**Simple and multiple linear regression in R**

Goal: Use linear regression to predict the miles per gallon (MPG) of a car based on its predictors/attributes/features using a real dataset.

Data: Download it from UCI at: <https://archive.ics.uci.edu/ml/datasets/auto+mpg>

**Table of Contents**

Part A: Simple Linear Regression ------ 1

Part B: Multiple Regression -------------- 5

**PART A: Simple Linear Regression**

**Step 1:** Import data using read.table

#read.table can handle a remote URL as well as a local file, be careful to only use trustworthy sites  
auto.mpg <- read.table('https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data', stringsAsFactors = FALSE)  
View(auto.mpg)  
head(auto.mpg,2)

## V1 V2 V3 V4 V5 V6 V7 V8 V9  
## 1 18 8 307 130.0 3504 12.0 70 1 chevrolet chevelle malibu  
## 2 15 8 350 165.0 3693 11.5 70 1 buick skylark 320  
# head()Returns the first or last parts of a vector, matrix, table, data frame or function.

Note these column names “V1…V9” are meaningless. Assign names use NAMES(), referring to the URL above.

names(auto.mpg) <- c("mpg", "cylinders", "displacement", "horsepower", "weight", "acceleration", "model.year", "origin", "name")

**Step 2:** Train/Test Split

Split out data into a training set and a test set, e.g., 80% for training and 20% for testing. Set a seed for the random number generator. In this case, the random number generator (rng) type is Mersenne-Twister by default. The seed number you pass to set.seed is for you to replicate this dataset. E.g.: set.seed(1)will ensure if you restart R and run your code again, the random number generator will output the same sequence of random draws.

Generate the training dataset by sampling 80% of the original data (without replacement) from the indices of our dataframe. training.data will receive 80% of the original dataset and testing.data the rest (-training.indices provides the remaining 20%)

set.seed(1)

Random.seed <- c("Mersenne-Twister", 1)

training.indices <- sample(1:nrow(auto.mpg), 0.8 \* nrow(auto.mpg), replace = FALSE)

training.data <- auto.mpg[training.indices, ]

View(training.data)

testing.data <- auto.mpg[-training.indices,]

View(testing.data)

Using lm(), run a simple linear regression model with one dimension/attribute/predictor. This model should predict the car’s *mpg* given its *weight* (mpg ~ weight). If weight changes, how does mpg change? Is weight a significant predictor for mpg? Use only the training dataset to train the model; then verify its predictive performance using the remaining testing data.

simple.model <- lm(mpg ~ weight, data = training.data)  
print(summary(simple.model))

##  
Call:

lm(formula = mpg ~ weight, data = training.data)

Residuals:

Min 1Q Median 3Q Max

-12.2394 -2.9167 -0.3989 2.4022 16.2517

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 46.7604729 0.9277116 50.40 <2e-16 \*\*\*

weight -0.0077783 0.0003012 -25.83 <2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.513 on 316 degrees of freedom

Multiple R-squared: 0.6785, Adjusted R-squared: 0.6775

F-statistic: 667 on 1 and 316 DF, p-value: < 2.2e-16

Note: Here, *weight* is a significant predictor, as denoted by “\*\*\*” in the Pr(>|t|) column. Usually, the predictor is interpreted as “statistically significant” when the value of these is less than 0.05, meaning that the *weight* can predict *mpg*. R-squared represents the proportion of the variance explained by the model; higher R-squared is better.

For our fitted model, the best fit line is expressed as:

mpg = 46.76 -0.0078 \* weight

This formula is derived from the coefficient estimates for (*Intercept*) and *weight* in the Estimate column. Linear regression attempts to estimate the mapping (Y ~ X) as a linear equation Y=B0+ B1X form (best fit line). B0 is the Intercept, 46.7604729, computed with a standard error of 0.93, and B1 is the coefficient associated with weight, given by -0.0077783.

**Step 3:** Calculate the RSE, R-squared, Adjusted R-squared, F-statistic **using formulas**

# degree of freedom = sample size minus the number of parameters being estimated

# we use 2 parameters in the simple model and have 318 obs. in training data

degree\_of\_freedom = 318 - 2

# Mean squared error: get the residual numbers from the model (the difference between the observed value and the predicted value by the model) and squared all of them and calculate the mean. The lower this is, the better (less error)

mse <- mean(residuals(simple.model)^2)

mse

[1] 20.23512

# Root mean squared error = square root of the mse

rmse <- sqrt(mse)

rmse

[1] 4.498346

# Residual sum of squares = calculate the sum of squared residuals;

rss <- sum(residuals(simple.model)^2)

rss

[1] 6434.768

# Residual standard error

# The residual standard error is the square root of the residual sum of # squares divided by the degrees of freedom

rse <- sqrt( rss / degree\_of\_freedom )

rse

[1] 4.512559

# R-squared: if R-squared is 0.8, it means 80% of the variation in the output variable is explained by the input variables; the higher the R squared, the more variation is explained by input variables, hence the better is the model.

# ssd = sum of squared deviation

temp = unlist(training.data["mpg"])

ssd = sum( (temp - mean(temp) )^2 )

r.squared <- (ssd - rss)/ssd

r.squared

[1] 0.6785269

# Adjusted R-square considers the number of predictors and penalizes you for adding variables which do not improve the existing model

p = 1 # number of predictors

n = 0.8\*nrow(auto.mpg)# number of of observations

adj.r.squared <- 1 - (1 - r.squared) \* (n - 1)/(n - p - 1)

adj.r.squared

[1] 0.6775109

# f statistic tell you if a group of variables are jointly statistically significant. Statistical significance is a measure of whether your research findings are meaningful.

F statistic = Mean of Squares for Model (msm) / Mean of Squares for Error (mse)

fstatistic = ( (ssd - rss)/p )/(rss/(n-p-1) )

fstatistic

[1] 667.8193

**Step 4:** Run the fitted simple linear model on **the testing data**set to check the prediction errors, ie. 20% of the original data.

#recall mpg = 46.76 -0.0078 \* weight from the training set and the output object called “simple.model”

simple.model.predictions <- predict(simple.model, testing.data)  
  
test.simple.model.ssl <- sum((testing.data$mpg - simple.model.predictions)^2)  
sprintf("SSL/SSR/SSE: %f", test.simple.model.ssl)

[1] "SSL/SSR/SSE: 1050.912019"

test.simple.model.mse <- test.simple.model.ssl / nrow(testing.data)  
sprintf("MSE: %f", test.simple.model.mse)

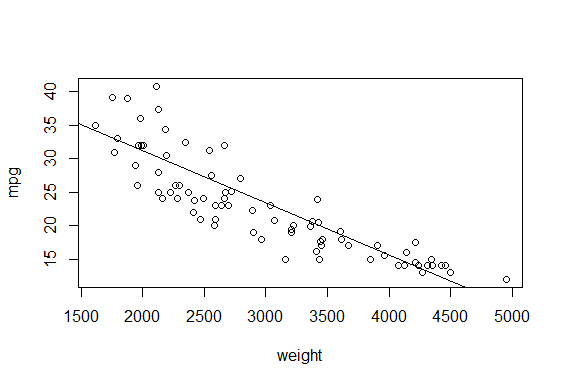
[1] "MSE: 13.136400"

test.simple.model.rmse <- sqrt(test.simple.model.mse)  
sprintf("RMSE: %f", test.simple.model.rmse)

[1] "RMSE: 3.624417"

Note: The following code plots the best fit line for the actual testing data (mpg ~ weight).

plot(mpg ~ weight, data=testing.data)  
abline(simple.model)



**PART B: Multiple Regression**

**Step 1:** Running the Model

Run a multiple linear regression model with more than two dimensions/attributes. Instead simply predicting the effect of *weight* on *mpg*, build a model which predicts *mpg* using all of the remaining attributes/features/predictors. Hence, add *cylinders*, *displacement*, *acceleration*, and *model.year*. Train on the same training set as the previous section (80% of the data)

multi.var.model <- lm(mpg ~ cylinders + displacement + weight + acceleration + model.year, data = training.data)  
print(summary(multi.var.model))

##

Call:

lm(formula = mpg ~ cylinders + displacement + weight + acceleration +

model.year, data = training.data)

Residuals:

Min 1Q Median 3Q Max

-8.4572 -2.4891 -0.1248 2.1045 14.0738

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -1.871e+01 4.792e+00 -3.904 0.000116 \*\*\*

cylinders -1.434e-01 4.028e-01 -0.356 0.722105

displacement 2.026e-03 8.483e-03 0.239 0.811396

weight -6.510e-03 6.726e-04 -9.679 < 2e-16 \*\*\*

acceleration 1.302e-01 9.010e-02 1.445 0.149432

model.year 7.880e-01 5.789e-02 13.611 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.522 on 312 degrees of freedom

Multiple R-squared: 0.8066, Adjusted R-squared: 0.8035

F-statistic: 260.2 on 5 and 312 DF, p-value: < 2.2e-16

Note: significant predictors are weight and model.year

Calculate the RSE, R-squared, Adjusted R-squared, F-statistic using formula

# we are now using SIX parameters on 318 observations

degree\_of\_freedom = 318 - 6

mse <- mean(residuals(multi.var.model)^2)

mse

[1] 12.17358

rmse <- sqrt(mse)

rmse

[1] 3.489065

rss <- sum(residuals(multi.var.model)^2)

rss

[1] 3871.198

rse <- sqrt( rss / degree\_of\_freedom )

rse

[1] 3.522454

temp = unlist(training.data["mpg"])

ssd = sum( (temp - mean(temp) )^2 )

r.squared <- (ssd - rss)/ssd

r.squared

[1] 0.8065997

p = 5 # number of predictors

n = 0.8\*nrow(auto.mpg)# number of of observations

adj.r.squared <- 1 - (1 - r.squared) \* (n - 1)/(n - p - 1)

adj.r.squared

[1] 0.8035044

fstatistic = ( (ssd - rss)/p )/(rss/(n-p-1) )

fstatistic

[1] 260.5806

**Step 2:** Run the fitted multiple linear model on the testing data (20% of the data) to check the prediction errors.

multi.var.predictions <- predict(multi.var.model, testing.data)  
  
test.multi.var.ssl <- sum((testing.data$mpg - multi.var.predictions)^2)  
sprintf("SSL/SSR/SSE: %f", test.multi.var.ssl)

[1] "SSL/SSR/SSE: 802.877376"

test.multi.var.mse <- test.multi.var.ssl / nrow(testing.data)  
sprintf("MSE: %f", test.multi.var.mse)

[1] "MSE: 10.035967"

test.multi.var.rmse <- sqrt(test.multi.var.mse)  
sprintf("RMSE: %f", test.multi.var.rmse)

[1] "RMSE: 3.167959"

Note: Compare the multiple regression MSE to the simple linear model MSE. Multiple regression model yields lower MSE and is therefore superior.

**Step 3:** Analyzing the full dataset

With multiple predictors, draw a scatter plot for each variable against the response Y, along with the line of best fit. Use the testing dataset to check the model and the best fit line. This can be done in two different ways:

1. Use scatter.smooth(x= testing.data$weight, y= testing.data$mpg, main="mpg ~ weight") and modify the variable to plot the other predictors besides weight.

Or

1. Use plot() and abline() as shown above for each predictor against Y.

Chart, scatter chart

Description automatically generated

Chart, scatter chart

Description automatically generated

Then, apply the full model to the entire dataset.

full.model <- lm(mpg ~ cylinders + displacement + weight + acceleration + model.year, data = auto.mpg)  
print(summary(full.model))

##   
Call:

lm(formula = mpg ~ cylinders + displacement + weight + acceleration +

model.year, data = auto.mpg)

Residuals:

Min 1Q Median 3Q Max

-8.6747 -2.3625 -0.1178 2.0375 14.3300

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -1.457e+01 4.138e+00 -3.521 0.00048 \*\*\*

cylinders -2.586e-01 3.286e-01 -0.787 0.43177

displacement 7.268e-03 7.146e-03 1.017 0.30977

weight -6.926e-03 5.963e-04 -11.614 < 2e-16 \*\*\*

acceleration 8.035e-02 7.839e-02 1.025 0.30604

model.year 7.553e-01 5.078e-02 14.875 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.44 on 392 degrees of freedom

Multiple R-squared: 0.8087, Adjusted R-squared: 0.8062

F-statistic: 331.4 on 5 and 392 DF, p-value: < 2.2e-16

Finally, report the parsimonious model: apply a model with only the significant predictors (*weight* and *model.year*). This is the “parsimonious” model.

signif.model <- lm(mpg ~ weight + model.year, data = training.data)

print(summary(signif.model))

##

Call:

lm(formula = mpg ~ weight + model.year, data = training.data)

Residuals:

Min 1Q Median 3Q Max

-8.6946 -2.5276 -0.2144 2.1973 14.2044

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -1.767e+01 4.579e+00 -3.858 0.000139 \*\*\*

weight -6.649e-03 2.552e-04 -26.054 < 2e-16 \*\*\*

model.year 8.011e-01 5.642e-02 14.199 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.592 on 315 degrees of freedom

Multiple R-squared: 0.7945, Adjusted R-squared: 0.7932

F-statistic: 609 on 2 and 315 DF, p-value: < 2.2e-16

mse <- mean(residuals(signif.model)^2)

mse

[1] 12.77835

rmse <- sqrt(mse)

rmse

[1] 3.574682

rss <- sum(residuals(signif.model)^2)

rss

[1] 4063.516

# Run the fitted Parsimonious model on the testing data to check the prediction errors, ie. 20% of the original data

signif.predictions <- predict(signif.model, testing.data)

test.signif.ssl <- sum((testing.data$mpg - signif.predictions)^2)

sprintf("SSL/SSR/SSE: %f", test.signif.ssl)

[1] "SSL/SSR/SSE: 607.186550"

test.signif.mse <- test.signif.ssl / nrow(testing.data)

sprintf("MSE: %f", test.signif.mse)

[1] "MSE: 7.589832"

test.signif.rmse <- sqrt(test.signif.mse)

sprintf("RMSE: %f", test.signif.rmse)

[1] "RMSE: 2.754965"

# Obtain results from training on full dataset

full.signif.model <- lm(mpg ~ weight + model.year, data = auto.mpg)

print(summary(full.signif.model))

##

Call:

lm(formula = mpg ~ weight + model.year, data = auto.mpg)

Residuals:

Min 1Q Median 3Q Max

-8.8777 -2.3140 -0.1211 2.0591 14.3330

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -1.420e+01 3.968e+00 -3.578 0.000389 \*\*\*

weight -6.664e-03 2.139e-04 -31.161 < 2e-16 \*\*\*

model.year 7.566e-01 4.898e-02 15.447 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.435 on 395 degrees of freedom

Multiple R-squared: 0.8079, Adjusted R-squared: 0.8069

F-statistic: 830.4 on 2 and 395 DF, p-value: < 2.2e-16

mse <- mean(residuals(full.signif.model)^2)

mse

[1] 11.70814

rmse <- sqrt(mse)

rmse

[1] 3.421715

rss <- sum(residuals(full.signif.model)^2)

rss

[1] 4659.838

Summary table comparing statistics between simple and multiple regression for training, testing, and full dataset:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Statistic | Simple Linear Regression | | | Multiple Regression | | | Multiple Regression  (only significant variables) | | |
| Train | Test | All | Train | Test | All | Train | Test | All |
| MSE | 20.2 | 13.1 | 18.8 | 12.2 | 10.0 | 11.7 | 12.8 | 7.59 | 11.7 |
| RMSE | 4.50 | 3.62 | 4.33 | 3.48 | 3.17 | 3.41 | 3.57 | 2.75 | 3.42 |
| RSS | 6434.8 | 1050.9 | 7474.8 | 3871.2 | 802.9 | 4639.8 | 4063.5 | 607.2 | 4659.8 |
| RSE | 4.51 | - | 4.35 | 3.52 | - | 3.44 | 3.592 | - | 3.44 |
| R^2 | 0.679 | - | 0.692 | 0.807 | - | 0.809 | 0.795 | - | 0.808 |
| Adj. R^2 | 0.678 | - | 0.691 | 0.804 | - | 0.806 | 0.793 | - | 0.807 |
| F-Statistic | 667.8 | - | 888.9 | 260.6 | - | 331.4 | 609.0 | - | 830.4 |