STATS415Project

Yunguo Cai

3/25/2018

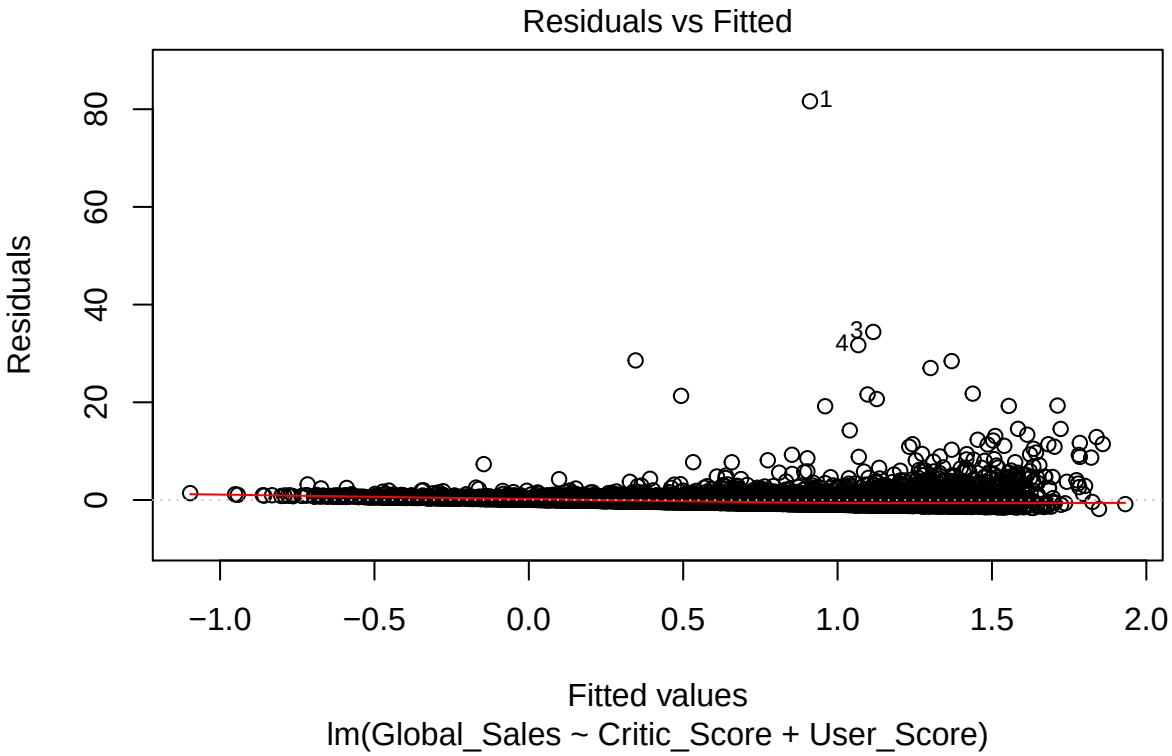
```r
data=read.csv("/Users/cyunguo/Desktop/Video_Games_Sales_as_at_22_Dec_2016.csv")
data=na.omit(data)
#summary(data$Global_Sales)
```

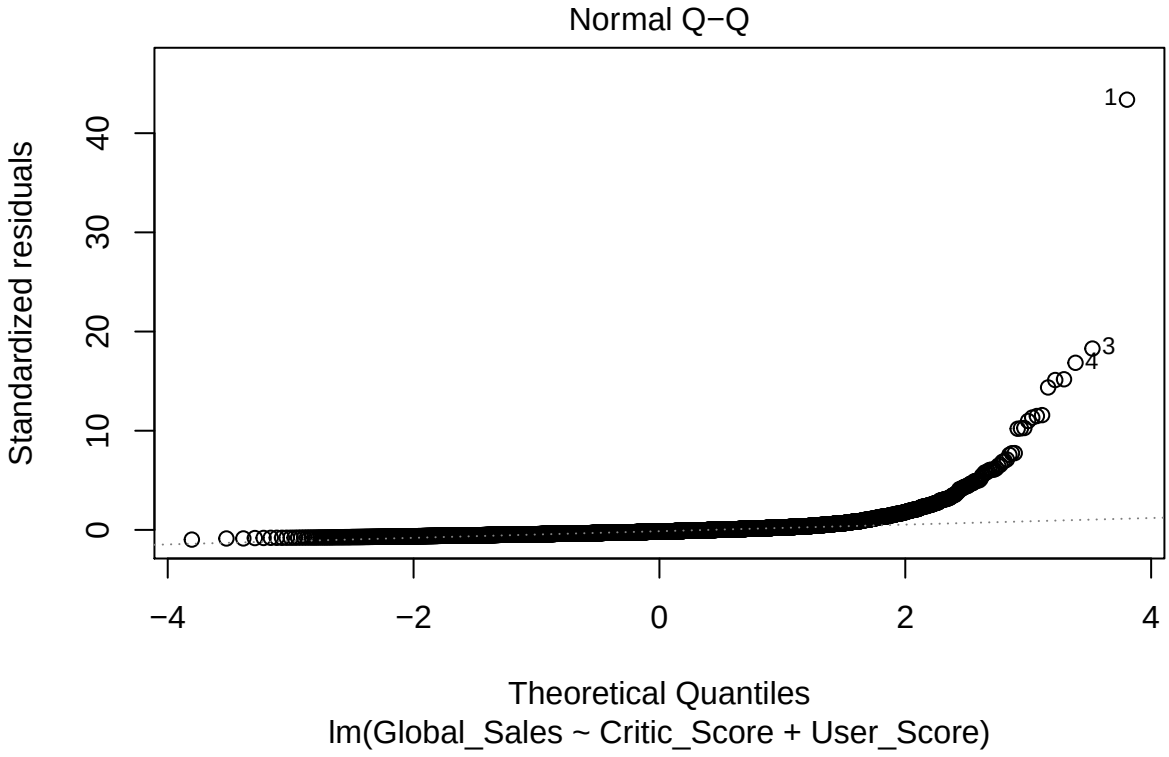
%model1 with critic score and user score

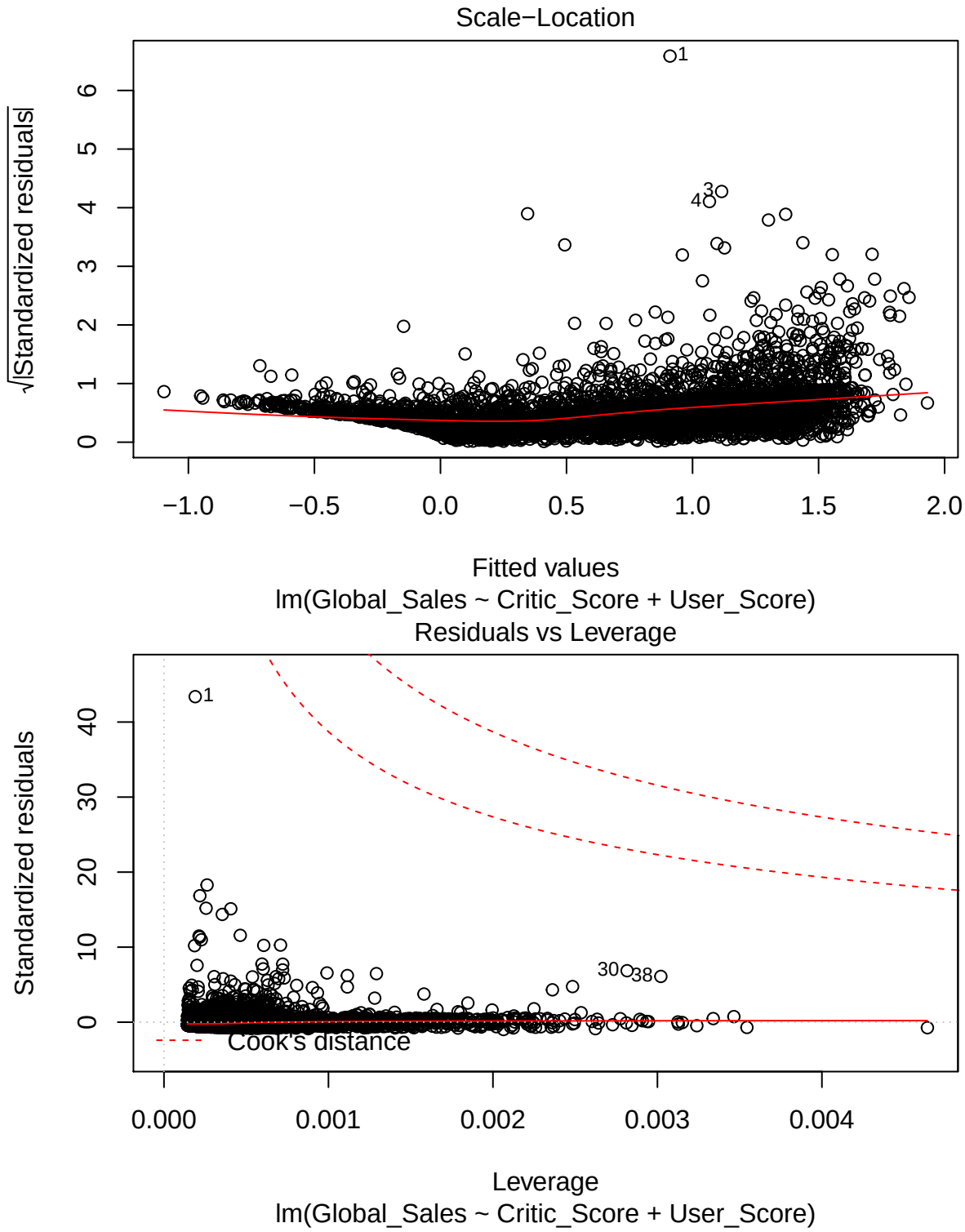
```r
data$User_Score <- as.numeric(data$User_Score)
data$Critic_Score <- as.numeric(data$Critic_Score)
mod1 <- lm(Global_Sales~Critic_Score+User_Score, data = data)
summary(mod1)
```

```
##
## Call:
## lm(formula = Global_Sales ~ Critic_Score + User_Score, data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.837 -0.670 -0.310  0.175 81.619
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.276029   0.128072  -9.963  < 2e-16 ***
## Critic_Score  0.039136   0.001988  19.683  < 2e-16 ***
## User_Score   -0.009971   0.001915  -5.206 1.98e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.882 on 7014 degrees of freedom
## Multiple R-squared:  0.05978,    Adjusted R-squared:  0.05951
## F-statistic:   223 on 2 and 7014 DF,  p-value: < 2.2e-16
```

```r
plot(mod1)
```


Residuals vs Fitted
Residuals
Fitted values
lm(Global_Sales ~ Critic_Score + User_Score)
1
3
4
0
20
40
60
80
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Global_Sales ~ Critic_Score + User_Score)
1
3
4
0
10
20
30
40
−4
−2
0
2
4



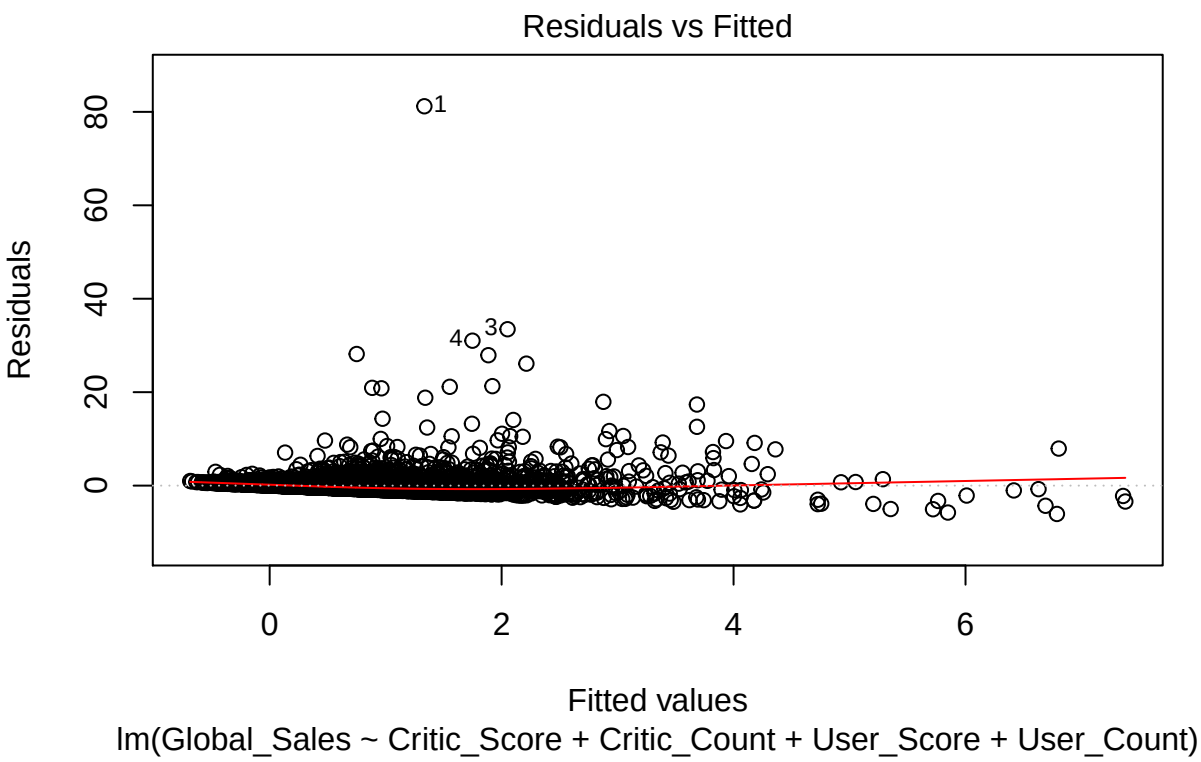
%mod2 with scores and counts

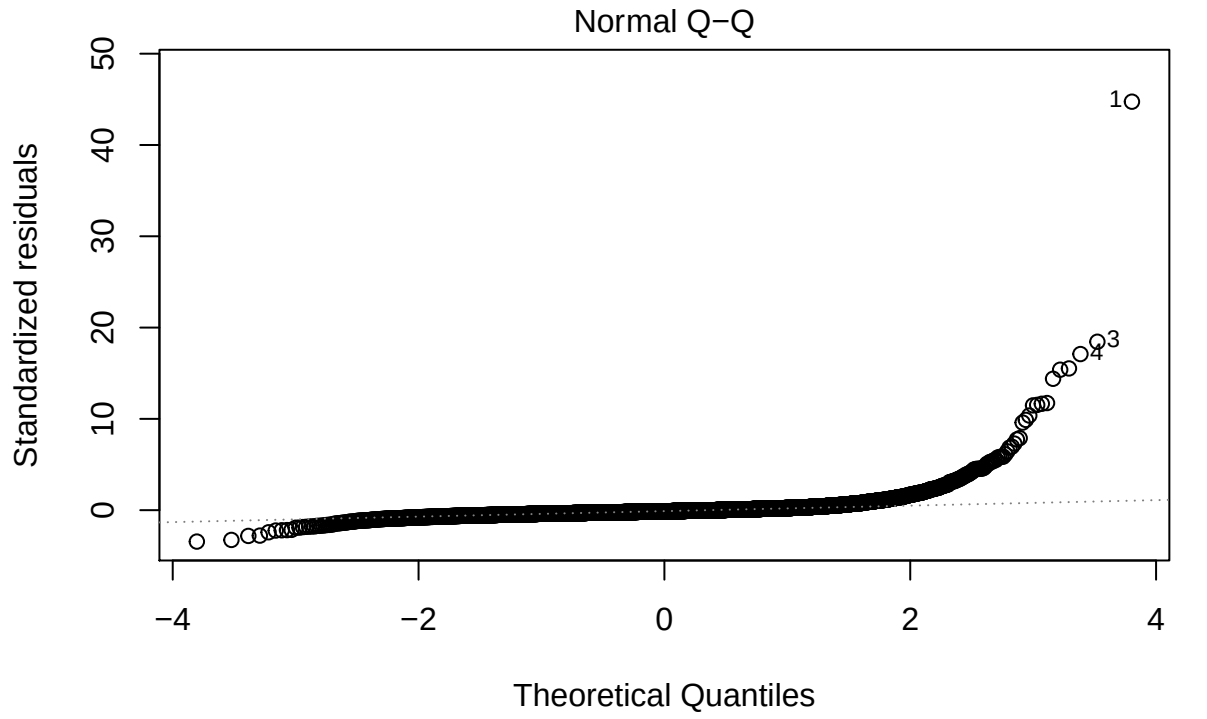
```
mod2 <- lm(Global_Sales~Critic_Score+Critic_Count+User_Score+User_Count, data = data)
summary(mod2)
```

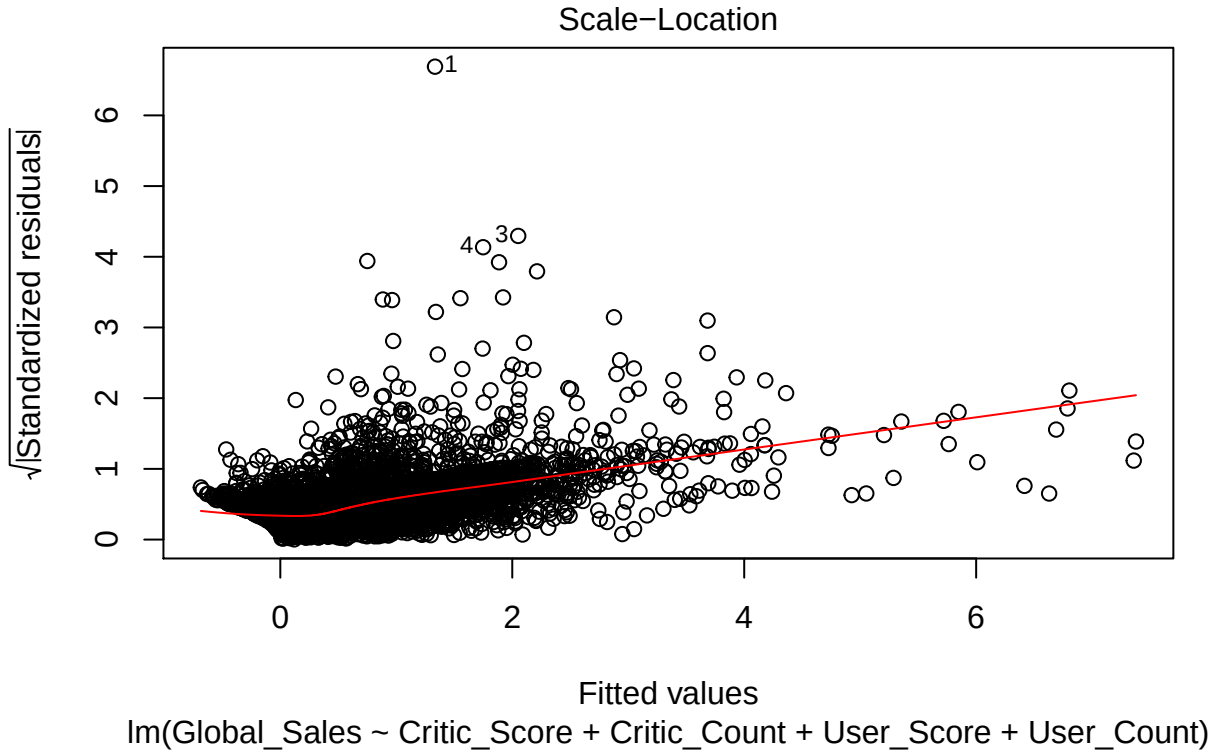
```
##
## Call:
```

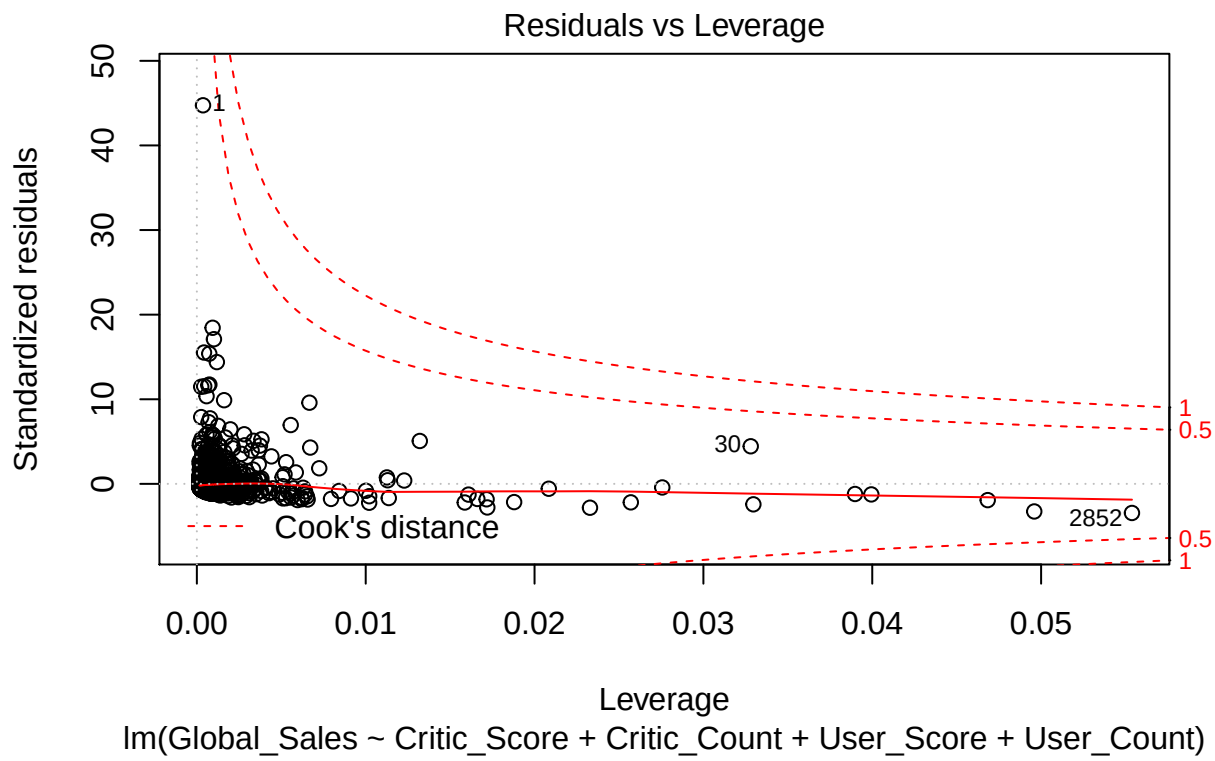
```
## lm(formula = Global_Sales ~ Critic_Score + Critic_Count + User_Score +
##     User_Count, data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.068 -0.557 -0.227  0.179 81.196
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -9.420e-01  1.245e-01  -7.567 4.31e-14 ***
## Critic_Score  1.983e-02  2.095e-03   9.469  < 2e-16 ***
## Critic_Count  1.840e-02  1.285e-03  14.320  < 2e-16 ***
## User_Score   -4.329e-03  1.875e-03  -2.308    0.021 *
## User_Count    5.340e-04  4.093e-05  13.047  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.815 on 7012 degrees of freedom
## Multiple R-squared:  0.1255, Adjusted R-squared:  0.125
## F-statistic: 251.5 on 4 and 7012 DF,  p-value: < 2.2e-16
```

```r
plot(mod2)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Global_Sales ~ Critic_Score + Critic_Count + User_Score + User_Count)
1
3
4


Scale−Location
√|Standardized residuals|
Fitted values
lm(Global_Sales ~ Critic_Score + Critic_Count + User_Score + User_Count)
1
3
4

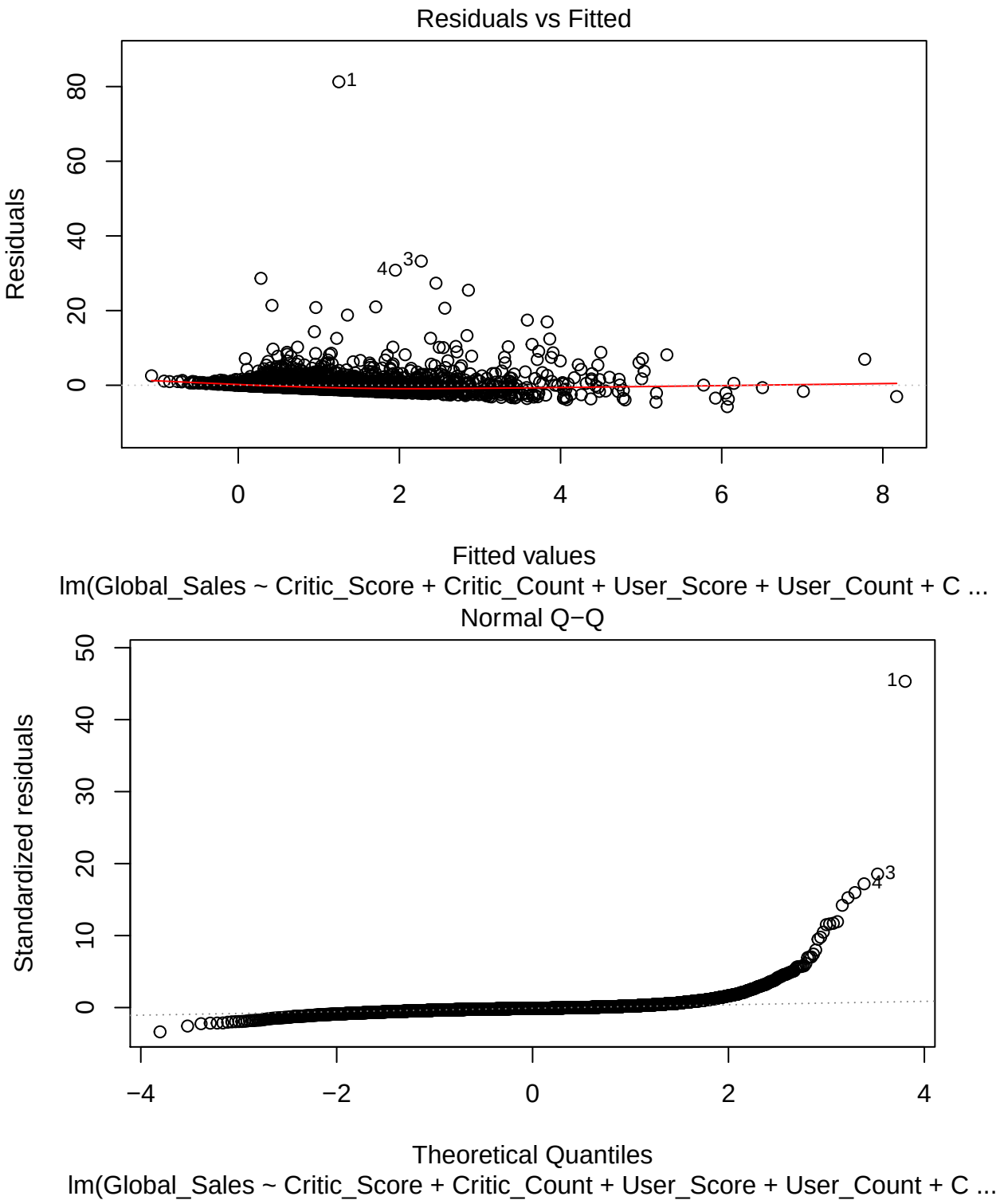


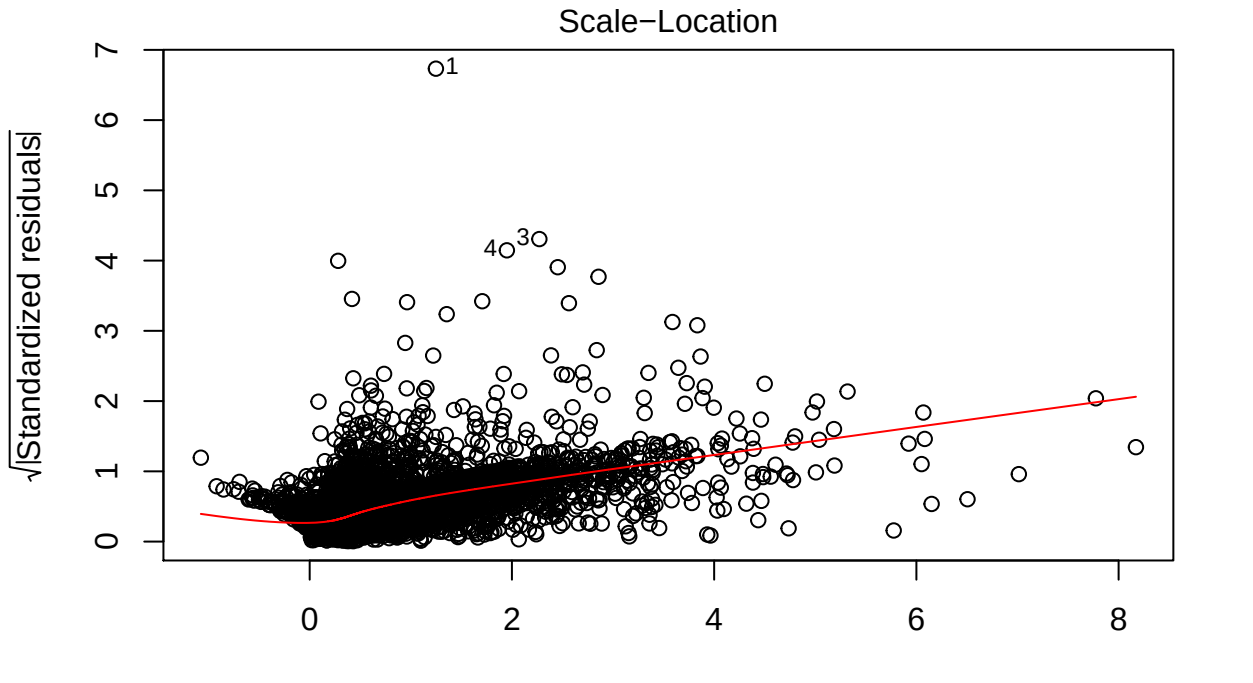
%mod3 with scores counts and interaction

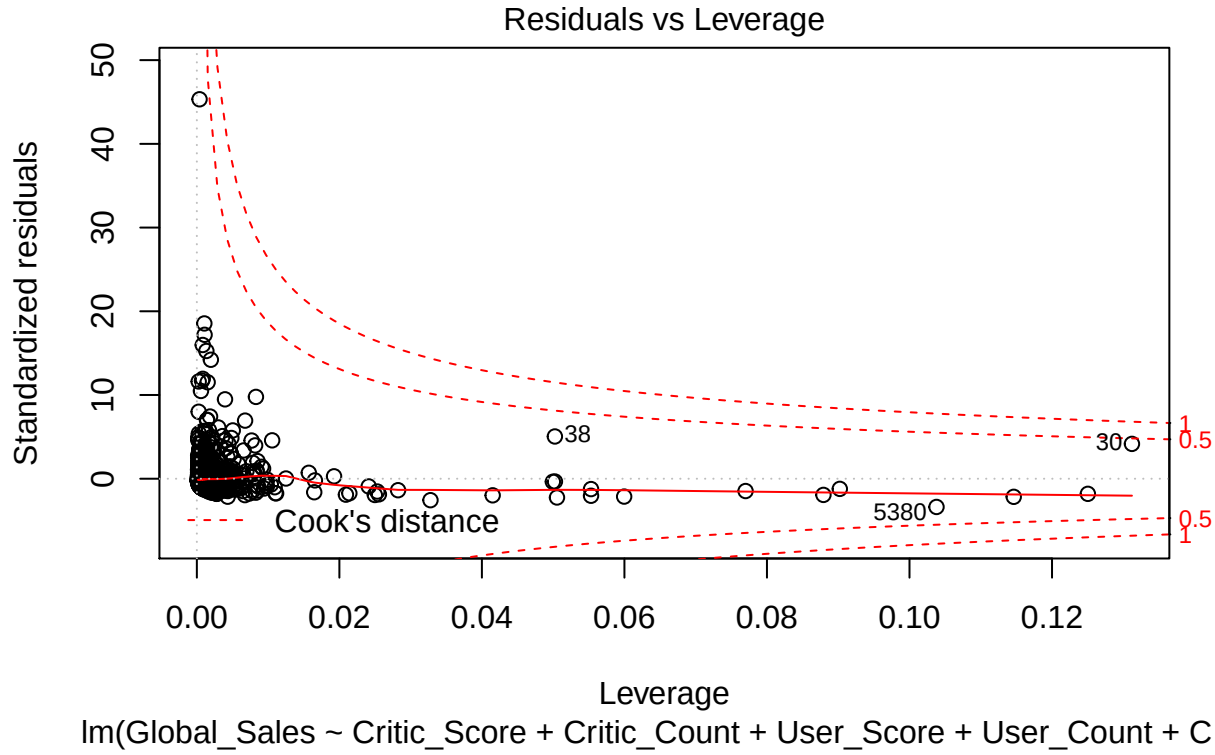
```
mod3 <- lm(Global_Sales~Critic_Score+Critic_Count+User_Score+User_Count+Critic_Score:Critic_Count+User_S
summary(mod3)
```

```
##
## Call:
## lm(formula = Global_Sales ~ Critic_Score + Critic_Count + User_Score +
##     User_Count + Critic_Score:Critic_Count + User_Score:User_Count,
##     data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.729 -0.466 -0.201  0.115 81.282
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 9.174e-01  1.904e-01   4.819 1.47e-06 ***
## Critic_Score               -6.170e-03  2.877e-03  -2.145   0.0320 *
## Critic_Count               -7.114e-02  6.987e-03 -10.181  < 2e-16 ***
## User_Score                 -3.751e-03  1.916e-03  -1.958   0.0503 .
## User_Count                  6.968e-04  1.315e-04   5.297 1.21e-07 ***
## Critic_Score:Critic_Count   1.190e-03  9.130e-05  13.032  < 2e-16 ***
## User_Score:User_Count      -4.402e-06  1.796e-06  -2.451   0.0143 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.794 on 7010 degrees of freedom
## Multiple R-squared:  0.1462, Adjusted R-squared:  0.1454
## F-statistic:   200 on 6 and 7010 DF,  p-value: < 2.2e-16
```

```r
plot(mod3)
```


Residuals vs Fitted
Residuals
Fitted values
lm(Global_Sales ~ Critic_Score + Critic_Count + User_Score + User_Count + C ...
Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Global_Sales ~ Critic_Score + Critic_Count + User_Score + User_Count + C ...





%split data

```
set.seed(12345)
trainid = sample(1:nrow(data), size = trunc(nrow(data) * 0.8))
trainData = data[trainid,]
testData = data[-trainid,]
trainx = model.matrix(Global_Sales ~ . - NA_Sales - EU_Sales - JP_Sales - Other_Sales - Global_Sales - I
```

```
trainy = trainData$Global_Sales
testx = model.matrix(Global_Sales ~ . - NA_Sales - EU_Sales - JP_Sales - Other_Sales - Global_Sales - Na
testy = testData$Global_Sales
```

%mse function

```
mse <- function(model, y, data) {
  yhat <- predict(model, data)
  mean((y - yhat)^2)
}
train_mse = mse(mod2, trainData$Global_Sales, trainData)
test_mse = mse(mod2, testData$Global_Sales, testData)
train_mse
```

```
## [1] 2.38982
```

```
test_mse
```

```
## [1] 6.898599
```

%KNN k=100 best

```
library("FNN")
k_range = c(95,96,97,98,99,100,101,102,103,104)
trainMSE = c()
for(i in 1:length(k_range)){
knnTrain <- knn.reg(train = scale(trainData[ ,c(11, 12, 13, 14)]), test = scale(trainData[ ,c(11, 12, 13
y = trainData$Global_Sales, k = k_range[i])
trainMSE[i] <- mean((trainData$Global_Sales - knnTrain$pred)^2)
}
testMSE = c()
for(i in 1:length(k_range)){
knnTest <- knn.reg(train = scale(trainData[ ,c(11, 12, 13, 14)]), test = scale(testData[ ,c(11, 12, 13,
y = trainData$Global_Sales, k = k_range[i])
testMSE[i] <- mean((testData$Global_Sales - knnTest$pred)^2)
}
trainMSE
```

```
##  [1] 2.251961 2.253331 2.254233 2.255306 2.254488 2.255318 2.257103
##  [8] 2.259159 2.259101 2.261870
```

```
testMSE
```

```
##  [1] 6.808890 6.813020 6.816129 6.817797 6.815294 6.798639 6.801113
##  [8] 6.802728 6.802670 6.805351
```

%lasso

```
library(glmnet)
```

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

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-13
```

```
cv.out = cv.glmnet(trainx, trainy, alpha = 1)
bestlambda = cv.out$lambda.min
model.lasso = glmnet(trainx, trainy, lambda = bestlambda,alpha = 1)
lasso_train_pred = predict(model.lasso, s = bestlambda, newx = trainx)
```

```r
lasso_train_mse = mean((lasso_train_pred - trainy) ^ 2)
lasso_test_pred = predict(model.lasso, s = bestlambda, newx = testx)
lasso_test_mse = mean((lasso_test_pred - testy) ^ 2)
bestlambda
```

```
## [1] 0.0009847606
```

```r
lasso_train_mse
```

```
## [1] 2.18442
```

```r
lasso_test_mse
```

```
## [1] 6.6081
```

%ridge regression

```r
library(glmnet)
grid=10^seq(10,-2,length=100)
ridge.mod=glmnet(trainx,trainy,alpha=0,lambda=grid)
set.seed(1)
cv.out_ridge=cv.glmnet(trainx,trainy,alpha=0)
bestlam=cv.out_ridge$lambda.min
bestlam
```

```
## [1] 0.060424
```

```r
ridge.pred_train=predict(ridge.mod,s=bestlam,newx=trainx)
ridgeTrainMSE=mean((ridge.pred_train-trainy)^2)
ridge.pred_test=predict(ridge.mod,s=bestlam,newx=testx)
ridgeTestMSE=mean((ridge.pred_test-testy)^2)
ridgeTrainMSE
```

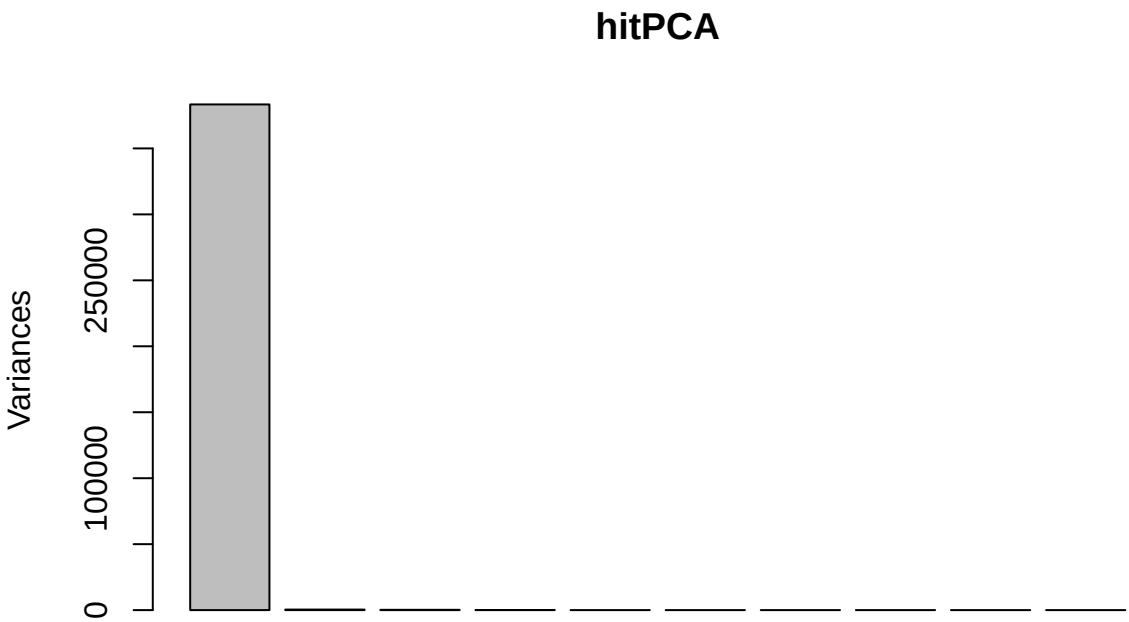
```
## [1] 2.18538
```

```r
ridgeTestMSE
```

```
## [1] 6.613405
```

%PCA

```r
library(ISLR)
hitPCA <- prcomp(x=trainx, center = T, scale = F)
plot(hitPCA)
```




```
#trainx.centered <- apply(trainx, 2, function(x) x - mean(x))
#hitEigen <- eigen(var(trainx.centered))
#cbind("eigen" = hitEigen$vectors[, 1], "prcomp" = hitPCA$rotation[, 1])
```

```
#summary(hitPCA)$x[,1]
```

%PCR

```
library(pls)
```

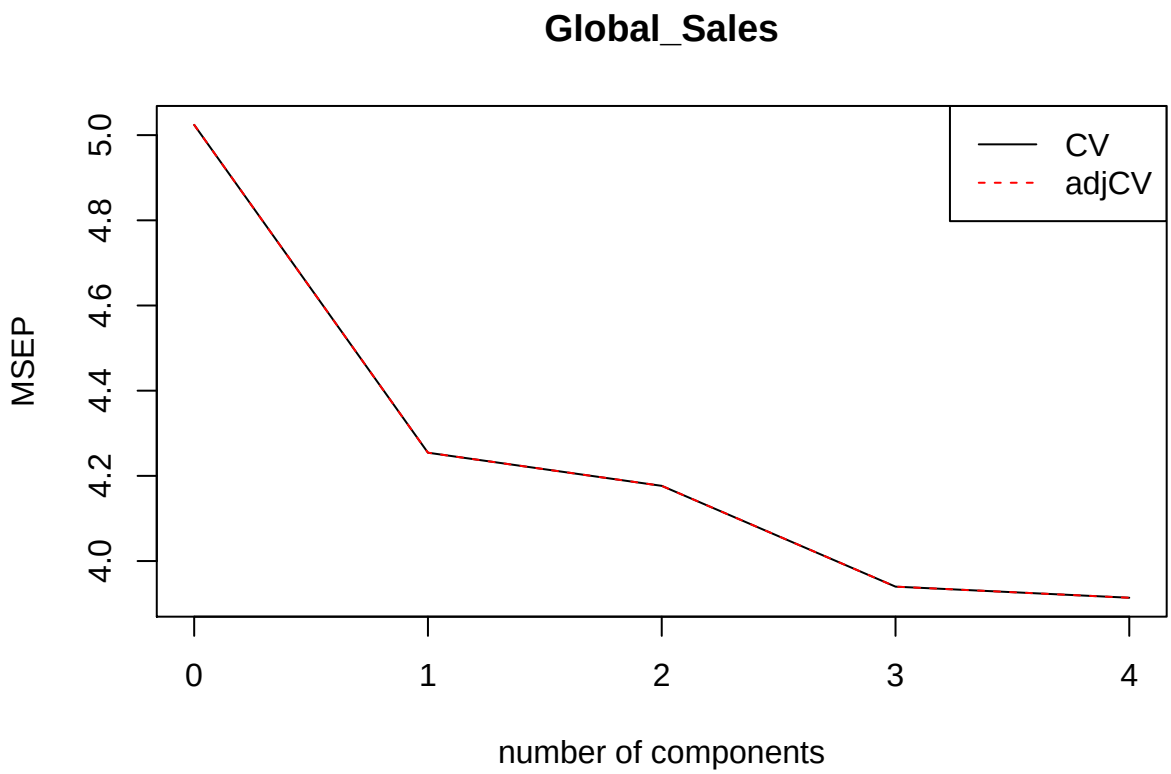
```
##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings
```

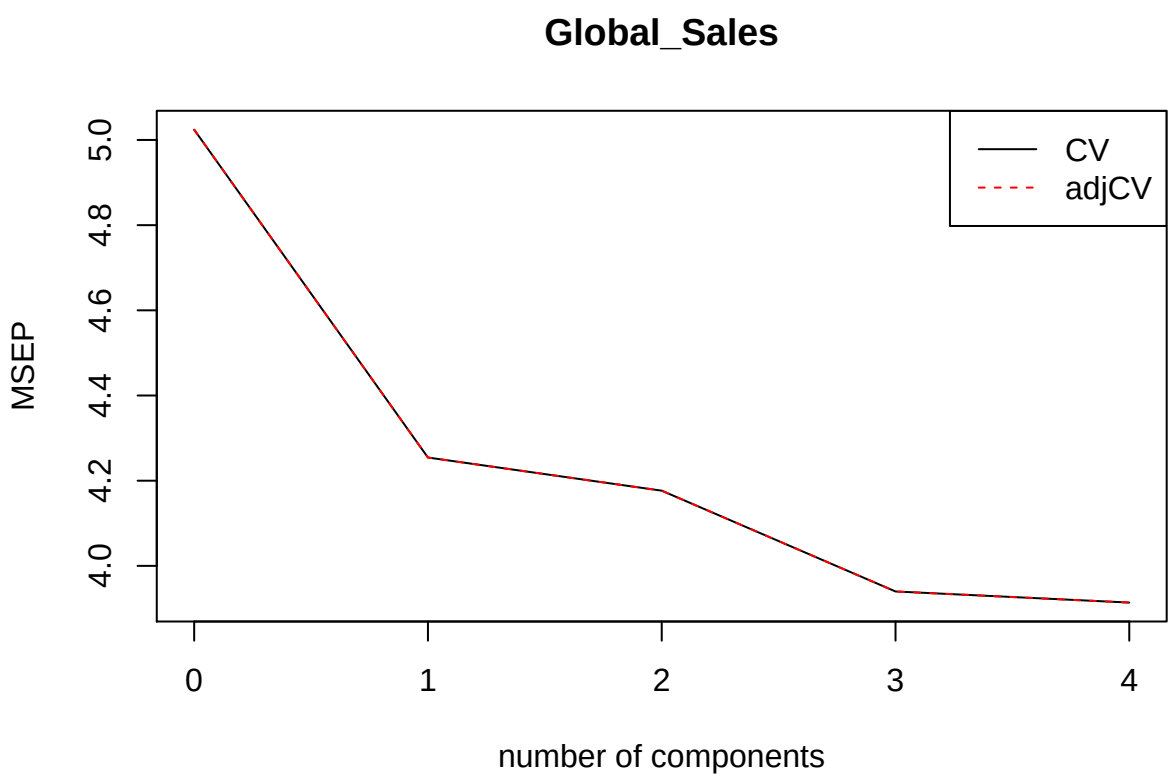
```
set.seed(1)
hitPCR <- pcr(Global_Sales ~ Critic_Score+Critic_Count+User_Score+User_Count, data = trainData, subset =
summary(hitPCR)
```

```
## Data:    X dimension: 33661 4
##  Y dimension: 33661 1
## Fit method: svdpc
## Number of components considered: 4
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps
## CV           2.241    2.063    2.044    1.985    1.978
## adjCV        2.241    2.063    2.044    1.985    1.978
##
## TRAINING: % variance explained
##               1 comps  2 comps  3 comps  4 comps
## X               51.55    78.87    91.12   100.00
## Global_Sales    15.32    16.87    21.59    22.11
```

```r
validationplot(hitPCR, val.type = "MSEP", legendpos = "topright")
```




```r
validationplot(hitPCR, val.type = "MSEP", legendpos = "topright")
```



```r
train_x = model.matrix(Global_Sales ~Critic_Score+Critic_Count+User_Score+User_Count , data = trainData
train_y = trainData$Global_Sales
test_x = model.matrix(Global_Sales ~Critic_Score+Critic_Count+User_Score+User_Count, data = testData)[,
test_y = testData$Global_Sales
hitPCR.pred <- predict(hitPCR, test_x, ncomp = 4)
PCRTestMSE <- mean((hitPCR.pred -test_y)^2)
PCRTestMSE
```

```
## [1] 7.371198
```

%PLS

```r
set.seed(1)
hitPLS <- plsr(Global_Sales ~Critic_Score+Critic_Count+User_Score+User_Count, data = trainData, subset =
summary(hitPLS)
```

```
## Data:    X dimension: 33661 4
##  Y dimension: 33661 1
## Fit method: kernelpls
## Number of components considered: 4
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps
## CV           2.241    2.184    1.985    1.978    1.978
## adjCV        2.241    2.184    1.985    1.978    1.978
##
## TRAINING: % variance explained
##               1 comps  2 comps  3 comps  4 comps
## X              99.293    99.68    99.92   100.00
## Global_Sales    5.063    21.59    22.10    22.11
```

```r
hitPLS.pred <- predict(hitPLS, test_x, ncomp = 4)
PLSTestMSE <- mean((hitPLS.pred -test_y)^2)
PLSTestMSE
```

```
## [1] 7.371198
```

```r
d <- data.frame("TestMSE" = c(PCRTestMSE, PLSTestMSE, ridgeTestMSE, lasso_test_mse,test_mse, testMSE[6]
rownames(d) <- c("PCR", "PLS", "Ridge", "Lasso","Linear Regression","KNN")
knitr::kable(d)
```

	TestMSE
PCR	7.371198
PLS	7.371198
Ridge	6.613405
Lasso	6.608100
Linear Regression	6.898599
KNN	6.798639

```r
anova(mod1,mod2)
```

```
## Analysis of Variance Table
##
## Model 1: Global_Sales ~ Critic_Score + User_Score
## Model 2: Global_Sales ~ Critic_Score + Critic_Count + User_Score + User_Count
```

```
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1   7014 24835
## 2   7012 23100  2    1735.3 263.38 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
anova(mod1,mod3)
```

```
## Analysis of Variance Table
##
## Model 1: Global_Sales ~ Critic_Score + User_Score
## Model 2: Global_Sales ~ Critic_Score + Critic_Count + User_Score + User_Count +
##     Critic_Score:Critic_Count + User_Score:User_Count
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1   7014 24835
## 2   7010 22553  4    2281.9 177.31 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
anova(mod2,mod3)
```

```
## Analysis of Variance Table
##
## Model 1: Global_Sales ~ Critic_Score + Critic_Count + User_Score + User_Count
## Model 2: Global_Sales ~ Critic_Score + Critic_Count + User_Score + User_Count +
##     Critic_Score:Critic_Count + User_Score:User_Count
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1   7012 23100
## 2   7010 22553  2    546.57 84.942 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
mod9 <- lm(trainData$Global_Sales~hitPCA$x[,1])
```