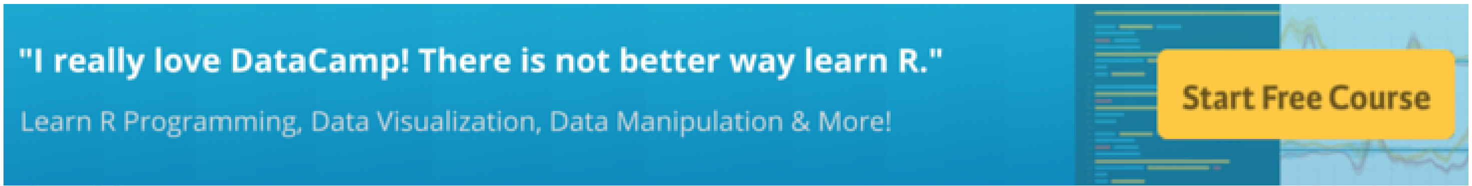


Learn R programming for data science

- [Home]
- [About Us]
- [Archives]
- [Contribute]
- [Free Account]
- Search | Search

We share R tutorials from scientists at academic and scientific institutions with a goal to give everyone in the world access to a free knowledge. Our tutorials cover different topics including statistics, data manipulation and visualization!

Introduction Getting Data Data Management Visualizing Data Basic Statistics Regression Models Advanced Modeling Programming Best R Packages Tips & Tricks

Regression Models

Using Linear Regression to Predict Energy Output of a Power Plant

by Teja Kodali on September 29, 2015 5 Comments

In this article, I will show you how to fit a linear regression to predict the energy output at a Combined Cycle Power Plant(CCPP). The dataset is obtained from the UCI Machine Learning Repository. The dataset contains five columns, namely, Ambient Temperature (AT), Ambient Pressure (AP), Relative Humidity (RH), Exhaust Vacuum (EV), and net hourly electrical energy output (PE) of the plant. The first four are the attributes, and are used to predict the output, PE.

Reading in the data and splitting

Since the data is in xlsx format, we will need the `xlsx` package. We will use the data from the first sheet of the file.

```
library(xlsx)
powerData <- read.xlsx('Folds5x2_pp.xlsx', 1)

head(powerData)
```

```
     AT     V      AP    RH     PE
1 14.96 41.76 1024.07 73.17 463.26
2 25.18 62.96 1020.04 59.08 444.37
3  5.11 39.40 1012.16 92.14 488.56
4 20.86 57.32 1010.24 76.64 446.48
5 10.82 37.50 1009.23 96.62 473.90
6 26.27 59.44 1012.23 58.77 443.67
```

Next, we need to split the data into a training set and a testing set. As their names imply, the training set is used to train and build the model, and then this model is tested on the testing set. Let's say we want to have about 75% of the data in the training set and 25% of the data in the testing set. It can be done as follows:

```
set.seed(123)
split <- sample(nrow(powerData), size = floor(0.75 * nrow(powerData)))
trainData <- powerData[split, ]
testData <- powerData[-split, ]
head(trainData)
```

```
        AT     V      AP    RH     PE
2752 29.14 67.45 1015.51 46.47 433.34
7542 24.67 70.94 1007.99 75.64 443.51
3913 20.84 51.19 1008.63 84.11 448.98
8447 31.73 74.67 1016.38 44.51 425.34
8995  4.44 38.44 1016.14 75.35 486.53
436   9.43 37.14 1013.03 74.99 473.57
```

```
head(testData)
```

```
      AT     V      AP    RH     PE
2  25.18 62.96 1020.04 59.08 444.37
4  20.86 57.32 1010.24 76.64 446.48
12 20.14 46.93 1014.66 64.22 453.99
13 24.34 73.50 1011.31 84.15 440.29
14 25.71 58.59 1012.77 61.83 451.28
17 18.21 45.00 1022.86 48.84 467.54
```

Let me explain what the above commands do.

- First, we set the seed so that the results are reproducible.
- Then, we create a sequence whose length is equal to the number of rows of the dataset. These numbers act as the indices of the dataset. We randomly select 75% of the numbers in the sequence and store it in the variable `split`.
- Lastly, we copy all the rows which correspond to the indices in split into `trainData` and all the remaining rows into `testData`.

Building the prediction model

Now, let's build the prediction model. We will use the `lm` function for this.

```
predictionModel <- lm(PE ~ AT + V + AP + RH, data = trainData)
```

The above function will try to predict PE based on the variables AT, V, AP, and RH. Since we are using all the variables present in the dataset, a shorter way to write the above command is: (this is very helpful when are are a large number of variables.)

```
predictionModel <- lm(PE ~ ., data = trainData)
```

We can now look at the summary of the model.

```
summary(predictionModel)
```

```
Call:
lm(formula = PE ~ ., data = trainData)

Residuals:
    Min      1Q  Median      3Q     Max
-43.363  -3.148  -0.140   3.162  17.763
```

```
Coefficients:
              Estimate Std. Error  t value Pr(>|t|)    
(Intercept) 450.764756  11.331102   39.781  < 2e-16 ***
AT           -1.978787   0.017599 -112.435  < 2e-16 ***
V            -0.232049   0.008415  -27.575  < 2e-16 ***
AP            0.065590   0.010993    5.967 2.54e-09 ***
RH           -0.155561   0.004829  -32.214  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.574 on 7171 degrees of freedom
Multiple R-squared:  0.9284,	Adjusted R-squared:  0.9284 
F-statistic: 2.326e+04 on 4 and 7171 DF,  p-value: < 2.2e-16
```

This will help us determine which variables to include in the model. A linear regression can be represented by the equation: **$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \cdots + \varepsilon$** where **$y_i$** represents the outcome we’re trying to predict (PE), **x_i** represent the various attributes (AT, V, AP, and RH), **β** represent their coefficients, and **ε** represents the constant term.
The first column in the summary, namely *Estimate* gives us these values. The first value corresponds to **ε**, and the rest of the values correspond to the various **β** values. If the coefficient for a particular attribute is 0 or close to 0, that means it has very little to no effect on the prediction, and hence, can be removed. The *standard error* column gives an estimation of how much the coefficients may vary from the estimate values. The **t** value is calculated by dividing the estimate by the standard error column. The last column gives a measure of how likely it is that the coefficient is 0 and is inversely proportional to the **t** value column. Hence, an attribute with a high absolute value of **t**, or a very low absolute value of **Pr(>|t|)** is desirable.

The easiest way to determine which variables are significant is by looking at the stars next to them. The scheme is explained at the bottom of the table. Variables with three stars are most significant, followed by two stars, and finally one. Variables with a period next to them may or may not be significant and are generally not included in prediction models, and variables with nothing next to them are not significant.

In our model, we can see that all our variables are highly significant, so we will leave our prediction model as it is. In case you are dealing with a dataset in which there are one or more variables which are non-significant, it is advisable to test the model by removing one variable at a time. This is because when two variables are highly correlated with each other, they may be non-significant to the model, but when one of them is removed, the other could become significant. This is due to multicollinearity. You can find out more about multicollinearity here.

The easiest way to check the accuracy of a model is by looking at the R-squared value. The summary provides two R-squared values, namely Multiple R-squared, and Adjusted R-squared. The Multiple R-squared is calculated as follows:

Multiple R-squared = 1 – SSE/SST where:

- SSE is the sum of square of residuals. Residual is the difference between the predicted value and the actual value, and can be accessed by `predictionModel$residuals`.
- SST is the total sum of squares. It is calculated by summing the squares of difference between the actual value and the mean value.

For example, lets say that we have 5, 6, 7, and 8, and a model predicts the outcomes as 4.5, 6.3, 7.2, and 7.9. Then, SSE can be calculated as: SSE = (5 – 4.5) ^ 2 + (6 – 6.3) ^ 2 + (7 – 7.2) ^ 2 + (8 – 7.9) ^ 2; and SST can be calculated as: mean = (5 + 6 + 7 + 8) / 4 = 6.5; SST = (5 – 6.5) ^ 2 + (6 – 6.5) ^ 2 + (7 – 6.5) ^ 2 + (8 – 6.5) ^ 2

The Adjusted R-squared value is similar to the Multiple R-squared value, but it accounts for the number of variables. This means that the Multiple R-squared will always increase when a new variable is added to the prediction model, but if the variable is a non-significant one, the Adjusted R-squared value will decrease. For more info, refer here.

An R-squared value of 1 means that it is a perfect prediction model, and an R-squared value of 0 means that it is of no improvement over the baseline model (the baseline model just predicts the output to always be equal to the mean). From the summary, we can see that our R-squared value is 0.9284, which is very high.

Testing the prediction model

We will now apply the prediction model to the test data.

```
prediction <- predict(predictionModel, newdata = testData)
```

Now, let’s look at the first few values of prediction, and compare it to the values of PE in the test data set.

```
head(prediction)
```

```
       2        4       12       13       14       17 
444.0433 450.5260 456.5837 438.7872 443.1039 463.7809
```

```
head(testData$PE)
```

```
[1] 444.37 446.48 453.99 440.29 451.28 467.54
```

This means that for the value of 444.37, our prediction was 444.0433, for 446.48, it was 450.5260, and so on.

We can calculate the value of R-squared for the prediction model on the test data set as follows:

```
SSE <- sum((testData$PE - prediction) ^ 2)
SST <- sum((testData$PE - mean(testData$PE)) ^ 2)
1 - SSE/SST
```

```
0.9294734
```

That brings us to the end of this article. I hope you enjoyed it, and found it valuable! If you have any questions, feel free to leave a comment or reach out to me on Twitter.

Tags Linear Regression

The Author

Teja is a writer for DataScience+, studying Master of Science in Technical Entrepreneurship and Management at the University of Rochester, NY. He enjoys working with data and building visualisations. In his spare time, Teja enjoys playing sports, running, and watching Silicon Valley.

LinkedIn Twitter

Disclosure

- Teja Kodali does not work or receive funding from any company or organization that would benefit from this article.

70
Shares

Like this article? Give it a share:

Facebook Twitter Google+ Linkedin Email this

5 Comments DataScience+ Hub 1 Login

Recommend Share Sort by Best

Join the discussion…

Alex M • a month ago

Hi Teja,

Thank you for writing this article. I found it very helpful in learning linear regression in R.

I am still very new to using statistical methods to analyze data and am seeking understanding of different methods of approaching problems.

I have a couple of questions on the following sections of the articles (denoted by quotation marks):

1.
"split <- sample(seq_len(nrow(powerData)), size = floor(0.75 * nrow(powerData)))"

Why is seq_len() being used here?

I found that just using the nrow() function appears to be sufficient by itself but maybe I am overlooking something.

2.
"We can calculate the value of R-squared for the prediction model on the test data set as follows:

see more

Reply • Share ›

Teja K Alex M • a month ago

Hello Alex, I'm glad you enjoyed the article.
1. You are correct, nrow is sufficient, I will fix that.
2. When we build the model on the training data set, it calculates the R^2 and displays it automatically. While this is a decent indicator of how good the model is, it is important to test the model on data that it has not seen before. If the model does not perform well on unseen data, then it is not very useful and cannot be used for predictions. This is why we also calculate the R^2 on test data. This basically is a measure of how closely the predicted data reflects the actual values in the test data. I hope that answers your question!

• Reply • Share ›

Alex M Teja K • a month ago

Okay, understood. Thanks!

• Reply • Share ›

Alan • 6 months ago

Really awesome article, thanks for sharing. Question - did you do anything to verify that the independent variables are linearly related to the dependent variable before fitting the model? Or did you go about fitting the model to test whether linear regression is the appropriate method?

• Reply • Share ›

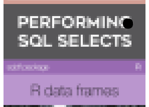
Teja K Alan • 6 months ago

Hello Alan, thank you for the kind words. You bring up a good point. I actually did not do anything to verify that the independent variables are linearly related to the dependent variable. I tried fitting the model first, and felt that the results were good enough for the purpose of this article. I hope that answers your question!

• Reply • Share ›

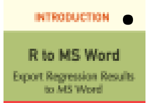
Trending Now on DataScience+

- Developing an R Tutorial shiny dashboard app



- Performing SQL selects on R data frames

- Learn R by Intensive Practice



- How to export Regression results from R to MS Word

- K Means Clustering in R

DataScience+Learn R programming for data science

>_ Site Links

- About Us
- Contribute
- Advertise
- Contact Us

>_ Legal

- Privacy Policy
- Terms of Use
- Account Terms
- Stylebook

>_ Other Sites

- R Bloggers

Connect with Us