Classwork 5 - MLR_Diagnostics_WrapUp_Blank-Copy1

March 9, 2025

0.0.1 MLR Diagnostics Wrap-Up

Exercise 1) State the MLR assumptions that are necessary for using ordinary least squares.

- constant variance of residuals (homoscedastic)
- linearity
- assume low/no collinearity/multi-collinearity
- independence in error/residuals
- normality of residuals/error

Exercise 2) How do you test for linear relationship, normality of residuals, no multicollinearity, and homoscedasticity?

constant variance: residuals vs fitted plot leveneTest how to fix: possibly rescaling of just response or of response and features

normality of residuals: QQ plot Shapiro-Wilk Test -H0 = the residuals are normal

linearity: residuals vs fitted plot, or fitted vs observed plot pairwise scatterplots

collinearity/multi-collinearity: pairwise scatter plots summary: NA row or some of the coefficients are counterintuitive, and p-values that indicate counterintuitive significance vif

independence: successive residual plot autocorrelation of order 1 Durbin-Watson test

Exercise 3) Diagnosing multicollinearity.

In MLR, when multicollinearity exists between variables, this means that there is redundancy between predictor variables. When multicollinearity is present, the solution of the regression model becomes unstable. It also might not be possible to estimate the coefficients.

- We've discussed looking for NA values in the table of regression coefficients.
- We can also assess multicollinearity by computing the variance inflation factor (VIF). This measures how much variance of a regression coefficient is inflated due to multicollinearity. Look for VIF $>$ 5-10. That is the threshold for a problematic amount of collinearity.
- When you find multicollinearity, you should remove the features that are causing it.

[1]:
```
library(tidyverse)
library(caret)
```

```
Warning message in system("timedatectl", intern = TRUE):
"running command 'timedatectl' had status 1"
  Attaching packages                      tidyverse
```

```
1.3.2
 ggplot2 3.4.0       purrr   1.0.0
 tibble  3.1.8       dplyr   1.0.10
 tidyr   1.2.1       stringr 1.5.0
 readr   2.1.3       forcats 0.5.2
  Conflicts
tidyverse_conflicts()
 dplyr::filter() masks stats::filter()
 dplyr::lag()    masks stats::lag()
Loading required package: lattice


Attaching package: 'caret'


The following object is masked from 'package:purrr':

    lift
```

Let's investigate multicollinearity using the folling data set. This data set is for predicting the median house value in Boston suburbs, based on multiple predictor variables.

Part A: First load the data.

[2]:
```
data("Boston", package = "MASS")
```

Part B: Next split the data into a training set and a test set with an 80/20 split.

[3]:
```
training.samples = Boston$medv %>%
    createDataPartition(p=0.8, list=FALSE)
train.data = Boston[training.samples,]
test.data = Boston[-training.samples,]
```

Part C: Build a regression model with all of the predictor variables. Analyze the model

[4]:
```
# 1) CODE HERE
# Build the model
model_full = lm(medv~.,data = train.data)
# Make predictions using the test data
predictions = model_full %>% predict(test.data)

# Find the RMSE and R2 values to analyze performance
data.frame(RMSE = RMSE(predictions,test.data$medv),R2=R2(predictions,test.
 ↪data$medv))
```

A data.frame: 1 × 2

RMSE	R2
<dbl>	<dbl>
4.70539	0.7415893

Part D: Use the `vif` score to detect multicollinearity in this regression model.

[5]:
```
# 2) CODE HERE
# library(car)
source("vif_function.r")
vif(model_full)
```

crim 1.70909195104463 **zn** 2.24791526097292 **indus** 4.15486955967486 **chas** 1.08774794801745 **nox** 4.35930650496483 **rm** 2.03358441479768 **age** 3.28952955627 **dis** 4.10208247556056 **rad** 7.31926759520643 **tax** 8.93529935968821 **ptratio** 1.69047125673128 **black** 1.35625737480865 **lstat** 3.0638913902352

This output shows each feature and its corresponding vif score. In general, you would want to remove any feature that has a high vif score, which is typically a value larger than 5-10.

Part E: Note that `tax` has the highest vif score. Remove the `tax` feature and re-fit the MLR model. Note that removing `rad` also may be a good idea.

[6]:
```
# 3) CODE HERE
# Re-build the model without the `tax` feature.
model_reduced = lm(medv~.-tax,data=train.data)

# Make predictions with the test set.

# Test the model performance by computing the RMSE and the R2 values.
predictions = model_reduced %>% predict(test.data)

# Find the RMSE and R2 values to analyze performance
data.frame(RMSE = RMSE(predictions,test.data$medv),R2=R2(predictions,test.
 ↪data$medv))
```

A data.frame: 1 × 2

RMSE	R2
<dbl>	<dbl>
4.675003	0.744865

Removing the `tax` feature did not affect the perfomance of the model significantly.

Reference for this problem: http://www.sthda.com/english/articles/39-regression-model-diagnostics/160-multicollinearity-essentials-and-vif-in-r/

Exercise 4)

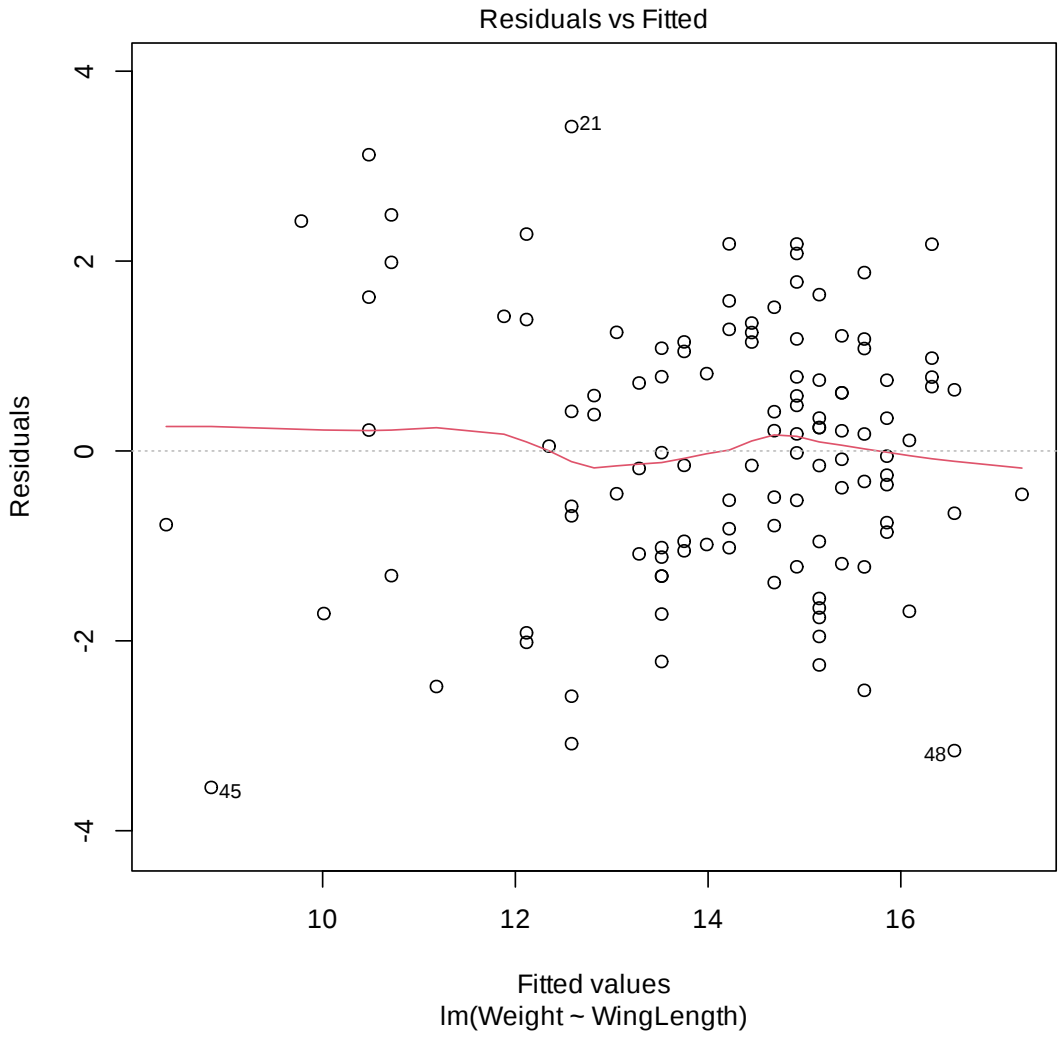
Part A: Construct and interpret the plots of the standardized residuals against the fitted values. Do you see evidence of nonconstant variance?

[7]:
```
# First we load the data set
dfSparrows = read.csv("Sparrows.csv")

# 4) CODE HERE
# Now we fit a regression model
# dfSparrows
```

```r
sparrow_model = lm(Weight~WingLength,data=dfSparrows)

# Plot the residuals versus fitted values.
plot(sparrow_model,1)
```



No evidence of nonconstant variance. There is more data for larger fitted values, however the spread of the data looks alright.

- We are looking at random variation above and below 0.
- We are looking for no apparent patterns.
- The width of the "band" of points is relatively constant.

[9]:
```
# 5) CODE HERE
source("leveneTest.r")
leveneTest(Weight~Treatment,data=dfSparrows)
```

```
Warning message in leveneTest.default(y = y, group = group, …):
"group coerced to factor."
```

A anova: 2 × 3

	Df	F value	Pr(>F)
	<int>	<dbl>	<dbl>
group	2	1.74936	0.1785612
	113	NA	NA

Heterogeneous variances are indicated by a non-random pattern in the residuals vs. fitted plot. With this categorical data, we are looking for even spread along the columns of dots. We appear to have that here, so what we see indicates homogeneity.

Part B: Now check the normality of the data with a Q-Q plot.

[ ]:
```
# 6) CODE HERE
```

This data looks reasonably normal.

Part C: Use the Shapiro-Wilk test to test for normality.

[ ]:
```
# Another test for normality of the residuals is the Shapiro-Wilk test.

# 7) CODE HERE
# Extract the residuals

# Run the Shapiro-Wilk test
```

With this non-significant p-value, we would conclude that our data upholds the normality condition.

Exercise 5) How do you correct the problem of nonconstant variance.

- Use transformations. Recall sqrt(), log(), inverse()
- Use weighted least squares (not covered). See link below.

Further study for weighted least squares: https://stats.oarc.ucla.edu/stata/ado/analysis/stata-analysis-toolsweighted-least-squares-regression/

Exercise 6) How do you diagnose violations of the uncorrelated errors assumption?

Exercise 7) Define outliers and influential observations. How do you find these points?

[ ]:
```

```

An **outlier** is a point with a large residual.

An **influential point** is a point that has a large effect on the regression coefficients.

Not all outliers are influential.

[ ]:
```

```