

# Simple template for R Markdown

## for Advanced Methods for Regression and Classification

Prof. Peter Filzmoser

01.10.2024

```
knitr::opts_chunk$set(echo = TRUE,warning=FALSE,message=FALSE)
data(College,package="ISLR")
str(College)
```

```
## 'data.frame': 777 obs. of 18 variables:
## $ Private : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
## $ Apps : num 1660 2186 1428 417 193 ...
## $ Accept : num 1232 1924 1097 349 146 ...
## $ Enroll : num 721 512 336 137 55 158 103 489 227 172 ...
## $ Top10perc : num 23 16 22 60 16 38 17 37 30 21 ...
## $ Top25perc : num 52 29 50 89 44 62 45 68 63 44 ...
## $ F.Undergrad: num 2885 2683 1036 510 249 ...
## $ P.Undergrad: num 537 1227 99 63 869 ...
## $ Outstate : num 7440 12280 11250 12960 7560 ...
## $ Room.Board : num 3300 6450 3750 5450 4120 ...
## $ Books : num 450 750 400 450 800 500 500 450 300 660 ...
## $ Personal : num 2200 1500 1165 875 1500 ...
## $ PhD : num 70 29 53 92 76 67 90 89 79 40 ...
## $ Terminal : num 78 30 66 97 72 73 93 100 84 41 ...
## $ S.F.Ratio : num 18.1 12.2 12.9 7.7 11.9 9.4 11.5 13.7 11.3 11.5 ...
## $ perc.alumni: num 12 16 30 37 2 11 26 37 23 15 ...
## $ Expend : num 7041 10527 8735 19016 10922 ...
## $ Grad.Rate : num 60 56 54 59 15 55 63 73 80 52 ...
```

```
summary(College)
```

For some Codeblocks, we hide the results as they were exploding the PDF File.

Our goal is to find a linear regression model which allows to predict the variable Apps, i.e. the number of applications received, using the remaining variables except of the variables Accept and Enroll.

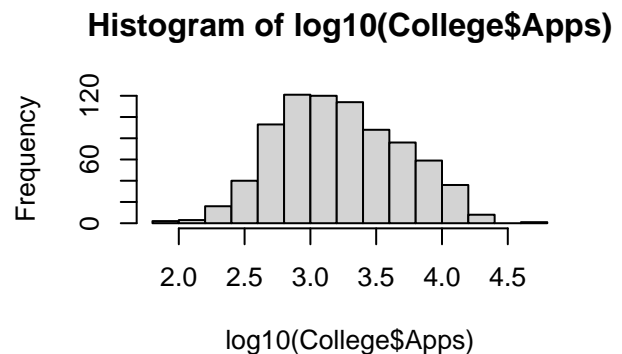
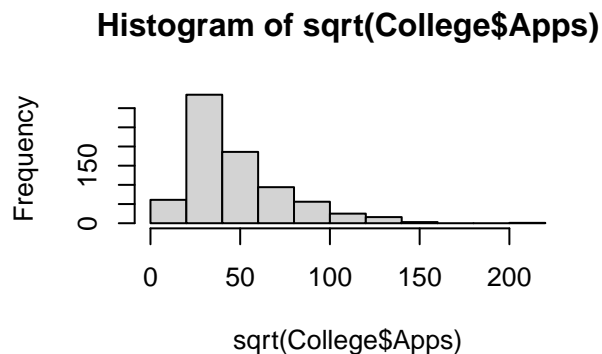
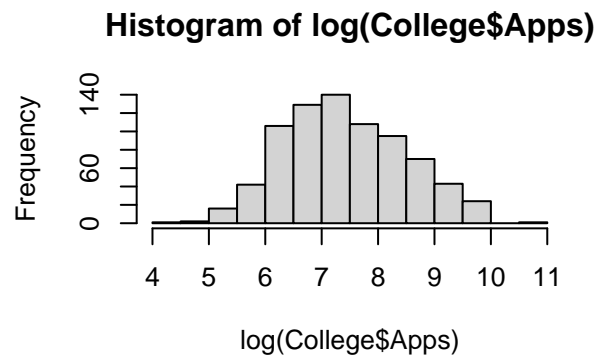
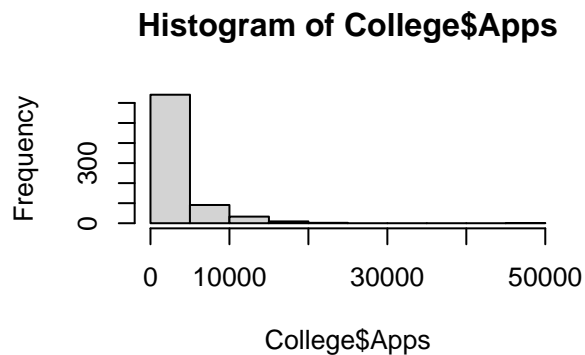
For the following tasks, split the data randomly into training and test data (about 2/3 and 1/3), build the model with the training data, and evaluate the model using the RMSE as a criterion.

split the data into training and test data:

```
n <- nrow(College)
set.seed(11835945)
train <- sample(1:n, 2*n/3)
test <- -train
```

1. Look first at your data. Is any preprocessing necessary or useful? Argue why a log-transformation of the response variable can be useful. Continue with  $\log(\text{Apps})$  as the response.

```
par(mfrow=c(2,2))
hist(College$Apps)
hist(log(College$Apps))
hist(sqrt(College$Apps))
hist(log10(College$Apps))
```



Logarithmic values are normal distributed.

```
College$logApps <- log(College$Apps)
College<-College[-c(2,3,4)]
train.data <- College[train,]
test.data <- College[test,]
#intersect(train.data,test.data)
```

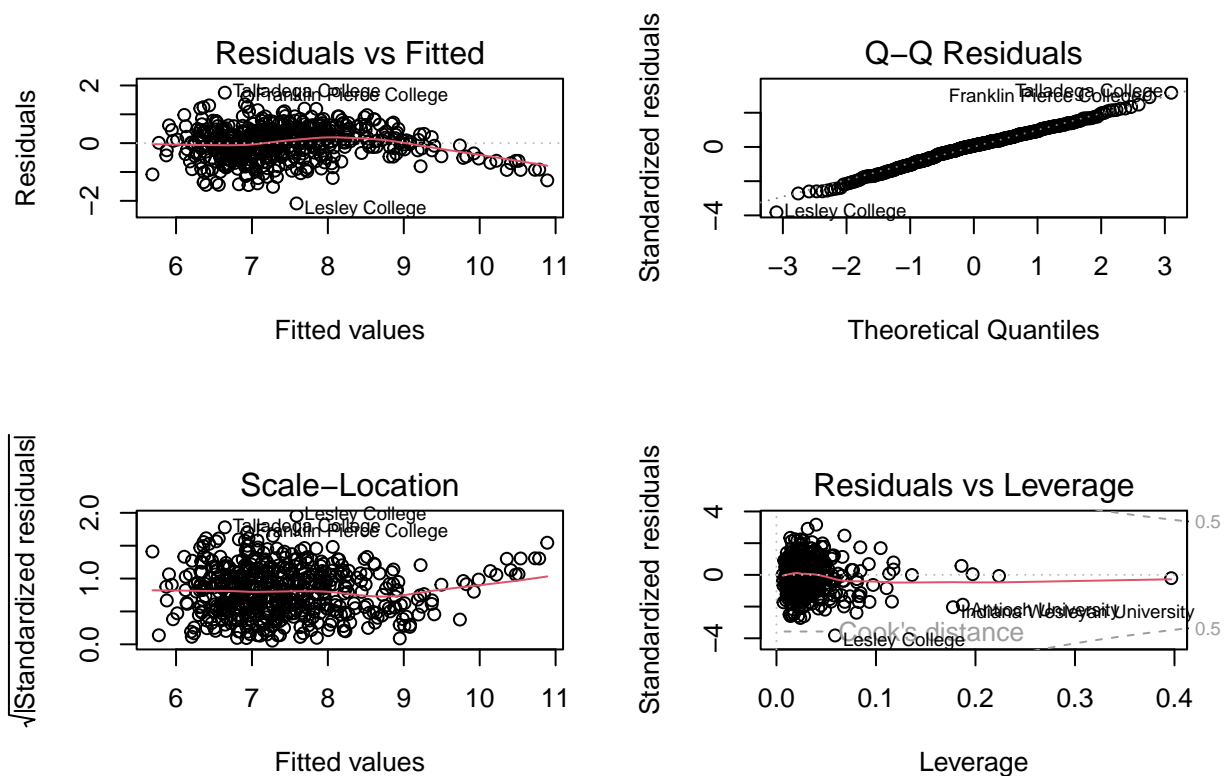
## 2. Full model: Estimate the full regression model and interpret the results.

(a) or that purpose, apply the function `lm()` to compute the estimator – for details see course notes. Interpret the outcome of `summary(res)`, where `res` is the output from the `lm()` function. Which variables contribute to explaining the response variable? Look at diagnostics plots with `plot(res)`. Are the model assumptions fulfilled?

```
par(mfrow=c(2,2))
res <- lm(logApps ~ ., data=train.data)
summary(res)

##
## Call:
## lm(formula = logApps ~ ., data = train.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.09017 -0.35901  0.04444  0.36923  1.75008
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.505e+00  2.627e-01  17.149  < 2e-16 ***
## PrivateYes   -5.458e-01  9.592e-02  -5.690  2.16e-08 ***
## Top10perc     8.395e-04  3.548e-03   0.237  0.813042
## Top25perc     1.784e-03  2.875e-03   0.621  0.535165
## F.Undergrad   1.096e-04  8.115e-06  13.500  < 2e-16 ***
## P.Undergrad   1.675e-05  1.908e-05   0.878  0.380440
## Outstate      4.363e-05  1.263e-05   3.456  0.000596 ***
## Room.Board    9.068e-05  3.146e-05   2.883  0.004111 **
## Books         4.742e-04  1.714e-04   2.767  0.005866 **
## Personal     -6.124e-05  4.438e-05  -1.380  0.168265
## PhD           8.776e-03  3.060e-03   2.868  0.004309 **
## Terminal      1.362e-04  3.341e-03   0.041  0.967492
## S.F.Ratio     3.887e-02  8.608e-03   4.516  7.86e-06 ***
## perc.alumni  -8.729e-03  2.656e-03  -3.287  0.001085 **
## Expend        2.758e-05  7.582e-06   3.638  0.000303 ***
## Grad.Rate     7.389e-03  1.931e-03   3.827  0.000146 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.564 on 502 degrees of freedom
## Multiple R-squared:  0.7225, Adjusted R-squared:  0.7143
## F-statistic: 87.15 on 15 and 502 DF,  p-value: < 2.2e-16
```

```
plot(res)
```



predict the number of applications for the test data:

```
pred <- predict(res, newdata=test.data)
```

calculate the RMSE:

```
rmse <- sqrt(mean((test.data$logApps - pred)^2))
rmse
```

```
## [1] 0.5939058
```

Now we check what variables are important for the prediction:

```
library(caret)
varImp(res)
```

```
##           Overall
## PrivateYes  5.68976620
## Top10perc   0.23662776
## Top25perc   0.62056807
## F.Undergrad 13.49986698
## P.Undergrad  0.87785930
## Outstate    3.45559629
## Room.Board   2.88280220
```

```
## Books      2.76699251
## Personal   1.37979118
## PhD        2.86766202
## Terminal   0.04077427
## S.F.Ratio  4.51608349
## perc.alumni 3.28655720
## Expend     3.63796291
## Grad.Rate  3.82657917
```

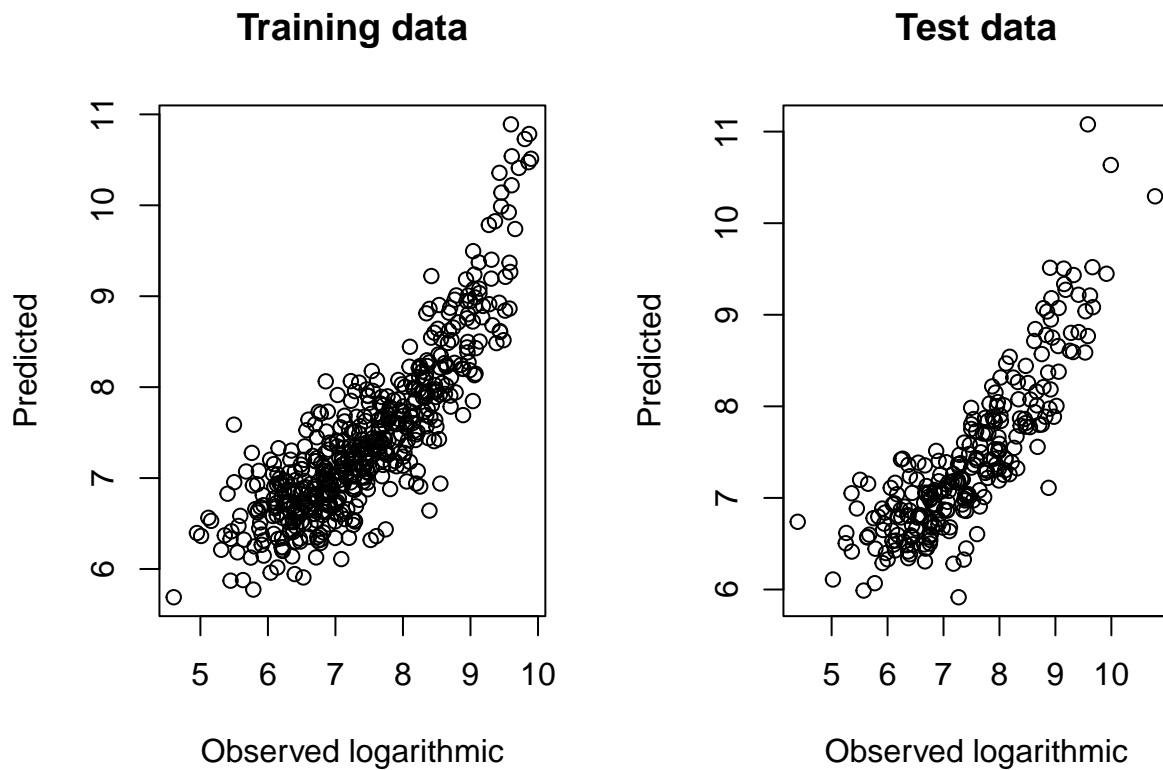
(b) Now we try to manually compute the LS coefficients, in the same way as `lm()`. Thus, replace from the above command `lm()` by `model.matrix()`. This gives you the matrix `X` as it is used to estimate the regression coefficients. Now apply the formula to compute the LS estimator. You can do matrix multiplication in R by `%*%`, and the inverse of a matrix is computed with `solve()`. How is R handling binary variables (Private), and how can you interpret the corresponding regression coefficient? Compare the resulting coefficients with those obtained from `lm()`. Do you get the same result?

```
X <- model.matrix(logApps ~ . , data=train.data)
y <- train.data$logApps
beta <- solve(t(X) %*% X) %*% t(X) %*% y
beta
```

```
##           [,1]
## (Intercept) 4.504733e+00
## PrivateYes  -5.457882e-01
## Top10perc    8.395119e-04
## Top25perc    1.783999e-03
## F.Undergrad  1.095526e-04
## P.Undergrad  1.675083e-05
## Outstate     4.363130e-05
## Room.Board   9.068202e-05
## Books        4.742280e-04
## Personal    -6.123653e-05
## PhD          8.775718e-03
## Terminal     1.362216e-04
## S.F.Ratio    3.887223e-02
## perc.alumni -8.729084e-03
## Expend       2.758132e-05
## Grad.Rate    7.389273e-03
```

(c) Compare graphically the observed and the predicted values of the response variable – once only for the training data, and once for the test data. What do you think about the prediction performance of your model?

```
par(mfrow=c(1,2))
plot(train.data$logApps, predict(res), xlab="Observed logarithmic", ylab="Predicted", main="Training data")
plot(test.data$logApps, pred, xlab="Observed logarithmic", ylab="Predicted", main="Test data")
```



(d) Compute the RMSE separately for training and test data, and compare the values. What do you conclude?

```
pred.train <- predict(res, newdata=train.data)
rmse.train <- sqrt(mean((train.data$logApps - pred.train)^2))
rmse.train
```

```
## [1] 0.5552257
```

```
rmse
```

```
## [1] 0.5939058
```

Since the model was fitted to the training data, i expect the RMSE of the test data set to be bigger. This is the also the case.

**3. Reduced model:** Exclude all input variables from the model which were not significant in 2(a), and compute the LS-estimator.

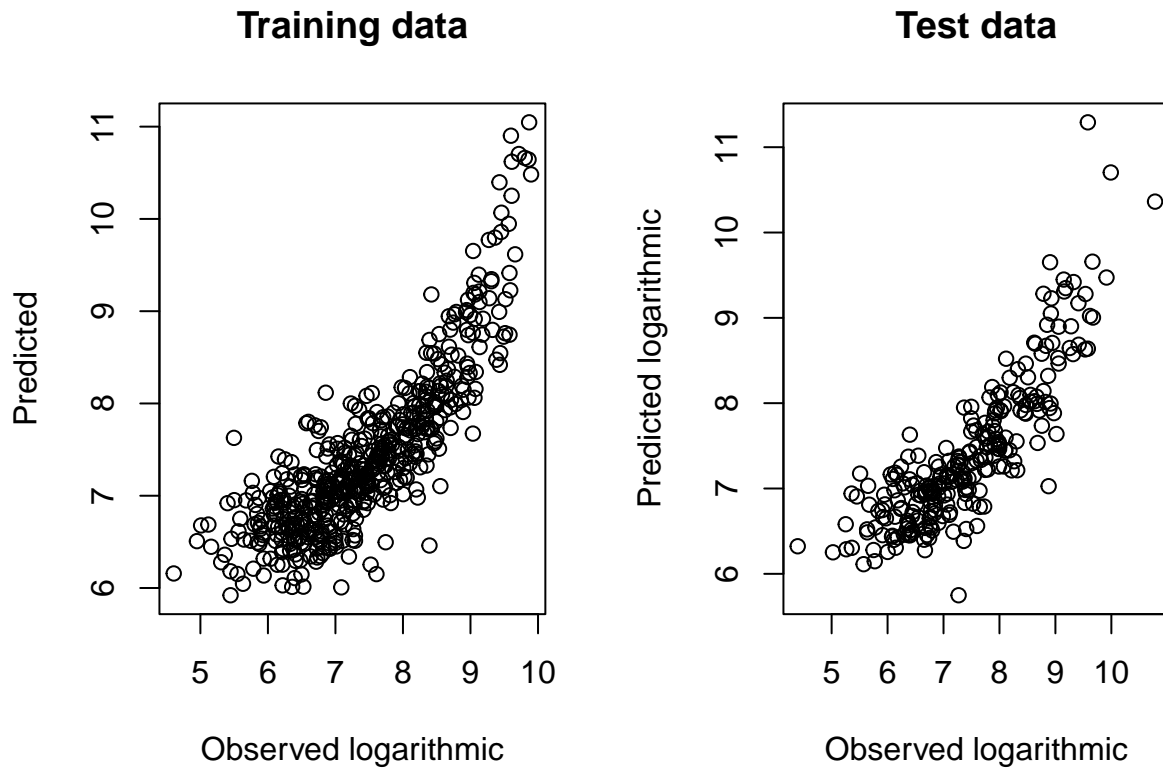
```
reduced.model<- lm(logApps ~ . -Top25perc -Top10perc -P.Undergrad -Personal -PhD -Terminal -perc.alumni)
summary(reduced.model)
```

```
##
## Call:
## lm(formula = logApps ~ . - Top25perc - Top10perc - P.Undergrad -
##      Personal - PhD - Terminal - perc.alumni, data = train.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.12992 -0.32849  0.05935  0.38432  1.93196
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.663e+00  2.217e-01  21.037  < 2e-16 ***
## PrivateYes   -6.938e-01  9.363e-02  -7.410  5.29e-13 ***
## F.Undergrad  1.199e-04  7.108e-06  16.867  < 2e-16 ***
## Outstate     5.309e-05  1.196e-05   4.438  1.12e-05 ***
## Room.Board   1.240e-04  3.122e-05   3.972  8.16e-05 ***
## Books        4.548e-04  1.695e-04   2.684  0.00752 **
## S.F.Ratio    4.444e-02  8.754e-03   5.076  5.41e-07 ***
## Expend       3.316e-05  7.063e-06   4.694  3.44e-06 ***
## Grad.Rate    8.100e-03  1.811e-03   4.471  9.59e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5796 on 509 degrees of freedom
## Multiple R-squared:  0.7029, Adjusted R-squared:  0.6983
## F-statistic: 150.6 on 8 and 509 DF,  p-value: < 2.2e-16
```

(a) Are now all input variables significant in the model? Why is this not to be expected in general?

Yes. Various Reasons such as Overfitting, colinearity, sample size limitations, noise and bias, etc. ### (b) Visualize the fit and the prediction from the new model, see 2(c).

```
par(mfrow=c(1,2))
pred <- predict(reduced.model, newdata=test.data)
plot(train.data$logApps, predict(reduced.model), xlab="Observed logarithmic", ylab="Predicted", main="Train data")
plot(test.data$logApps, pred, xlab="Observed logarithmic", ylab="Predicted logarithmic", main="Test data")
```



(c) Compute the RMSE for the new model, see 2(d). What would we expect?

```
pred.train <- predict(reduced.model, newdata=train.data)
rmse.train <- sqrt(mean((train.data$logApps - pred.train)^2))
cat("RMSE of native model",rmse.train)
```

```
## RMSE of native model 0.5745158
```

```
rmse <- sqrt(mean((test.data$logApps - pred)^2))
cat("RMSE of reduced model",rmse)
```

```
## RMSE of reduced model 0.5908559
```

I expect the new model to have a higher RMSE, even though we only removed insignificant variables. However, the error is smaller since with all variables we did fit the model to the noise of the model (Overfitting)

### (d) Compare the two models with anova(). What can you conclude?

```
anova(res,reduced.model)
```

```
## Analysis of Variance Table
```

```
##
```

```
## Model 1: logApps ~ Private + Top10perc + Top25perc + F.Undergrad + P.Undergrad +
```



```
##      Outstate + Room.Board + Books + Personal + PhD + Terminal +
##      S.F.Ratio + perc.alumni + Expend + Grad.Rate
## Model 2: logApps ~ (Private + Top10perc + Top25perc + F.Undergrad + P.Undergrad +
##      Outstate + Room.Board + Books + Personal + PhD + Terminal +
##      S.F.Ratio + perc.alumni + Expend + Grad.Rate) - Top25perc -
##      Top10perc - P.Undergrad - Personal - PhD - Terminal - perc.alumni
##      Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1      502 159.69
## 2      509 170.97 -7    -11.289 5.0697 1.413e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4. Perform variable selection based on stepwise regression, using the function `step()`, see help file and course notes. Perform both, forward selection (start from the empty model) and backward selection (start from the full model). Compare the resulting models with the RMSE, and with plots of response versus predicted values.

```
full_model <- lm(logApps ~ ., data=train.data)
empty_model <- lm(logApps ~ 1, data = train.data)
forward_model <- step(empty_model, direction = "forward", scope=formula(full_model))
backward_model <- step(full_model, direction = "backward")
anova(forward_model, backward_model)
```

```
anova(forward_model, backward_model)
```

```
## Analysis of Variance Table
##
## Model 1: logApps ~ F.Undergrad + PhD + Room.Board + Private + Outstate +
##      Grad.Rate + S.F.Ratio + Expend + perc.alumni + Books
## Model 2: logApps ~ Private + F.Undergrad + Outstate + Room.Board + Books +
##      PhD + S.F.Ratio + perc.alumni + Expend + Grad.Rate
##      Res.Df    RSS Df    Sum of Sq F Pr(>F)
## 1      507 160.96
## 2      507 160.96  0 -1.1369e-13
```

```
# Function to calculate RMSE
rmse <- function(model){
  predictions <- predict(model, train.data)
  sqrt(mean((test.data$logApps - predictions)^2))
}
rmse_forward <- rmse(forward_model)
rmse_backward <- rmse(backward_model)

cat("RMSE of Forward Model:", rmse_forward)
```

```
## RMSE of Forward Model: 1.447649
```

```
cat("RMSE of Backward Model:", rmse_backward)
```

```
## RMSE of Backward Model: 1.447649
```

```
require(gridExtra)
plot_model <- function(model, title) {
  predictions <- predict(model, newdata = test.data)
  ggplot(test.data, aes(x = predictions, y = logApps)) +
    geom_point() +
    geom_smooth(method = "lm", color = "blue") +
    labs(title = title, x = "Predicted Values", y = "Actual Values") +
    theme_minimal() } # Plotting both models
plot1 <- plot_model(forward_model, "Forward Selection Model")
plot2 <- plot_model(backward_model, "Backward Selection Model")
grid.arrange(plot1, plot2, ncol=2)
```

