Exercise 6 - Cross Validation of Models

12433732 - Stefan Merdian

2024-11-28

Task 1

1)

```
library(ISLR)
df <- Auto
head(df)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                        name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

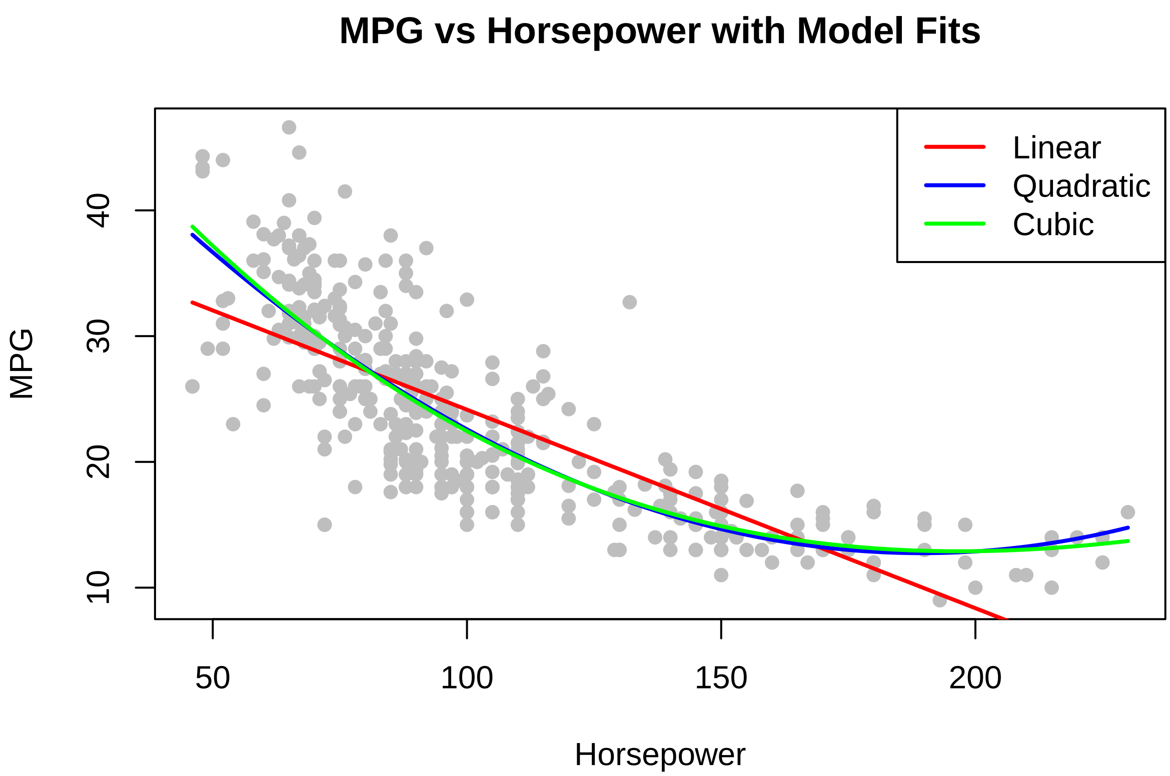
```
summary(df)
```

```
##       mpg          cylinders      displacement     horsepower        weight
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5   Median :2804
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5   Mean   :2978
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140
##
##   acceleration        year           origin                      name
##  Min.   : 8.00   Min.   :70.00   Min.   :1.000   amc matador       :  5
##  1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000   ford pinto        :  5
##  Median :15.50   Median :76.00   Median :1.000   toyota corolla    :  5
##  Mean   :15.54   Mean   :75.98   Mean   :1.577   amc gremlin       :  4
```

```
##  3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000   amc hornet         :  4
##  Max.   :24.80   Max.   :82.00   Max.   :3.000   chevrolet chevette:  4
##                                                  (Other)           :365
```

```r
df$name <- NULL
df$origin <- NULL
```

```r
model1 <- lm(mpg ~ horsepower, data = df)

model2 <- lm(mpg ~ poly(horsepower, 2), data = df)

model3 <- lm(mpg ~ poly(horsepower, 3), data = df)

plot(df$horsepower, df$mpg, main = "MPG vs Horsepower with Model Fits",
     xlab = "Horsepower", ylab = "MPG", pch = 16, col = "grey")


horsepower_grid <- seq(min(df$horsepower), max(df$horsepower), length.out =100)

lines(horsepower_grid, predict(model1, newdata = data.frame(horsepower = horsepower_grid)),
      col = "red", lwd = 2)

lines(horsepower_grid, predict(model2, newdata = data.frame(horsepower = horsepower_grid)),
      col = "blue", lwd = 2)

lines(horsepower_grid, predict(model3, newdata = data.frame(horsepower = horsepower_grid)),
      col = "green", lwd = 2)

legend("topright", legend = c("Linear", "Quadratic", "Cubic"),
       col = c("red", "blue", "green"), lwd = 2)
```



2)

First we create a function to save our metrics.

```
library(Metrics)

calculate_metrics <- function(model, test_data) {
  predictions <- predict(model, newdata = test_data)
  actuals <- test_data$mpg
  rmse_val <- rmse(actuals, predictions)
  mse_val <- mse(actuals, predictions)
  mad_val <- mad(actuals - predictions)
  return(list(RMSE = rmse_val, MSE = mse_val, MAD = mad_val))
}
```

Than we split data. One train/test split of 50%/50% and once 70%/30%

```
set.seed(123)

sample70 <- sample(c(TRUE, FALSE), nrow(df), replace=TRUE, prob=c(0.7,0.3))
train_data70 <- df[sample70, ]
test_data70 <- df[!sample70, ]

sample50 <- sample(c(TRUE, FALSE), nrow(df), replace=TRUE, prob=c(0.5,0.5))
train_data50 <- df[sample50, ]
test_data50 <- df[!sample50, ]

model1_50 <- glm(mpg ~ horsepower, data = train_data50)
```

```
model2_50 <- glm(mpg ~ poly(horsepower, 2), data = train_data50)
model3_50 <- glm(mpg ~ poly(horsepower, 3), data = train_data50)

model1_70 <- glm(mpg ~ horsepower, data = train_data70)
model2_70 <- glm(mpg ~ poly(horsepower, 2), data = train_data70)
model3_70 <- glm(mpg ~ poly(horsepower, 3), data = train_data70)
```

We produce the prediction with the test-data and calculate RMSE, MSE and MAD.

```
metrics_50_model1 <- calculate_metrics(model1_50, test_data50)
metrics_50_model2 <- calculate_metrics(model2_50, test_data50)
metrics_50_model3 <- calculate_metrics(model3_50, test_data50)

metrics_70_model1 <- calculate_metrics(model1_70, test_data70)
metrics_70_model2 <- calculate_metrics(model2_70, test_data70)
metrics_70_model3 <- calculate_metrics(model3_70, test_data70)

all_metrics <- rbind(metrics_50_model1, metrics_50_model2, metrics_50_model3,
                     metrics_70_model1, metrics_70_model2, metrics_70_model3)

print(all_metrics)
```

```
##                   RMSE     MSE      MAD
## metrics_50_model1 5.026192 25.26261 5.069911
## metrics_50_model2 4.37068  19.10285 3.856378
## metrics_50_model3 4.42424  19.5739  4.005665
## metrics_70_model1 4.974376 24.74442 4.543917
## metrics_70_model2 4.523842 20.46515 3.900278
## metrics_70_model3 4.539099 20.60342 4.039891
```

Based on the evaluation metrics—Root Mean Squared Error (RMSE), Mean Squared Error (MSE), and Median Absolute Deviation (MAD)—the optimal model is the cubic model trained and tested with a 50%/50% data split, referred to as metrics_50_model3.

- **metrics_50_model3** with RMSE: 4.42424 MSE:19.5739 MAD:4.005665

3)

1. Leave-one-out Cross Validation

```
library(boot)
```

```
## Warning: package 'boot' was built under R version 4.4.2
```

```
loocv1_50 <- cv.glm(train_data50, model1_50)$delta
loocv2_50 <- cv.glm(train_data50, model2_50)$delta
loocv3_50 <- cv.glm(train_data50, model3_50)$delta
# For 70% training data split
loocv1_70 <- cv.glm(train_data70, model1_70)$delta
```

```r
loocv2_70 <- cv.glm(train_data70, model2_70)$delta
loocv3_70 <- cv.glm(train_data70, model3_70)$delta

all_metrics_loocv <- rbind(loocv1_50, loocv2_50, loocv3_50,
                          loocv1_70, loocv2_70, loocv3_70)
all_metrics_loocv
```

```
##               [,1]     [,2]
## loocv1_50 23.24453 23.24305
## loocv2_50 19.68590 19.68449
## loocv3_50 19.80351 19.80157
## loocv1_70 24.19620 24.19547
## loocv2_70 18.85311 18.85250
## loocv3_70 18.90403 18.90315
```

Based on the average error also here, the cubic model with the 50%/50% data split has the lowest value.

2.1 5-fold Cross Validation

```r
cv.err1_50_5fold <- cv.glm(train_data50, model1_50, K = 5)$delta
cv.err2_50_5fold <- cv.glm(train_data50, model2_50, K = 5)$delta
cv.err3_50_5fold <- cv.glm(train_data50, model3_50, K = 5)$delta

# For 70% training data split
cv.err1_70_5fold <- cv.glm(train_data70, model1_70, K = 5)$delta
cv.err2_70_5fold <- cv.glm(train_data70, model2_70, K = 5)$delta
cv.err3_70_5fold <- cv.glm(train_data70, model3_70, K = 5)$delta

# Combine results
cv_5fold_results <- rbind(
  cv.err1_50_5fold, cv.err2_50_5fold, cv.err3_50_5fold,
  cv.err1_70_5fold, cv.err2_70_5fold, cv.err3_70_5fold
)
print(cv_5fold_results)
```

```
##                      [,1]     [,2]
## cv.err1_50_5fold 23.24642 23.18280
## cv.err2_50_5fold 19.62871 19.57315
## cv.err3_50_5fold 19.74558 19.66639
## cv.err1_70_5fold 24.12276 24.08643
## cv.err2_70_5fold 19.05558 18.99550
## cv.err3_70_5fold 19.41055 19.29241
```

2.2 10-fold Cross Validation

```r
# For 50% training data split
cv.err1_50_10fold <- cv.glm(train_data50, model1_50, K = 10)$delta
cv.err2_50_10fold <- cv.glm(train_data50, model2_50, K = 10)$delta
cv.err3_50_10fold <- cv.glm(train_data50, model3_50, K = 10)$delta
```

```r
# For 70% training data split
cv.err1_70_10fold <- cv.glm(train_data70, model1_70, K = 10)$delta
cv.err2_70_10fold <- cv.glm(train_data70, model2_70, K = 10)$delta
cv.err3_70_10fold <- cv.glm(train_data70, model3_70, K = 10)$delta

# Combine results
cv_10fold_results <- rbind(
  cv.err1_50_10fold, cv.err2_50_10fold, cv.err3_50_10fold,
  cv.err1_70_10fold, cv.err2_70_10fold, cv.err3_70_10fold
)
print(cv_10fold_results)
```

```
##                       [,1]     [,2]
## cv.err1_50_10fold 23.40395 23.36395
## cv.err2_50_10fold 19.99446 19.94862
## cv.err3_50_10fold 19.68199 19.64761
## cv.err1_70_10fold 24.22118 24.19858
## cv.err2_70_10fold 18.87191 18.85288
## cv.err3_70_10fold 18.80215 18.78172
```

4)

```r
models <- c("model1_50", "model2_50", "model3_50",
            "model1_70", "model2_70", "model3_70")

all_metrics_clean <- as.data.frame((all_metrics))
all_metrics_clean$RMSE <- unlist(all_metrics_clean$RMSE)
all_metrics_clean$MSE <- unlist(all_metrics_clean$MSE)
all_metrics_clean$MAD <- unlist(all_metrics_clean$MAD)
rownames(all_metrics_clean) <- NULL

cv_loocv_clean <- as.data.frame(all_metrics_loocv)
names(cv_loocv_clean) <- c("loocv R","loocv Adj R")
cv_loocv_clean$Model <- rownames(cv_loocv_clean)
cv_loocv_clean[,3]<- NULL
rownames(cv_loocv_clean) <- NULL

cv_5fold_clean <- as.data.frame(cv_5fold_results)
names(cv_5fold_clean) <- c("5-Fold R","5-Flod Adj R")
cv_5fold_clean$Model <- rownames(cv_5fold_results)
cv_5fold_clean[,3]<- NULL
rownames(cv_5fold_clean) <- NULL

cv_10fold_clean <- as.data.frame(cv_10fold_results)
names(cv_10fold_clean) <- c("10-Fold R", "10-Fold Adj R")
cv_10fold_clean$Model <- rownames(cv_10fold_results)
cv_10fold_clean[,3]<- NULL
rownames(cv_10fold_clean) <- NULL

final_results <- cbind(models,all_metrics_clean,cv_loocv_clean,cv_5fold_clean, cv_10fold_clean)
```

```r
final_results
```

```
##      models     RMSE      MSE      MAD  loocv R loocv Adj R 5-Fold R
## 1 model1_50 5.026192 25.26261 5.069911 23.24453    23.24305 23.24642
## 2 model2_50 4.370680 19.10285 3.856378 19.68590    19.68449 19.62871
## 3 model3_50 4.424240 19.57390 4.005665 19.80351    19.80157 19.74558
## 4 model1_70 4.974376 24.74442 4.543917 24.19620    24.19547 24.12276
## 5 model2_70 4.523842 20.46515 3.900278 18.85311    18.85250 19.05558
## 6 model3_70 4.539099 20.60342 4.039891 18.90403    18.90315 19.41055
##   5-Flod Adj R 10-Fold R 10-Fold Adj R
## 1     23.18280  23.40395      23.36395
## 2     19.57315  19.99446      19.94862
## 3     19.66639  19.68199      19.64761
## 4     24.08643  24.22118      24.19858
## 5     18.99550  18.87191      18.85288
## 6     19.29241  18.80215      18.78172
```

Based on all error metrics, the quadratic model with the 50/50 split demonstrates the best overall performance. This is because the quadratic model aligns well with the shape of the dependent variable, as evidenced by the earlier plot showing it as the most appropriate fit. Additionally, the 50/50 split quadratic model has the highest adjusted R2, indicating it explains more of the variation in the data compared to other models.

Typically, we expect the 70/30 split to perform better because it has more training data to generalize effectively. However, in this case, the 50/50 split model benefits from having more test data, reducing variability and improving its performance metrics. For the raw cross-validation errors (5-Fold and 10-Fold R), the 70/30 split quadratic model (model2_70) performs slightly better, as the increased training data allows for better generalization during cross-validation.

In general, using more data for training is preferred as it helps the model generalize better. However, when there is enough data, allocating too much to training can lead to overfitting, resulting in slightly worse metrics on the test set. This trade-off is evident in the comparison between the 50/50 and 70/30 splits for the quadratic model.

Task 2

1)

```r
library(ggplot2)

df_eco <- economics

head(df_eco)
```

```
## # A tibble: 6 x 6
##   date         pce    pop psavert uempmed unemploy
##   <date>     <dbl>  <dbl>   <dbl>   <dbl>    <dbl>
## 1 1967-07-01  507. 198712    12.6     4.5     2944
## 2 1967-08-01  510. 198911    12.6     4.7     2945
## 3 1967-09-01  516. 199113    11.9     4.6     2958
```

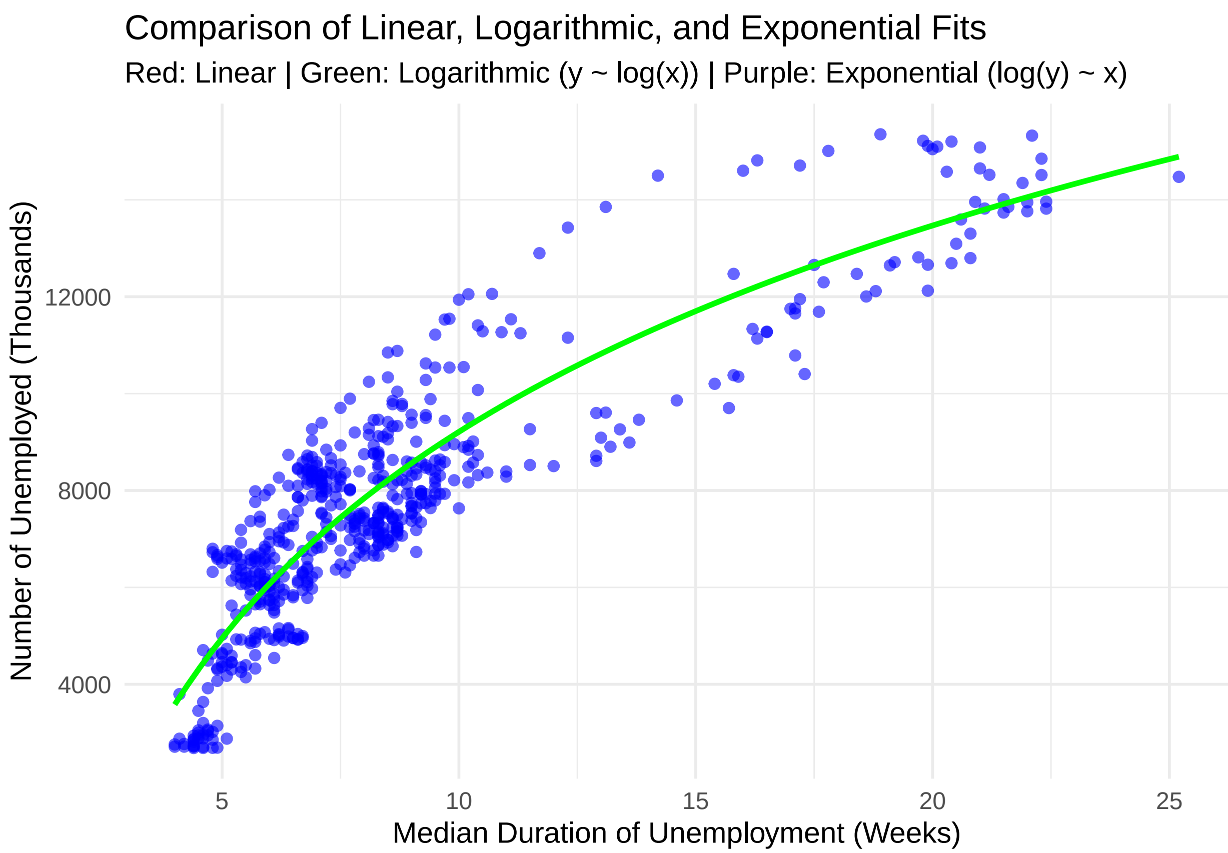
```
## 4 1967-10-01  512. 199311    12.9      4.9      3143
## 5 1967-11-01  517. 199498    12.8      4.7      3066
## 6 1967-12-01  525. 199657    11.8      4.8      3018
```

Plot uempmed vs. unemploy

We will check first what is more appropriate for the second task.

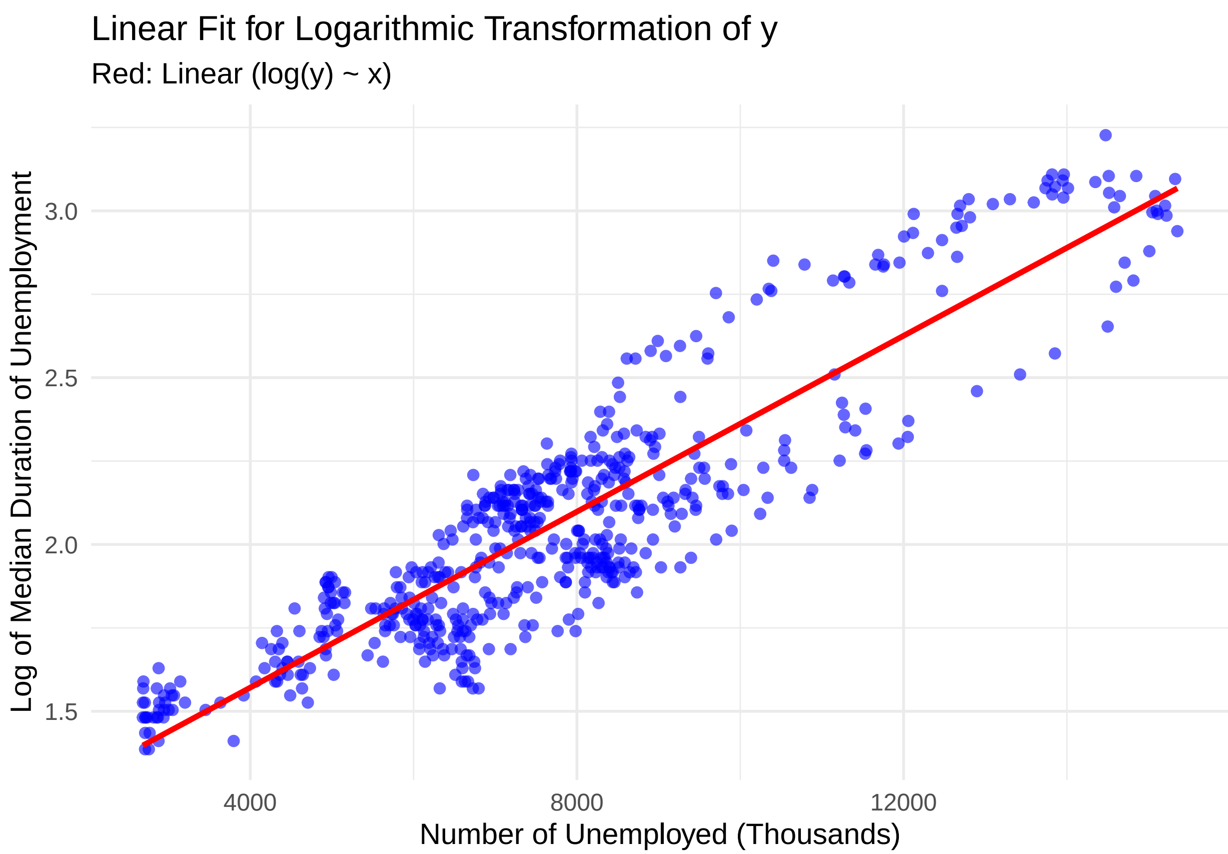
```r
library(ggplot2)


ggplot(df_eco, aes(x = uempmed, y = unemploy)) +
  geom_point(color = "blue", alpha = 0.6) +
  geom_smooth(method = "lm", formula = y ~ log(x), color = "green", se = FALSE, size = 1) +  # Logarith
  labs(title = "Comparison of Linear, Logarithmic, and Exponential Fits",
       subtitle = "Red: Linear | Green: Logarithmic (y ~ log(x)) | Purple: Exponential (log(y) ~ x)",
       x = "Median Duration of Unemployment (Weeks)",
       y = "Number of Unemployed (Thousands)") +
   theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



When "unemploy" (number of unemployed in thousands) is the dependent variable and "uempmed" (median duration of unemployment) is the independent variable, the data points follow a logarithmic pattern. This means that applying a logarithmic transformation to the "uempmed" variable aligns well with the shape of the data, as shown in the plot.

```
ggplot(df_eco, aes(x = unemploy, y = log(uempmed))) +
  geom_point(color = "blue", alpha = 0.6) +
  geom_smooth(method = "lm", formula = y ~ x, color = "red", se = FALSE, size = 1) +
  labs(
    title = "Linear Fit for Logarithmic Transformation of y",
    subtitle = "Red: Linear (log(y) ~ x)",
    x = "Number of Unemployed (Thousands)",
    y = "Log of Median Duration of Unemployment"
  ) +
  theme_minimal()
```



We can see, by transforming y to log(y), we linearized the relationship between "unemploy" and "uempmed," improving the model's ability to capture the exponential trend in the data more accurately.

linear model

```
linear_model <- glm(unemploy ~ uempmed, data = df_eco)
linear_model_reverse <- glm(uempmed ~ unemploy, data = df_eco)
```

log model

```r
log_model <- glm( unemploy ~ log(uempmed), data = df_eco)
log_model_reverse <- glm( log(unemploy) ~ uempmed, data = df_eco)
```

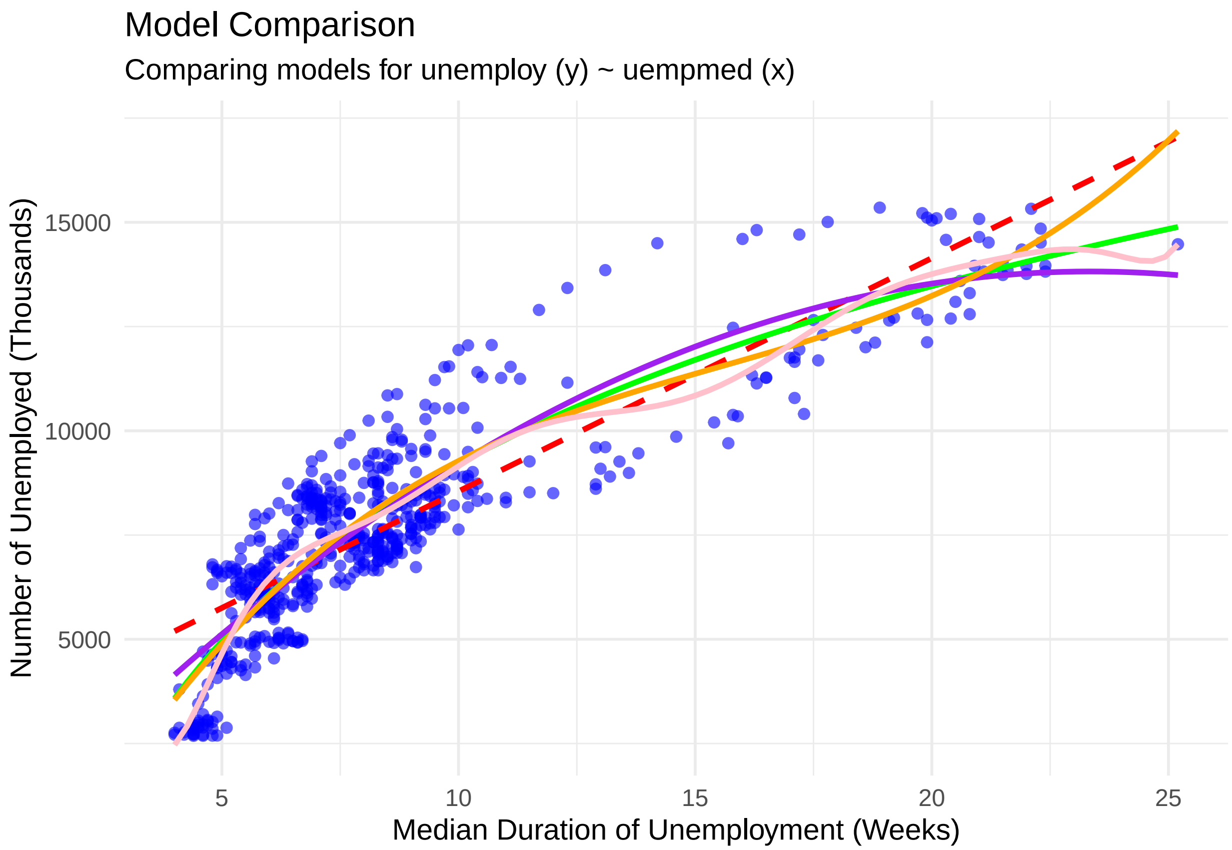
polynomial model

```r
poly_model_2 <- glm(unemploy ~ poly(uempmed, 2), data = df_eco)
poly_model_3 <- glm(unemploy ~ poly(uempmed, 3), data = df_eco)
poly_model_10 <- glm(unemploy ~ poly(uempmed, 10), data = df_eco)

poly_model_reverse_2 <- glm(uempmed ~ poly(unemploy, 2), data = df_eco)
poly_model_reverse_3 <- glm(uempmed ~ poly(unemploy, 3), data = df_eco)
poly_model_reverse_10 <- glm(uempmed ~ poly(unemploy, 10), data = df_eco)
```

2)

Plot and compare models.

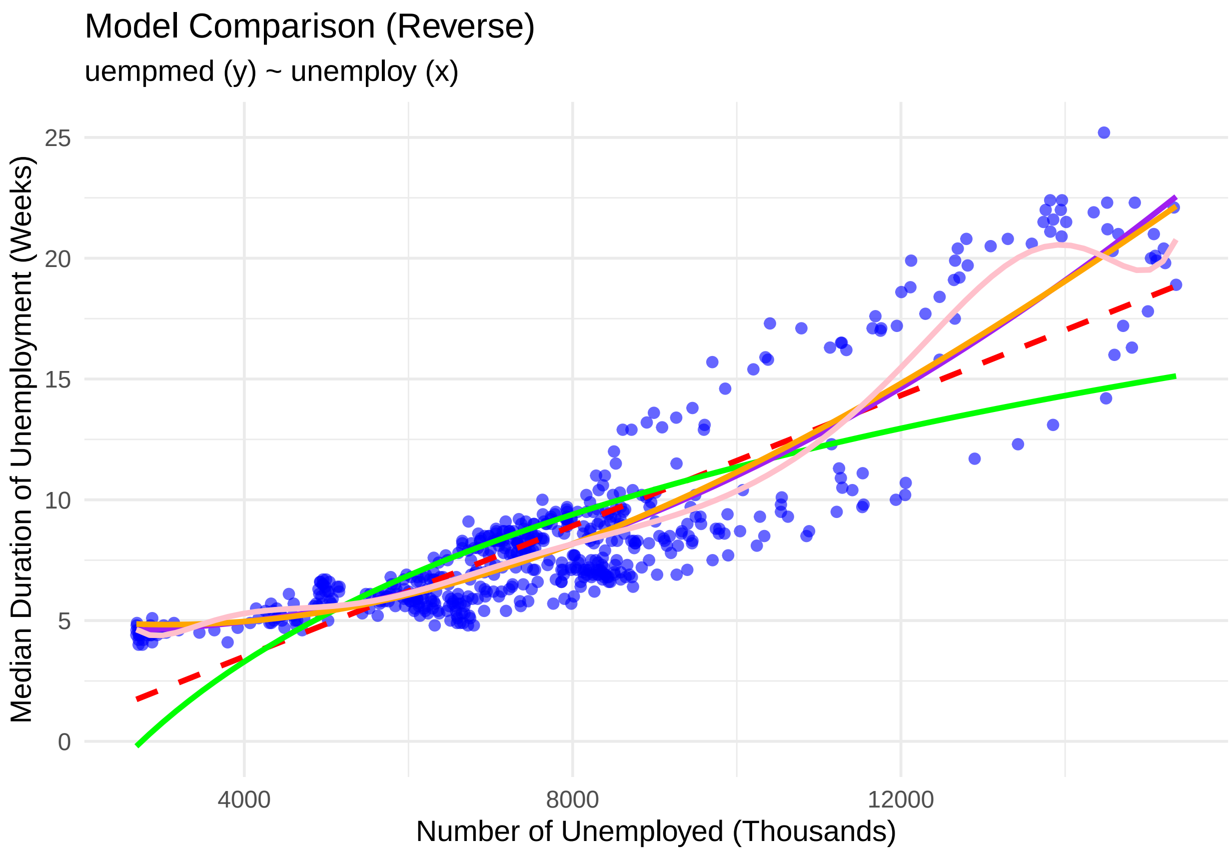
```r
library(ggplot2)

p <- ggplot(df_eco, aes(x = uempmed, y = unemploy)) +
  geom_point(color = "blue", alpha = 0.6) +
  labs(
    title = "Model Comparison",
    subtitle = "Comparing models for unemploy (y) ~ uempmed (x)",
    x = "Median Duration of Unemployment (Weeks)",
    y = "Number of Unemployed (Thousands)"
  ) +
  theme_minimal()
# linear
p <- p + geom_smooth(method = "lm", formula = y ~ x, color = "red", se = FALSE, size = 1, linetype = "da
# log
p <- p + geom_smooth(method = "lm", formula = y ~ log(x), color = "green", se = FALSE, size = 1)
# poly 2,3,10
p <- p + geom_smooth(method = "lm", formula = y ~ poly(x, 2), color = "purple", se = FALSE, size = 1) +
  geom_smooth(method = "lm", formula = y ~ poly(x, 3), color = "orange", se = FALSE, size = 1) +
  geom_smooth(method = "lm", formula = y ~ poly(x, 10), color = "pink", se = FALSE, size = 1)

print(p)
```



The 2. degree polynomial model (Purple) and the logarithmic model (Green) provide the best fit for the data in this case. The other models deviate a bit from the overall trend. Higher-degree polynomial models are overly to wavy, while the linear model don't fit the non-linear pattern present in the data.

```
library(ggplot2)

p_reverse <- ggplot(df_eco, aes(x = unemploy, y = uempmed)) +
  geom_point(color = "blue", alpha = 0.6) +
  labs(
    title = "Model Comparison (Reverse)",
    subtitle = "uempmed (y) ~ unemploy (x)",
    x = "Number of Unemployed (Thousands)",
    y = "Median Duration of Unemployment (Weeks)"
  ) +
  theme_minimal()

# linear
p_reverse <- p_reverse + geom_smooth(method = "lm", formula = y ~ x, color = "red", se = FALSE, size = 1

# log
p_reverse <- p_reverse + geom_smooth(method = "lm", formula = y ~ log(x), color = "green", se = FALSE, s

# polynomial
p_reverse <- p_reverse + geom_smooth(method = "lm", formula = y ~ poly(x, 2), color = "purple", se = FAL
  geom_smooth(method = "lm", formula = y ~ poly(x, 3), color = "orange", se = FALSE, size = 1) +
  geom_smooth(method = "lm", formula = y ~ poly(x, 10), color = "pink", se = FALSE, size = 1)
```

```r
print(p_reverse)
```



For this model, the purple (2nd-degree polynomial) and orange (3rd-degree polynomial) models fit the data most appropriately. The pink (10th-degree polynomial) model overfits the data, capturing minor fluctuations at the cost of missing the overall trend. The linear model (Red) performs worse here compared to the other model, as it cannot capture the clear non-linear relationship. The logarithmic model (Green) is unsuitable in this case because the data exhibits a stronger increase as x grows, which is not well represented by a logarithmic transformation. This behavior makes the logarithmic model inappropriate, while the 2nd-degree polynomial captures the pattern effectively.

3)

Preparing data frames for saving the results.

```r
library(boot)

models <- list(
  linear_model = linear_model,
  log_model = log_model,
  poly_model_2 = poly_model_2,
  poly_model_3 = poly_model_3,
  poly_model_10 = poly_model_10
)

models_reverse <- list(
  linear_model_reverse = linear_model_reverse,
```

```r
  log_model_reverse = log_model_reverse,
  poly_model_reverse_2 = poly_model_reverse_2,
  poly_model_reverse_3 = poly_model_reverse_3,
  poly_model_reverse_10 = poly_model_reverse_10
)


results <- data.frame(
  Model = names(models),
  RMSE = numeric(length(models)),
  MSE = numeric(length(models))
)

results_reverse <- data.frame(
  Model = names(models_reverse),
  RMSE = numeric(length(models_reverse)),
  MSE = numeric(length(models_reverse))
)
```

1. Leave-one-out Cross Validation

```r
for (i in seq_along(models)) {
  cv_result <- cv.glm(df_eco, models[[i]])
  mse <- round(cv_result$delta[1],4)
  rmse <- round(sqrt(mse),4)
  results$MSE[i] <- mse
  results$RMSE[i] <- rmse
}

for (i in seq_along(models_reverse)) {
  cv_result <- cv.glm(df_eco, models_reverse[[i]])
  mse <- round(cv_result$delta[1],4)
  rmse <- round(sqrt(mse),4)
  results_reverse$MSE[i] <- mse
  results_reverse$RMSE[i] <- rmse
  print(cv_result$delta[1])
}
```

```
## [1] 4.159797
## [1] 0.05311919
## [1] 3.005112
## [1] 3.009934
## [1] 2.832303
```

```r
results_combined <- rbind(
  cbind(Direction = "Original", results),
  cbind(Direction = "Reverse", results_reverse)
)
print(results_combined)
```

```
##    Direction                    Model      RMSE           MSE
```

```
## 1   Original          linear_model 1309.6606 1715210.8049
## 2   Original             log_model 1154.9877 1333996.6577
## 3   Original           poly_model_2 1196.8837 1432530.6144
## 4   Original           poly_model_3 1168.9331 1366404.5907
## 5   Original          poly_model_10 2128.5531 4530738.2779
## 6    Reverse  linear_model_reverse    2.0396       4.1598
## 7    Reverse     log_model_reverse    0.2304       0.0531
## 8    Reverse  poly_model_reverse_2    1.7335       3.0051
## 9    Reverse  poly_model_reverse_3    1.7349       3.0099
## 10   Reverse poly_model_reverse_10    1.6829       2.8323
```

1. 5 - Fold Cross Validation

```r
for (i in seq_along(models)) {
  cv_result <- cv.glm(df_eco, models[[i]],  K = 5)
  mse <- round(cv_result$delta[1],4)
  rmse <- round(sqrt(mse),4)
  results$MSE[i] <- mse
  results$RMSE[i] <- rmse
}

for (i in seq_along(models_reverse)) {
  cv_result <- cv.glm(df_eco, models_reverse[[i]],  K = 5)
  mse <- round(cv_result$delta[1],4)
  rmse <- round(sqrt(mse),4)
  results_reverse$MSE[i] <- mse
  results_reverse$RMSE[i] <- rmse
  print(cv_result$delta[1])
}
```

```
## [1] 4.183983
## [1] 0.05307314
## [1] 3.021365
## [1] 2.99448
## [1] 2.834467
```

```r
results_combined <- rbind(
  cbind(Direction = "Original", results),
  cbind(Direction = "Reverse", results_reverse)
)
print(results_combined)
```

```
##    Direction                 Model      RMSE          MSE
## 1   Original          linear_model 1316.3254 1.732713e+06
## 2   Original             log_model 1155.1999 1.334487e+06
## 3   Original           poly_model_2 1198.3129 1.435954e+06
## 4   Original           poly_model_3 1177.3063 1.386050e+06
## 5   Original          poly_model_10 6976.8631 4.867662e+07
## 6    Reverse  linear_model_reverse    2.0455 4.184000e+00
## 7    Reverse     log_model_reverse    0.2304 5.310000e-02
## 8    Reverse  poly_model_reverse_2    1.7382 3.021400e+00
```

```
## 9    Reverse  poly_model_reverse_3    1.7305 2.994500e+00
## 10   Reverse poly_model_reverse_10    1.6836 2.834500e+00
```

1. 10 - Fold Cross Validation

```r
for (i in seq_along(models)) {
  cv_result <- cv.glm(df_eco, models[[i]],  K = 10)
  mse <- round(cv_result$delta[1],4)
  rmse <- round(sqrt(mse),4)
  results$MSE[i] <- mse
  results$RMSE[i] <- rmse
}

for (i in seq_along(models_reverse)) {
  cv_result <- cv.glm(df_eco, models_reverse[[i]],  K = 10)
  mse <- round(cv_result$delta[1],4)
  rmse <- round(sqrt(mse),4)
  results_reverse$MSE[i] <- mse
  results_reverse$RMSE[i] <- rmse
  print(cv_result$delta[1])
}
```

```
## [1] 4.142628
## [1] 0.05301392
## [1] 3.001368
## [1] 3.001934
## [1] 2.856523
```

```r
results_combined <- rbind(
  cbind(Direction = "Original", results),
  cbind(Direction = "Reverse", results_reverse)
)
print(results_combined)
```

```
##    Direction                 Model      RMSE          MSE
## 1   Original          linear_model 1308.8966 1713210.1843
## 2   Original             log_model 1153.3074 1330117.8550
## 3   Original          poly_model_2 1198.9871 1437570.0712
## 4   Original          poly_model_3 1173.4486 1376981.7282
## 5   Original         poly_model_10 1961.6180 3847945.1320
## 6    Reverse  linear_model_reverse    2.0353       4.1426
## 7    Reverse     log_model_reverse    0.2302       0.0530
## 8    Reverse  poly_model_reverse_2    1.7325       3.0014
## 9    Reverse  poly_model_reverse_3    1.7326       3.0019
## 10   Reverse poly_model_reverse_10    1.6901       2.8565
```

##4) The significant difference in error values between the original and reversed models is primarily due to the different scales of the dependent variables. In the reversed models, the dependent variable, "uempmed," has a much smaller range compared to "unemploy" in the original models. Since the data was not normalized, the reversed models naturally produce much lower error values, even if their relative performance is similar.

Orginal: As expected, both the logarithmic model and the quadratic (2nd-degree polynomial) model perform well for the original models. This aligns with the visualizations, where these models captured the nonlinear trend in the data effectively. Interestingly, the 3rd-degree polynomial model performed slightly better than the quadratic model. This was also evident in the plots, where the cubic model fit the data closely without excessive overfitting.

The linear model, however, is clearly underfitted. It fails to capture the non-linear nature of the relationship between "unemploy" and "uempmed," leading to higher error values compared to the nonlinear models.

In contrast, the 10th-degree polynomial model overfits the data. While it may achieve very low bias in specific areas, its excessive flexibility leads to high variance, making it less generalizable. This overfitting is reflected in its higher RMSE and MSE values.

Reverse:

The linear model significantly underfits the data, failing to capture the nonlinear pattern, and has the worst error results among all models. This confirms that a simple linear relationship is not appropriate for this data.

The logarithmic model performs well, as reflected in the significantly lower RMSE and MSE values compared to other models. This indicates that the log-transformation on the dependet variable has successfully linearized the relationship, making the logarithmic model highly suitable for capturing the data's trend. The logarithmic model now provides the best fit among all the reverse models.

Both the quadratic (poly_model_reverse_2) and cubic (poly_model_reverse_3) models show nearly identical RMSE and MSE values. As seen in the plot, they closely follow the data's trend. They strike a good balance between fitting the data and maintaining generalization.

The 10th-degree polynomial model (poly_model_reverse_10) achieves the lowest RMSE and MSE values. However, as clearly observed in the plot, it overfits the data significantly. It lacks generalization and shows exaggerated curvature, especially at the higher range of the x-axis, which would likely lead to higher error values on unseen data due to its excessive complexity.