Choi, Asgm4

Yeonsil Choi

07/12/2021

Q1 For this question, you will use the prostate data from the faraway package. Use lpsa as the response variable and the others as the possible predictors. You may use the step function in R for the questions below and no need to show all the iterative output. Use "trace=0" in the function to suppress the output for each step. This simply stores the final model.

(a) Using the backward selection approach (with AIC), find a good subset of predictors to predict lpsa.

```
# Use function 'step' for variable selection.
library(faraway)
fit_all = lm(lpsa ~ ., data = prostate)
#summary(fit_all)
fit_back_aic = step(fit_all, direction = "backward", trace=0)
#step(fit_all, direction = "backward")
# The resulting model is stored at fit_back_aic

# predictors: lcavol, lweight, age, lbph, svi
fit_back_aic
```

```
##
## Call:
## lm(formula = lpsa ~ lcavol + lweight + age + lbph + svi, data = prostate)
##
## Coefficients:
## (Intercept)       lcavol      lweight          age         lbph          svi
##     0.95100      0.56561      0.42369     -0.01489      0.11184      0.72095
```

(b) Repeat (a) with BIC. By which criterion the smaller subset was obtained? Was the result surprising? Briefly explain.

```
n = nrow(prostate)
fit_back_bic = step(fit_all, direction = "backward", k=log(n), trace=0)
```

```
# predictors: lcavol, lweight, svi
fit_back_bic
```

```
##
## Call:
## lm(formula = lpsa ~ lcavol + lweight + svi, data = prostate)
##
## Coefficients:
## (Intercept)       lcavol      lweight          svi
##     -0.2681       0.5516       0.5085       0.6662
```

Backward selection approach with BIC has smaller subsets. This is not surprising because it is an expected result as BIC has a larger penalty than AIC. When n >= 8, BIC will prefer a smaller model as compared to AIC.

(c) We want to compare the resulting models from (a) and (b) using an F-test. Report the full model, reduced model and the null hypothesis for the test.

$H_0 = \beta_{age} = \beta_{lbph} = 0$ $H_1 =$ At least one of $\beta_j \neq 0$ where j = age, lbph

```
# Under H0,
reduced_model = lm(lpsa ~ lcavol+ lweight + svi, data = prostate)
summary(reduced_model)
```

```
##
## Call:
## lm(formula = lpsa ~ lcavol + lweight + svi, data = prostate)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.72964 -0.45764  0.02812  0.46403  1.57013
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.26809    0.54350  -0.493  0.62298
## lcavol       0.55164    0.07467   7.388  6.3e-11 ***
## lweight      0.50854    0.15017   3.386  0.00104 **
## svi          0.66616    0.20978   3.176  0.00203 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7168 on 93 degrees of freedom
## Multiple R-squared:  0.6264, Adjusted R-squared:  0.6144
## F-statistic: 51.99 on 3 and 93 DF,  p-value: < 2.2e-16
```

```
# Under H1,
full_model = lm(lpsa ~ lcavol+ lweight + age + lbph + svi, data = prostate)
summary(full_model)
```

```
##
```

```
## Call:
## lm(formula = lpsa ~ lcavol + lweight + age + lbph + svi, data = prostate)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.83505 -0.39396  0.00414  0.46336  1.57888
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.95100    0.83175   1.143 0.255882
## lcavol       0.56561    0.07459   7.583 2.77e-11 ***
## lweight      0.42369    0.16687   2.539 0.012814 *
## age         -0.01489    0.01075  -1.385 0.169528
## lbph         0.11184    0.05805   1.927 0.057160 .
## svi          0.72095    0.20902   3.449 0.000854 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7073 on 91 degrees of freedom
## Multiple R-squared:  0.6441, Adjusted R-squared:  0.6245
## F-statistic: 32.94 on 5 and 91 DF,  p-value: < 2.2e-16
```

(d) Obtain the F-value (or F-statistic) for the test in (c) and make a conclusion at alpha = 0.05.

```
anova(reduced_model, full_model)
```

```
## Analysis of Variance Table
##
## Model 1: lpsa ~ lcavol + lweight + svi
## Model 2: lpsa ~ lcavol + lweight + age + lbph + svi
##   Res.Df    RSS Df Sum of Sq     F Pr(>F)
## 1     93 47.785
## 2     91 45.526  2    2.2593 2.258 0.1104
```

```
F_crit = qf(0.95, df1 = 2, df2 = 91)
F_crit
```

```
## [1] 3.096553
```

```
# F0 = 2.258 < F_crit = 3.096553
```

Therefore, we fail to reject H_0. That is, β_{age} and β_{lbph} may be dropped.

Q2 For question 2, you will use the College data in the ISLR package. The goal is to predict the number of applications received using the other variables in the College data set. Use the following traning/test data split:

(a) - (c)

```r
#install.packages('ISLR', repos = "http://cran.us.r-project.org")
library(ISLR)
#install.packages('glmnet', repos = "http://cran.us.r-project.org")
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-3
```

```r
set.seed (10)
a <- sample(777,300,rep=FALSE)
college_trn <- College [a ,]
college_tst <- College[-a,]

# (a) Fit a linear model on the training set, and report the mean square error on the test data.
fit_ts = lm(Apps ~ ., college_trn)
summary(fit_ts)
```

```
##
## Call:
## lm(formula = Apps ~ ., data = college_trn)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5136.1  -501.7   -56.3   362.8  6946.4
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -732.52984  831.92426  -0.881 0.379325
## PrivateYes  -693.55138  230.88291  -3.004 0.002905 **
## Accept         1.72572    0.05684  30.360  < 2e-16 ***
## Enroll        -0.93931    0.31143  -3.016 0.002794 **
## Top10perc     51.86123   10.17581   5.097 6.35e-07 ***
## Top25perc    -12.72975    7.96733  -1.598 0.111220
## F.Undergrad   -0.02715    0.06127  -0.443 0.658019
## P.Undergrad    0.03826    0.07058   0.542 0.588131
## Outstate      -0.10600    0.03184  -3.329 0.000988 ***
## Room.Board     0.11725    0.09074   1.292 0.197371
## Books         -0.10617    0.36319  -0.292 0.770259
## Personal       0.15812    0.12468   1.268 0.205774
## PhD          -13.52079    9.43274  -1.433 0.152854
## Terminal       0.56998   10.82371   0.053 0.958039
## S.F.Ratio     26.61956   25.95282   1.026 0.305916
## perc.alumni    3.61937    7.30765   0.495 0.620784
## Expend         0.09478    0.02348   4.036 7.00e-05 ***
## Grad.Rate     12.78728    5.65881   2.260 0.024602 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1156 on 282 degrees of freedom
## Multiple R-squared:  0.941,  Adjusted R-squared:  0.9374
## F-statistic: 264.5 on 17 and 282 DF,  p-value: < 2.2e-16
```

```
pred_ts = predict(fit_ts, newdata = college_tst) # prediction for test data
mse_ts = mean((pred_ts - college_tst$Apps)^2) # mean squared error (mse) for test data
mse_ts
```

```
## [1] 1012976
```

```
# (b) Use the backward selection (in terms of AIC) to reduce the number of effective predictors. Does t
fit_backward = step(fit_ts, direction="backward", trace=FALSE)
length(fit_backward$coefficients) # = #(beta)
```

```
## [1] 10
```

```
pred_backward = predict(fit_backward, newdata = college_tst)
mse_backward = mean((pred_backward - college_tst$Apps)^2) # mse for the test data
mse_backward
```

```
## [1] 1026868
```

```
mse_ts > mse_backward
```

```
## [1] FALSE
```

```
## No, the resulting model does not achieve a better predictive performance.

# (c) Fit a ridge regression model on the training set, with lamda chosen by cross-validation. Report t
#install.packages("glmnet")

data(College, package = "ISLR")
# To find the best lambda, use the cv.glmnet function (10 folds cv by default).
X = model.matrix(Apps~., College)[, -1] #the first column (for intercept) is eliminated
y = College$Apps

X_tr = X[a,]
y_tr <- y[a]

y_ts <- y[-a]
X_ts <- X[-a,]

# To find the best lambda, use the cv.glmnet function (10 folds cv by default).
fit_ridge_cv = cv.glmnet(X_tr, y_tr, alpha = 0)
```

```r
# The plot illustrates the MSE for the lambda considered.
#plot(fit_ridge_cv)

bestlam = fit_ridge_cv$lambda.min
bestlam
```

```
## [1] 437.6464
```

```r
log(bestlam)
```

```
## [1] 6.081411
```

```r
# This is the ridge regression fit using the best lambda
fit_ridge_best = glmnet(X_tr, y_tr, alpha = 0, lambda = bestlam)

# predict Y for the test data
pred_ridge = predict(fit_ridge_best, s = bestlam, newx = X_ts)
mse_ridge = mean((pred_ridge - y_ts)^2)
mse_ridge
```

```
## [1] 959735.6
```

(d) Fit a lasso model on the training set, with lamda chosen by cross validation. Report the test error obtained, along with the number of non-zero coefficient estimates.

```r
# For the LASSO, use alpha=1
library(glmnet)
#install.packages("glmnet")
fit_lasso_cv = cv.glmnet(X_tr, y_tr, alpha = 1)

bestlam = fit_lasso_cv$lambda.min
bestlam
```

```
## [1] 38.06425
```

```r
log(bestlam)
```

```
## [1] 3.639276
```

```r
fit_lasso_best = glmnet(X_tr, y_tr, alpha = 1, lambda = bestlam)
coef(fit_lasso_best)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept) -838.82807481
## PrivateYes  -486.93558801
## Accept          1.57654294
## Enroll         -0.55264334
## Top10perc      32.26837080
## Top25perc       .
## F.Undergrad     .
## P.Undergrad     .
## Outstate       -0.06611547
## Room.Board      0.04733608
## Books           .
## Personal        0.05523525
## PhD            -6.29043532
## Terminal        .
## S.F.Ratio       4.20172890
## perc.alumni     .
## Expend          0.08006997
## Grad.Rate       8.94321570
```

```
sum(coef(fit_lasso_best) != 0)
```

```
## [1] 12
```

```
pred_lasso = predict(fit_lasso_best, s = bestlam, newx = X_ts)
mse_lasso = mean((pred_lasso-y_ts)^2)
mse_lasso
```

```
## [1] 950903.3
```

Q3

(a) Obtain the probability of Y = 1 at $x_1 = 1$ and $x_2 = 0.5$.

$$p(x) = \frac{e^{g(x)}}{1+e^{g(x)}} = \frac{1}{1+e^{-g(x)}}$$

where

$$g(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

. Since $x_1 = 1$, $x_2 = 0.5$, and β_0, β_1, β_2 are given from summary table, we can plug all.

$$g(x) = -2.7399 + 3.0287 \times 1 + (-1.2081) \times 0.5 = -0.31525$$

$$p(x) = \frac{1}{1+e^{0.31525}} = 0.421833793$$

(b) Test H_0 :$\beta_2 = 0$ vs H_1 :$\beta_2 \neq 0$ at $\alpha = 0.05$.

$$z^* = \frac{\beta_2 - 0}{SE(\hat{\beta}_2)} = \frac{-1.2081}{0.4620} = -2.614935064$$

p-val $= 2 \times p(z > |z^*|)$

```r
2*(1-pnorm(abs(-2.614935065), 0, 1))
```

```
## [1] 0.008924442
```

Since p-value < 0.05, we reject H_0 and conclude that β_2 is significant.

(c) Test $H_0 : \beta_1 = \beta_2 = 0$ vs $H_1 : H_0$ is false, at $\alpha = 0.05$.

$$D = Dev_{reduced} - Dev_{full} = 110.216 - 56.436 = 53.78$$

p-val = p($\chi^2_k > D$) where k = p - q = 3 - 1 = 2. Plug D and k. p-val = p($\chi^2_2 > 53.73$)

```r
1-pchisq(53.73, 2)
```

```
## [1] 2.151168e-12
```

Since p-val < 0.05, we reject H_0.

Q4 For this problem, you use the ILPD (Indian Liver Patient Dataset) data. Import the data from https://raw.githubusercontent.com/hgweon2/data/main/ILPD2.txt. The imported dataset contains 579 observations with 11 variables. Use Selector as the response variable and the others as predictors. Additional information about the variables can be found at: https://archive.ics.uci.edu/ml/datasets/ILPD+%28Indian+Liver+Patient+Dataset%29.

For the following questions, use the first 400 observations of the data set as the training data and the rest (179 observations) as the test data.

```r
q4_data <- read.csv("https://raw.githubusercontent.com/hgweon2/data/main/ILPD2.txt")
q4_data$Selector <- (q4_data$Selector == 1) + 0
a = 1:400
tr = q4_data[a,]
head(tr)
```

```
##   Age Gender  TB  DB Alkphos Sgpt Sgot  TP ALB A.G Selector
## 1  65   Male 1.4 0.6     260   28   24 5.2 2.2 0.7        0
## 2  38   Male 1.7 0.7     859   89   48 6.0 3.0 1.0        1
## 3  50   Male 7.3 3.6    1580   88   64 5.6 2.3 0.6        0
## 4  17   Male 0.9 0.2     279   40   46 7.3 4.0 1.2        0
## 5  31 Female 1.1 0.3     190   26   15 7.9 3.8 0.9        1
## 6  34   Male 4.1 2.0     289  875  731 5.0 2.7 1.1        1
```

```
nrow(tr)
```

```
## [1] 400
```

```
ts = q4_data[-a,]
head(ts)
```

```
##     Age Gender  TB  DB Alkphos Sgpt Sgot  TP ALB  A.G Selector
## 401  62   Male 7.3 4.1     490   60   68 7.0 3.3 0.89        1
## 402  26 Female 0.9 0.2     154   16   12 7.0 3.5 1.00        1
## 403  74 Female 1.1 0.4     214   22   30 8.1 4.1 1.00        1
## 404  25   Male 0.6 0.1     183   91   53 5.5 2.3 0.70        0
## 405  40 Female 0.9 0.3     293  232  245 6.8 3.1 0.80        1
## 406  51   Male 2.9 1.3     482   22   34 7.0 2.4 0.50        1
```

```
nrow(ts)
```

```
## [1] 179
```

(a) Fit a logistic model to the training data. Use the trained model to predict the test data (cutoff 0.5). Using the prediction results, make a confusion matrix between Y and Y hat and report the accuracy, sensitivity (recall), specificity and precision of the prediction.

```
# For logistic regression, specify the family to binomial(link = "logit")
fit_logistic = glm(Selector ~ ., data = tr, family = binomial(link = "logit"))
summary(fit_logistic)
```

```
##
## Call:
## glm(formula = Selector ~ ., family = binomial(link = "logit"),
##     data = tr)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0296  -1.0716   0.4312   0.8865   1.5109
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.944648   1.531540  -1.923  0.05452 .
## Age          0.012462   0.007634   1.632  0.10259
## GenderMale  -0.135601   0.287871  -0.471  0.63761
## TB          -0.387509   0.358441  -1.081  0.27965
## DB           0.991516   0.733523   1.352  0.17647
## Alkphos      0.001557   0.000990   1.572  0.11586
## Sgpt         0.010187   0.005601   1.819  0.06897 .
## Sgot         0.002184   0.003064   0.713  0.47605
## TP           1.145910   0.439200   2.609  0.00908 **
## ALB         -2.330976   0.872682  -2.671  0.00756 **
## A.G          2.453004   1.318927   1.860  0.06291 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 474.36  on 399  degrees of freedom
## Residual deviance: 401.22  on 389  degrees of freedom
## AIC: 423.22
##
## Number of Fisher Scoring iterations: 7
```

```
#fit_logistic = glm(Selector ~ ., data = tr, family = binomial)
#summary(fit_logistic)
#pred_ts = predict(fit_glm, newdata = ts) # prediction for test data

# probability outcomes for test data
prob_tst <- predict(fit_logistic, newdata=ts, type="response")

# Obtain Y_hat values for the data observation (cutoff=0.5)
n = nrow(ts)
cutoff = 0.5
y_hat = rep(0,n)
idx = which(prob_tst>cutoff)
y_hat[idx] = 1
y_hat
```

```
##   [1] 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1
##  [38] 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1
##  [75] 1 0 1 0 1 1 1 0 1 0 1 1 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1 0 0 1 1 0 1 0 1 1
## [112] 1 1 1 1 1 1 0 1 0 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 0 1 1
## [149] 0 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
# confusion matrix at cutoff=0.5
conf_mat = table(predicted = y_hat, actual = ts$Selector)
conf_mat
```

```
##          actual
## predicted   0   1
##         0  16  13
##         1  37 113
```

```
# Evaluate the prediction outcomes at cutoff 0.5
mean(y_hat == ts$Selector) # Accuracy
```

```
## [1] 0.7206704
```

```
conf_mat[2, 2] / sum(conf_mat[, 2]) # Sensitivity
```

```
## [1] 0.8968254
```

```
conf_mat[1, 1] / sum(conf_mat[, 1]) # Specificity
```

```
## [1] 0.3018868
```

```
conf_mat[2, 2] / sum(conf_mat[2, ]) # Precision
```

```
## [1] 0.7533333
```

(b) Repeat (a) using cutoff = 0.8.

```
# Obtain Y_hat values for the data observation (cutoff=0.8)
n = nrow(ts)
cutoff = 0.8
y_hat = rep(0,n)
idx = which(prob_tst>cutoff)
y_hat[idx] = 1
y_hat
```

```
##   [1] 1 0 0 0 1 1 1 1 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 1 1 0 1 0 0 1 1 1 0 1 0 0 1
##  [38] 0 0 0 1 0 0 0 0 0 1 1 0 0 1 1 0 1 0 1 1 1 0 0 0 0 0 0 0 1 0 1 0 0 1 1 0 0
##  [75] 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0
## [112] 0 0 0 0 1 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0
## [149] 0 0 0 0 1 0 1 1 1 0 1 0 1 0 0 0 1 1 0 0 1 0 1 0 1 1 1 1 1 1 1
```

```
# confusion matrix at cutoff=0.8
conf_mat = table(predicted = y_hat, actual = ts$Selector)
conf_mat
```

```
##          actual
## predicted  0  1
##         0 52 64
##         1  1 62
```

```
# Evaluate the prediction outcomes at cutoff 0.8
mean(y_hat == ts$Selector) # Accuracy
```

```
## [1] 0.6368715
```

```
conf_mat[2, 2] / sum(conf_mat[, 2]) # Sensitivity
```

```
## [1] 0.4920635
```

```
conf_mat[1, 1] / sum(conf_mat[, 1]) # Specificity
```

```
## [1] 0.9811321
```

```
conf_mat[2, 2] / sum(conf_mat[2, ]) # Precision
```

```
## [1] 0.984127
```

(c) If we want to increase the sensitivity of prediction (using the same logistic model), how should the cutoff be changed from 0.5 (decrease/increase)? Briefly explain.

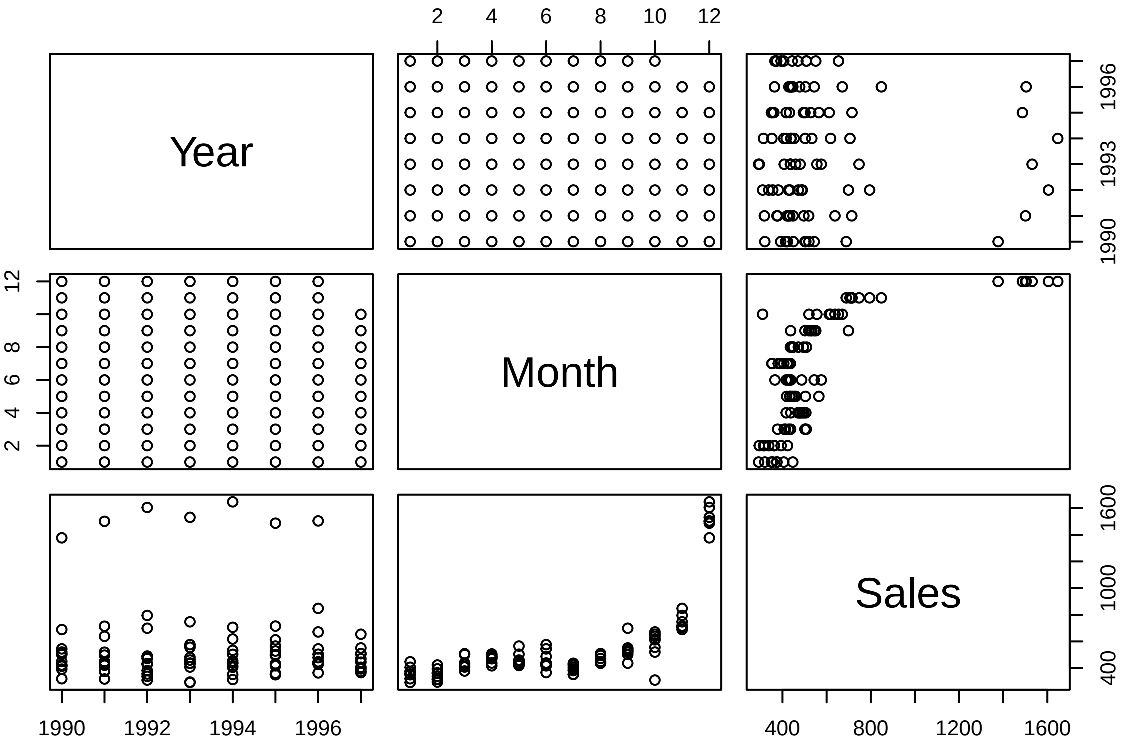
The cutoff should be changed from 0.5 decrease. We can control the number of $\hat{Y} = 1$ by changing the cutoff value. If cutoff value is increasing, the number of $\hat{Y} = 1$ is decreasing, and the number of $\hat{Y} = 0$ is increasing. If cutoff value is decreasing, there should be many $\hat{Y} = 1$ and less $\hat{Y} = 0$. That is, many oberservations with Y=1 will be correctly predicted.

Q5 Import the data from https://raw.githubusercontent.com/hgweon2/ss3859/master/hw4-data1.csv The imported dataset contains monthly sales (94 observations) for a bookstore. All data are in $100.

```
q5_data <- read.csv("https://raw.githubusercontent.com/hgweon2/ss3859/master/hw4-data1.csv")
```

(a) We want to regress sales on the time variables year and month. Check the scatterplot between sales and month, and comment on the monthly sales pattern. Fit models A and B and compare them in terms of adjusted R2.

```
# Check the scatterplot between sales and month, and comment on the monthly sales pattern.
pairs(~.,data = q5_data)
```



```r
# model A - both year and month are used as numerical predictors
model_a_fit = lm(Sales ~. , data = q5_data)

# model B - year is numerical but month is used as a categorical predictor.
model_b = q5_data
model_b$Month  = as.factor(q5_data$Month)
model_b_fit = lm(Sales~., data=model_b)

summary(model_a_fit)$adj.r.squared
```

```
## [1] 0.4321569
```

```r
summary(model_b_fit)$adj.r.squared
```

```
## [1] 0.9581081
```

From the scatterplot, sales are increasing as month is increasing. model_b_fit is preferred as adjusted R^2 is larger than model_a_fit.

(b) Using model B, describe the yearly trend and the seasonal pattern. Use this model to predict the sales for the next 12 months. Discuss all model assumptions used in your predictions.

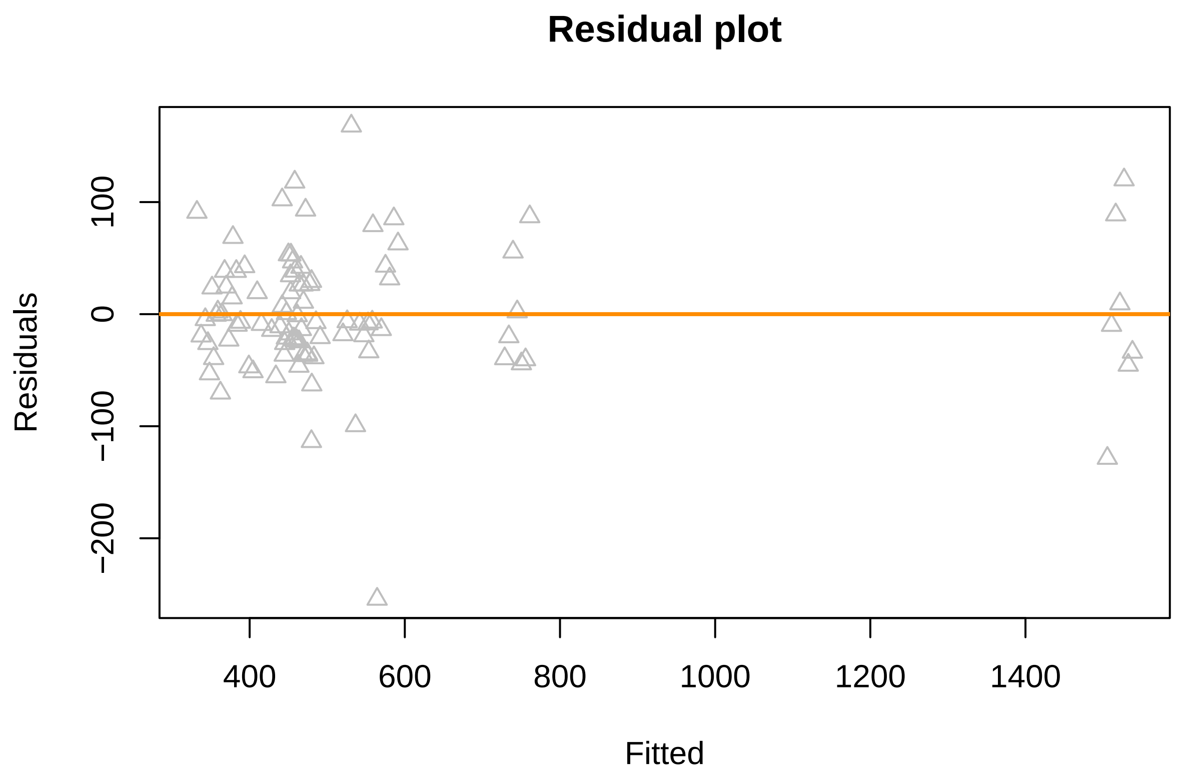
```
#model_b_fit
#yearly => increasing
#predict(model_b_fit, newdata = model_b)

year.new = c(1997, 1997, rep(1998,10))
month.new = as.factor(c(11,12,1:10))
x.new = data.frame(Year=year.new, Month=month.new)
predict(model_b_fit, newdata = x.new)
```

```
##        1        2        3        4        5        6        7        8
##  766.395 1543.252  389.230  375.105  471.480  496.230  488.230  484.980
##        9       10       11       12
##  420.480  485.105  563.355  596.605
```

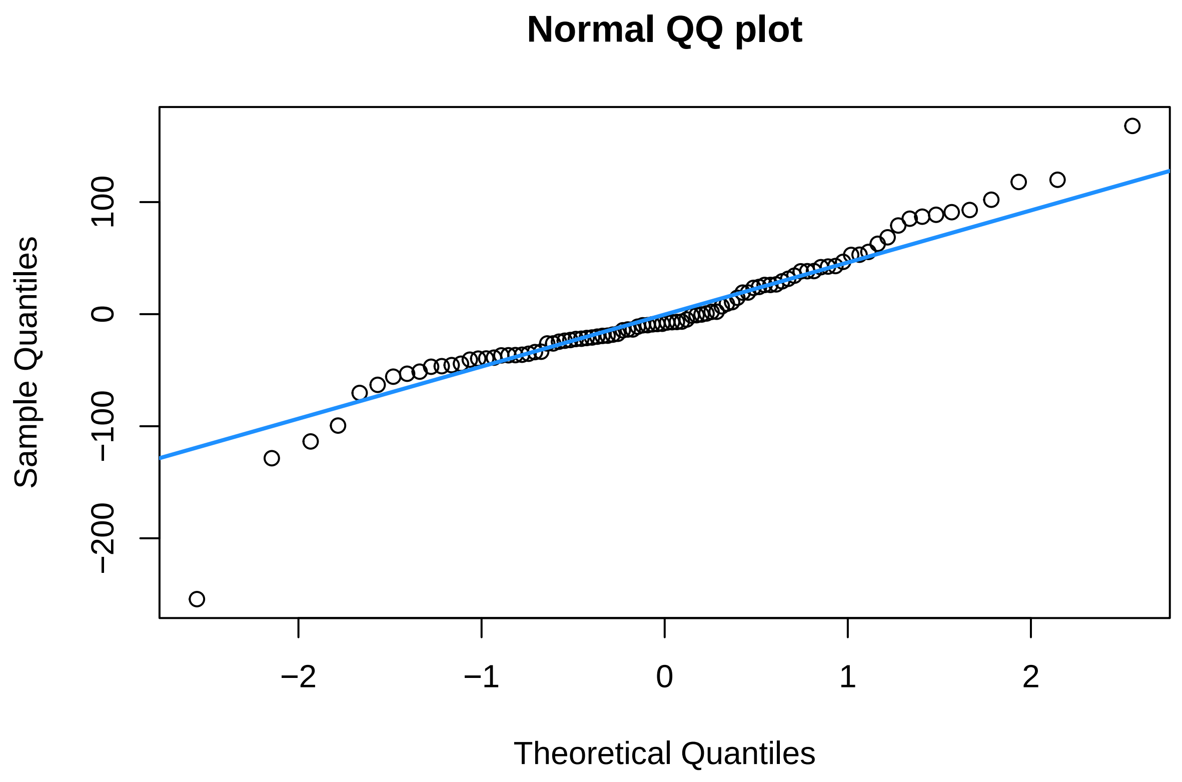
We Assume random errors are independent. Three assumptions: linearity, normality and equal variance.

(c) Check the model assumptions (model B). In particular, investigate whether adjacent residuals (lag 1) are correlated, using the Durbin-Watson test.

```
plot(fitted(model_b_fit), resid(model_b_fit), col="grey", pch=2, xlab = "Fitted", ylab="Residuals", mai
abline(h=0, col="darkorange", lwd=2)
```



```
qqnorm(resid(model_b_fit), main = "Normal QQ plot")
qqline(resid(model_b_fit), col="dodgerblue", lwd = 2)
```



```
#from the residual plot: the mean of the residuals is roughly close to zero at any fitted value.
#hence the linearity assumption holds.

library(lmtest)
```

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

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

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

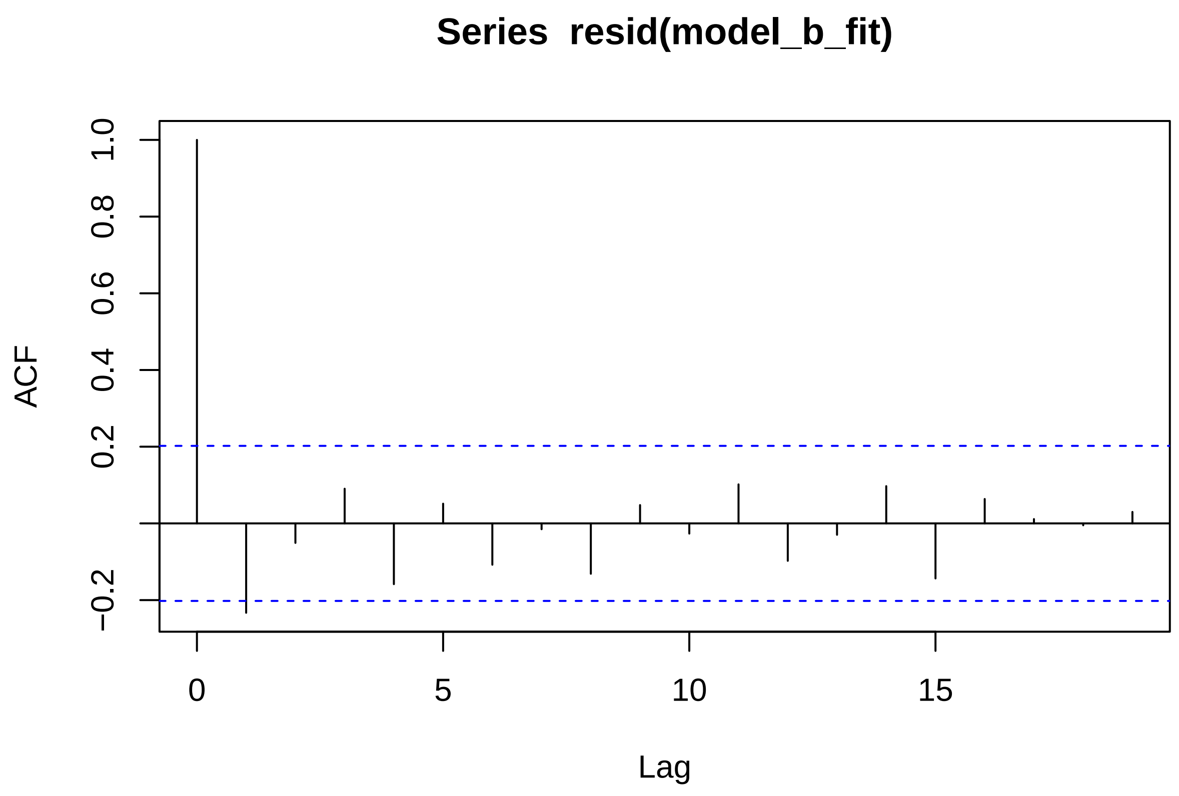
```
bptest(model_b_fit)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  model_b_fit
## BP = 13.9, df = 12, p-value = 0.3071
```

```
#since p-value of the test was greater than 0.05, there was no evidence against the null hypothesis. Tr

shapiro.test(resid(model_b_fit))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(model_b_fit)
## W = 0.93187, p-value = 0.0001059
```

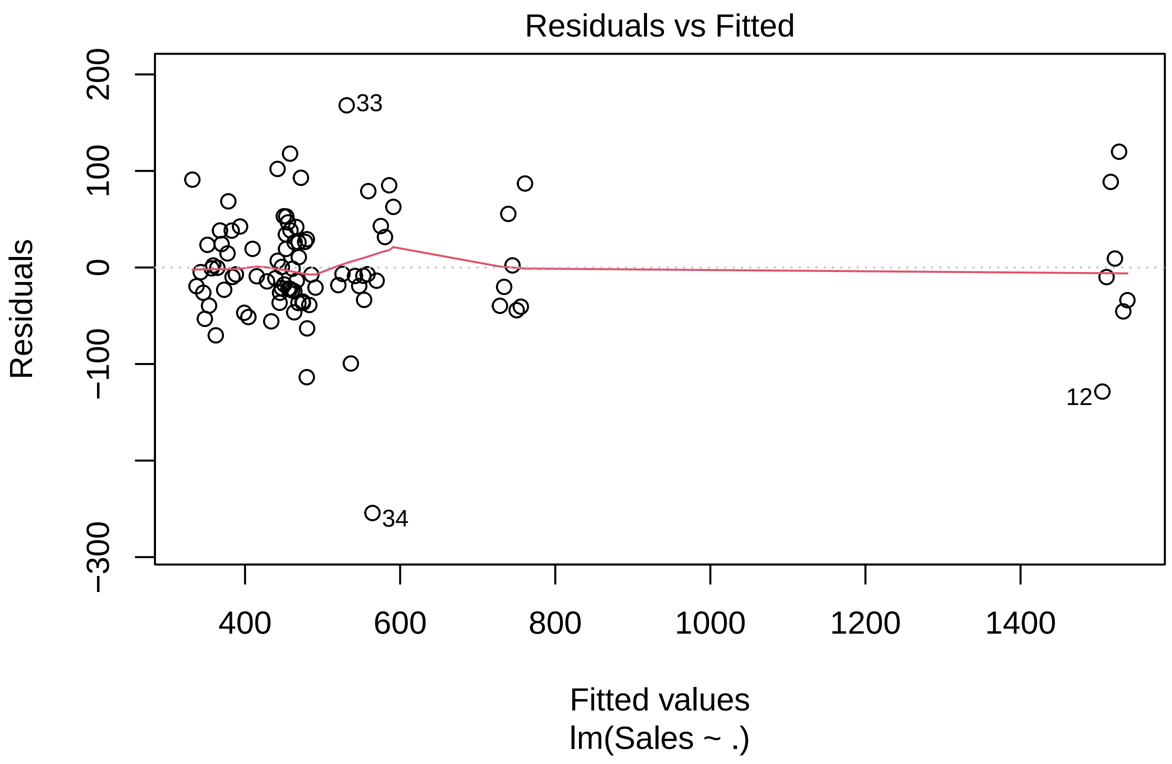
```
#since p-value of the test was lower than 0.05, we reject the null hypothesis and conclude that
#the normal assumption is violated

acf(resid(model_b_fit))
```

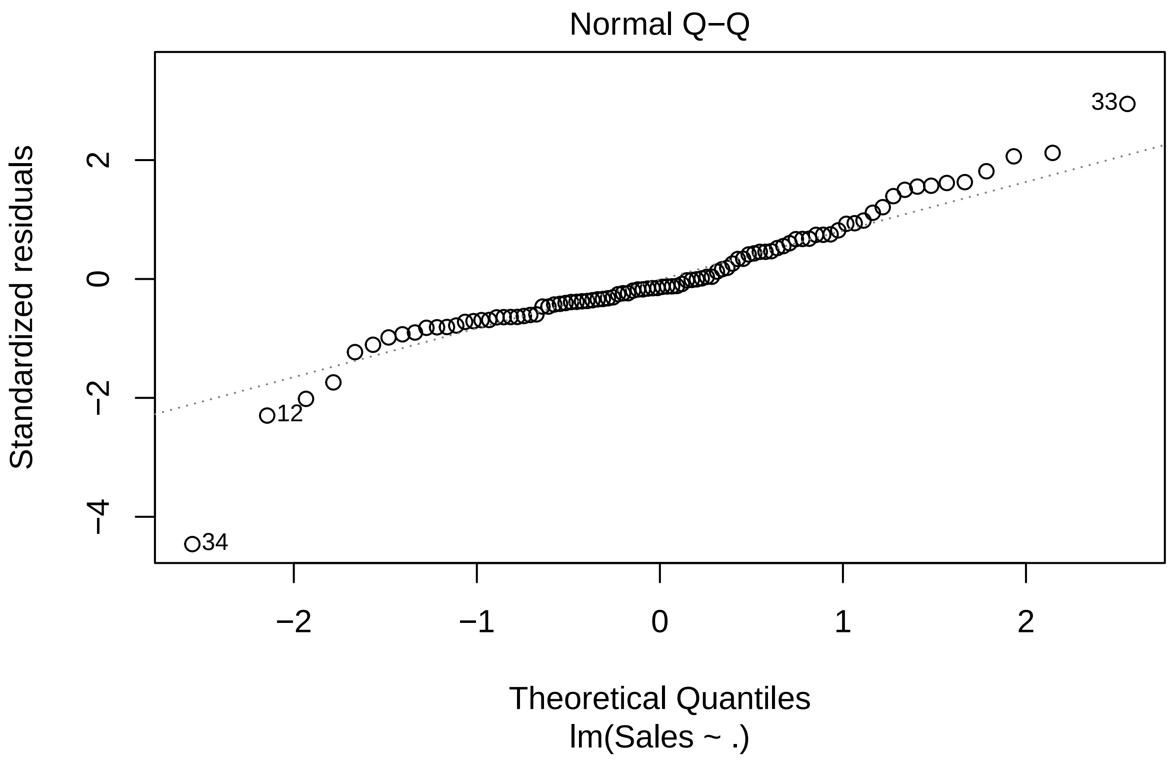


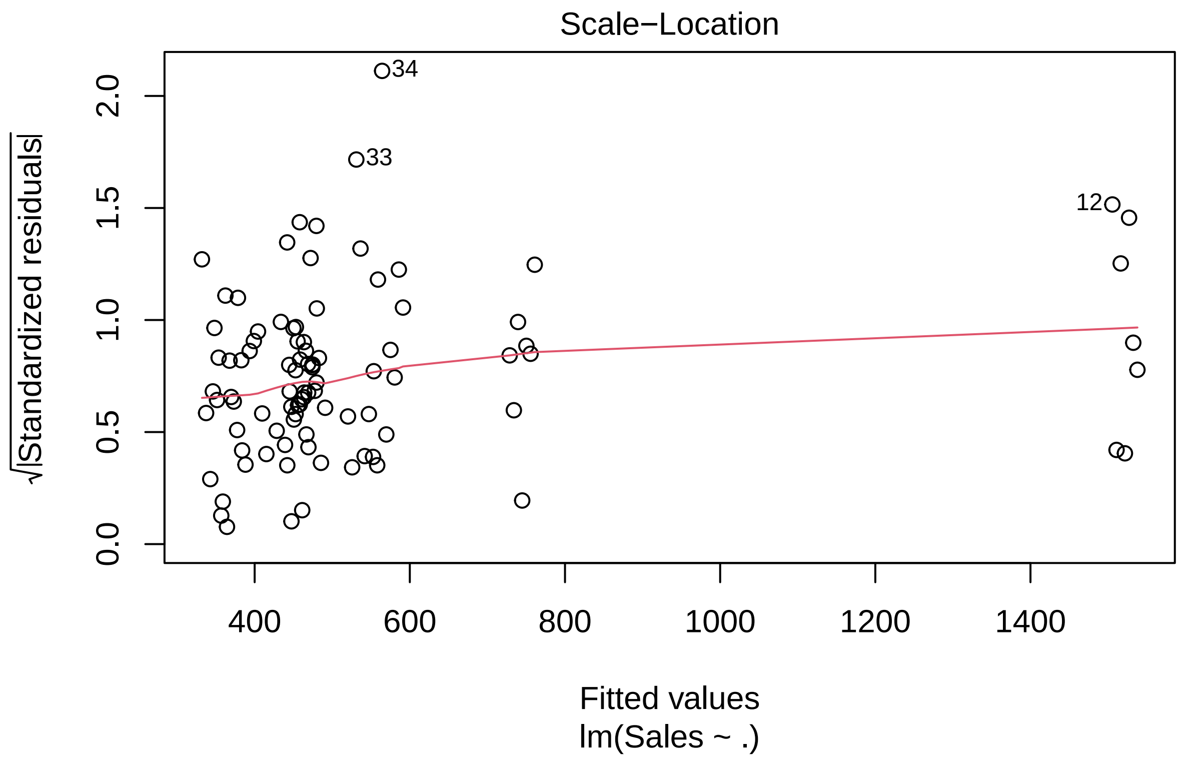
```
#From ACF plot, it is a negative autocorrelation.

dwtest(model_b_fit, alternative="two.sided")
```

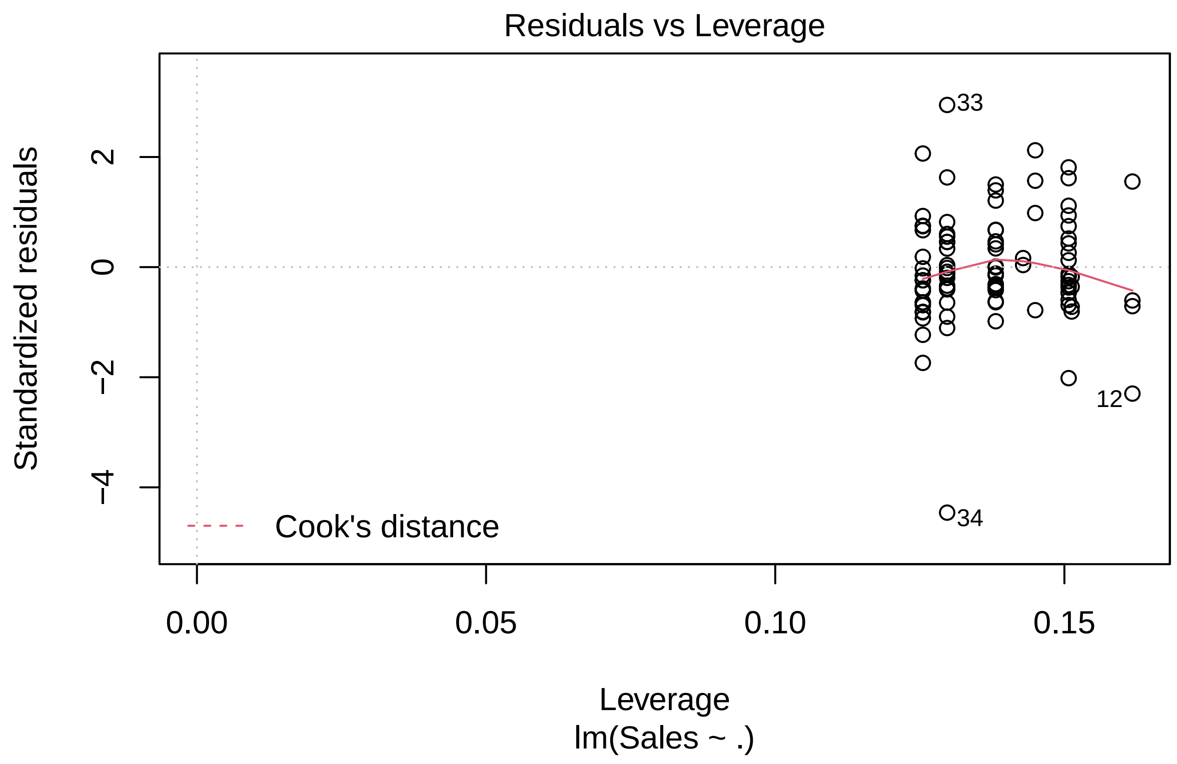
```
##
##  Durbin-Watson test
##
## data:  model_b_fit
## DW = 2.4509, p-value = 0.03902
## alternative hypothesis: true autocorrelation is not 0
```

```
# since p-value of the test is lower than 0.05, we reject the null hypothesis and conclude
#that rho is not 0.

plot(model_b_fit)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Sales ~ .)
33
12
34
−4
−2
0
2
−2
−1
0
1
2


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
400
600
800
1000
1200
1400
34
33
12
Fitted values
lm(Sales ~ .)



(d) Assuming that the errors follow a first-order autoregressive model, estimate the lag 1 autocorrelation ρ. Using this estimate, fit another model (model C) that results in the best linear unbiased estimator of β. Use the ACF plot to check whether the error independence assumption is met in this model. Compare model B and C in terms of AIC.

```r
#estimate the lag 1 autocorrelation rho
rho_hat_dw = (1-dwtest(model_b_fit)$statistic/2)
rho_hat_dw
```

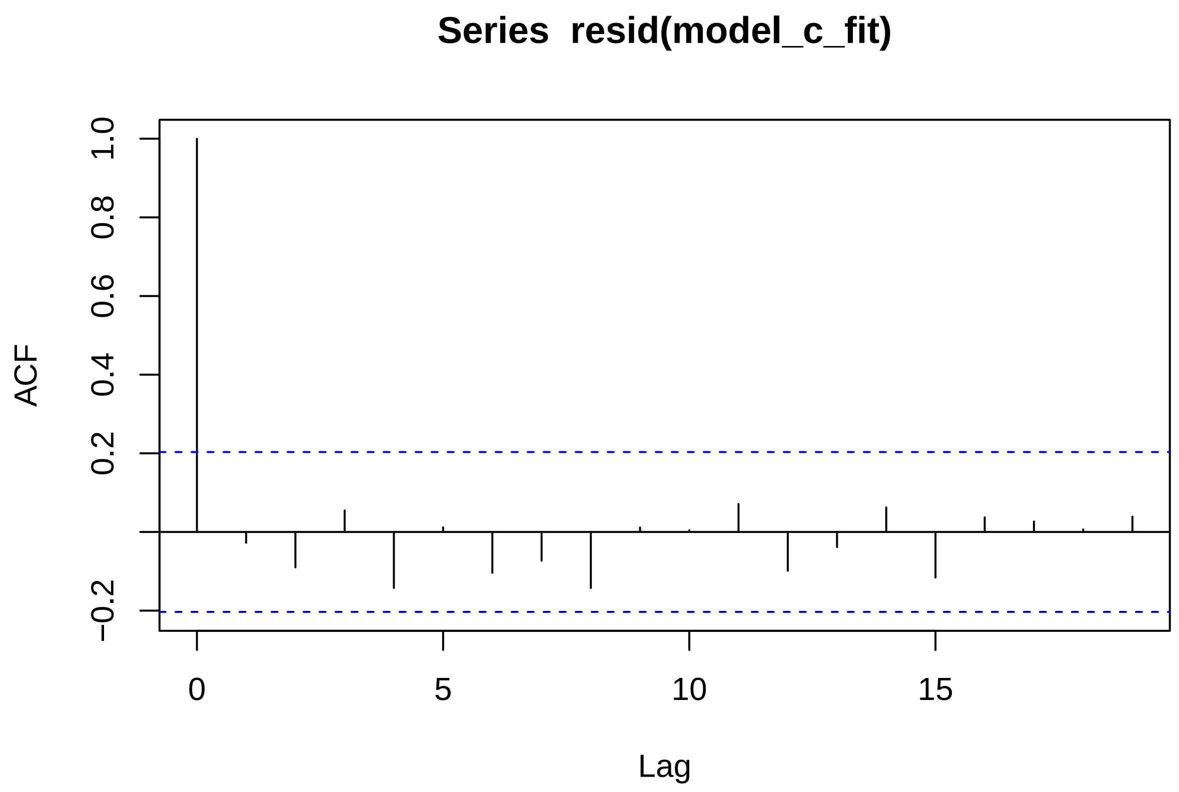
```
##        DW
## -0.225457
```

```r
#using this estimate, fit another model (model C)
num_obs = nrow(q5_data)
y_t = q5_data$Sales[-1]
y_t_1 = q5_data$Sales[-num_obs]
y_new = y_t - rho_hat_dw*y_t_1 # transformed y

year_t = q5_data$Year[-1]
year_t_1 = q5_data$Year[-num_obs]
year_new = year_t - rho_hat_dw*year_t_1
```

```r
month_t = q5_data$Month[-1]
month_t_1 = q5_data$Month[-num_obs]
month_new = month_t - rho_hat_dw*month_t_1

x_new = data.frame(Year = year_new, Month = month_new) # transformed x

q5_new = data.frame(Sales = y_new, Year = year_new, Month = month_new)

q5_new$Month = as.factor(q5_new$Month)

model_c_fit = lm(Sales ~ ., data=q5_new)

dwtest(model_c_fit, alternative = "two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  model_c_fit
## DW = 2.0167, p-value = 0.9509
## alternative hypothesis: true autocorrelation is not 0
```

```r
# Use the ACF plot to check whether the error independence assumption is met in this model
acf(resid(model_c_fit))
```



```
#acf(resid(model_a_fit))
#acf(resid(model_b_fit))
#dwtest(model_c_fit, alternative="two.sided")

# ACF plots and DW tests confirm that model_c_fit has no error dependence issue.

# Compare model B and C in terms of AIC.
AIC(model_b_fit)
```

```
## [1] 1054.002
```

```
AIC(model_c_fit)
```

```
## [1] 1038.544
```

```
# model_c_fit is preferred as it has smaller AIC than model_b_fit
```