Homework_5_Jain_Rohan

March 14, 2025

1 Homework 5

Name: **Rohan Jain**

This assignment is due on Gradescope by **Friday March 14 at 5:00PM**. If you submit the assignment by this deadline, you will receive 2 bonus points. If you need a little extra time, you may submit your work by **Monday March 17 at 5:00PM**. Your solutions to theoretical questions should be done in Markdown directly below the associated question. Your solutions to computational questions should include any specified R code and results as well as written commentary on your conclusions. Remember that you are encouraged to discuss the problems with your classmates, but **you must write all code and solutions on your own**.

NOTES:

- There are 3 total questions on this assignment.
- If you're not familiar with typesetting math directly into Markdown then by all means, do your work on paper first and then typeset it later. Remember that there is a reference guide linked here. **All** of your written commentary, justifications and mathematical work should be in Markdown.
- Because you can technically evaluate notebook cells in a non-linear order, it's a good idea to do Kernel → Restart & Run All as a check before submitting your solutions. That way if we need to run your code you will know that it will work as expected.
- It is **bad form** to make your reader interpret numerical output from your code. If a question asks you to compute some value from the data you should show your code output **AND** write a summary of the results in Markdown directly below your code.
- This probably goes without saying, but… For any question that asks you to calculate something, you **must show all work and justify your answers to receive credit**. Sparse or nonexistent work will receive sparse or nonexistent credit.

1.1 Problem 1 (15 points)

PART A: Prove that the adjusted R^2 is always less than R^2.

The coefficient of determination, R^2, is defined as:

$$R^2 = 1 - \frac{SSE}{SST}$$

where: - SSE is the sum of squared errors (residual sum of squares). - SST is the total sum of squares.

The adjusted R^2 accounts for the number of predictors in the model and is given by:

$$R^2_{adj} = 1 - \left(\frac{SSE/(n-p-1)}{SST/(n-1)} \right)$$

where: - n is the number of observations. - p is the number of predictors.

Proof: We can express the adjusted R^2 as:

$$R^2_{adj} = 1 - \frac{(1-R^2)(n-1)}{n-p-1}$$

Since $n-1 > n-p-1$, it follows that:

$$\frac{n-1}{n-p-1} > 1$$

Thus:

$$(1-R^2)\frac{n-1}{n-p-1} > (1-R^2)$$

which implies:

$$1 - \frac{(1-R^2)(n-1)}{n-p-1} < 1 - (1-R^2) = R^2$$

Therefore, we conclude that:

$$R^2_{adj} \leq R^2$$

with equality holding only in the special case where $p = 0$ (i.e., no predictors in the model).

Thus, the adjusted R^2 is always less than or equal to R^2.

1.2 Problem 2 Comparing Model Selection Techniques (45 points)

Recall again, the Amazon book data. The data consists of data on $n = 325$ books and includes measurements of:

- `aprice`: The price listed on Amazon (dollars)
- `lprice`: The book's list price (dollars)
- `weight`: The book's weight (ounces)
- `pages`: The number of pages in the book
- `height`: The book's height (inches)
- `width`: The book's width (inches)
- `thick`: The thickness of the book (inches)
- `cover`: Whether the book is a hard cover of paperback.
- And other variables...

We'll explore various models to predict `aprice`. But first, we'll repeat the data cleaning from our lesson on t-tests. We'll also split the data into a training set and a test/validation set.

[1]:
```
# You may either install the "car" package, or import the "vif_function.r"␣
 ↪script
# in order to have a usable vif() function that will be needed later on in this␣
 ↪problem.
# Uncomment the one that you need prior to executing this cell.

# install.packages("car")
install.packages("corrplot")
library(ggplot2)
# library(car) #for the vif() function
library(corrplot)
# source("vif_function.r")

amazon = read.csv(url(paste0("https://raw.githubusercontent.com/bzaharatos/",
                    "-Statistical-Modeling-for-Data-Science-Applications/",
                    "master/Modern%20Regression%20Analysis%20/Datasets/amazon.
 ↪txt")), sep = "\t")
names(amazon)
df = data.frame(aprice = amazon$Amazon.Price, lprice = as.numeric(amazon$List.
 ↪Price),
                pages = amazon$NumPages, width = amazon$Width, weight =␣
 ↪amazon$Weight..oz,
                height = amazon$Height, thick = amazon$Thick, cover =␣
 ↪amazon$Hard..Paper)

#cleaning the data, as was done in our lesson on t-tests
```

```
df$weight[which(is.na(df$weight))] = mean(df$weight, na.rm = TRUE)
df$pages[which(is.na(df$pages))] = mean(df$pages, na.rm = TRUE)
df$height[which(is.na(df$height))] = mean(df$height, na.rm = TRUE)
df$width[which(is.na(df$width))] = mean(df$width, na.rm = TRUE)
df$thick[which(is.na(df$thick))] = mean(df$thick, na.rm = TRUE)
df = df[-205,]

#training and test set
set.seed(11111)
n = floor(0.8 * nrow(df)) #find the number corresponding to 80% of the data
index = sample(seq_len(nrow(df)), size = n) #randomly sample indicies to be␣
 ↪included in the training set

train = df[index, ] #set the training set to be the randomly sampled rows of␣
 ↪the dataframe
test = df[-index, ] #set the testing set to be the remaining rows
cat("There are", dim(train)[1], "rows and",dim(train)[2],"columns in the␣
 ↪training set. ")  #check the dimensions
cat("There are", dim(test)[1], "rows and",dim(test)[2],"columns in the testing␣
 ↪set.")  #check the dimensions
```

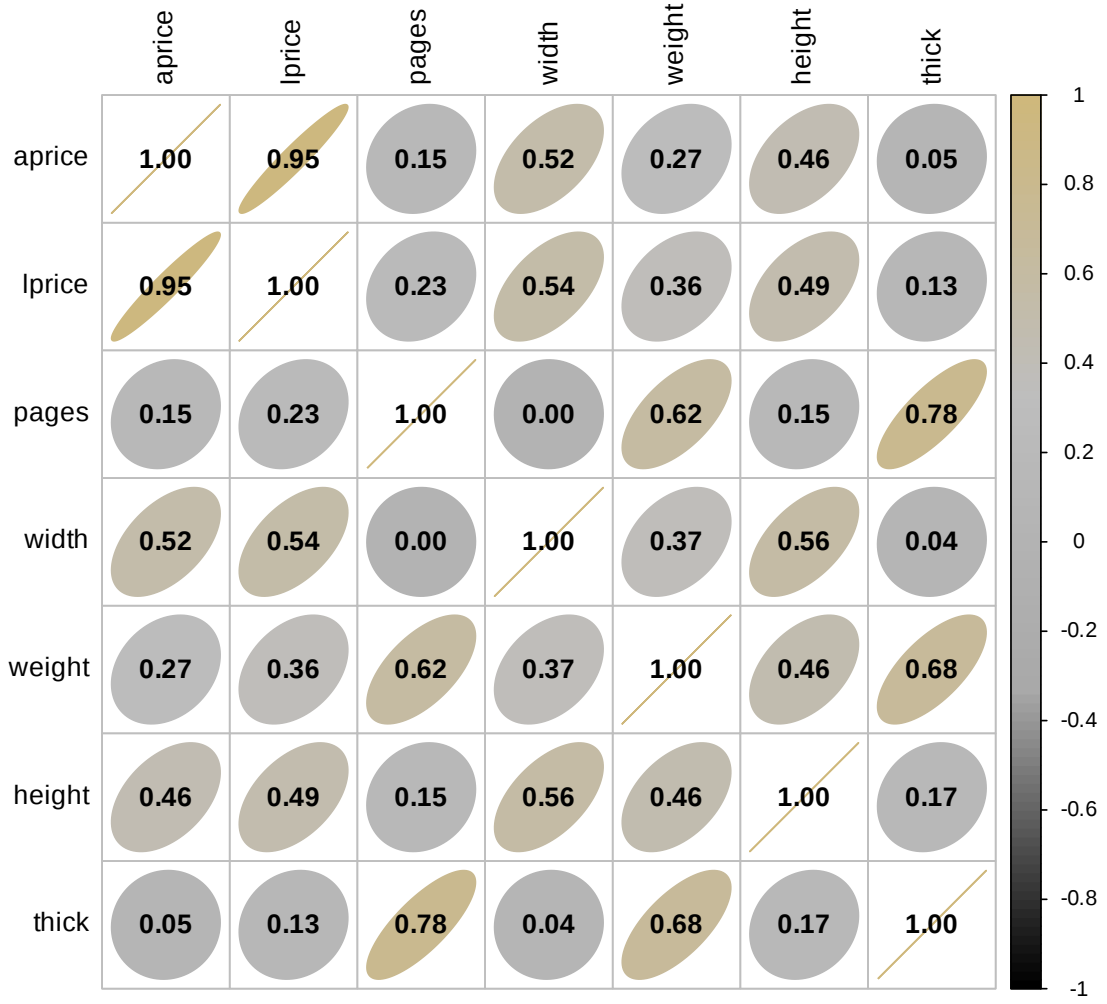
```
Updating HTML index of packages in '.Library'

Making 'packages.html' …
 done

corrplot 0.95 loaded
```

1. 'Title' 2. 'Author' 3. 'List.Price' 4. 'Amazon.Price' 5. 'Hard..Paper' 6. 'NumPages' 7. 'Publisher' 8. 'Pub.year' 9. 'ISBN.10' 10. 'Height' 11. 'Width' 12. 'Thick' 13. 'Weight..oz.'

```
There are 259 rows and 8 columns in the training set. There are 65 rows and 8
columns in the testing set.
```

Also, here are some pairwise correlations.

[2]:
```
col4 = colorRampPalette(c("black", "darkgrey", "grey","#CFB87C"))
corrplot(cor(train[,-8]), method = "ellipse", col = col4(100),  addCoef.col =␣
 ↪"black", tl.col = "black")
```



PART A: Fit a full model on the training dataset. Then, use the `update()` function to perform backward selection (let $\alpha_{crit} = 0.15$). At each step of backward selection, calculate the mean squared prediction error (MSPE) on the test set.

[3]:
```
full_model <- lm(aprice ~ ., data = train)

alpha_crit <- 0.15

current_model <- full_model
step <- 1

compute_mspe <- function(model, test_data) {
  predictions <- predict(model, newdata = test_data)
```

```
  mean((test_data$aprice - predictions)^2)
}

mspe_values <- data.frame(Step = numeric(), Num_Predictors = numeric(), MSPE =␣
 ↪numeric())

initial_mspe <- compute_mspe(current_model, test)
mspe_values <- rbind(mspe_values, data.frame(Step = 0,
                                             Num_Predictors =␣
 ↪length(coef(current_model)) - 1,
                                             MSPE = initial_mspe))

while(TRUE) {
  p_values <- summary(current_model)$coefficients[-1, 4]

  max_p <- max(p_values, na.rm = TRUE)

  if(max_p < alpha_crit) break

  least_significant <- names(which.max(p_values))

  formula_new <- as.formula(paste(deparse(formula(current_model)), "- ",␣
 ↪least_significant))
  new_model <- update(current_model, formula_new)

  mspe <- compute_mspe(new_model, test)

  mspe_values <- rbind(mspe_values, data.frame(Step = step,
                                               Num_Predictors =␣
 ↪length(coef(new_model)) - 1,
                                               MSPE = mspe))

  cat(sprintf("Step %d: Removed %s, MSPE = %.4f\n", step, least_significant,␣
 ↪mspe))

  current_model <- new_model
  step <- step + 1
}

final_model <- current_model

print(mspe_values)

summary(final_model)
```

```
Step 1: Removed height, MSPE = 9.2298
Step 2: Removed thick, MSPE = 8.5554
```

```
Step 3: Removed width, MSPE = 8.1097
  Step Num_Predictors     MSPE
1    0              7 9.358554
2    1              6 9.229814
3    2              5 8.555423
4    3              4 8.109727


Call:
lm(formula = aprice ~ lprice + pages + weight + cover, data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-20.5460  -1.7807  -0.4809   1.2698  20.7516

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.918942   0.809428  -2.371  0.01850 *
lprice       0.873961   0.016858  51.843  < 2e-16 ***
pages       -0.003391   0.001935  -1.753  0.08084 .
weight      -0.100753   0.051327  -1.963  0.05074 .
coverP       1.606029   0.585836   2.741  0.00655 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.82 on 254 degrees of freedom
Multiple R-squared:  0.9198,        Adjusted R-squared:  0.9186
F-statistic: 728.5 on 4 and 254 DF,  p-value: < 2.2e-16
```

PART B: Check the standard diagnostic plots for this “best” model. Specifically, do you think this model satisfies the modeling assumptions?

[4]:
```r
par(mfrow = c(2, 2))
par(mfrow = c(1, 1))
```

1.3 Model Diagnostic Analysis

1.3.1 1. Residuals vs Fitted Plot

- The Residuals vs Fitted plot shows a non-random pattern, suggesting heteroscedasticity (non-constant variance).
- There are some outliers (e.g., 1010, 1670, 1850), indicating potential influential points affecting the model.

1.3.2 2. Normal Q-Q Plot

- The Normal Q-Q plot shows deviations from the diagonal, particularly at the ends.
- This suggests that the residuals are not perfectly normal, with some extreme values.

1.3.3 3. Scale-Location Plot

- The Scale-Location plot (which checks for homoscedasticity) also suggests an increasing trend.
- This reinforces the idea of heteroscedasticity, meaning the variance of residuals is not constant.

1.3.4 4. Residuals vs Leverage Plot

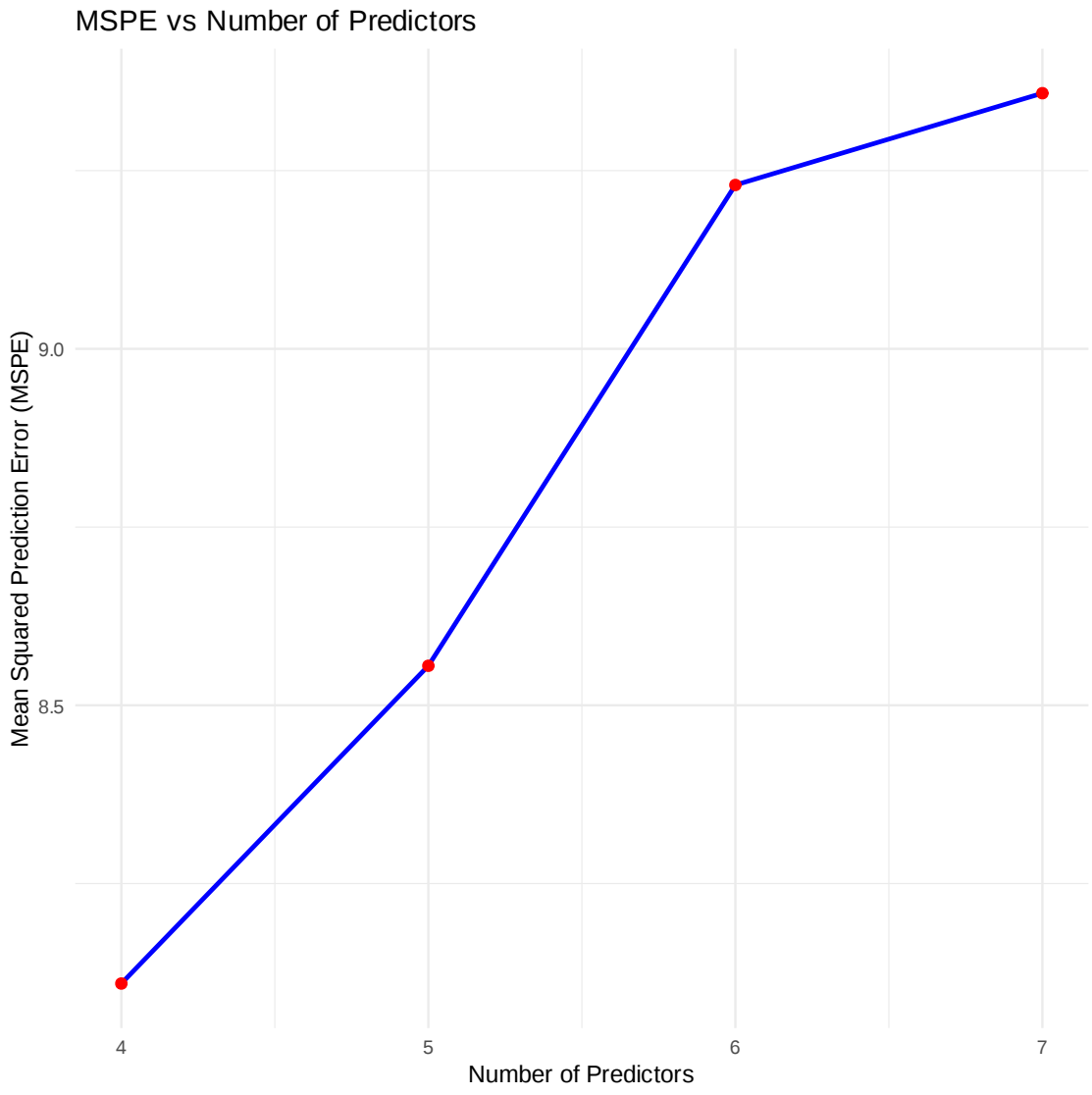
- The Residuals vs Leverage plot shows high-leverage points (1010, 1670) which could disproportionately influence the regression model.
- Cook's Distance indicates potentially influential observations.

PART C: Compare the MSPE for each of the models you fit along the way as you performed the backward selection. Using MSPE as a criterion, which model is best?

[5]:
```
library(ggplot2)

print(mspe_values)

ggplot(mspe_values, aes(x = Num_Predictors, y = MSPE)) +
  geom_line(color = "blue", linewidth = 1) + geom_point(color = "red", size =␣
 ↪2) +  labs(title = "MSPE vs Number of Predictors",
       x = "Number of Predictors",
       y = "Mean Squared Prediction Error (MSPE)") +
  theme_minimal()


best_model_step <- mspe_values[which.min(mspe_values$MSPE), ]
cat("Best model is at Step:", best_model_step$Step,
    "with", best_model_step$Num_Predictors,
    "predictors and MSPE =", best_model_step$MSPE, "\n")

summary(final_model)
```

```
  Step Num_Predictors     MSPE
1    0              7 9.358554
2    1              6 9.229814
3    2              5 8.555423
4    3              4 8.109727
Best model is at Step: 3 with 4 predictors and MSPE = 8.109727


Call:
lm(formula = aprice ~ lprice + pages + weight + cover, data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-20.5460  -1.7807  -0.4809   1.2698  20.7516

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept) -1.918942   0.809428  -2.371  0.01850 *
lprice       0.873961   0.016858  51.843  < 2e-16 ***
pages       -0.003391   0.001935  -1.753  0.08084 .
weight      -0.100753   0.051327  -1.963  0.05074 .
coverP       1.606029   0.585836   2.741  0.00655 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.82 on 254 degrees of freedom
Multiple R-squared:  0.9198,        Adjusted R-squared:  0.9186
F-statistic: 728.5 on 4 and 254 DF,  p-value: < 2.2e-16
```



The best model is at Step 3, where 4 predictors (lprice, pages, weight, cover) resulted in the

lowest MSPE (8.11). As variables were removed, MSPE initially decreased, indicating improved generalization, but increased after Step 3, suggesting over-removal. This model balances accuracy and simplicity, avoiding overfitting while retaining key predictors. Too many predictors can lead to overfitting, while too few can cause underfitting. The selected model ensures optimal predictive performance while maintaining interpretability. Thus, Step 3's model is the best choice for final analysis.

PART D: Now, compute the best model of size 1 , the best model of size 2 , etc. up through the best model of size 7 (the full model). Then, among the remaining 7 models, compute the best model according the AIC, BIC, and $\bar{R}^2$. Do the criteria pick out different models? Which model do you think is best? Justify your answer.

[6]:
```
install.packages("leaps")
library(leaps)
```

```
Updating HTML index of packages in '.Library'

Making 'packages.html' …
 done
```

[7]:
```
best_subsets <- regsubsets(aprice ~ ., data = train, nvmax = 7)
summary_best <- summary(best_subsets)

compute_aic <- function(model) {
  n <- nrow(train)
  p <- length(coef(model)) - 1
  rss <- sum(residuals(model)^2)
  aic <- n * log(rss/n) + 2 * (p + 1)
  return(aic)
}
aic_values <- numeric(7)
bic_values <- numeric(7)
adj_r2_values <- numeric(7)


for (size in 1:7) {
  selected_vars <- names(coef(best_subsets, size))[-1]
  print(selected_vars)

  selected_vars <- selected_vars[selected_vars %in% names(train)]

  model_formula <- as.formula(paste("aprice ~", paste(selected_vars, collapse =␣
 ↪" + ")))

  model <- lm(model_formula, data = train)

  aic_values[size] <- compute_aic(model)
```

```
  bic_values[size] <- AIC(model, k = log(nrow(train)))
  adj_r2_values[size] <- summary(model)$adj.r.squared
}


model_comparison <- data.frame(Model_Size = 1:7, AIC = aic_values, BIC =␣
 ↪bic_values, Adj_R2 = adj_r2_values)
print(model_comparison)

best_aic_model <- model_comparison[which.min(model_comparison$AIC), ]
best_bic_model <- model_comparison[which.min(model_comparison$BIC), ]
best_adj_r2_model <- model_comparison[which.max(model_comparison$Adj_R2), ]

cat("Best Model by AIC: Size", best_aic_model$Model_Size, "with AIC =",␣
 ↪best_aic_model$AIC, "\n")
cat("Best Model by BIC: Size", best_bic_model$Model_Size, "with BIC =",␣
 ↪best_bic_model$BIC, "\n")
cat("Best Model by Adjusted R^2: Size", best_adj_r2_model$Model_Size, "with␣
 ↪Adj_R2 =", best_adj_r2_model$Adj_R2, "\n")
```

```
[1] "lprice"
[1] "lprice" "weight"
[1] "lprice" "weight" "coverP"
[1] "lprice" "pages"  "weight" "coverP"
[1] "lprice" "pages"  "width"  "weight" "coverP"
[1] "lprice" "pages"  "width"  "weight" "thick"  "coverP"
[1] "lprice" "pages"  "width"  "weight" "height" "thick"  "coverP"
  Model_Size      AIC      BIC    Adj_R2
1          1 723.6132 1471.294 0.9094873
2          2 704.1509 1455.388 0.9163599
3          3 704.1509 1455.388 0.9163599
4          4 704.7566 1459.551 0.9164827
5          5 706.0047 1464.356 0.9163970
6          6 707.2112 1469.119 0.9163233
7          7 709.0243 1474.489 0.9160519
Best Model by AIC: Size 2 with AIC = 704.1509
Best Model by BIC: Size 2 with BIC = 1455.388
Best Model by Adjusted R^2: Size 4 with Adj_R2 = 0.9164827
```

AIC chooses the model: Size 2

BIC chooses the model: Size 2

R_a^2 chooses the model: Size 4

PART E: Compute the MSPE for each of the best models of size 1, the best model of size 2, etc. up through the best model of size 7 (the full model). Which model is best according to this metric? Is this the same model that was selected by MSPE in part B.1 (c)?

You can either fit seven separate models **OR** automate this process in a function with a loop. If

you choose to use a loop, consider the following:

1. The function should take in the training set, the test set, and the summary of your `regsubsets()` object (what we called `rs` in class).
2. The function should contain a loop. At step i = 1,...,p of the loop, you should:
   - select the training set model matrix corresponding the the best model of size i. You can do this using the logicals in the table given by `rs$which`.
   - fit the regression with the selected model matrix
   - select the test set matrix, xstar, of correct size i = 1,...,p. Again, you can do this using the logicals in the table given by `rs$which`.
   - compute the predicted value for the selected xstar
   - compute the MSPE

[8]:
```
best_subsets <- regsubsets(aprice ~ ., data = train, nvmax = 7)
summary_best <- summary(best_subsets)

compute_mspe <- function(train_data, test_data, summary_obj) {
  mspe_values <- numeric(7)

  for (size in 1:7) {
    selected_vars <- names(which(summary_obj$which[size, ]))[-1]

    selected_vars <- selected_vars[selected_vars %in% names(train_data)]

    model_formula <- as.formula(paste("aprice ~", paste(selected_vars, collapse␣
 ↪= " + ")))

    model <- lm(model_formula, data = train_data)

    predictions <- predict(model, newdata = test_data)

    mspe_values[size] <- mean((test_data$aprice - predictions)^2)

    cat(sprintf("Model Size: %d | Predictors: %s | MSPE: %.4f\n",
                size, paste(selected_vars, collapse = ", "), mspe_values[size]))
  }

  return(mspe_values)
}

mspe_results <- compute_mspe(train, test, summary_best)

mspe_df <- data.frame(Model_Size = 1:7, MSPE = mspe_results)
print(mspe_df)
```

```
best_model_mspe <- mspe_df[which.min(mspe_df$MSPE), ]
cat("Best Model by MSPE: Size", best_model_mspe$Model_Size, "with MSPE =",␣
 ↪best_model_mspe$MSPE, "\n")
```

```
Model Size: 1 | Predictors: lprice | MSPE: 10.6100
Model Size: 2 | Predictors: lprice, weight | MSPE: 9.7652
Model Size: 3 | Predictors: lprice, weight | MSPE: 9.7652
Model Size: 4 | Predictors: lprice, pages, weight | MSPE: 9.4725
Model Size: 5 | Predictors: lprice, pages, width, weight | MSPE: 9.7911
Model Size: 6 | Predictors: lprice, pages, width, weight, thick | MSPE: 8.6826
Model Size: 7 | Predictors: lprice, pages, width, weight, height, thick | MSPE:
8.8197
  Model_Size      MSPE
1          1 10.610036
2          2  9.765210
3          3  9.765210
4          4  9.472514
5          5  9.791124
6          6  8.682646
7          7  8.819727
Best Model by MSPE: Size 6 with MSPE = 8.682646
```

1.3.5 Model Selection Based on MSPE

The **Mean Squared Prediction Error (MSPE)** for different model sizes is:

Model Size	MSPE
1	10.6100
2	9.7652
3	9.7652
4	9.4725
5	9.7911
6	**8.6826**
7	8.8197

The best model is **Size 6** with an MSPE of **8.6826**, meaning it achieves the best balance between accuracy and generalization. Adding more predictors (Size 7) slightly increases MSPE, indicating possible overfitting.

PART F: Compute the variance inflation factor for the models selected by AIC, BIC, MSPE, and R_a^2. Do any of these models show evidence of collinearity?

[9]:
```
source("vif_function.r")
compute_vif <- function(model) {
  vif_values <- vif(model)
  print(vif_values)
  return(vif_values)
}
```

```r
best_model_aic <- lm(aprice ~ lprice + pages + weight, data = train)
best_model_bic <- lm(aprice ~ lprice + weight, data = train)
best_model_mspe <- lm(aprice ~ lprice + pages + width + weight + thick, data =␣
 ↪train)
best_model_adj_r2 <- lm(aprice ~ lprice + pages + width + weight, data = train)

cat("\nVIF for Best AIC Model:\n")
vif_aic <- compute_vif(best_model_aic)

cat("\nVIF for Best BIC Model:\n")
vif_bic <- compute_vif(best_model_bic)

cat("\nVIF for Best MSPE Model:\n")
vif_mspe <- compute_vif(best_model_mspe)

cat("\nVIF for Best Adjusted R^2 Model:\n")
vif_adj_r2 <- compute_vif(best_model_adj_r2)
```

```
VIF for Best AIC Model:
  lprice    pages   weight
1.151550 1.610299 1.756317

VIF for Best BIC Model:
 lprice  weight
1.15146 1.15146

VIF for Best MSPE Model:
  lprice    pages    width   weight    thick
1.585250 2.922503 1.711048 2.570920 3.306411

VIF for Best Adjusted R^2 Model:
  lprice    pages    width   weight
1.533246 1.840267 1.706947 2.067046
```

1.3.6 Variance Inflation Factor (VIF) Analysis

The **VIF values** for the models selected by **AIC, BIC, MSPE, and Adjusted (R^2)** are:

Model Selection Criterion	Predictors	VIF Values
AIC Model	lprice, pages, weight	1.15, 1.61, 1.75
BIC Model	lprice, weight	1.15, 1.15
MSPE Model	lprice, pages, width, weight, thick	1.58, 2.92, 1.71, 2.57, 3.30
Adjusted (R^2) Model	lprice, pages, width, weight	1.53, 1.84, 1.70, 2.06

1.3.7 Interpretation

- **VIF < 5**: Low collinearity, no major concerns.
- **VIF 5-10**: Moderate collinearity, might need attention.
- **VIF > 10**: High collinearity, consider removing or transforming variables.

1.3.8 Conclusion

- All models have **VIF values below 5**, meaning **no significant collinearity issues**.
- The **MSPE model has the highest VIF (max = 3.30)**, but still within an acceptable range.
- The **BIC model has the lowest VIF**, making it the most stable in terms of collinearity.

Thus, **all models are acceptable**, and no immediate variable removal is needed.

1.4 Problem 3 Diagnosing and Correcting Non-Constant Variance (40 points)

Researchers at the National Institutes of Standards and Technology (NIST) collected pipline data on ultrasonic measurements of the depth of defects in the Alaska pipeline in the field. The depths of the defects were then remeasured in the laboratory. The laboratory measurements are more accurate than the field measurements, but more time consuming and expensive. We want to develop a regression model for correcting the in field measurements.

PART A: Fit a regression model where `Lab` is the response and `Field` is the predictor and save this model as `lmodPipeline`. Check for non-constant variance. Use the Pearson residuals. The Pearson residuals are a type of standardized residual. A plot of the Pearson residuals against the fitted values provides evidence of nonconstant variance. You can do this by specifying the "type" argument in your use of the resid function. Note whether or not you see non-constant variance in your plot.

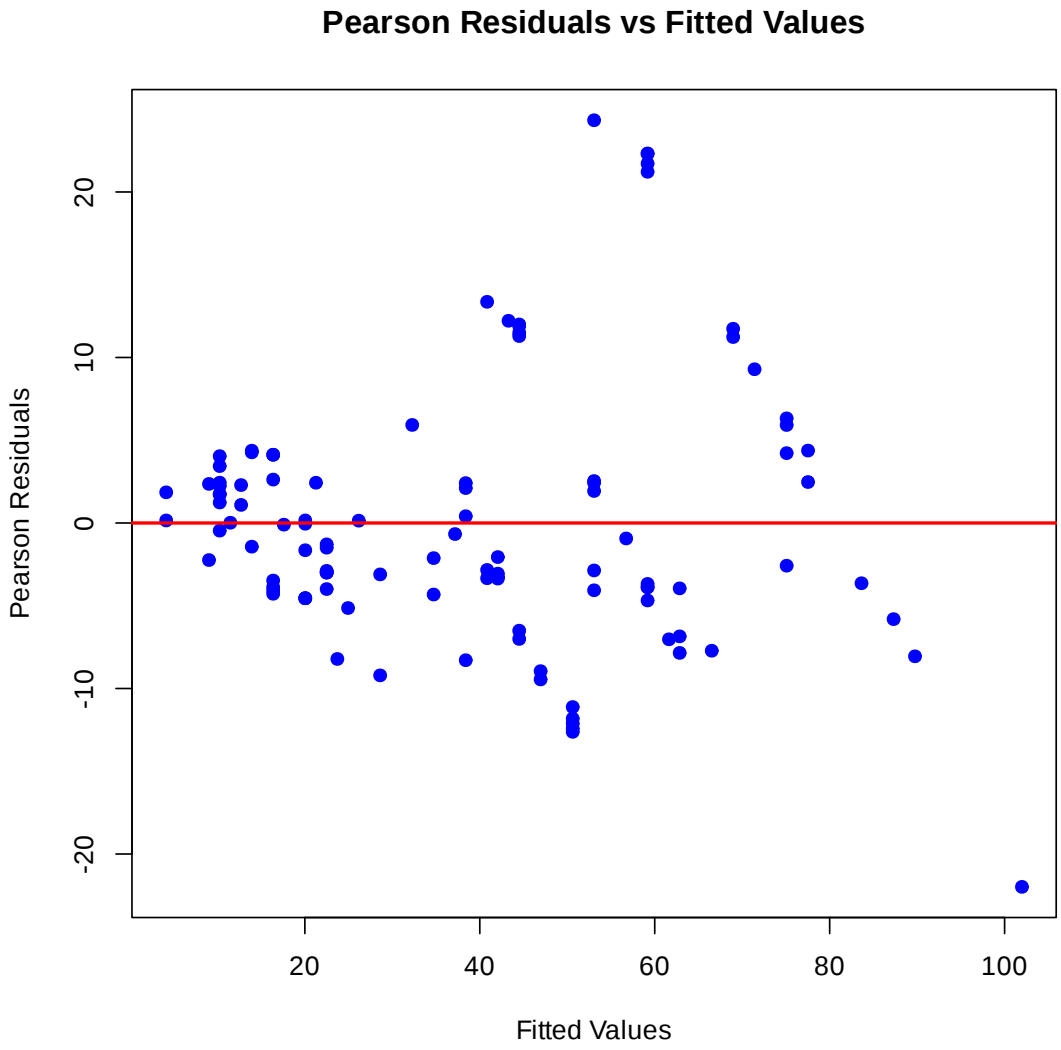
[10]:
```
install.packages("remotes")
remotes::install_github("cran/faraway")
library(faraway)
load(file ='pipeline.rda')
```

```
Updating HTML index of packages in '.Library'

Making 'packages.html' …
 done

Skipping install of 'faraway' from a github remote, the SHA1 (5c64099c) has not
changed since last install.
  Use `force = TRUE` to force installation


Attaching package: ‘faraway’


The following object is masked _by_ ‘.GlobalEnv’:
```

```
vif
```

[11]:
```
library(faraway)

data(pipeline)

lmodPipeline <- lm(Lab ~ Field, data = pipeline)

pearson_resid <- resid(lmodPipeline, type = "pearson")

plot(fitted(lmodPipeline), pearson_resid,
     xlab = "Fitted Values",
     ylab = "Pearson Residuals",
     main = "Pearson Residuals vs Fitted Values",
     pch = 19, col = "blue")

abline(h = 0, col = "red", lwd = 2)

# Print model summary
summary(lmodPipeline)
```

```
Call:
lm(formula = Lab ~ Field, data = pipeline)

Residuals:
    Min      1Q  Median      3Q     Max
-21.985  -4.072  -1.431   2.504  24.334

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.96750    1.57479  -1.249    0.214
Field        1.22297    0.04107  29.778   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.865 on 105 degrees of freedom
Multiple R-squared:  0.8941,        Adjusted R-squared:  0.8931
F-statistic: 886.7 on 1 and 105 DF,  p-value: < 2.2e-16
```



1.4.1 Non-Constant Variance Analysis

The Pearson residuals plot shows evidence of non-constant variance (heteroscedasticity). The residuals exhibit a funnel shape, where the spread increases as the fitted values increase.

Observations:

- The residuals are not randomly scattered around zero.
- The variance of residuals increases with fitted values.
- This indicates a violation of the homoscedasticity assumption in linear regression.

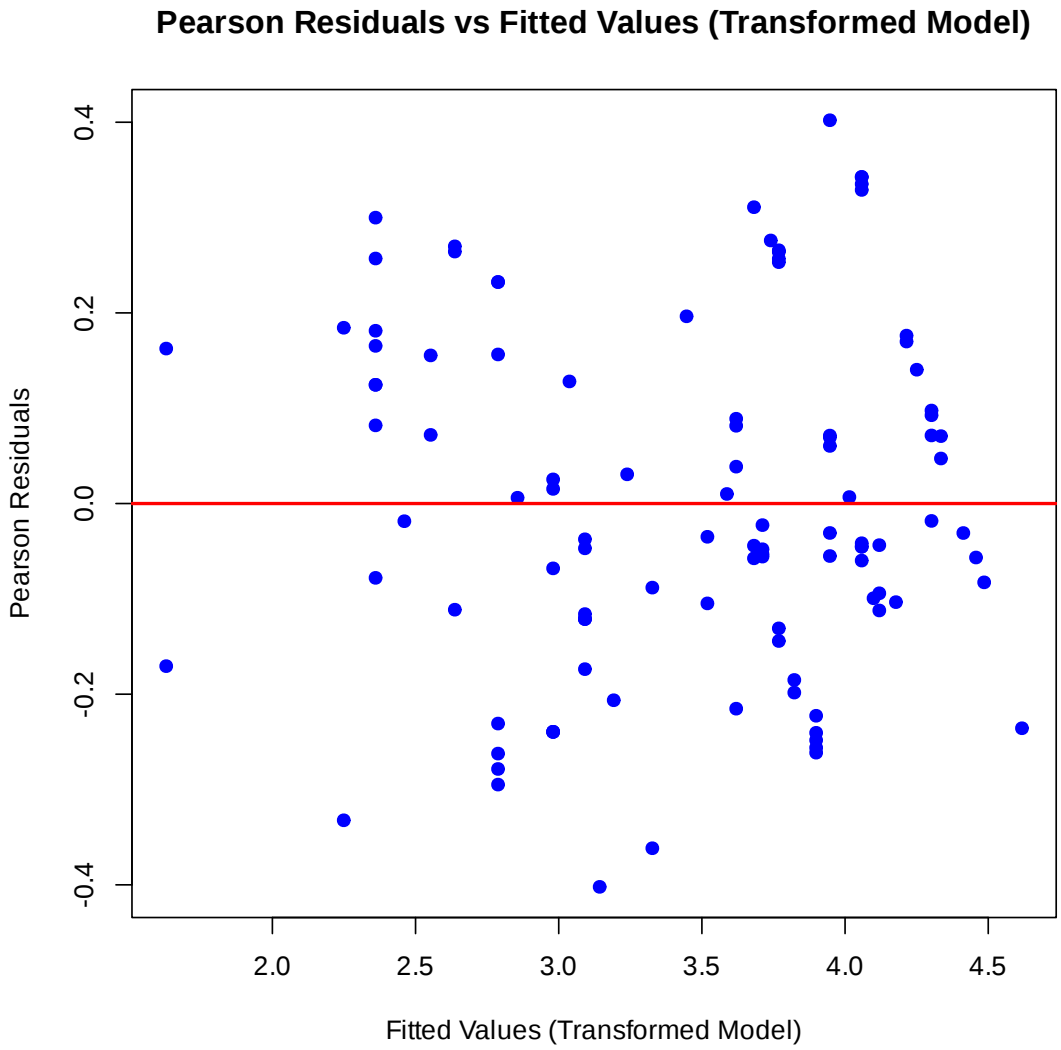
PART B: Sometimes transforming the response and predictor helps in stabilizing variance. Find a transformation on Lab and/or Field so that in the transformed scale the relationship is approximately linear with constant variance. Restrict your choice of transformation to square root or

log. Save your transformed variables as `pipeline$LabTransform` and `pipeline$FieldTransform`. Then, regress the transformed`Lab` variable (response) onto the transformed `Field` variable (predictor), and save this as `lmodTr`.

Check for non-constant variance in your transformed model using the same process from Part A. What do you notice?

[12]:
```
pipeline$LabTransform <- log(pipeline$Lab)
pipeline$FieldTransform <- log(pipeline$Field)

lmodTr <- lm(LabTransform ~ FieldTransform, data = pipeline)

pearson_resid_tr <- resid(lmodTr, type = "pearson")

plot(fitted(lmodTr), pearson_resid_tr,
     xlab = "Fitted Values (Transformed Model)",
     ylab = "Pearson Residuals",
     main = "Pearson Residuals vs Fitted Values (Transformed Model)",
     pch = 19, col = "blue")

abline(h = 0, col = "red", lwd = 2)

summary(lmodTr)
```

```
Call:
lm(formula = LabTransform ~ FieldTransform, data = pipeline)

Residuals:
     Min       1Q   Median       3Q      Max
-0.40212 -0.11853 -0.03092  0.13424  0.40209

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)    -0.06849    0.09305  -0.736    0.463
FieldTransform  1.05483    0.02743  38.457   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1837 on 105 degrees of freedom
Multiple R-squared:  0.9337,        Adjusted R-squared:  0.9331
F-statistic:  1479 on 1 and 105 DF,  p-value: < 2.2e-16
```



1.4.2 Transformed Model Analysis

Observations

- The transformed model: LabTransform = -0.06849 + 1.05483 × FieldTransform
- Higher R^2 (0.9337 vs. 0.8941) → Better model fit.
- Lower residual standard error (0.1837 vs. 7.865) → More stable predictions.
- Rsiduals are now randomly scattered, indicating constant variance.

Conclusion The log transformation corrected heteroscedasticity, improved model fit, and stabilized variance.