Exercise 1

105.707 Advanced Methods for Regression and Classification, 2024W

11912007 - Yahya Jabary

22.10.2024

Contents

1 Preprocessing

Question: *Is any preprocessing necessary or useful? Argue why a log-transformation of the response variable can be useful.*

We don't have any missing values to handle.

But a log-transformation of the response variable `Apps` can be useful for:

- Fixing skewed data: Log-transformation can help reduce this right-skewness and make the distribution more symmetric. The summary statistics show that the `Apps` variable is highly skewed, with a median (1558) much lower than the mean (3002) and a very large maximum value (48094) compared to the minimum (81).
- Scale compression: Log-transformation compresses the scale of variables that span several orders of magnitude, as is the case with Apps (from 81 to 48094).
- Variance stabilization, outlier impact reduction: Log transformations are useful when you have data that's spread out over a wide range. It helps bring very large numbers closer together and spreads out very small numbers, making the data easier to work with.
- Linearization of relationships: Log transforming can often make relationships simpler and more linear, which can improve the fit of linear regression models.
- Interpretation in terms of relative changes: When using log-transformed data, coefficients can be interpreted in terms of percentage changes rather than absolute changes. This can be more intuitive sometimes.

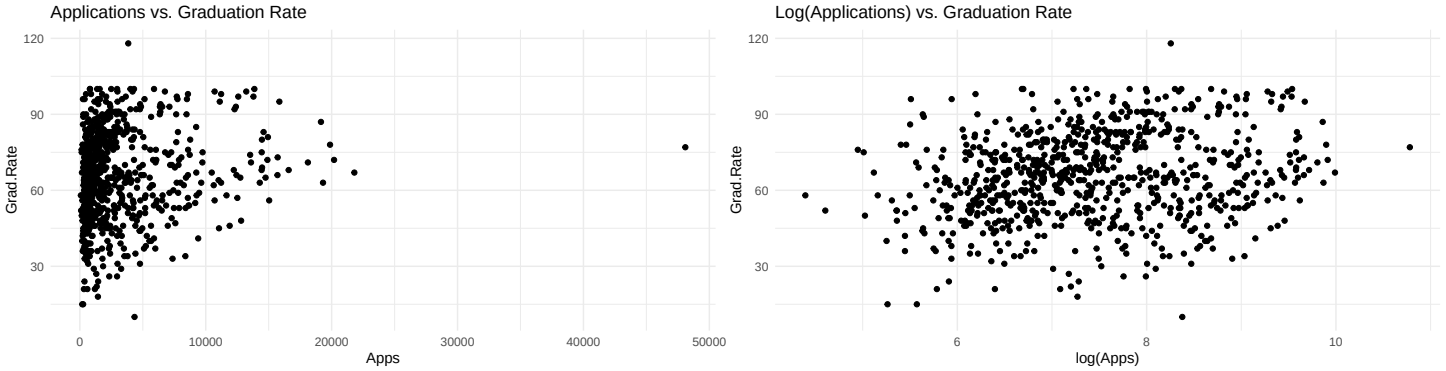
There are some caveats to keep in mind however:

- It doesn't make sense for all variables.
- After log transforming, you'll need to interpret your results differently (e.g., transforming back to the original scale).
- You can't log transform zero or negative numbers directly.

```
data(College)

# no values to drop
cat("num of missing vals: ", sum(is.na(College)), "\n")
```

```
## num of missing vals:  0
```

```
# log transform fixes skewness
before <- ggplot(College, aes(Apps, Grad.Rate)) + geom_point() + theme_minimal() + labs(title = "Applications vs. Graduation Rate")
after <- ggplot(College, aes(log(Apps), Grad.Rate)) + geom_point() + theme_minimal() + labs(title = "Log(Applications) vs. Graduation Rate")
print(before + after)
```




```
summary(College$Apps)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      81     776    1558    3002    3624   48094
```

2 Training Full model

Question: *Interpret the outcome of `summary(res)`, where `res` is the output from the `lm()` function. Which variables contribute to explaining the response variable?*

The significance codes are denoted by the number of stars next to the variable names. They show the significance to explaining the response variable `log(Apps)` in the model. Some are highly significant with p-values as low as 2e-16, while others are significant with p-values less than 0.01.

The model has an Adjusted R-squared of 0.7155, indicating that it explains about 71.55% of the variance in `log(Apps)`.

Question: *Look at diagnostics plots with `plot(res)`. Are the model assumptions fulfilled?*

- Residuals vs Fitted: There's a slight curve in the red line, suggesting some non-linearity might be present. Q-Q Residuals: The residuals mostly follow the diagonal line, indicating approximate normality, with some deviation at the tails.
- Scale-Location: There's a slight upward trend, suggesting some heteroscedasticity (meaning the spread of errors / residuals is not constant across all values of the predictor variables).

- Residuals vs Leverage: No points fall outside Cook's distance (dotted lines), indicating no highly influential points (outliers) in the data.

While not perfect, the model assumptions are reasonably well met, with some minor concerns about non-linearity and heteroscedasticity.

Question: *How is R handling binary variables (Private), and how can you interpret the corresponding regression coefficient?*

R handles the binary variable `Private` using dummy coding. The coefficient for `PrivateYes` represents the average difference in `log(Apps)` between private and public institutions, holding other variables constant. Private institutions are associated with approximately $\exp(-0.630) = 0.532 = 53\%$ fewer applications compared to public institutions, all else being equal.

Question: *Compare the resulting coefficients with those obtained from `lm()`. Do you get the same result?*

The test case passes with a tolerance of 1e-10.

Note: We get the warning "names for current but not for target" which means they are not in the same order - but that doesn't matter for the comparison.

Question: *What do you think about the prediction performance of your model (using a graphical representation)?*

The scatter plots of observed vs. predicted applications show a strong positive correlation between observed and predicted values for both training and test data.

We can also see that the model tends to underpredict for very high numbers of applications, especially in the test data. Prediction accuracy seems slightly better for the training data compared to the test data, which is expected.

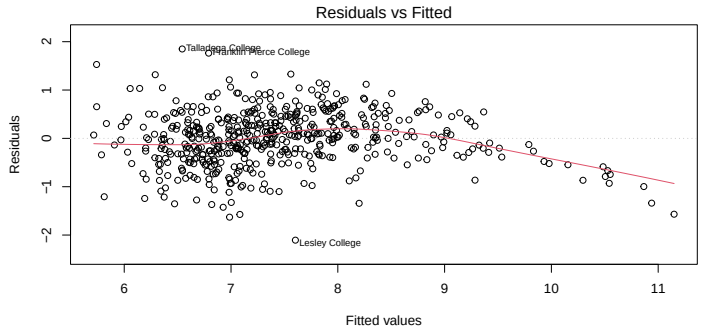
Question: *Compute the RMSE separately for training and test data, and compare the values. What do you conclude?*

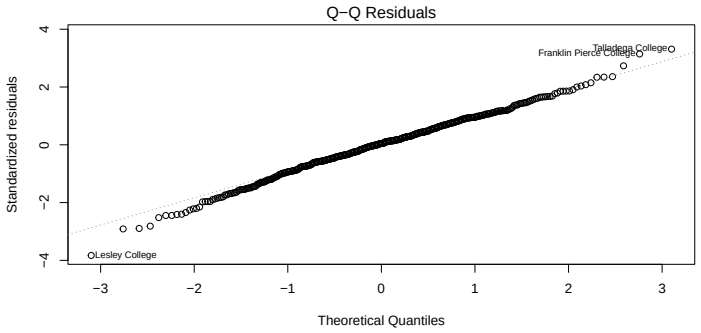
The model didn't generalize well to the test data, with a difference of ≈ 911 between the training and test RMSE values.

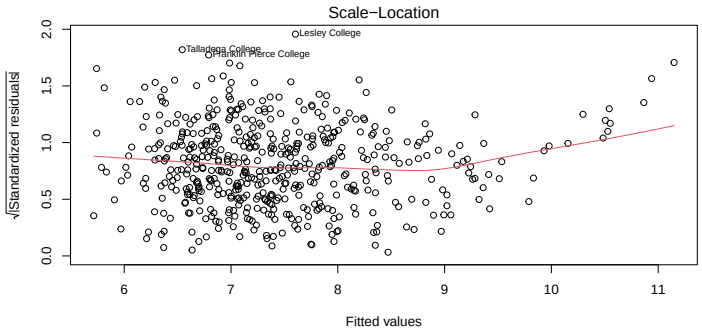
```r
data(College)

# holdout 2/3 split
train_index <- sample(1:nrow(College), 2/3 * nrow(College))
train_data <- College[train_index, ]
test_data <- College[-train_index, ]

# fit model
full_model <- lm(log(Apps) ~ . - Accept - Enroll, data = train_data)
summary(full_model)
```

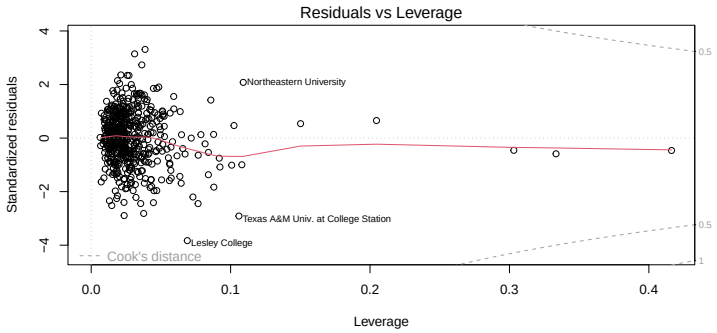
```
##
## Call:
## lm(formula = log(Apps) ~ . - Accept - Enroll, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.10573 -0.32671  0.02545  0.38679  1.84955
##
## Coefficients:
##                  Estimate   Std. Error t value             Pr(>|t|)
## (Intercept)  4.122121753  0.290045020  14.212 < 0.0000000000000002 ***
## PrivateYes  -0.630894166  0.092589954  -6.814      0.0000000000273 ***
## Top10perc   -0.000995426  0.003688635  -0.270              0.787377
## Top25perc    0.005979155  0.002941881   2.032              0.042636 *
## F.Undergrad  0.000101549  0.000007995  12.702 < 0.0000000000000002 ***
## P.Undergrad  0.000020071  0.000019525   1.028              0.304461
## Outstate     0.000051803  0.000012931   4.006      0.0000710658377 ***
## Room.Board   0.000057750  0.000034141   1.692              0.091359 .
## Books        0.000346539  0.000163506   2.119              0.034545 *
## Personal     0.000005098  0.000044666   0.114              0.909169
## PhD          0.008150631  0.003316154   2.458              0.014314 *
## Terminal    -0.000325757  0.003680042  -0.089              0.929499
## S.F.Ratio    0.052654881  0.009500523   5.542      0.0000000482817 ***
## perc.alumni -0.006161222  0.002829263  -2.178              0.029895 *
## Expend       0.000033337  0.000009027   3.693              0.000246 ***
## Grad.Rate    0.008702689  0.002042044   4.262      0.0000242308877 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5698 on 502 degrees of freedom
## Multiple R-squared:  0.727,  Adjusted R-squared:  0.7188
## F-statistic: 89.11 on 15 and 502 DF,  p-value: < 0.00000000000000022
```

```r
par(mfrow=c(2,2))
plot(full_model)
```









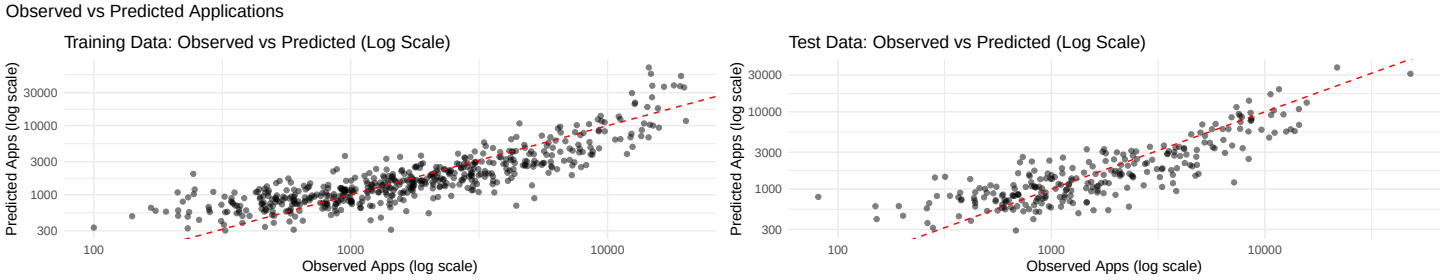

```
# solve analytically
X <- model.matrix(~ . - Accept - Enroll - Apps, data = train_data) # model matrix X
y <- log(train_data$Apps) # response vector y
beta_hat <- solve(t(X) %*% X) %*% t(X) %*% y # get LS estimate of beta analytically
all.equal(as.vector(beta_hat), coef(full_model), 1e-10) # compare coeffs
```

```
## [1] "names for current but not for target"
```

```
# get rmse error
train_error <- rmse(train_data$Apps, exp(predict(full_model)))
test_error <- rmse(test_data$Apps, exp(predict(full_model, newdata = test_data)))
cat("train RMSE: ", train_error, "vs. test RMSE: ", test_error, "\n")
```

```
## train RMSE:  4332.683 vs. test RMSE:  2288.898
```

```
# plot observed vs predicted
train_plot_data <- data.frame(Observed = train_data$Apps, Predicted = exp(predict(full_model, newdata = train_data)))
train_plot <- ggplot(train_plot_data, aes(x = Observed, y = Predicted)) +
  geom_point(alpha = 0.5) +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  scale_x_log10() + scale_y_log10() +
  labs(title = "Training Data: Observed vs Predicted (Log Scale)", x = "Observed Apps (log scale)", y = "Predicted Apps (log scale)") +
  theme_minimal()
test_plot_data <- data.frame(Observed = test_data$Apps, Predicted = exp(predict(full_model, newdata = test_data)))
test_plot <- ggplot(test_plot_data, aes(x = Observed, y = Predicted)) +
  geom_point(alpha = 0.5) +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  scale_x_log10() + scale_y_log10() +
  labs(title = "Test Data: Observed vs Predicted (Log Scale)", x = "Observed Apps (log scale)", y = "Predicted Apps (log scale)") +
  theme_minimal()
combined_plot <- train_plot + test_plot + plot_layout(ncol = 2) + plot_annotation(title = "Observed vs Predicted Applications")
print(combined_plot)
```



3 Training Reduced model

Question: *Are now all input variables significant in the model? Why is this not to be expected in general?*

In the reduced model, all input variables are now significant at the 0.05 level or better.

However, this is not generally expected for several reasons:

- Variable Selection Bias: By selecting only significant variables from the full model, we introduce a selection bias that can artificially inflate the significance of the remaining variables.

- Overfitting: The reduced model may be overfitting to the specific dataset, capturing noise rather than true underlying relationships.
- Multicollinearity (meaning that some variables are highly correlated with each other): Removing some variables can change the relationships between the remaining variables, potentially altering their significance levels.

Question: *Compute the RMSE for the new model. What would we expect?*

The reduced model's RMSE values are: 7549.269 (Train), 8459.517 (Test). These values are higher than those of the full model 4332.683 (Train), 2288.898 (Test).

This increase in RMSE is expected because (1) the reduced model has fewer predictors, potentially losing some explanatory power and (2) the full model may have been capturing some important relationships that are now omitted.

However, the reduced model could potentially generalize better to new data, despite the higher RMSE on the current test set given its lower complexity.

Question: *Compare the two models with* `anova()`*. What can you conclude?*

The ANOVA test comparing the reduced and full models yields a highly significant result (p-value < 0.001). This indicates that there is a statistically significant difference between the two models / the full model explains significantly more variance in the data than the reduced model.

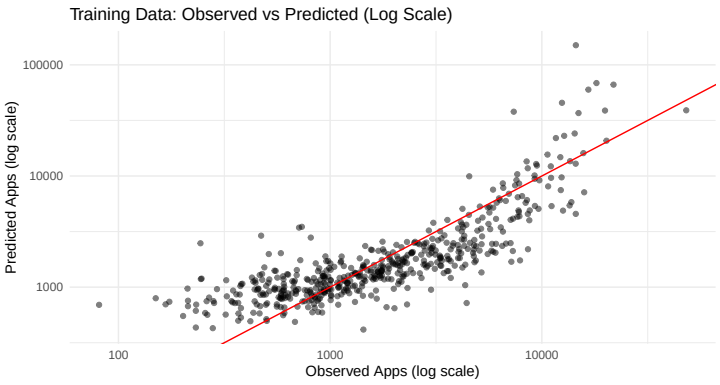
However, it's important to note that statistical significance doesn't always translate to practical significance.

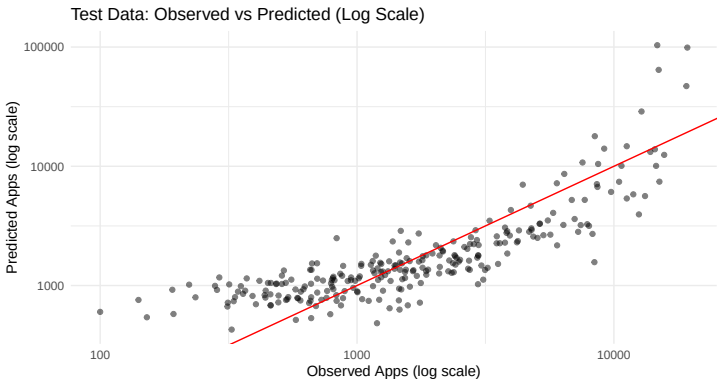
In conclusion the reduced model, while explaining less variance and performing worse on the test set might be more interpretable and generalizable. The choice between the two models would depend on the specific goals of the analysis, balancing between model complexity, interpretability and predictive power.

```
data(College)

# holdout 2/3 split
train_index <- sample(1:nrow(College), 2/3 * nrow(College))
train_data <- College[train_index, ]
test_data <- College[-train_index, ]

# get coeffs and p-values < 0.05
full_model <- lm(log(Apps) ~ . - Accept - Enroll, data = train_data)
summary_full <- summary(full_model)
coef_summary <- summary_full$coefficients
significant_vars <- rownames(coef_summary)[coef_summary[, "Pr(>|t|)"] < 0.05]
significant_vars <- significant_vars[!grepl("^\\(Intercept\\)|Yes$", significant_vars)] # drop intercept
reduced_formula <- as.formula(paste("log(Apps) ~", paste(significant_vars, collapse = " + ")))
# fit reduced model
reduced_model <- lm(reduced_formula, data = train_data)
summary(reduced_model)
```

```
##
## Call:
## lm(formula = reduced_formula, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3395 -0.3326  0.0606  0.3744  1.8136
##
## Coefficients:
##                 Estimate   Std. Error t value             Pr(>|t|)
## (Intercept)  4.491881807  0.215320793  20.861 < 0.0000000000000002 ***
## F.Undergrad  0.000153902  0.000006323  24.340 < 0.0000000000000002 ***
## Outstate     0.000034217  0.000012772   2.679              0.00762 **
## Room.Board   0.000068912  0.000032829   2.099              0.03630 *
## S.F.Ratio    0.054667250  0.009126658   5.990        0.00000000397 ***
## perc.alumni -0.007931156  0.002853195  -2.780              0.00564 **
## Expend       0.000044957  0.000007580   5.931        0.00000000556 ***
## Grad.Rate    0.010629885  0.002089706   5.087        0.00000051262 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6221 on 510 degrees of freedom
## Multiple R-squared:  0.6667, Adjusted R-squared:  0.6621
## F-statistic: 145.7 on 7 and 510 DF,  p-value: < 0.00000000000000022
```

```
# plot observed vs predicted
train_data_plot <- data.frame(Observed = train_data$Apps, Predicted = exp(predict(reduced_model, train_data)))
p1 <- ggplot(train_data_plot, aes(x = Observed, y = Predicted)) +
  geom_point(alpha = 0.5) +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  scale_x_log10() + scale_y_log10() +
  labs(title = "Training Data: Observed vs Predicted (Log Scale)", x = "Observed Apps (log scale)", y = "Predicted Apps (log scale)") +
  theme_minimal()
test_data_plot <- data.frame(Observed = test_data$Apps, Predicted = exp(predict(reduced_model, test_data)))
p2 <- ggplot(test_data_plot, aes(x = Observed, y = Predicted)) +
  geom_point(alpha = 0.5) +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  scale_x_log10() + scale_y_log10() +
  labs(title = "Test Data: Observed vs Predicted (Log Scale)", x = "Observed Apps (log scale)", y = "Predicted Apps (log scale)") +
  theme_minimal()
grid.arrange(p1, p2, ncol = 2)
```






```r
# get rmse error
train_error <- rmse(train_data$Apps, exp(predict(reduced_model)))
test_error <- rmse(test_data$Apps, exp(predict(reduced_model, newdata = test_data)))
cat("train RMSE: ", train_error, "vs. test RMSE: ", test_error, "\n")
```

```
## train RMSE:  7549.269 vs. test RMSE:  8459.517
```

```r
# anova
anova_result <- anova(reduced_model, full_model)
print(anova_result)
```

```
## Analysis of Variance Table
##
## Model 1: log(Apps) ~ F.Undergrad + Outstate + Room.Board + S.F.Ratio +
##     perc.alumni + Expend + Grad.Rate
## Model 2: log(Apps) ~ (Private + Accept + Enroll + Top10perc + Top25perc +
##     F.Undergrad + P.Undergrad + Outstate + Room.Board + Books +
##     Personal + PhD + Terminal + S.F.Ratio + perc.alumni + Expend +
##     Grad.Rate) - Accept - Enroll
##   Res.Df    RSS Df Sum of Sq      F                Pr(>F)
## 1    510 197.39
## 2    502 165.53  8    31.859 12.077 0.000000000000000007838 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4 Variable selection

Finally we perform forward and backward selection to find the best model and visualize the results.

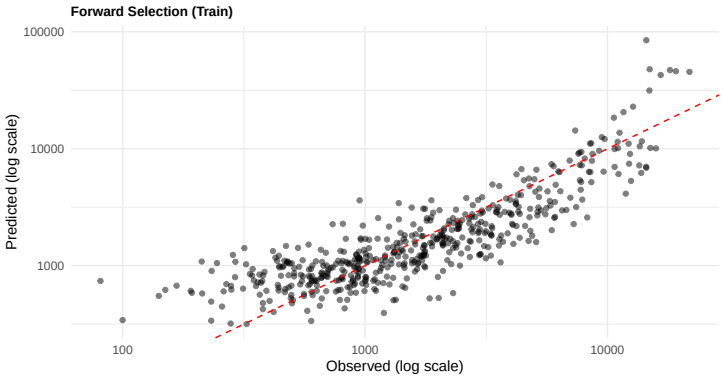
```r
data(College)
College <- na.omit(College)

# holdout 2/3 split
train_index <- sample(1:nrow(College), 2/3 * nrow(College))
train_data <- College[train_index, ]
test_data <- College[-train_index, ]

# forward / backward selection
empty_model <- lm(log(Apps) ~ 1, data = train_data)
full_model <- lm(log(Apps) ~ . - Accept - Enroll, data = train_data)
forward_model <- step(empty_model, direction = "forward", scope = formula(full_model), trace = 0)
backward_model <- step(full_model, direction = "backward", trace = 0)

# rmse
calculate_rmse <- function(model, data) {
    predictions <- exp(predict(model, newdata = data))
    actual <- data$Apps
    rmse <- sqrt(mean((actual - predictions)^2))
    return(rmse)
}
cat("forward selection RMSE:", calculate_rmse(forward_model, train_data), "(train) vs.", calculate_rmse(forward_model, test_data), "(test)\n")
```

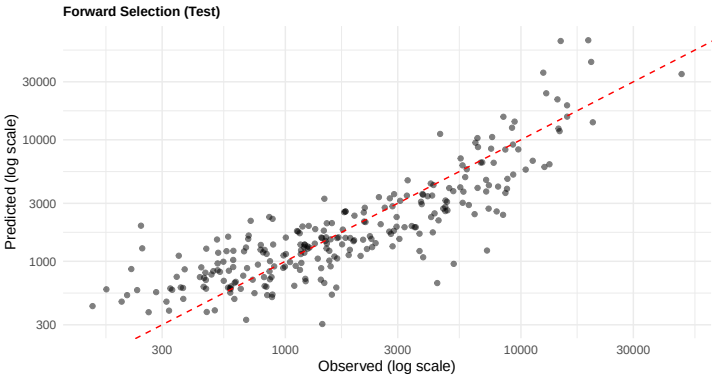
```
## forward selection RMSE: 4503.19 (train) vs. 5196.237 (test)
```

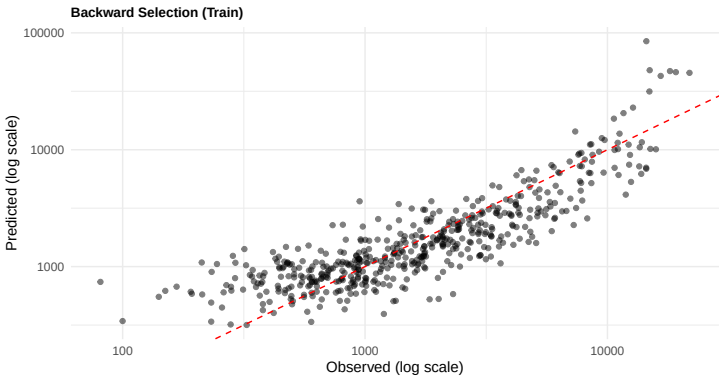
```r
cat("backward selection RMSE:", calculate_rmse(backward_model, train_data), "(train) vs.", calculate_rmse(backward_model, test_data), "(test)\n")
```

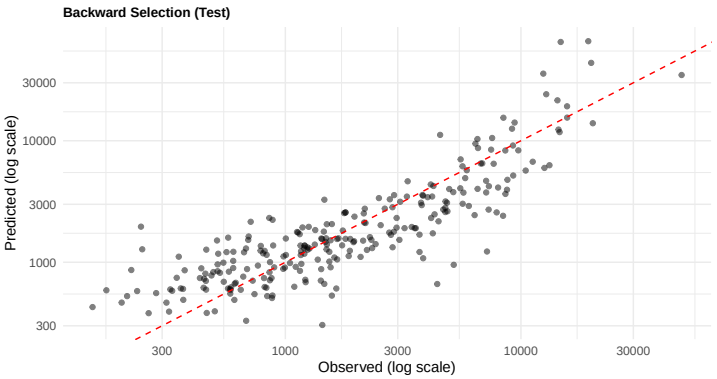
```
## backward selection RMSE: 4503.19 (train) vs. 5196.237 (test)
```

```r
# plot
create_plot <- function(model, data, title) {
predictions <- exp(predict(model, newdata = data))
ggplot(data.frame(observed = data$Apps, predicted = predictions), aes(x = observed, y = predicted)) +
    geom_point(alpha = 0.5, color = "black") +
    geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") + scale_x_log10() + scale_y_log10() +
    labs(title = title, x = "Observed (log scale)", y = "Predicted (log scale)") +
    theme_minimal() +
    theme(plot.title = element_text(size = 10, face = "bold"))
}
p1 <- create_plot(forward_model, train_data, "Forward Selection (Train)")
p2 <- create_plot(forward_model, test_data, "Forward Selection (Test)")
p3 <- create_plot(backward_model, train_data, "Backward Selection (Train)")
p4 <- create_plot(backward_model, test_data, "Backward Selection (Test)")
combined_plot <- (p1 + p2) / (p3 + p4)
print(combined_plot)
```


Forward Selection (Train)
Predicted (log scale)
Observed (log scale)
100000
10000
1000
100
1000
10000


Forward Selection (Test)
Predicted (log scale)
Observed (log scale)
30000
10000
3000
1000
300
300
1000
3000
10000
30000


Backward Selection (Train)
Predicted (log scale)
Observed (log scale)
100000
10000
1000
100
1000
10000


Backward Selection (Test)
Predicted (log scale)
Observed (log scale)
30000
10000
3000
1000
300
300
1000
3000
10000
30000