r
bp_init <- bptest(mod_init)
cat("Breusch-Pagan test p-value (raw model):", format.pval(bp_raw$p.value, digits = 3), "\n")
```

```
## Breusch-Pagan test p-value (raw model): <2e-16
```

```r
cat("Breusch-Pagan test p-value (transformed model):", format.pval(bp_init$p.value, digits = 3), "\n")
```

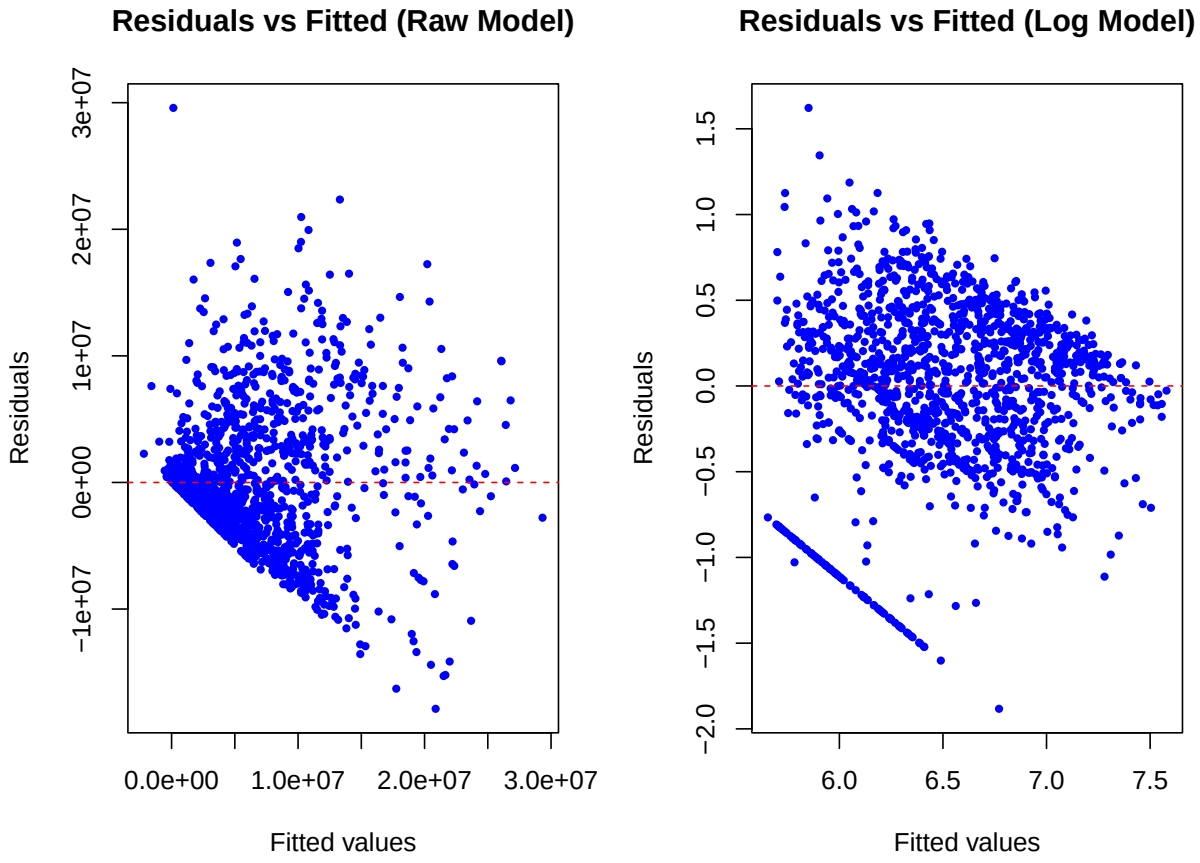
```
## Breusch-Pagan test p-value (transformed model): 4.8e-15
```

```r
# 4. Residual QQ plots comparison
par(mfrow = c(1, 2))
qqnorm(residuals(mod_raw), main = "QQ Plot - Raw Model")
qqline(residuals(mod_raw), col = "red")
qqnorm(residuals(mod_init), main = "QQ Plot - Log-Transformed Model")
qqline(residuals(mod_init), col = "red")
```

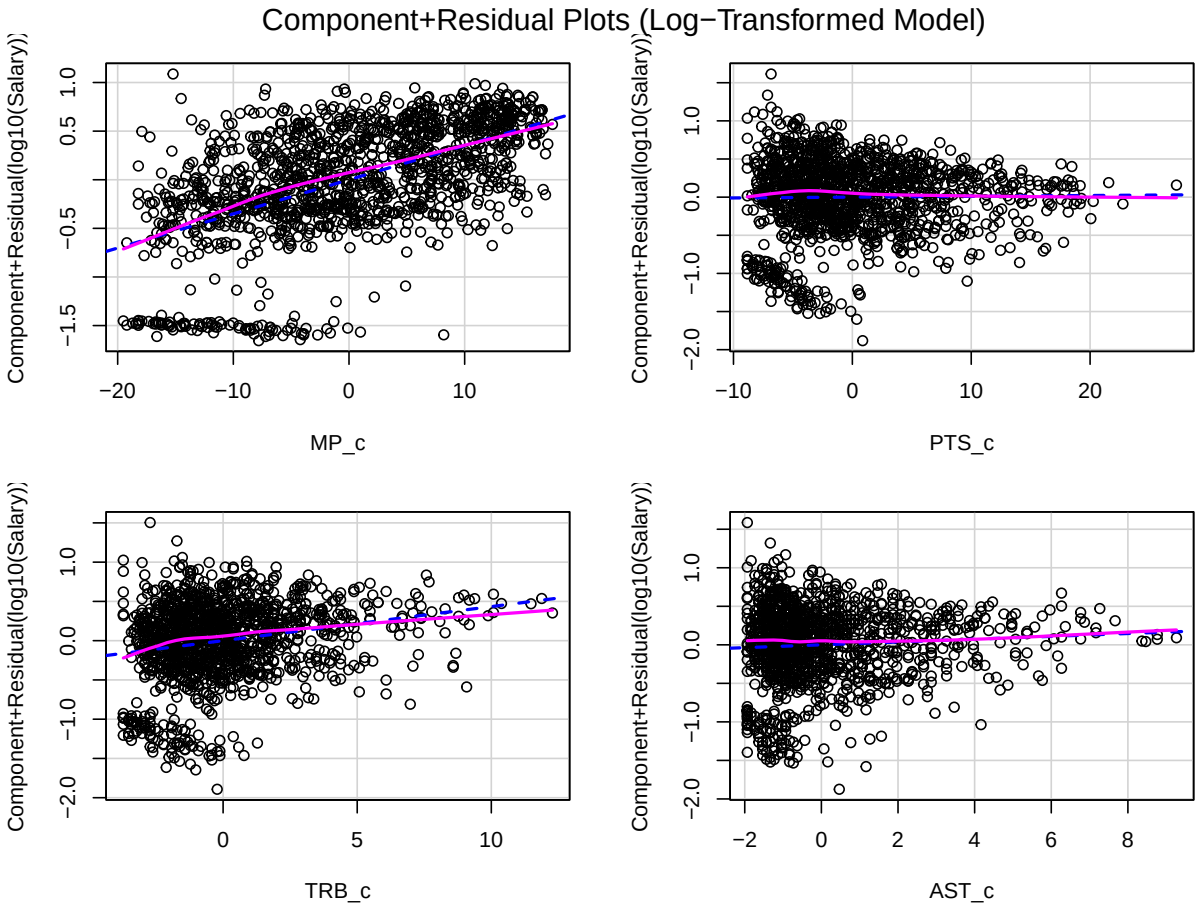



```r
par(mfrow = c(1, 1))

# 5. Residuals vs Fitted values comparison
par(mfrow = c(1, 2))
plot(fitted(mod_raw), residuals(mod_raw),
```

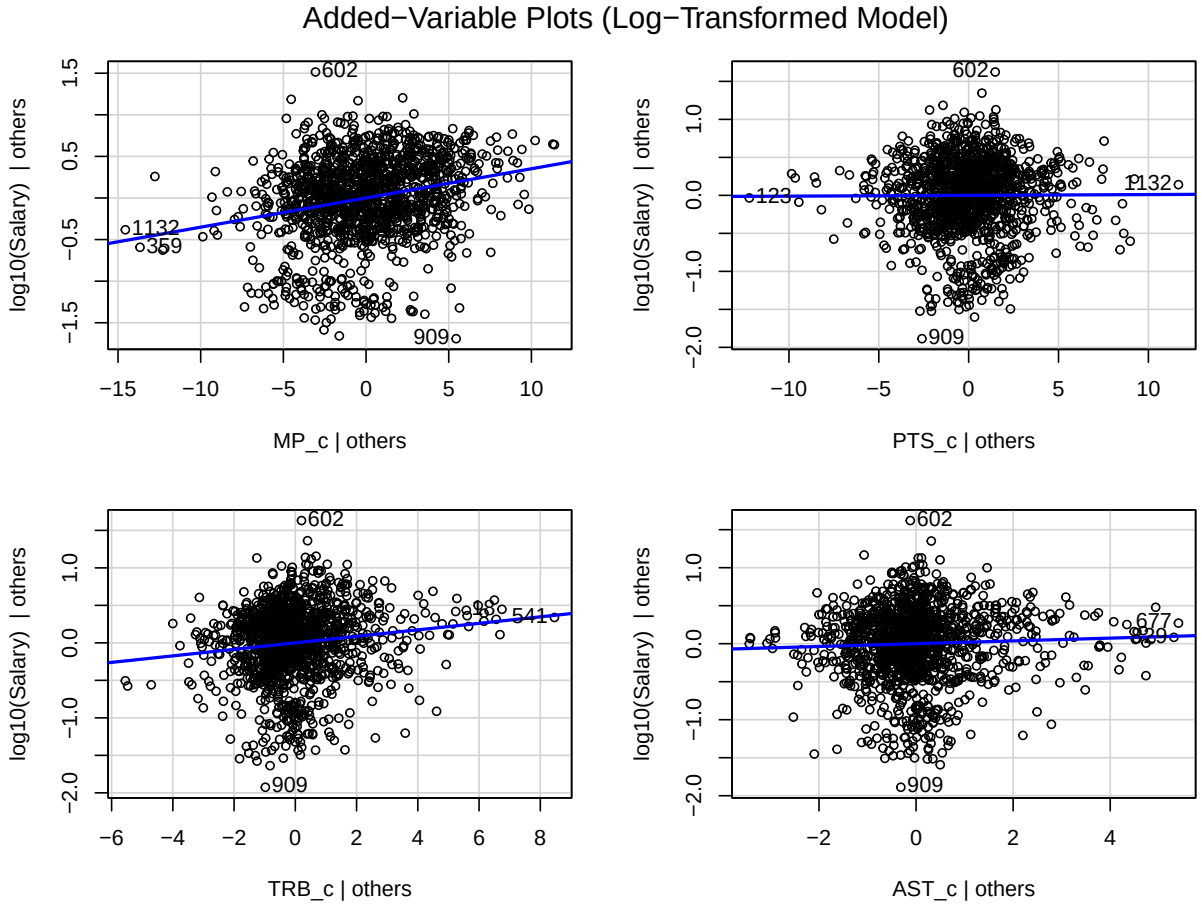
```
     main = "Residuals vs Fitted (Raw Model)",
     xlab = "Fitted values", ylab = "Residuals",
     pch = 16, col = "blue", cex = 0.7)
abline(h = 0, col = "red", lty = 2)
plot(fitted(mod_init), residuals(mod_init),
     main = "Residuals vs Fitted (Log Model)",
     xlab = "Fitted values", ylab = "Residuals",
     pch = 16, col = "blue", cex = 0.7)
abline(h = 0, col = "red", lty = 2)
```



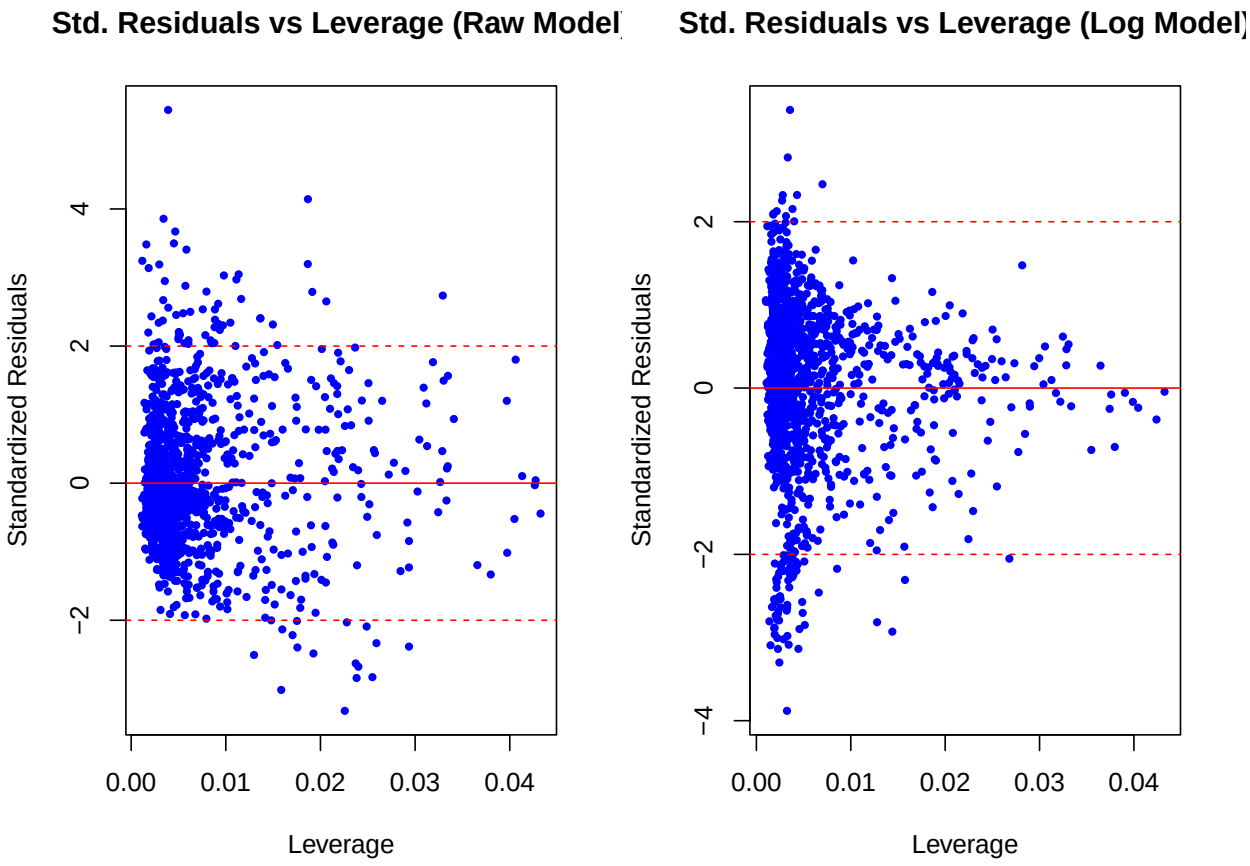

```
par(mfrow = c(1, 1))

# 6. Component+Residual plots for mod_init
crPlots(mod_init, terms = ~ MP_c + PTS_c + TRB_c + AST_c,
        main = "Component+Residual Plots (Log-Transformed Model)")
```



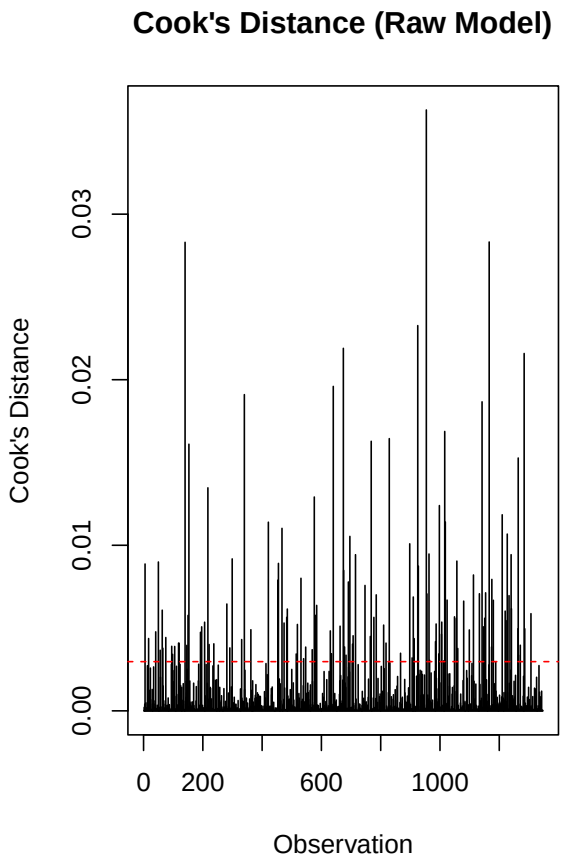

```
# 7. Added-Variable plots for mod_init
avPlots(mod_init, terms = ~ MP_c + PTS_c + TRB_c + AST_c,
        main = "Added-Variable Plots (Log-Transformed Model)")
```

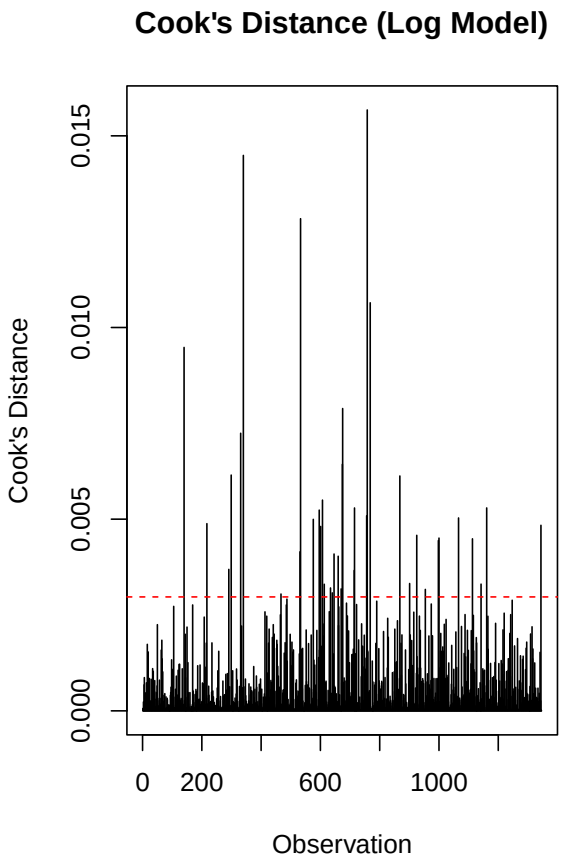



```r
# 8. Standardized residuals vs leverage
par(mfrow = c(1, 2))
plot(hatvalues(mod_raw), rstandard(mod_raw),
     main = "Std. Residuals vs Leverage (Raw Model)",
     xlab = "Leverage", ylab = "Standardized Residuals",
     pch = 16, col = "blue", cex = 0.7)
abline(h = c(-2, 0, 2), col = "red", lty = c(2, 1, 2))
plot(hatvalues(mod_init), rstandard(mod_init),
     main = "Std. Residuals vs Leverage (Log Model)",
     xlab = "Leverage", ylab = "Standardized Residuals",
     pch = 16, col = "blue", cex = 0.7)
abline(h = c(-2, 0, 2), col = "red", lty = c(2, 1, 2))
```

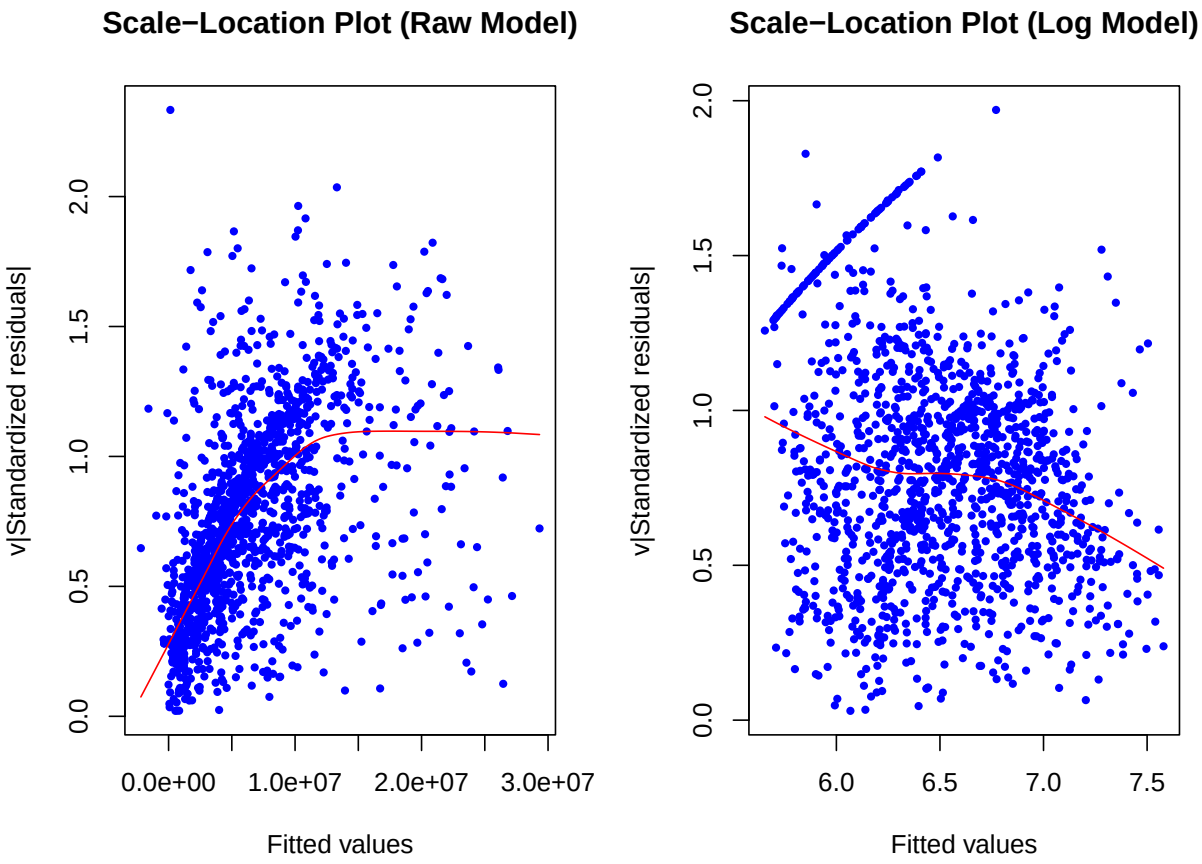



```r
par(mfrow = c(1, 1))

# 9. Cook's distance comparison
par(mfrow = c(1, 2))
cooksd_raw <- cooks.distance(mod_raw)
cooksd_init <- cooks.distance(mod_init)
cutoff <- 4/nrow(nba)
plot(cooksd_raw, type = "h",
     main = "Cook's Distance (Raw Model)",
     ylab = "Cook's Distance", xlab = "Observation")
abline(h = cutoff, col = "red", lty = 2)
plot(cooksd_init, type = "h",
     main = "Cook's Distance (Log Model)",
     ylab = "Cook's Distance", xlab = "Observation")
abline(h = cutoff, col = "red", lty = 2)
```






```r
par(mfrow = c(1, 1))

# 10. Scale-Location plots comparison
par(mfrow = c(1, 2))
plot(fitted(mod_raw), sqrt(abs(rstandard(mod_raw))),
     main = "Scale-Location Plot (Raw Model)",
     xlab = "Fitted values",
     ylab = "√|Standardized residuals|",
     pch = 16, col = "blue", cex = 0.7)
lines(lowess(fitted(mod_raw), sqrt(abs(rstandard(mod_raw)))), col = "red")
plot(fitted(mod_init), sqrt(abs(rstandard(mod_init))),
     main = "Scale-Location Plot (Log Model)",
     xlab = "Fitted values",
     ylab = "√|Standardized residuals|",
     pch = 16, col = "blue", cex = 0.7)
lines(lowess(fitted(mod_init), sqrt(abs(rstandard(mod_init)))), col = "red")
```




```
par(mfrow = c(1, 1))
```

Residual Diagnostics Analysis

1. **Raw Model vs Log-Transformed Model**
   - The raw model shows severe heteroskedasticity with residuals fanning out at higher fitted values.
   - The log-transformed model greatly improves residual homoskedasticity.
   - QQ plots show the log transformation produces more normally distributed residuals.
2. **Heteroskedasticity Tests**
   - Breusch-Pagan test confirms significant heteroskedasticity in the raw model ($p < 0.001$).
   - Log transformation reduces but doesn't completely eliminate heteroskedasticity.
3. **Influential Observations**
   - Cook's distance plots identify more extreme influential points in the raw model.
   - Log transformation reduces the influence of high-salary observations.
4. **Linearity Assessment**
   - Component-residual plots show improved linearity in the log-transformed model.
   - Some non-linearity remains in the relationships with key predictors.
5. **Leverage Points**

- Both models have observations with high leverage.
- The log transformation reduces the impact of these high-leverage points.

6. **Overall Assessment**

- The log transformation substantially improves model diagnostics.
- Removing influential observations (as done in our clean models) further improves model fit.
- The quadratic term for Minutes Played helps address remaining non-linearity.