Exercise 2

Tobias Raidl, 11717659

2023-10-02

Preprocessing

Set up train and test set Omit NAs Log transform Remove variables “Accept” and “Enroll”

```
library(dplyr)
```

```
## Warning: Paket 'dplyr' wurde unter R Version 4.1.3 erstellt
```

```
##
## Attache Paket: 'dplyr'
```

```
## Die folgenden Objekte sind maskiert von 'package:stats':
##
##     filter, lag
```

```
## Die folgenden Objekte sind maskiert von 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# df name is College
data(College,package="ISLR")
College = na.omit(College)
df = select(College, -Accept, -Enroll)
df$Private = ifelse(df$Private == "Yes", 1, 0)
apps = log(df$Apps)
df$Apps = apps
#df = data.frame(scale(select(df,-Private)))
#df = cbind("Private"=private,df)
df$Apps = log(df$Apps)
set.seed(11717659)
sample = sample(c(TRUE, FALSE), size=nrow(df), replace=TRUE, prob=c(2/3,1/3))
train = df[sample, ]
y_train = train$Apps
test = df[!sample, ]
y_test = test$Apps
test = select(test, -Apps)
```

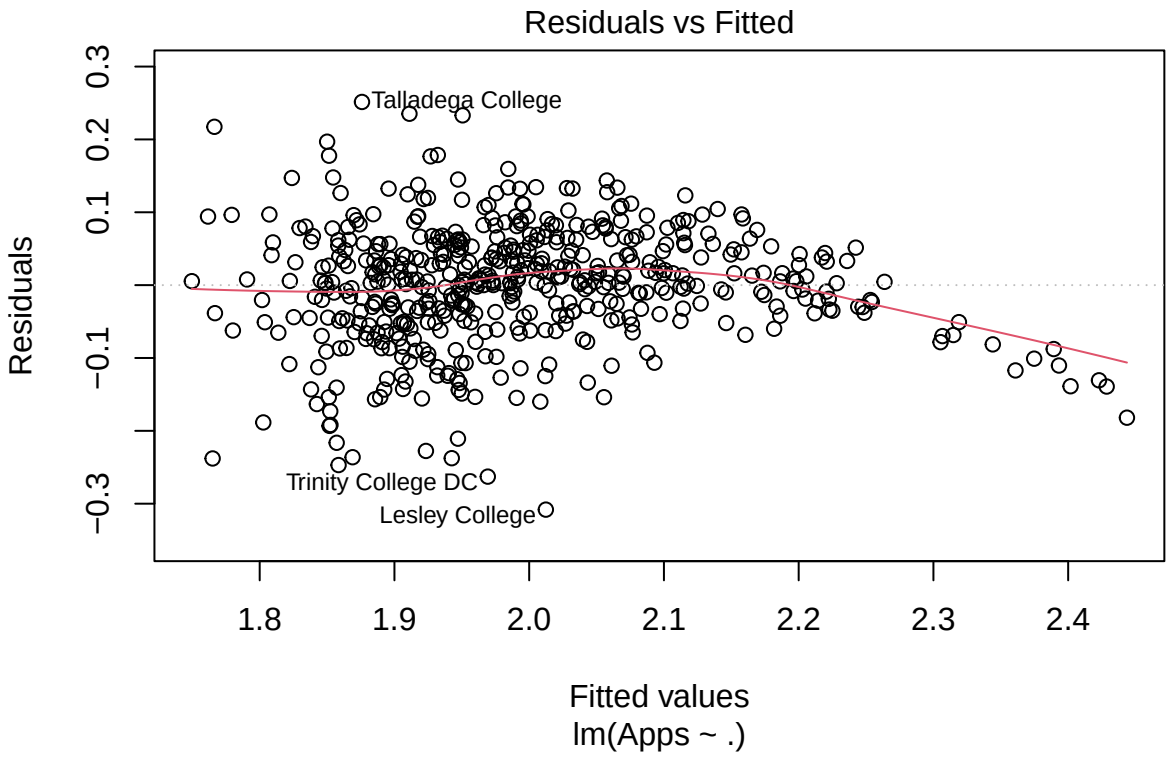
1.a

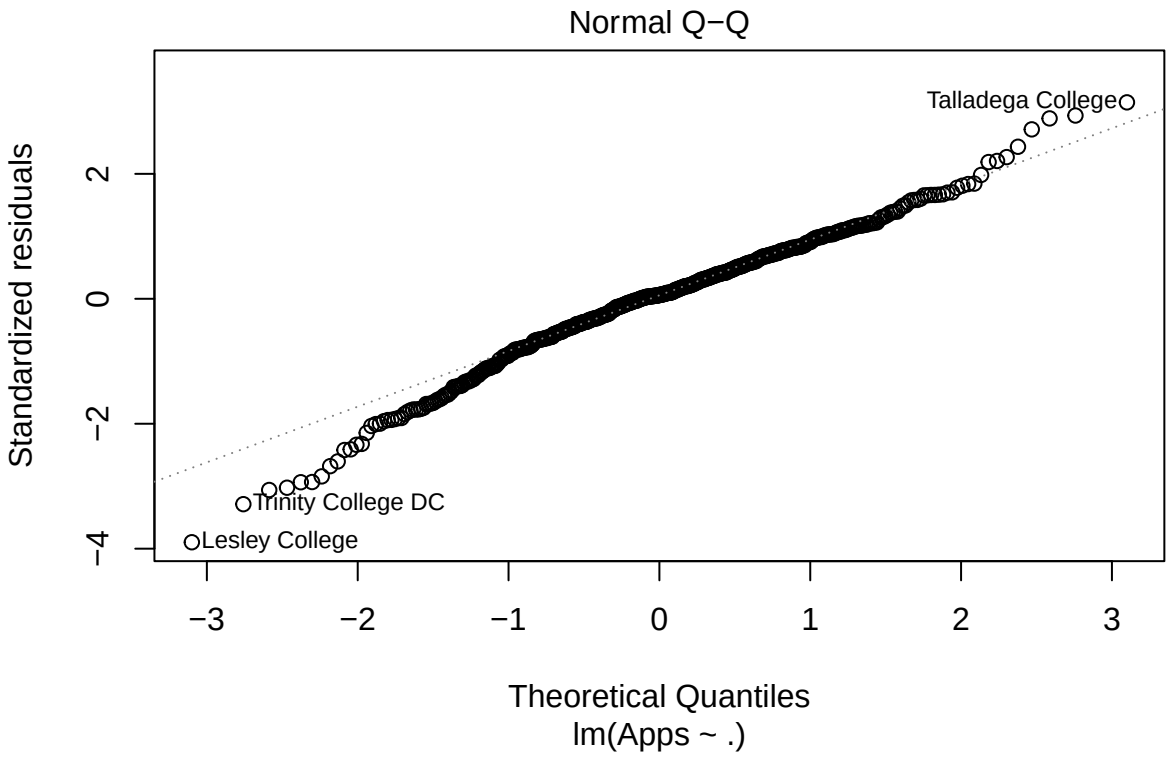
The variables contributing to explaining the variable "Apps" the most are no necessarily the ones with the highest absolute coefficient. Say we use an alpha of 0.05, all the variables with at least one asterisk to their name are perceived as statistically significant. The assumptions are not valid, because for high fitted values the variability is far lower than for small ones. Also the qq plot shows that the distribution is not normal and therefore this requirement is not fulfilled. The leverage plot shows possible outliars (outliars indicated by cooks distance > 1), we got none. Preprocessing the observed variables in addition to the response variable could make these plots more meaningful.

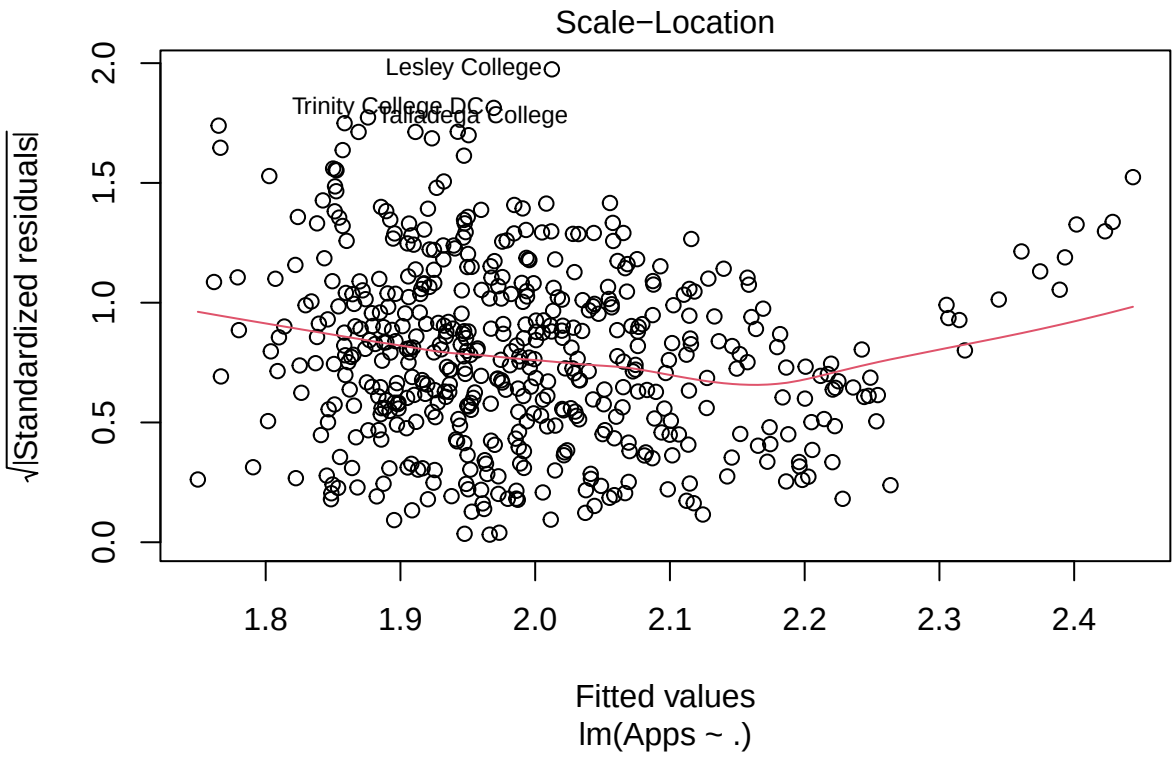
```
my_model = lm(Apps ~., data=train)
summary(my_model)
```

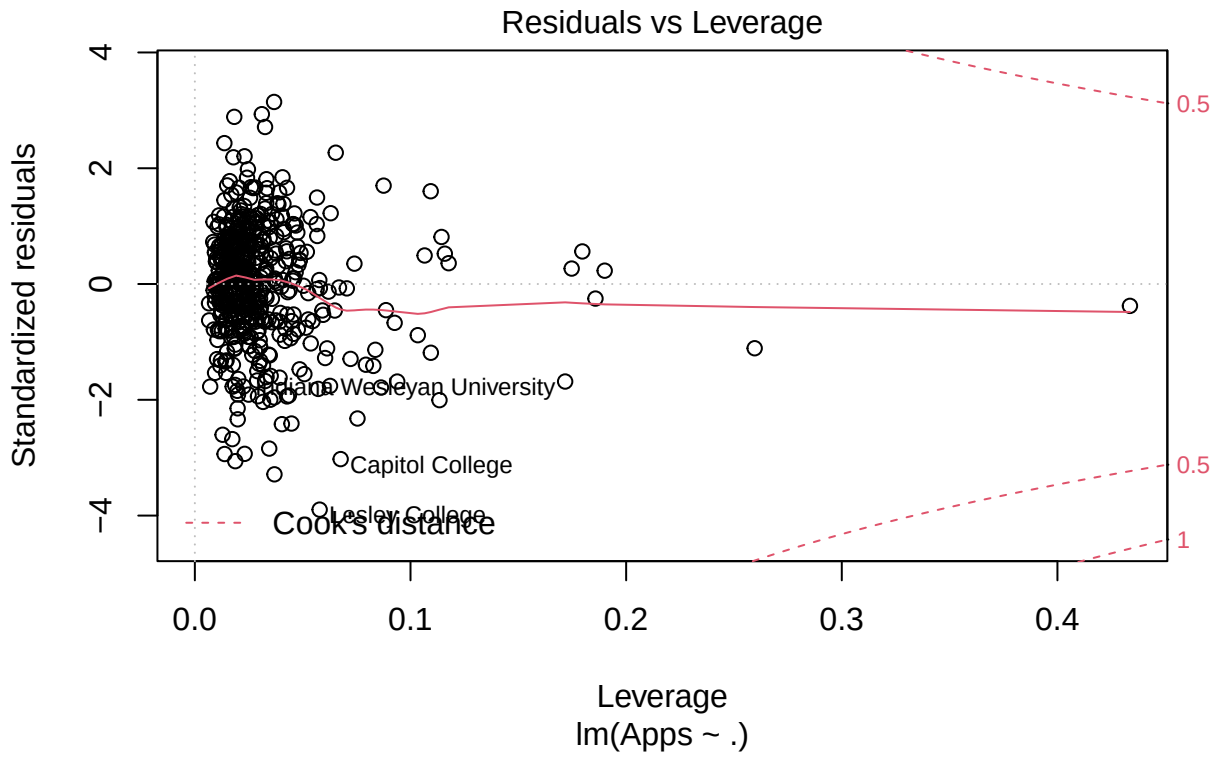
```
##
## Call:
## lm(formula = Apps ~ ., data = train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.308139 -0.044032  0.004912  0.053068  0.251431
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.561e+00  3.799e-02  41.081  < 2e-16 ***
## Private     -9.165e-02  1.367e-02  -6.705 5.45e-11 ***
## Top10perc    4.672e-04  5.427e-04   0.861 0.389720
## Top25perc    3.486e-04  4.261e-04   0.818 0.413624
## F.Undergrad  1.240e-05  1.125e-06  11.029  < 2e-16 ***
## P.Undergrad  3.047e-06  2.813e-06   1.083 0.279294
## Outstate     6.640e-06  1.860e-06   3.570 0.000391 ***
## Room.Board   1.287e-05  4.669e-06   2.757 0.006040 **
## Books        5.105e-05  2.310e-05   2.210 0.027568 *
## Personal     1.575e-06  6.320e-06   0.249 0.803324
## PhD          9.945e-04  4.337e-04   2.293 0.022253 *
## Terminal     2.731e-04  4.700e-04   0.581 0.561510
## S.F.Ratio    5.725e-03  1.237e-03   4.626 4.75e-06 ***
## perc.alumni -1.122e-03  3.888e-04  -2.886 0.004076 **
## Expend       2.416e-06  1.298e-06   1.861 0.063396 .
## Grad.Rate    1.402e-03  2.717e-04   5.161 3.55e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08146 on 500 degrees of freedom
## Multiple R-squared:  0.6954, Adjusted R-squared:  0.6862
## F-statistic: 76.08 on 15 and 500 DF,  p-value: < 2.2e-16
```

```
plot(my_model)
```


Residuals vs Fitted
Residuals
0.3
0.2
0.1
−0.1
−0.3
Talladega College
Trinity College DC
Lesley College
1.8
1.9
2.0
2.1
2.2
2.3
2.4
Fitted values
lm(Apps ~ .)


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Apps ~ .)
Talladega College
Trinity College DC
Lesley College
−4
−2
0
2
−3
−2
−1
0
1
2
3


Scale−Location
Lesley College
Trinity College DC
Talladega College
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
1.8
1.9
2.0
2.1
2.2
2.3
2.4
Fitted values
lm(Apps ~ .)



1.b

I encoded the boolean variable "Private" to Yes=1 and No=0. Because I did only log transform the Apps variable, the coefficients do not necessarily tell how much each variable describes the "Apps" variable.

```
X = model.matrix(Apps ~., data=train)
b = solve(t(X) %*% X) %*% t(X) %*% y_train
b
```

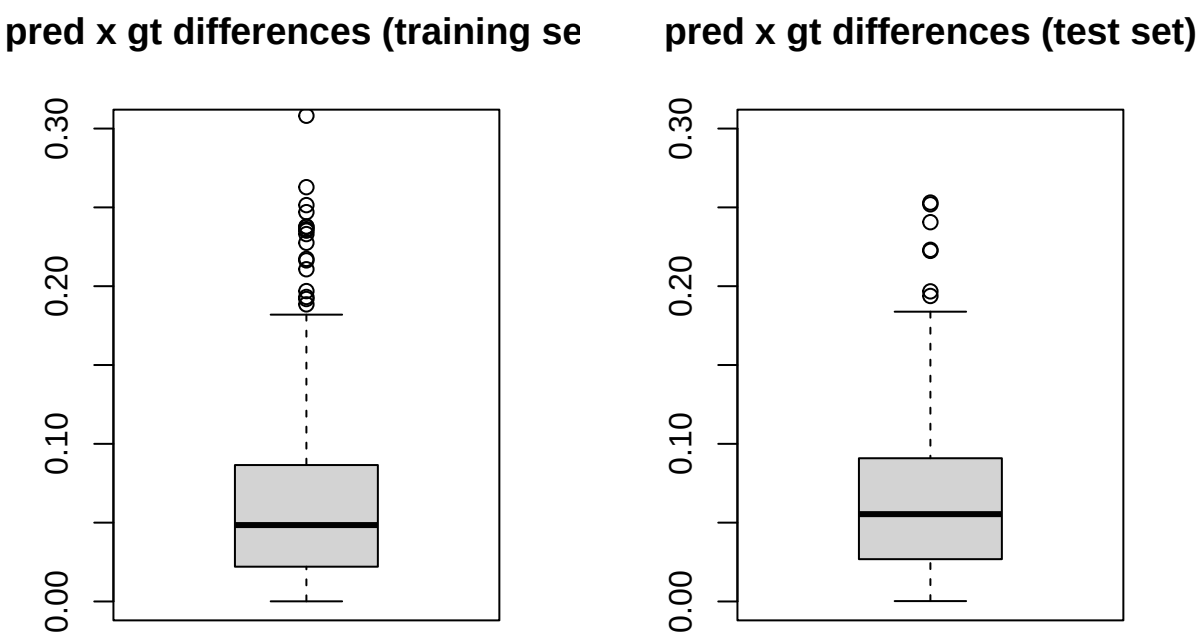
```
##                      [,1]
## (Intercept)  1.560626e+00
## Private     -9.164903e-02
## Top10perc    4.672343e-04
## Top25perc    3.486262e-04
## F.Undergrad  1.240231e-05
## P.Undergrad  3.047094e-06
## Outstate     6.640191e-06
## Room.Board   1.287329e-05
## Books        5.105389e-05
## Personal     1.574884e-06
## PhD          9.945032e-04
## Terminal     2.730726e-04
## S.F.Ratio    5.724574e-03
## perc.alumni -1.121788e-03
```

```
## Expend       2.415880e-06
## Grad.Rate    1.402323e-03
```

1.c

I chose to visualize the predicted values for train and test set by using a boxplot for the distribution of the absolute differences between ground truth and each dataset. As expected, the median of the train set differences goes down. Interestingly the 3rd median and the outliars are going up. As outliars do not belong to the same distribution though, they can be neglected.

```
y_hat_train = unlist(predict(my_model, select(train, -Apps)))
diffs = abs(y_train-y_hat_train)
par(mfrow = c(1, 2))
boxplot(main="pred x gt differences (training set)", diffs, ylim=c(0,0.3))

y_hat_test = unlist(predict(my_model, test))
diffs = abs(y_test-y_hat_test)
boxplot(main="pred x gt differences (test set)", diffs, ylim=c(0,0.3))
```



1.d

The error for the evaluation on the test set is higher due to the model being fitted on the exact train set data. This means it has a bias towards the train set and therefore a lower error.

```r
get_rmse = function(model, n, X, y) {
  y_hat = unlist(predict(model, X))
  rmse = sqrt((1/n) * sum((y-y_hat)^2))
  return(rmse)
}

train_rmse = get_rmse(my_model, nrow(train), select(train, -Apps), y_train)
test_rmse = get_rmse(my_model, nrow(test), test, y_test)

cat(paste("Train RMSE: ", train_rmse, "\nTest RMSE: ", test_rmse))
```

```
## Train RMSE:  0.0801878972960775
## Test RMSE:  0.0859935169851962
```

2.a

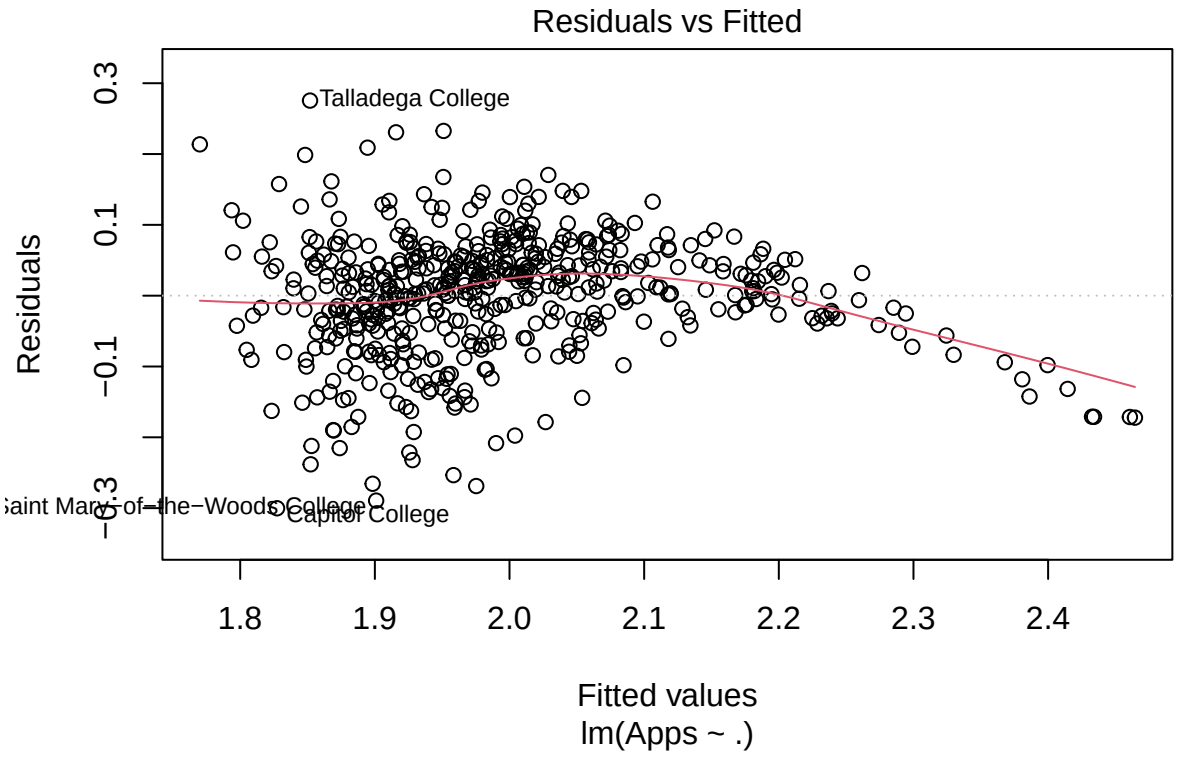
Say alpha=0.05 Not all variables that were previously significant are the same now. Expend is is not significant. It is not necessarily expected, because the test statistic is a new one too, meaning different p-values.

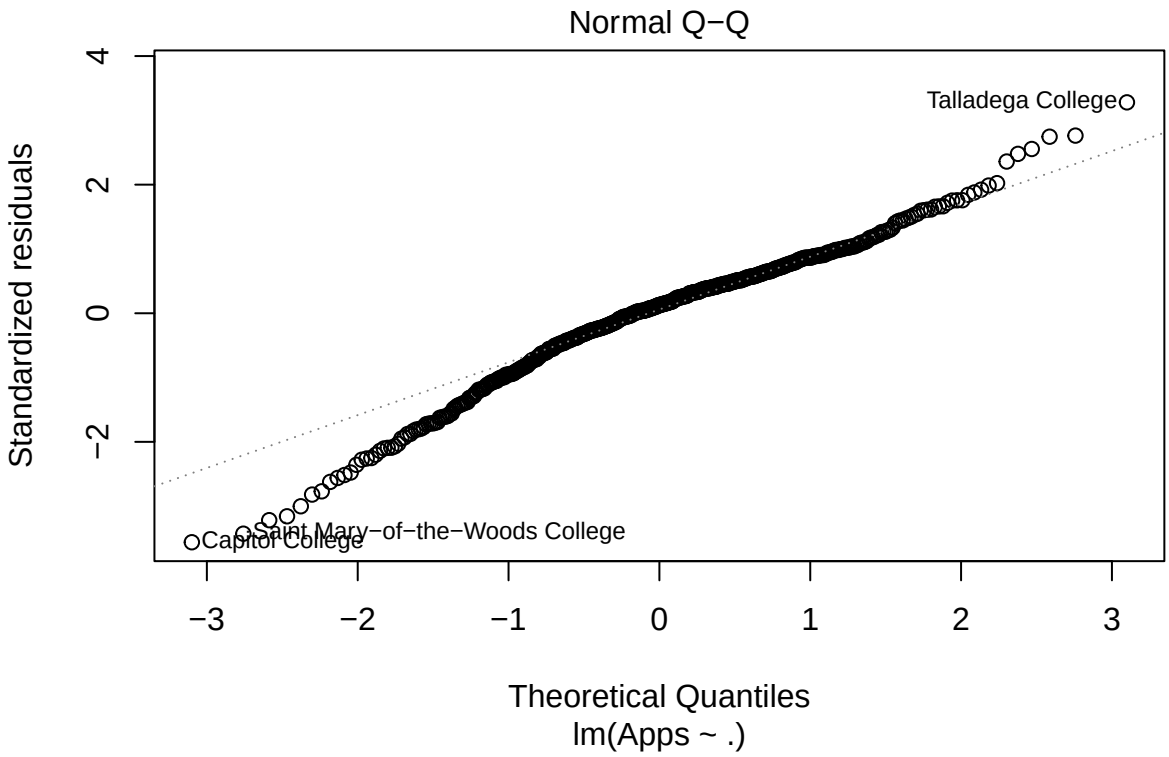
```r
small_train = select(train, c(Apps, Private,Top10perc, F.Undergrad, Outstate, Room.Board,
↪  perc.alumni, Expend, Grad.Rate))
small_test = select(test, c(Private,Top10perc, F.Undergrad, Outstate, Room.Board,
↪  perc.alumni, Expend, Grad.Rate))
small_model = lm(Apps ~., data=small_train)
summary(small_model)
```

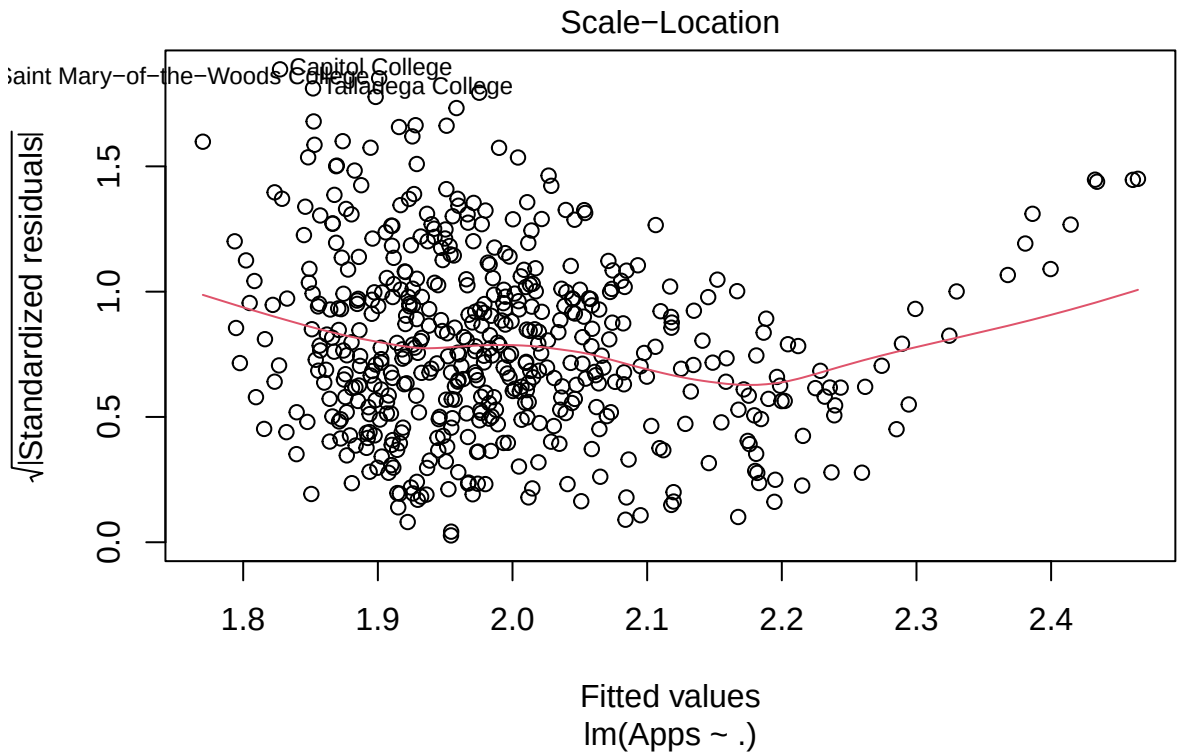
```
##
## Call:
## lm(formula = Apps ~ ., data = small_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.30018 -0.04155  0.01068  0.05096  0.27547
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.772e+00  1.922e-02  92.180  < 2e-16 ***
## Private     -1.237e-01  1.311e-02  -9.436  < 2e-16 ***
## Top10perc    1.220e-03  3.244e-04   3.761 0.000189 ***
## F.Undergrad  1.407e-05  1.038e-06  13.559  < 2e-16 ***
## Outstate     7.387e-06  1.878e-06   3.933 9.55e-05 ***
## Room.Board   1.785e-05  4.739e-06   3.766 0.000185 ***
## perc.alumni -1.056e-03  3.999e-04  -2.640 0.008552 **
## Expend       3.432e-07  1.227e-06   0.280 0.779830
## Grad.Rate    1.412e-03  2.771e-04   5.096 4.90e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08473 on 507 degrees of freedom
```

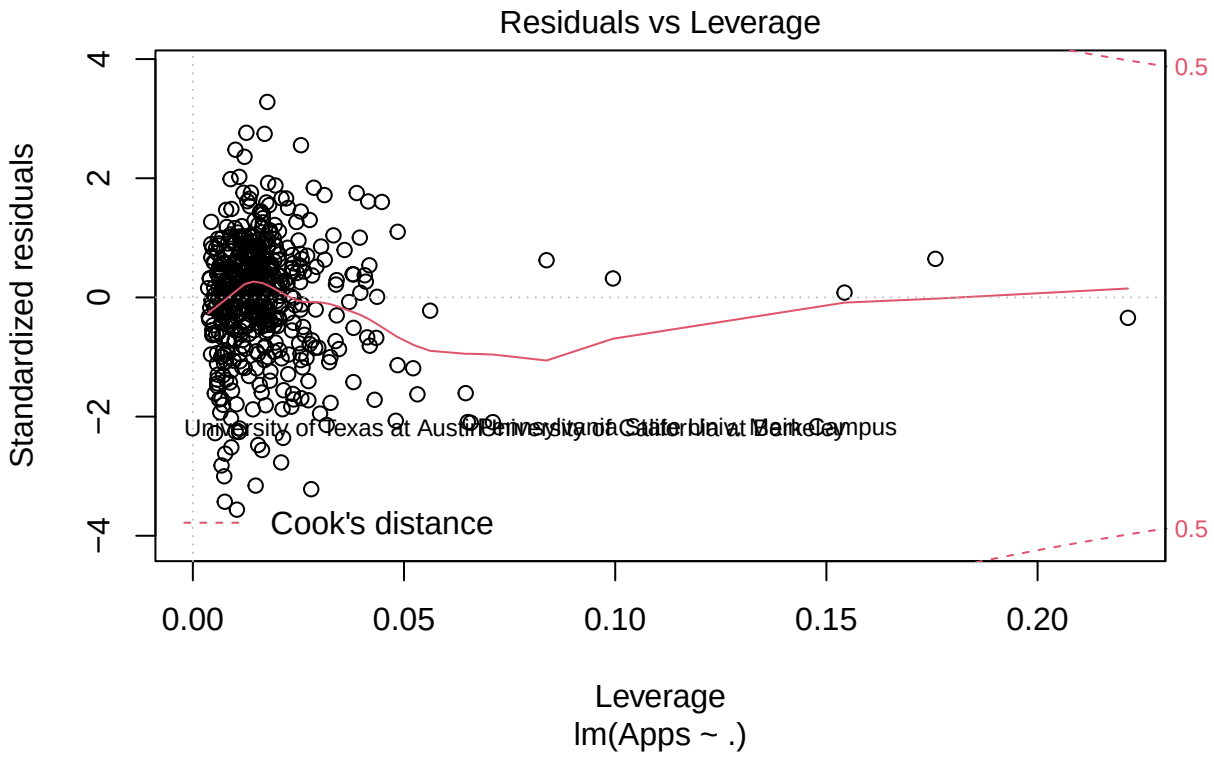
```
## Multiple R-squared:  0.6658, Adjusted R-squared:  0.6605
## F-statistic: 126.2 on 8 and 507 DF,  p-value: < 2.2e-16
```

```
plot(small_model)
```



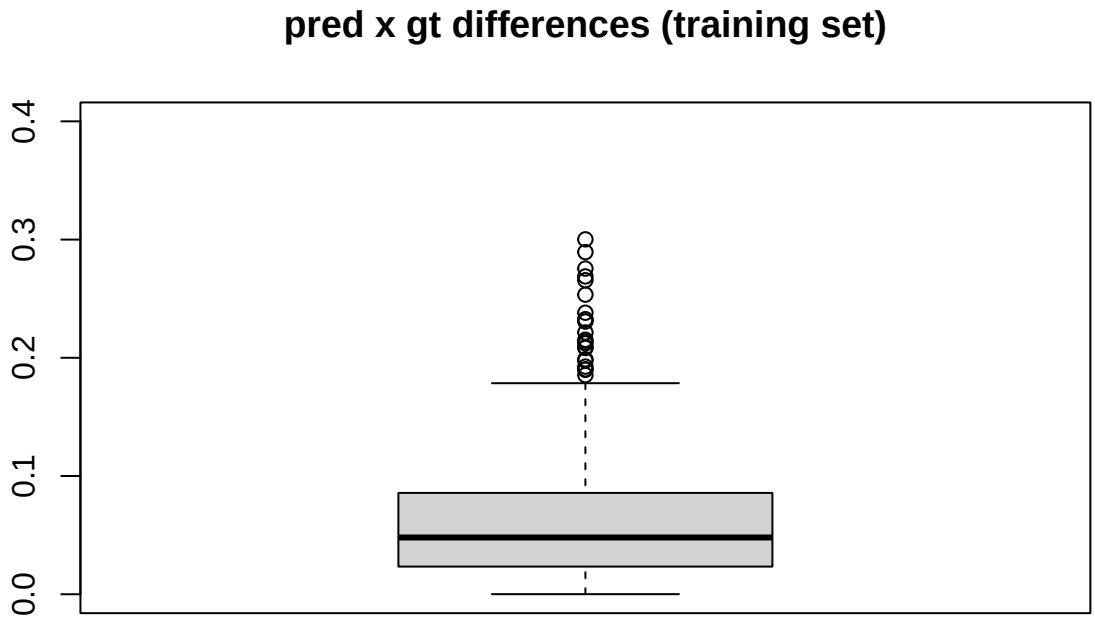

Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Apps ~ .)
Talladega College
Saint Mary−of−the−Woods College
Capitol College
4
2
0
−2
−3
−2
−1
0
1
2
3


Scale−Location
Saint Mary−of−the−Woods College
Capitol College
Talladega College
√|Standardized residuals|
0.0
0.5
1.0
1.5
1.8
1.9
2.0
2.1
2.2
2.3
2.4
Fitted values
lm(Apps ~ .)

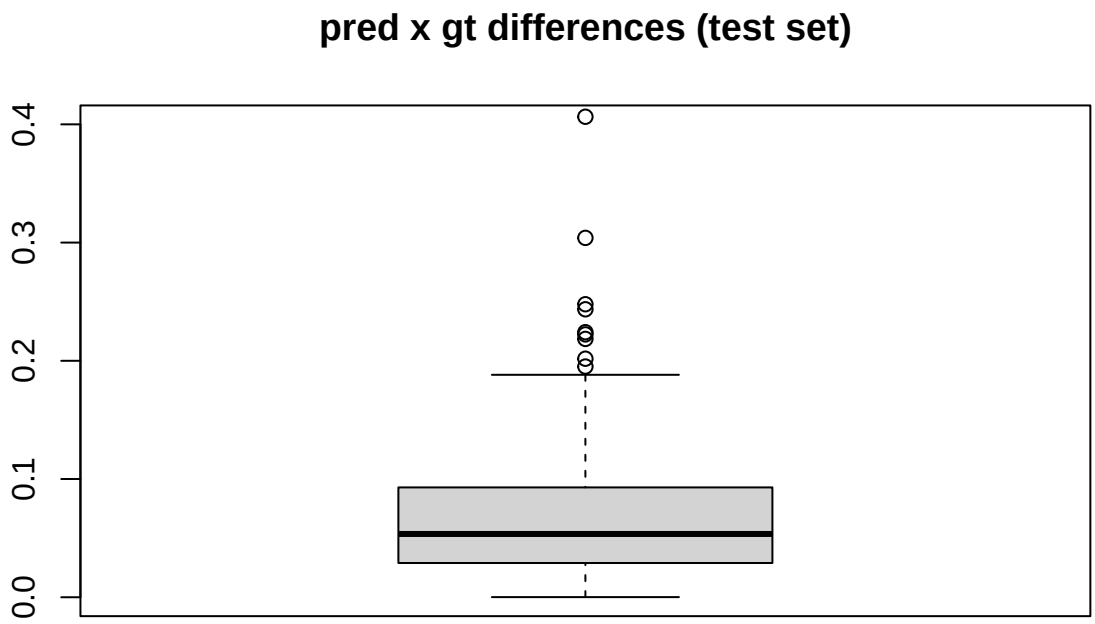


2.b

```
y_hat_small_train = unlist(predict(small_model, select(small_train, -Apps)))
diffs = abs(y_train-y_hat_small_train)
#par(mfrow = c(1, 2))
boxplot(main="pred x gt differences (training set)", diffs, ylim=c(0,0.4))
```




```
y_hat_small_test = unlist(predict(small_model, small_test))
diffs = abs(y_test-y_hat_small_test)
boxplot(main="pred x gt differences (test set)", diffs, ylim=c(0,0.4))
```



2.c

Again we evaluate for both the training and the test set, expecting the model to perform better on the train set. The errors are pretty similar to the ones of the model with all variables. The chosen variables seem to cover a big amount of the explanatory power for Apps.

```
get_rmse = function(model, n, X, y) {
  y_hat = unlist(predict(model, X))
  rmse = sqrt((1/n) * sum((y-y_hat)^2))
  return(rmse)
}

train_rmse = get_rmse(small_model, nrow(small_train), select(small_train, -Apps),
↪  y_train)
test_rmse = get_rmse(small_model, nrow(small_test), small_test, y_test)

cat(paste("Train RMSE: ", train_rmse, "\nTest RMSE: ", test_rmse))
```

```
## Train RMSE:  0.0839916737403742
## Test RMSE:  0.0898723686292986
```

2.d

ANOVA for checking if the two samples. The extremely low p-value means that the difference of the models is. Google says an F-value of over 2.5 would suggest to reject the null hypothesis. Ours i ~6.9.

I think the abbreviations mean the following: RSS: sum of squared errors (between each model prediction and observed value), Sum of Sq: sum of squared differences between models

```
anova(my_model, small_model)
```

```
## Analysis of Variance Table
##
## Model 1: Apps ~ Private + Top10perc + Top25perc + F.Undergrad + P.Undergrad +
##     Outstate + Room.Board + Books + Personal + PhD + Terminal +
##     S.F.Ratio + perc.alumni + Expend + Grad.Rate
## Model 2: Apps ~ Private + Top10perc + F.Undergrad + Outstate + Room.Board +
##     perc.alumni + Expend + Grad.Rate
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    500 3.3179
## 2    507 3.6402 -7  -0.32224 6.9373 6.756e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.

Both the RMSE aswell as the y vs. y hat plots are similar even though they backward and forward step algorithm came not to the same conclusion of the "optimal" features.

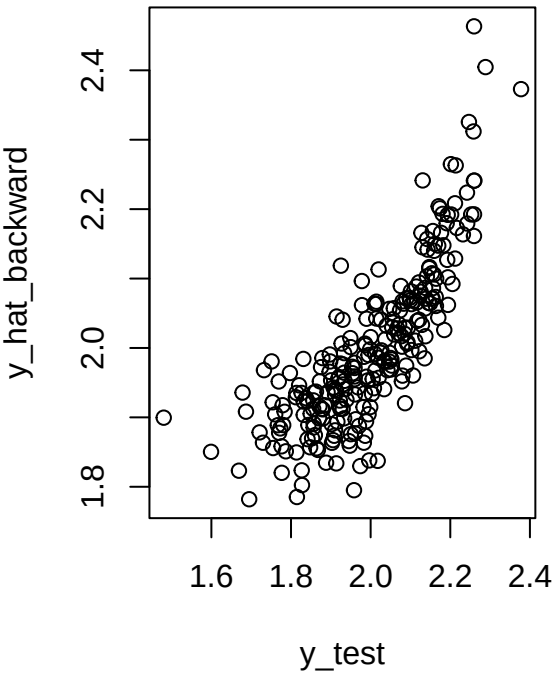
```
model_empty = lm(Apps ~ 1, data=train)
model_backward = step(my_model, direction="backward", trace=0)
model_forward = step(model_empty, scope=formula(my_model), direction="forward", trace=0)

y_hat_backward = predict(model_backward, test)
y_hat_forward = predict(model_forward, test)

cat(paste("backward model rmse: ", get_rmse(model_backward, nrow(test), test, y_test)))
```

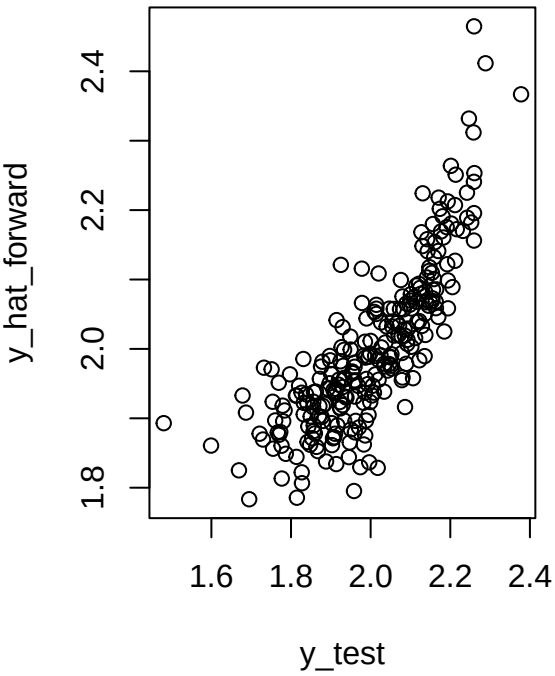
```
## backward model rmse:  0.0857424455031643
```

```
cat(paste("\nforward model rmse: ", get_rmse(model_forward, nrow(test), test, y_test)))
```

```
##
## forward model rmse:  0.086065876780987
```

```
par(mfrow=c(1,2))
plot(x=y_test, y=y_hat_backward)
plot(x=y_test, y=y_hat_forward)
```


y_hat_backward
1.8
2.0
2.2
2.4
1.6
1.8
2.0
2.2
2.4
y_test


y_hat_forward
1.8
2.0
2.2
2.4
1.6
1.8
2.0
2.2
2.4
y_test