Exercise 2

Tobias Raidl, 11717659

2023-10-02

Set up train and test set Omit NAs Log transform and standardize features "Accept" and "Enroll"

```r
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
```

```r
# 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)
private = df$Private
df = data.frame(scale(select(df,-Private)))
df = cbind("Private"=private,df)
set.seed(11717659)
sample = sample(c(TRUE, FALSE), size=nrow(df), replace=TRUE, prob=c(2/3,1/3))
train = df[sample, ]
y_train = train$Apps
test = df[!sample, ]
y_test = test$Apps
test = select(test, -Apps)
```

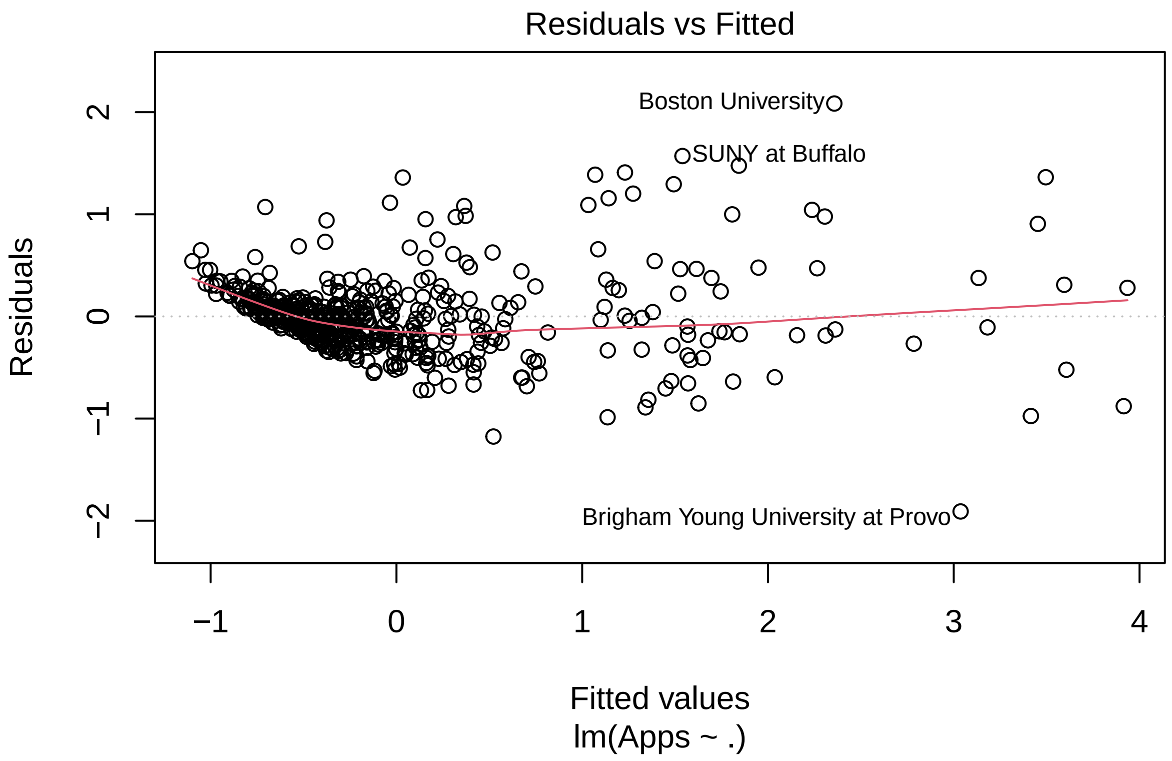
1.a

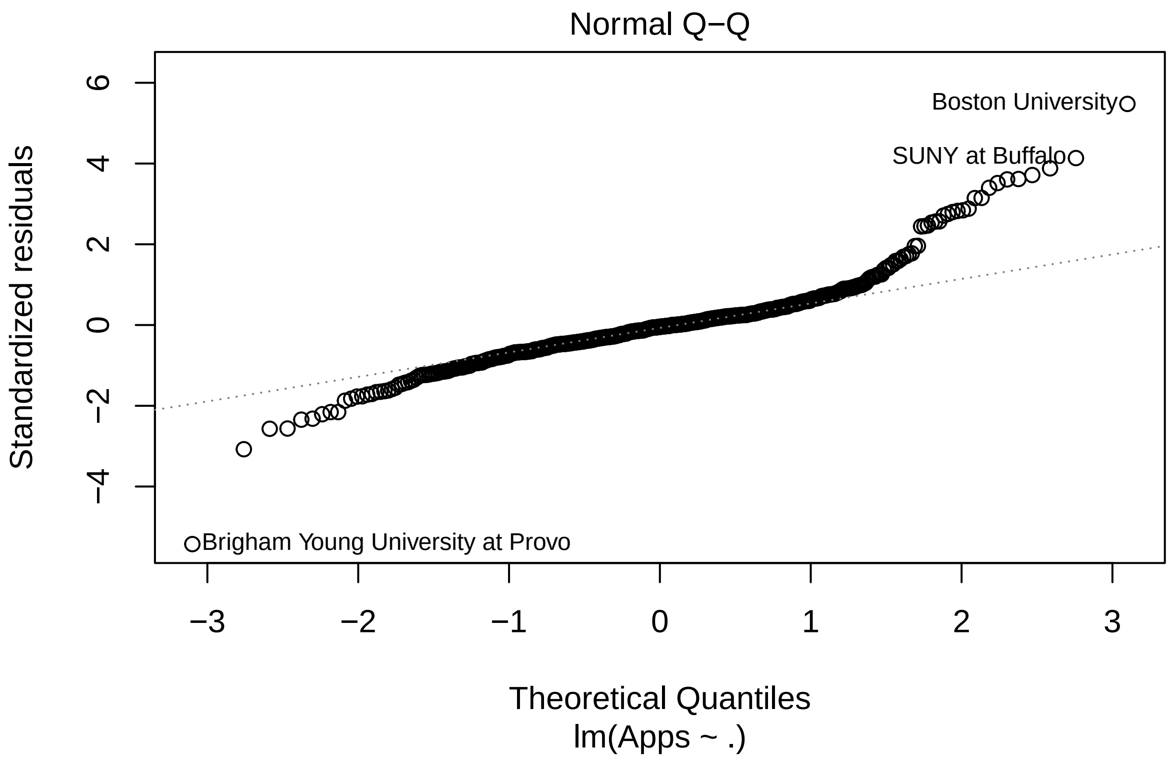
The variables contributing to explaining the variable "Apps" the most are the ones with the highest absolute coefficient. In our case the top 3 are "F. Undergrad", "Expend" and "Private". Say we use an alpha of 0.05, all the variables with at least one asterisk to their name are perceived as statistically significant. Yes the assumptions of the diagnostics plots with `plot(res)` are valid.

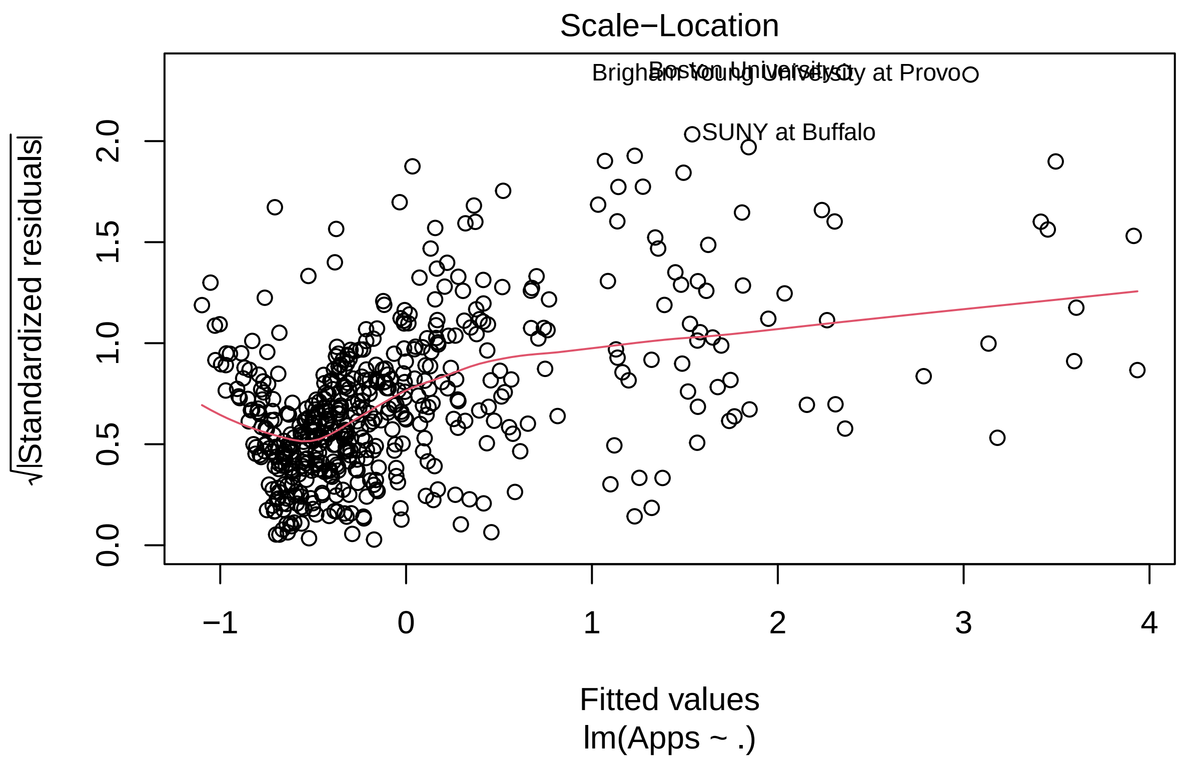
```
my_model = lm(Apps ~., data=train)
summary(my_model)
```

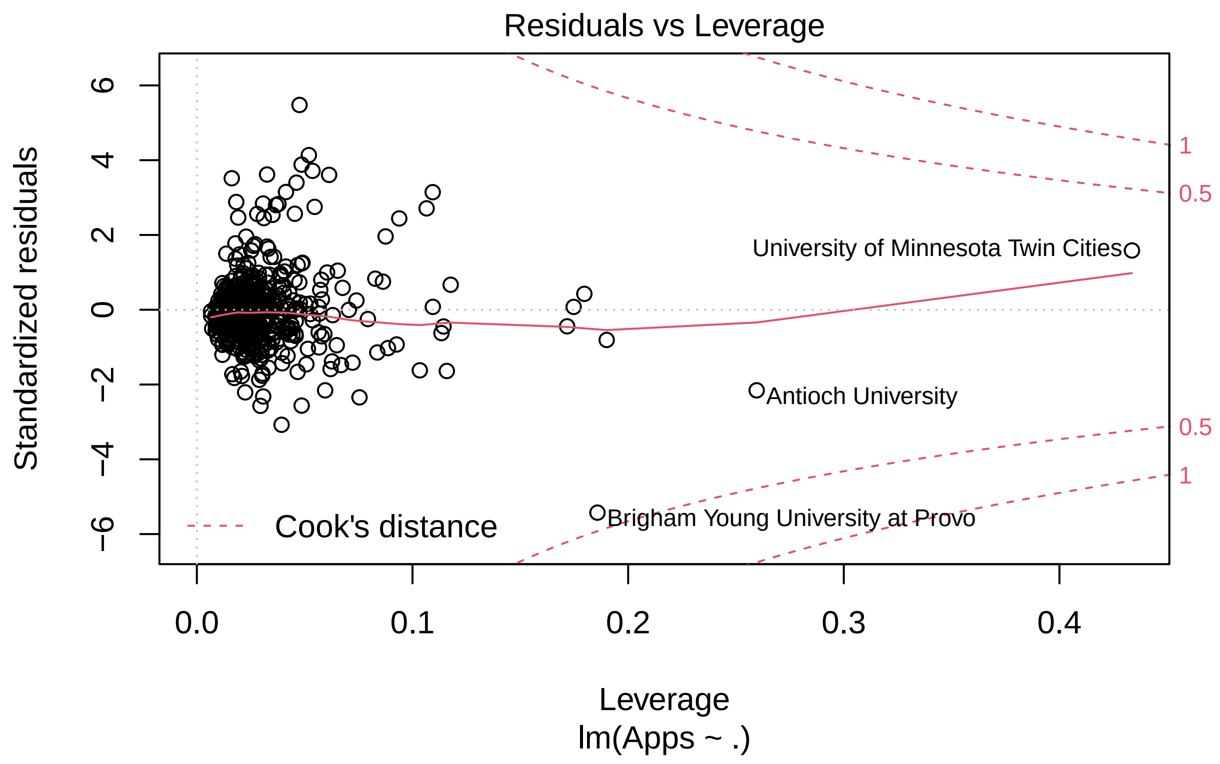
```
##
## Call:
## lm(formula = Apps ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.90952 -0.18492 -0.01665  0.13087  2.08449
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.153274   0.051338   2.986 0.002969 **
## Private     -0.226757   0.065437  -3.465 0.000575 ***
## Top10perc    0.178830   0.045836   3.902 0.000109 ***
## Top25perc   -0.070688   0.040398  -1.750 0.080770 .
## F.Undergrad  0.733215   0.026113  28.078  < 2e-16 ***
## P.Undergrad -0.017482   0.020505  -0.853 0.394311
## Outstate     0.094500   0.035822   2.638 0.008598 **
## Room.Board   0.098323   0.024512   4.011 6.97e-05 ***
## Books        0.002179   0.018261   0.119 0.905063
## Personal    -0.012884   0.020487  -0.629 0.529705
## PhD         -0.001482   0.033901  -0.044 0.965141
## Terminal    -0.050837   0.033128  -1.535 0.125518
## S.F.Ratio    0.005799   0.023449   0.247 0.804769
## perc.alumni -0.093202   0.023063  -4.041 6.16e-05 ***
## Expend       0.125576   0.032460   3.869 0.000124 ***
## Grad.Rate    0.092360   0.022345   4.133 4.19e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.39 on 500 degrees of freedom
## Multiple R-squared:  0.832,  Adjusted R-squared:  0.8269
## F-statistic: 165.1 on 15 and 500 DF,  p-value: < 2.2e-16
```

```
plot(my_model)
```


Residuals vs Fitted
Boston University
SUNY at Buffalo
Brigham Young University at Provo
Residuals
Fitted values
lm(Apps ~ .)
2
1
0
−1
−2
−1
0
1
2
3
4


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Apps ~ .)
Boston University
SUNY at Buffalo
Brigham Young University at Provo
6
4
2
0
−2
−4
−3
−2
−1
0
1
2
3


Scale−Location
Boston University
Brigham Young University at Provo
SUNY at Buffalo
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
−1
0
1
2
3
4
Fitted values
lm(Apps ~ .)



1.b

I encoded the boolean variable "Private" to Yes=1 and No=0. Because I standardized all the variables in advance (besides private), the coefficients tell how much each variable describes the "Applications" variable. A high absolute coefficient in comparison to the other coefficients means that its corresponding variable has high explanatory power. "F. Undergrad" for example has a high coefficient, which implies that colleges with a high number in "F. Undergrad" would be estimated by this linear regression model to have a higher "Applications" number on average.

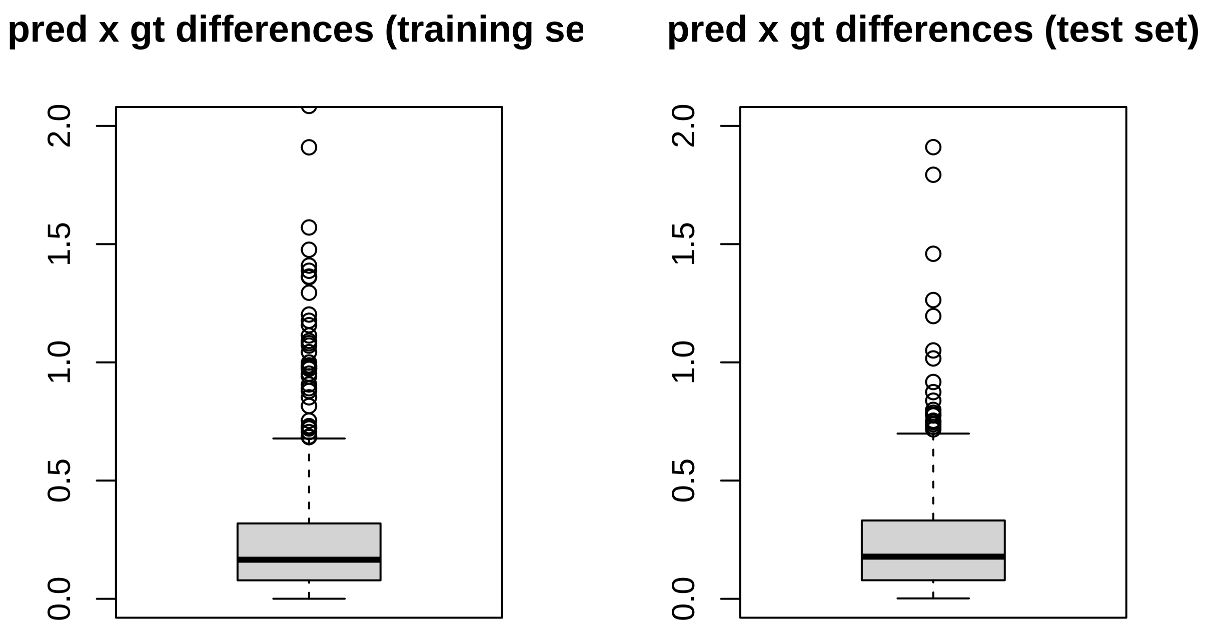
```r
X = model.matrix(Apps ~., data=train)
b = solve(t(X) %*% X) %*% t(X) %*% y_train
b
```

```
##                      [,1]
## (Intercept)  0.153273896
## Private     -0.226757395
## Top10perc    0.178829853
## Top25perc   -0.070688143
## F.Undergrad  0.733214676
## P.Undergrad -0.017481921
## Outstate     0.094500340
## Room.Board   0.098323358
## Books        0.002179073
## Personal    -0.012883961
## PhD          -0.001482334
## Terminal     -0.050836979
## S.F.Ratio     0.005799118
## perc.alumni -0.093201901
## Expend        0.125576011
## Grad.Rate     0.092359969
```

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.

```r
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(0,2))

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(0,2))
```



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:  0.383892782619763
## Test RMSE:  0.667468082778117
```

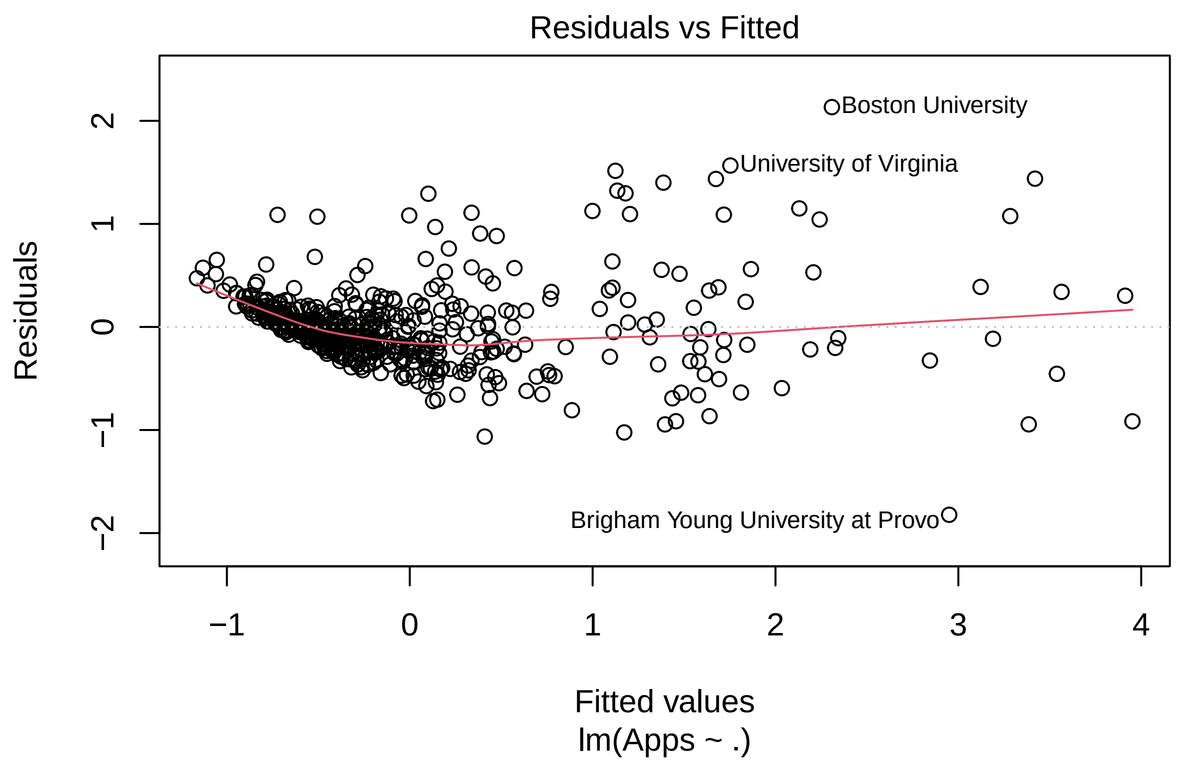
2.a

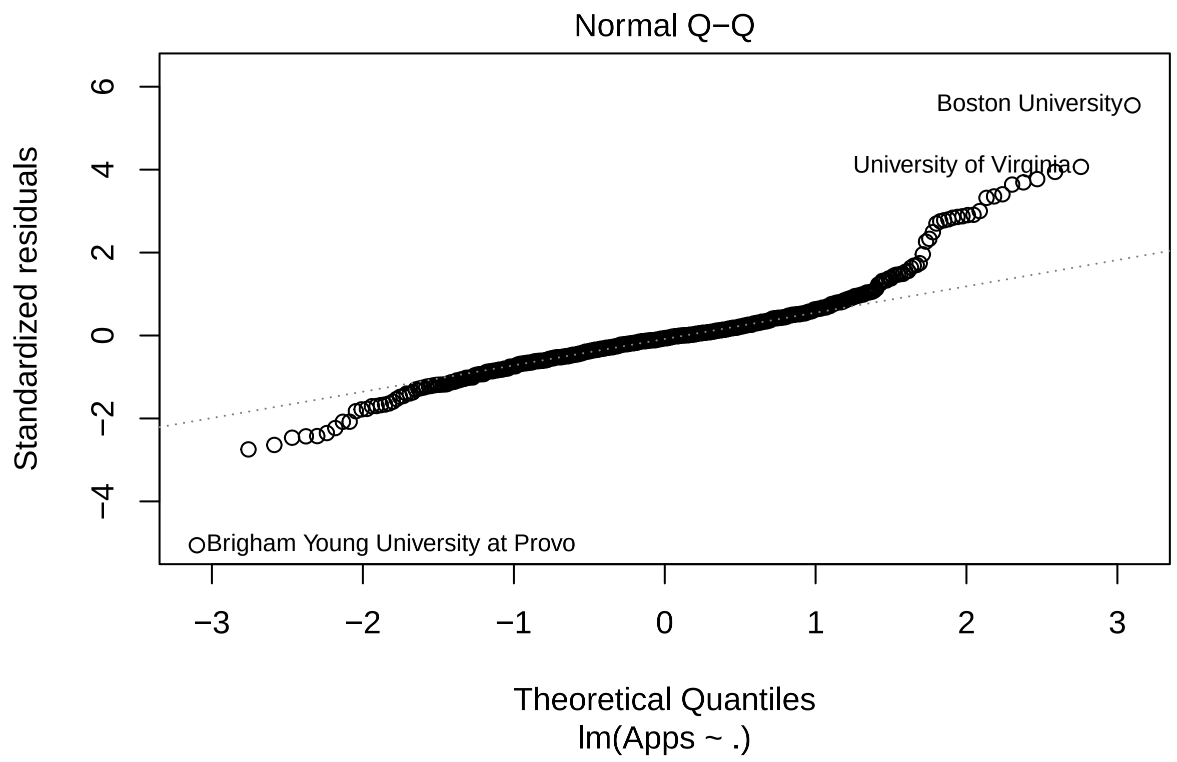
Say alpha=0.05 Now all variables are significant. It is not necessarily expected, because the test statistik is a new one too, meaning different p-values.

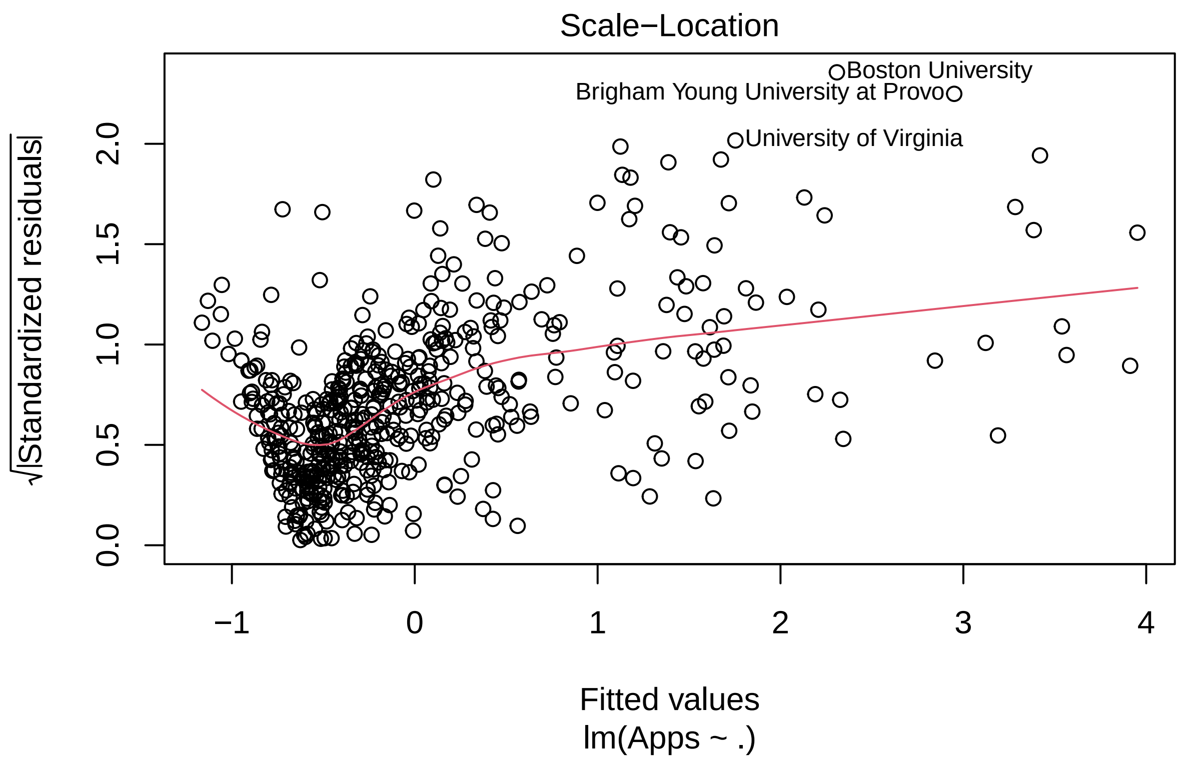
```r
small_train = select(train, c(Apps, Private,Top10perc, F.Undergrad, Outstate, Room.Board,
↪  perc.alumni, Expend, Grad.Rate))
small_test = select(test, c(Private,Top10perc, F.Undergrad, Outstate, Room.Board,
↪  perc.alumni, Expend, Grad.Rate))
small_model = lm(Apps ~., data=small_train)
summary(small_model)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

```
##
## Call:
## lm(formula = Apps ~ ., data = small_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.82251 -0.19959 -0.02542  0.13410  2.13321
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.11859    0.04824   2.458 0.014301 *
## Private     -0.17728    0.06061  -2.925 0.003602 **
## Top10perc    0.10445    0.02647   3.947 9.05e-05 ***
## F.Undergrad  0.70951    0.02328  30.477  < 2e-16 ***
## Outstate     0.07713    0.03494   2.208 0.027716 *
## Room.Board   0.08748    0.02404   3.640 0.000301 ***
## perc.alumni -0.10084    0.02291  -4.401 1.32e-05 ***
## Expend       0.12839    0.02963   4.333 1.77e-05 ***
## Grad.Rate    0.09277    0.02201   4.215 2.95e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3918 on 507 degrees of freedom
## Multiple R-squared:  0.828,  Adjusted R-squared:  0.8253
## F-statistic: 305.1 on 8 and 507 DF,  p-value: < 2.2e-16
```

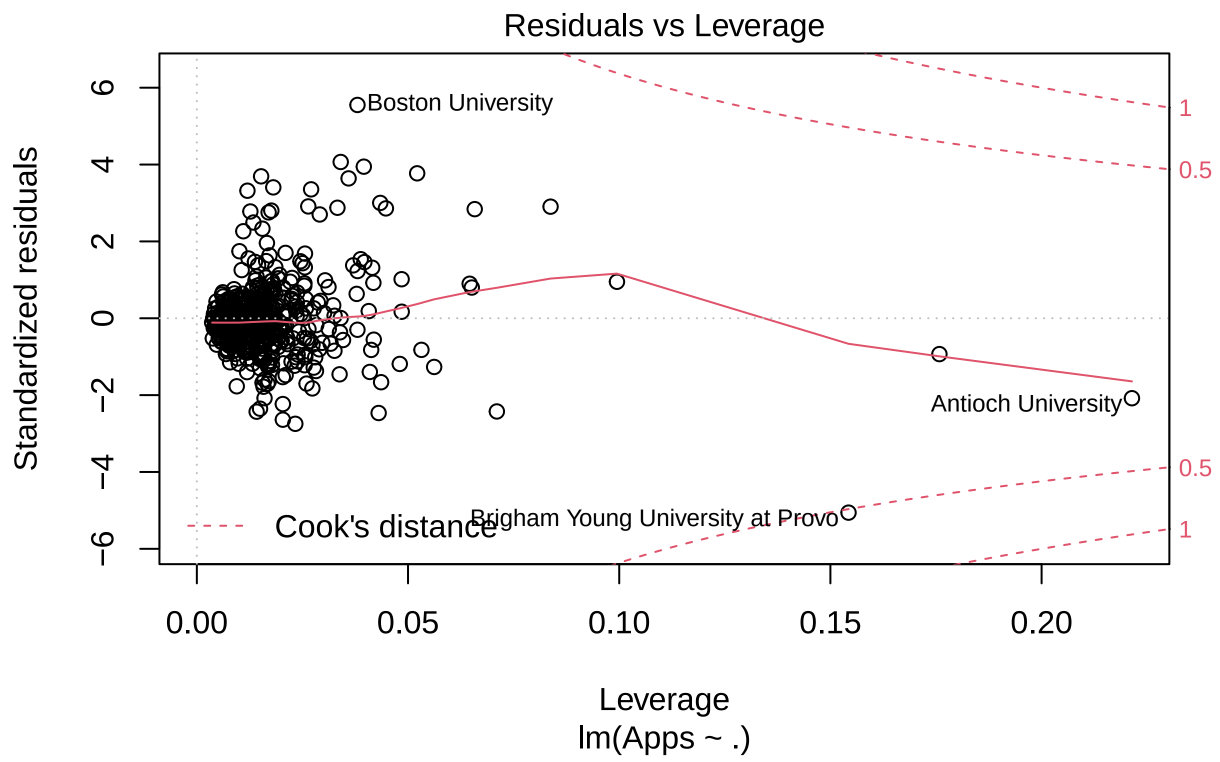
```
plot(small_model)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Apps ~ .)
Boston University
University of Virginia
Brigham Young University at Provo
6
4
2
0
−2
−4
−3
−2
−1
0
1
2
3

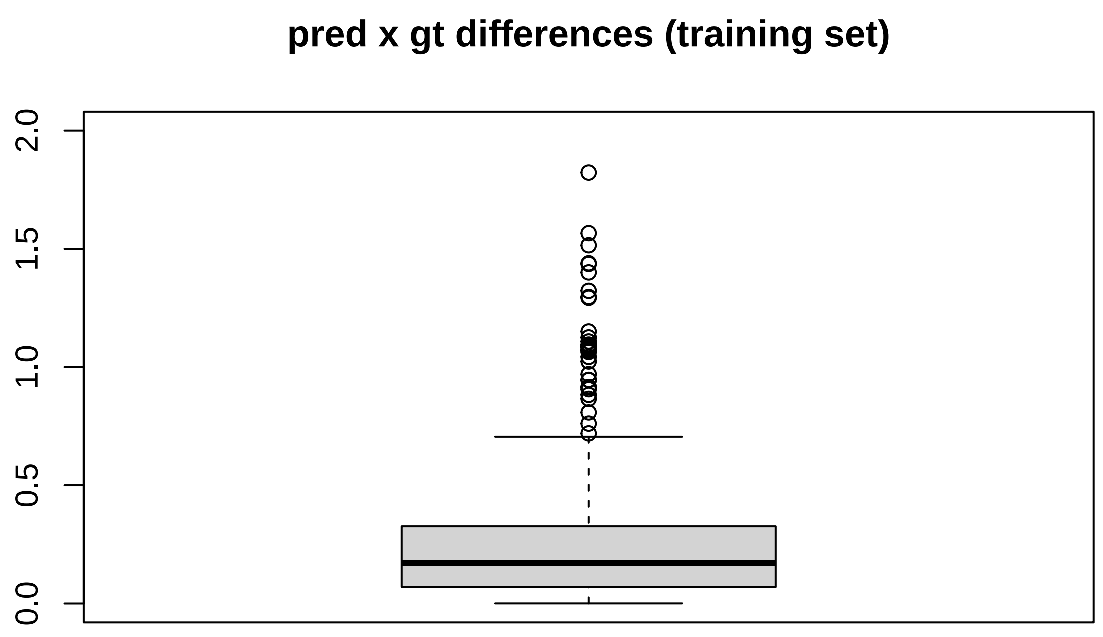


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
Boston University
Brigham Young University at Provo
University of Virginia
−1
0
1
2
3
4
Fitted values
lm(Apps ~ .)

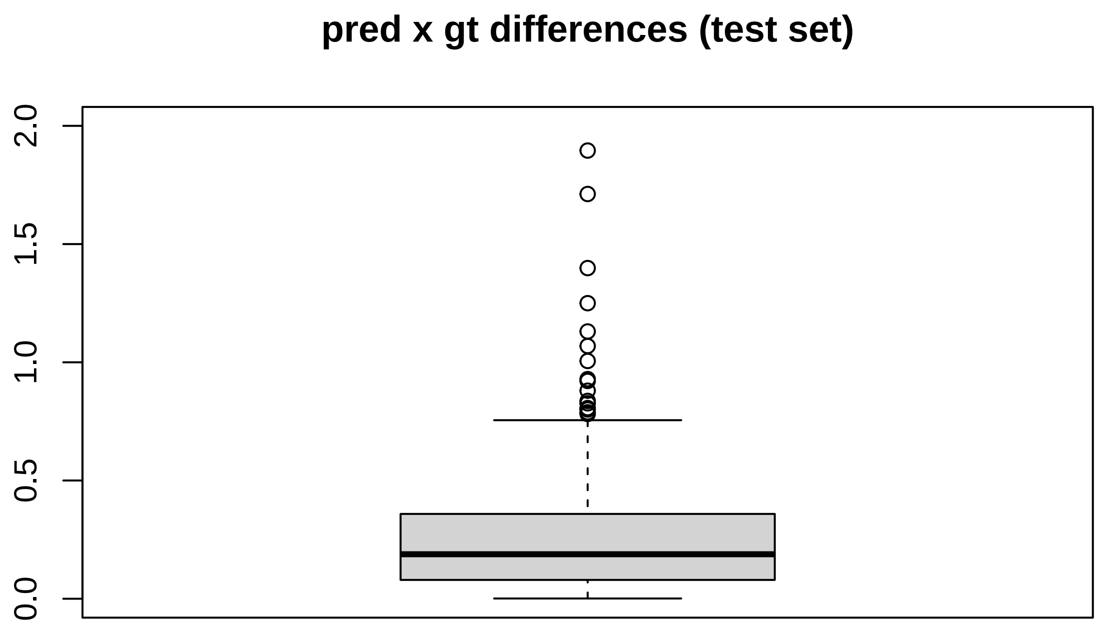


2.b

```r
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(0,2))
```



```r
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(0,2))
```



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 huge amount of the explanatory power for Apps. Dimension reduction successful.

```
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:  0.388397824099111
## Test RMSE:  0.666455196611784
```

2.d

ANOVA for checking if the two samples. Say alpha=0.05, the p-value of 0.11 means that the difference of the models is not significant. Google says an F-value of over 2.5 would suggest to reject the null hypothesis. Ours is 1.69.

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 My questions for oct 19th: What was the model evaluated on for receiving these metrics?

```r
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 76.045
## 2    507 77.840 -7   -1.7953 1.6863   0.11
```

3.

Both the RMSE aswell as the y vs. y hat plots are similar.

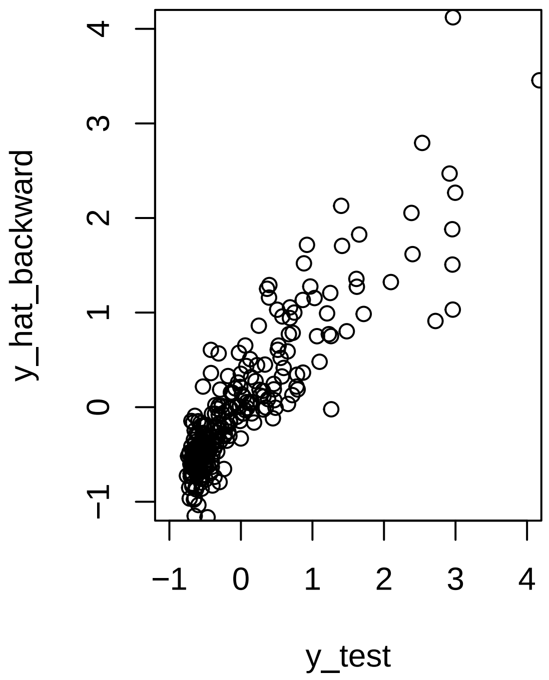
```r
model_backward = step(my_model, direction="backward", trace=0)
model_forward = step(my_model, direction="forward", trace=0)

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)))
```

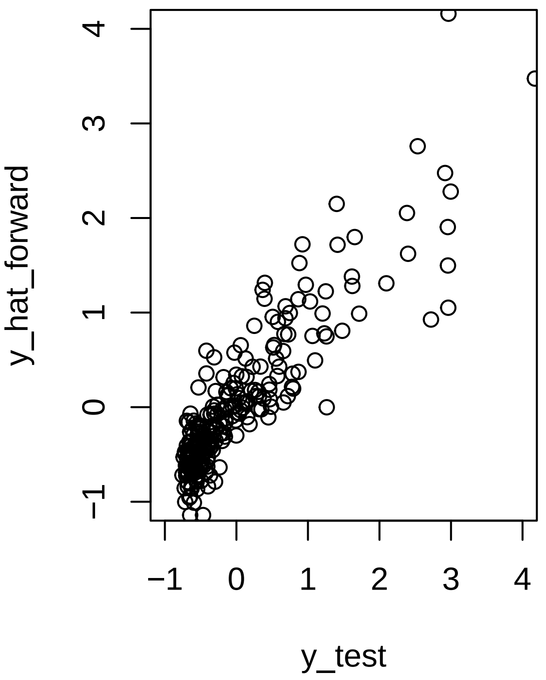
```
## backward model rmse:  0.669844584406186
```

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

```
##
## forward model rmse:  0.667468082778117
```

```r
par(mfrow=c(1,2))
plot(x=y_test, y=y_hat_backward, xlim=c(-1,4), ylim=c(-1,4))
plot(x=y_test, y=y_hat_forward, xlim=c(-1,4), ylim=c(-1,4))
```


y_hat_backward
y_test
−1
0
1
2
3
4


y_hat_forward
y_test
−1
0
1
2
3
4