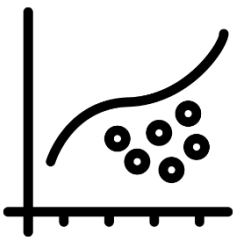
Montogue



Tutorial ST1
Linear and Nonlinear Regression with R

Lucas Monteiro Nogueira

• Summary •	
Problem 1	Simple linear regression
Problem 2	Multiple linear regression – Two predictors
Problem 3	Multiple linear regression – More than two predictors
Problem 4	Nonlinear regression

PROBLEMS

▶ Datasets

Download datasets *bostonData.xlsx* and *algae.txt* in our Google Drive folder.

▶ Problem 1 – Simple linear regression

Problems 1 to 3 use the **`Boston`** dataset in the **`MASS`** package. This dataset contains estimated housing values for the suburbs of Boston and 13 measurements that may be related to housing value, including average number of rooms, crime rate, and concentration of nitrogen oxides (a proxy for atmospheric pollution). The variables are summarized below.

Symbol	Definition
MEDV	Median value of owner-occupied homes in $1000s
CRIM	Per capita crime rate by town
ZN	Proportion of a town's residential land zoned for lots greater than 25,000 square feet
INDUS	Proportion of nonretail business acres per town
CHAS	Charles River dummy variable with value 1 if tract bounds on Charles River or 0 otherwise
NOX	Nitrogen oxides concentration (parts per 10 million)
RM	Average number of rooms per dwelling
AGE	Proportion of owner-occupied units built prior to 1940
DIS	Weighted distances to five employment centers in the Boston region
RAD	Index of accessibility to radial highways
TAX	Full-value property-tax rate (per $10,000)
PTRATIO	Pupil-teacher ratio by town
BLACK	$(Bk - 0.63)^2$, where Bk is the proportion of blacks in the population
LSTAT	Proportion of the population that is lower status (percent)

Let's begin by fitting **`medv`** to **`lstat`**, the proportion of lower-status population in a neighborhood. The command to use is **`lm`**:

```
> model1 = lm(medv~lstat, data=Boston)
> summary(model1)
Call:
lm(formula = medv ~ lstat, data = Boston)
Residuals:
    Min      1Q  Median      3Q     Max
-15.168  -3.990  -1.318   2.034  24.500
Coefficients:
             Estimate  Std. Error t value  Pr(>|t|)
(Intercept) 34.55384    0.56263   61.41   <2e-16 ***
lstat       -0.95005    0.03873  -24.53   <2e-16 ***
```


© 2022 Montogue Quiz

```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1
‘ ’ 1
Residual standard error: 6.216 on 504 degrees of freedom
Multiple R-squared:  0.5441,    Adjusted R-squared:  0.5432
F-statistic: 601.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

The estimated coefficients are highlighted in red. Another way to recover the coefficients is to type

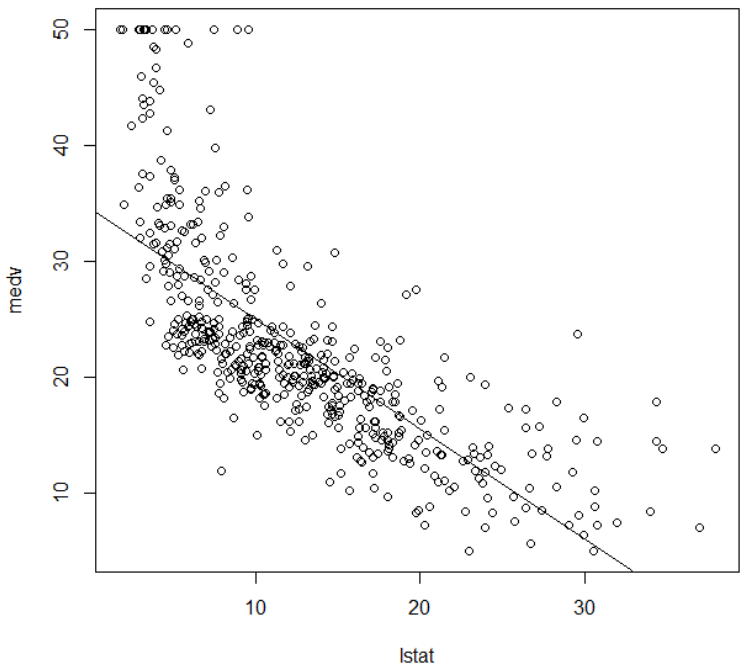
```
> model1$coefficients
(Intercept)       lstat
34.5538409  -0.9500494
```

Clearly, the median value of owner-occupied homes, **medv**, is related to the lower status of the population, **lstat**, by an equation of the form

$$\text{medv} = 34.554 - 0.950 \times \text{lstat}$$

This equation indicates that the median value of a housing unit decreases by $950 for every additional percent point of lower-status population, all else being fixed. One obvious next step is to plot the data, which can be done via the **plot** command, along with the line described by the fit above, which can be done via the **abline** command:

```
> plot(lstat,medv)
> abline(model1)
```



We can compute confidence intervals for the estimated coefficients with the **confint** command:

```
> confint(model1)
                2.5 %     97.5 %
(Intercept) 33.448457 35.6592247
lstat       -1.026148 -0.8739505
```

Note that R returns the 95% CI by default. To produce a custom CI range, add a **level** argument to the command:

```
> confint(model1, level=0.99)
                0.5 %     99.5 %
(Intercept) 33.099101 36.0085810
lstat       -1.050199 -0.8498995
```

The **predict** function can be used to yield confidence intervals and prediction intervals for the prediction of **medv** that corresponds to a given value of **lsat**. The following code yields the predicted housing costs for four values of **lsat**, along with the corresponding 95% confidence intervals:

```
>
predict(model1,data.frame(lstat=(c(2.5,5,7.5,10))),interval
="confidence")
      fit      lwr      upr
1 32.17872 31.23442 33.12301
```


© 2022 Montogue Quiz

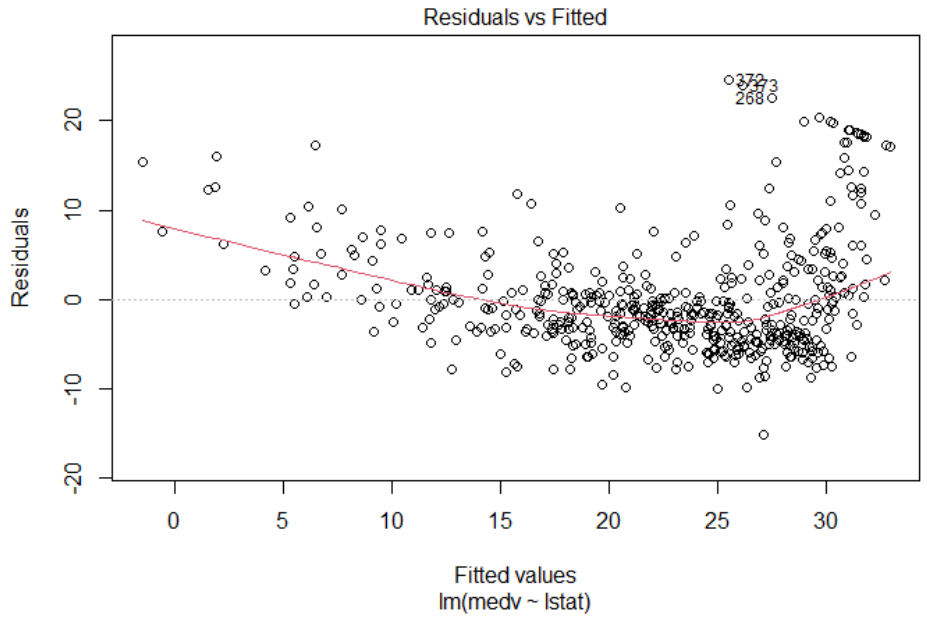
```
2 29.80359 29.00741 30.59978
3 27.42847 26.75877 28.09818
4 25.05335 24.47413 25.63256
```

The results predict a cost of approximately $29,804 for a home in which the low-status population percentage is 5%. The corresponding 95% CI is ($29,007; $30,600).

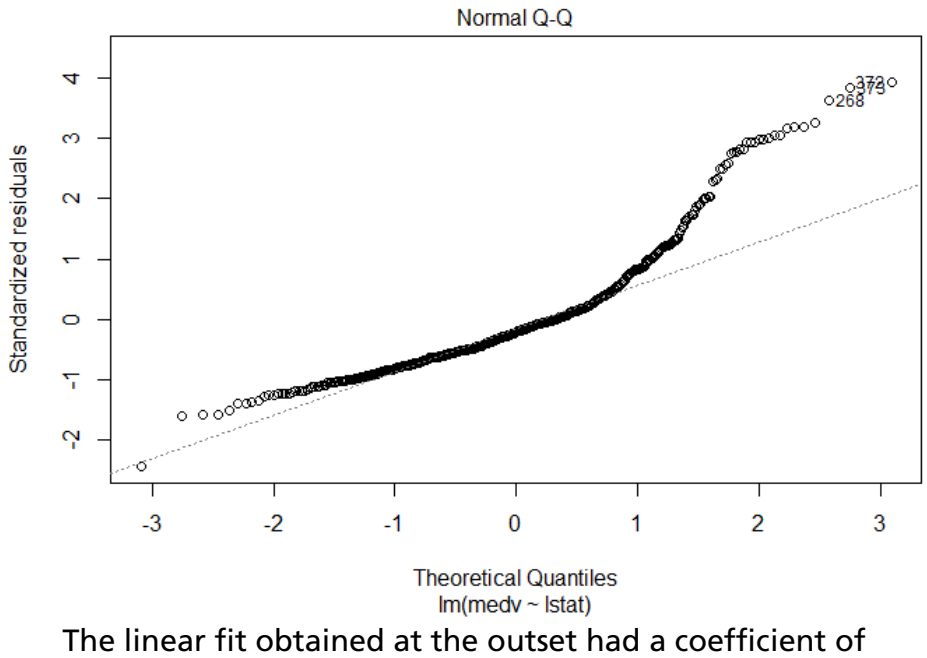
The command **`plot()`** can be used to generate simple diagnostic plots for the linear model at hand. **`plot`** normally generates four diagnostic plots, but, for a simple linear regression model, only the first two are particularly relevant:

```
> Plot(model1,which=1:2)
```

The first diagnostic graph is the residual plot. Ideally, residuals should be randomly distributed and more or less symmetric about the horizontal line at the origin. This is not the case here. The slight "U" shape of the residuals indicates that a quadratic or higher-order term may be missing from the fit.



The second diagnostic plot is the normal probability plot with the residuals on the vertical axis and the normal quantiles on the horizontal axis. The dotted diagonal line has the standard deviation s of the model for its slope. When the residuals are approximately normal, the points will be close to this diagonal line. The model at hand deviates substantially from the reference line, indicating that errors are not all normally distributed.



The linear fit obtained at the outset had a coefficient of determination of 0.5441, which indicates that about 54% of the total variation of the response variable (housing price) about its mean is explained by the linear association between the predictor variables. This is somewhat poor. Note that the data plot provided earlier indeed indicates a significant deviation from the linear trend we've assumed thus far, especially at low values of **`lstat`**. One way to improve the fit is

© 2022 Montogue Quiz

to introduce a transformation of the predictor variable by including a quadratic term:

$$\text{medv}\left(\text{lstat}\right) = \beta_0 + \beta_1 \times \text{lsat} + \beta_2 \times \text{lsat}^2$$

Fitting to this model with **`lm`** is similar to fitting an ordinary linear model, the difference being that we must include a term for **`lsat`**^{2} wrapped around the **`I()`** function:

```
> model2 = lm(medv~lstat+I(lstat^2),data=Boston)
> summary(model2)
Call:
lm(formula = medv ~ lstat + I(lstat^2), data = Boston)
Residuals:
     Min       1Q   Median       3Q      Max
-15.2834  -3.8313  -0.5295   2.3095  25.4148
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 42.862007   0.872084   49.15   <2e-16 ***
lstat       -2.332821   0.123803  -18.84   <2e-16 ***
I(lstat^2)   0.043547   0.003745   11.63   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1
‘ ’ 1
Residual standard error: 5.524 on 503 degrees of freedom
Multiple R-squared:  0.6407,    Adjusted R-squared:  0.6393
F-statistic: 448.5 on 2 and 503 DF,  p-value: < 2.2e-16
```

Note that the new model has a higher *R*-squared, and that the `I(lstat^2)` term is associated with a low *p*-value, which means that it is a significant component. A more formal way to assess whether an added predictor is significant is to use the **`anova()`** function:

```
> anova(model1,model2)
Analysis of Variance Table
Model 1: medv ~ lstat
Model 2: medv ~ lstat + I(lstat^2)
Res.Df RSS Df    Sum of Sq  F    Pr(>F)
1      504 19472
2      503 15347  1    4125.1 135.2 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1
‘ ’ 1
```

▶ Problem 2 – Multiple linear regression – Two predictors

As a first example of multiple linear regression, we attempt to relate **`medv`** to predictors **`lstat`**, which is the percentage of low-status population, and **`age`**, which is the proportion of owner-occupied units built prior to 1940. The syntax is straightforward:

```
> model3 <- lm(medv~lstat+age, data=Boston)
> summary(model3)
Call:
lm(formula = medv ~ lstat + age, data = Boston)

Residuals:
    Min      1Q  Median      3Q     Max
-15.981  -3.978  -1.283   1.968  23.158

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 33.22276    0.73085  45.458  < 2e-16 ***
lstat       -1.03207    0.04819 -21.416  < 2e-16 ***
age          0.03454    0.01223   2.826  0.00491 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1
‘ ’ 1
Residual standard error: 6.173 on 503 degrees of freedom
Multiple R-squared:  0.5513,    Adjusted R-squared:  0.5495
F-statistic:   309 on 2 and 503 DF,  p-value: < 2.2e-16
```


© 2022 Montogue Quiz

Note that the **age** predictor is associated with a low *p*-value (albeit greater than that of **lstat**), and hence constitutes a significant predictor in the ensuing model.

Importantly, **lstat** and **age** may not be independent predictors, and including a term $lsat \times age$ may yield an improved model. To introduce such a term in the model, we may either type

```
> lm(medv~lstat+age+lstat:age, data=Boston)
```

or, equivalently,

```
> lm(medv~lstat*age, data=Boston)
```

The latter is a viable alternative because **lstat*age** is a shorthand for **lstat+age+lstat:age** in the so-called Wilkinson notation. Let us summarize the statistics of the new model:

```
> model4 <- lm(medv~lstat+age+lstat:age, data=Boston)
> summary(model4)
Call:
lm(formula = medv ~ lstat + age + lstat:age, data = Boston)
Residuals:
    Min      1Q  Median      3Q     Max
-15.806  -4.045  -1.333   2.085  27.552
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 36.0885359  1.4698355  24.553  < 2e-16 ***
lstat       -1.3921168  0.1674555  -8.313 8.78e-16 ***
age         -0.0007209  0.0198792  -0.036   0.9711
lstat:age    0.0041560  0.0018518   2.244   0.0252 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
' ' 1
Residual standard error: 6.149 on 502 degrees of freedom
Multiple R-squared:  0.5557,    Adjusted R-squared:  0.5531
F-statistic: 209.3 on 3 and 502 DF, p-value: < 2.2e-16
```

Notice that, in the model with interactions, predictor **age** is not significant. Thus, we can generate a new model without the age term (consider also the **update** command):

```
> model4upd <- lm(medv~lstat+lstat:age,data=Boston)
> summary(model4upd)
Call:
lm(formula = medv ~ lstat + lstat:age, data = Boston)
Residuals:
    Min      1Q  Median      3Q     Max
-15.815  -4.039  -1.335   2.086  27.491
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 36.041514   0.691334  52.133  < 2e-16 ***
lstat       -1.388161   0.126911 -10.938  < 2e-16 ***
lstat:age    0.004103   0.001133   3.621 0.000324 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
' ' 1
Residual standard error: 6.142 on 503 degrees of freedom
Multiple R-squared:  0.5557,    Adjusted R-squared:  0.554
F-statistic: 314.6 on 2 and 503 DF,  p-value: < 2.2e-16
```

▶ Problem 3 – Multiple linear regression – More than two predictors

Let's take the analysis one step further and generate a linear model with all 13 predictors in the **Boston** dataset. Instead of typing all predictors one by one, we can simply add a dot (".") after the tilde in the **lm** command:

```
> model5 <- lm(medv~.,data=Boston)
> summary(model5)
Call:
lm(formula = medv ~ ., data = Boston)
Residuals:
    Min      1Q  Median      3Q     Max
-15.595  -2.730  -0.518   1.777  26.199
```


© 2022 Montogue Quiz

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
crim        -1.080e-01  3.286e-02  -3.287 0.001087 **
zn           4.642e-02  1.373e-02   3.382 0.000778 ***
indus        2.056e-02  6.150e-02   0.334 0.738288
chas         2.687e+00  8.616e-01   3.118 0.001925 **
nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
age          6.922e-04  1.321e-02   0.052 0.958229
dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
tax         -1.233e-02  3.760e-03  -3.280 0.001112 **
ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
black        9.312e-03  2.686e-03   3.467 0.000573 ***
lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
' ' 1
Residual standard error: 4.745 on 492 degrees of freedom
Multiple R-squared:  0.7406,    Adjusted R-squared:  0.7338
F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

Clearly, **indus** and **age** have high *p*-values and hence constitute potential candidates for removal.

Collinearity, also called multicollinearity, is a condition where the model's predictor variables are highly intercorrelated. A consequence of this situation is the inability to estimate the model's regression coefficients with acceptable precision. One simple approach to assess collinearity is to compute the *variance inflation factor*, *VIF*, for each predictor variable. Collinearity is a potential problem if a given predictor's *VIF* is greater than about 5. The **vif()** function in the **HH** package can be used to compute *VIF*s for each predictor in a linear model. In the model at hand:

```
> vif(model5)
crim       zn      indus     chas      nox       rm
1.792192 2.298758 3.991596 1.073995 4.393720 1.933744
age
3.100826
dis        rad       tax   ptratio    black    lstat
3.955945 7.484496 9.008554 1.799084 1.348521 2.941491
```

Notice that **rad** and **tax** both have *VIF*'s greater than 5. Also, **indus** has a high *VIF*, which, combined with the fact that the **indus** coefficient has a high *p*-value, encourages us to eliminate this variable:

```
> model5 <- lm(medv~.-indus,data=Boston)
> summary(model5)
Call:
lm(formula = medv ~ . - indus, data = Boston)
Residuals:
    Min      1Q  Median      3Q     Max
-15.587  -2.737  -0.506   1.742  26.212
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.636e+01  5.091e+00   7.143 3.30e-12 ***
crim        -1.084e-01  3.281e-02  -3.304 0.001022 **
zn           4.593e-02  1.364e-02   3.368 0.000816 ***
chas         2.716e+00  8.562e-01   3.173 0.001605 **
nox         -1.743e+01  3.681e+00  -4.735 2.87e-06 ***
rm           3.797e+00  4.158e-01   9.132  < 2e-16 ***
age          6.971e-04  1.320e-02   0.053 0.957898
dis         -1.490e+00  1.948e-01  -7.648 1.08e-13 ***
rad          2.999e-01  6.367e-02   4.710 3.22e-06 ***
tax         -1.178e-02  3.378e-03  -3.489 0.000529 ***
ptratio     -9.471e-01  1.296e-01  -7.308 1.10e-12 ***
black        9.282e-03  2.682e-03   3.461 0.000586 ***
lstat       -5.235e-01  5.052e-02 -10.361  < 2e-16 ***
---
```


© 2022 Montogue Quiz

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1
‘ ’ 1
Residual standard error: 4.741 on 493 degrees of freedom
Multiple R-squared:  0.7406,    Adjusted R-squared:  0.7343
F-statistic: 117.3 on 12 and 493 DF,  p-value: < 2.2e-16
```

Notice that the new model has essentially the same *R*-squared as the previous model, indicating that **`indus`** may in fact be indifferent for the situation at hand. Let's compute the *VIF*s a second time:

```
> vif(model5)
 crim       zn     chas      nox       rm      age
1.789724 2.272761 1.062600 4.087564 1.917402 3.100822
  dis
3.779656
  rad      tax    ptratio
6.904480 7.280729 1.768721
 black    lstat
1.347029 2.924416
```

A pair of variables that merit removal is **`age`**, which has a high *p*-value, or **`tax`**, which has a high VIF. We thus construct two new models, namely **`model5alt1`**, which has no **`age`** variable, and **`model5alt2`**, which has no **`tax`** variable.

```
> model5alt1 <- lm(medv~.-indus-age,data=Boston)
> summary(model5)
Call:
lm(formula = medv ~ . - indus - age, data = Boston)
Residuals:
     Min       1Q   Median       3Q      Max
-15.5984  -2.7386  -0.5046   1.7273  26.2373
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  36.341145   5.067492   7.171 2.73e-12 ***
crim         -0.108413   0.032779  -3.307 0.001010 **
zn            0.045845   0.013523   3.390 0.000754 ***
chas          2.718716   0.854240   3.183 0.001551 **
nox         -17.376023   3.535243  -4.915 1.21e-06 ***
rm            3.801579   0.406316   9.356  < 2e-16 ***
dis          -1.492711   0.185731  -8.037 6.84e-15 ***
rad           0.299608   0.063402   4.726 3.00e-06 ***
tax          -0.011778   0.003372  -3.493 0.000521 ***
ptratio      -0.946525   0.129066  -7.334 9.24e-13 ***
black         0.009291   0.002674   3.475 0.000557 ***
lstat        -0.522553   0.047424 -11.019  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1
‘ ’ 1
Residual standard error: 4.736 on 494 degrees of freedom
Multiple R-squared:  0.7406,    Adjusted R-squared:  0.7348
F-statistic: 128.2 on 11 and 494 DF,  p-value: < 2.2e-16
> model5alt2 <- lm(medv~.-indus-tax,data=Boston)
> summary(model5)
Call:
lm(formula = medv ~ . - indus - tax, data = Boston)
Residuals:
    Min      1Q  Median      3Q     Max
-16.274  -2.980  -0.497   1.806  26.279
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.469e+01  5.125e+00   6.768 3.72e-11 ***
crim        -1.048e-01  3.317e-02  -3.161 0.001668 **
zn           3.653e-02  1.352e-02   2.702 0.007123 **
chas         2.971e+00  8.627e-01   3.443 0.000624 ***
nox         -2.025e+01  3.632e+00  -5.575 4.08e-08 ***
rm           3.983e+00  4.170e-01   9.551  < 2e-16 ***
age         -8.615e-04  1.334e-02  -0.065 0.948527
dis         -1.433e+00  1.963e-01  -7.301 1.15e-12 ***
rad          1.285e-01  4.097e-02   3.137 0.001807 **
ptratio     -1.014e+00  1.296e-01  -7.825 3.10e-14 ***
black        9.711e-03  2.709e-03   3.584 0.000372 ***
```


© 2022 Montogue Quiz

```
lstat       -5.270e-01  5.108e-02 -10.317  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1
‘ ’ 1
Residual standard error: 4.794 on 494 degrees of freedom
Multiple R-squared:  0.7342,    Adjusted R-squared:  0.7283
F-statistic:   124 on 11 and 494 DF, p-value: < 2.2e-16
```

Both models have similar *R*-squared values and *F*-statistics. Hence, a model that includes neither **age** nor **tax** predictors should be worth considering:

```
> model5alt3 <- lm(medv~.-indus-tax-age,data=Boston)
> summary(model5alt3)
Call:
lm(formula = medv ~ . - indus - tax - age, data = Boston)
Residuals:
     Min       1Q   Median       3Q      Max
-16.2609  -2.9888  -0.5083   1.8041  26.2482
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  34.712342   5.102742   6.803 2.97e-11 ***
crim         -0.104843   0.033132  -3.164 0.001650 **
zn            0.036634   0.013412   2.731 0.006532 **
chas          2.967868   0.860830   3.448 0.000614 ***
nox         -20.314416   3.472292  -5.850 8.92e-09 ***
rm            3.977104   0.407731   9.754  < 2e-16 ***
dis          -1.429370   0.186922  -7.647 1.08e-13 ***
rad           0.128761   0.040788   3.157 0.001692 **
ptratio      -1.014914   0.129006  -7.867 2.30e-14 ***
black         0.009700   0.002701   3.591 0.000363 ***
lstat        -0.528147   0.047930 -11.019  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1
‘ ’ 1
Residual standard error: 4.79 on 495 degrees of freedom
Multiple R-squared:  0.7342,    Adjusted R-squared:  0.7288
F-statistic: 136.7 on 10 and 495 DF, p-value: < 2.2e-16
```

This latter model has about the same *R*-squared and *F*-statistic as either `model5alt1` or `model5alt2`, with the added benefit of one less predictor variable. Hence, we take `model5alt3` as our final formulation. This model predicts the mean value of housing of a certain Boston district as a function of the form

$$\text{medv} = 34.712 - 0.105 \times \text{crim} + 0.0366 \times \text{zn} + 2.968 \times \text{chas}$$
$$-20.314 \times \text{nox} + 3.977 \times \text{rm} - 1.429 \times \text{dis} + 0.129 \times \text{rad} - 1.015 \times \text{ptratio}$$
$$+0.0097 \times \text{black} - 0.528 \times \text{lstat}$$

A home with a greater number of rooms or a home whose tract bounds on the Charles River are associated with greater values, whereas a home in a zone with high nitrogen oxide concentration (i.e., a region subject to greater atmospheric pollution) or a home far removed from Boston's employment centers are associated with lower values. The model further indicates that the proportion of nonretail business acres per town (represented by the variable **indus**), the proportion of owner-occupied units built prior to 1940 (variable **age**), and the full-value property-tax rate (variable **tax**) are not good predictors of mean housing value in a typical Boston neighborhood.

▶ Problem 4 – Nonlinear regression

Algal cells have been proposed as a potential source of renewable fuels. A key aspect of algal dynamics is that, as in the case of other phototrophic organisms, their growth depends on sunlight intensity. The following data provides the relationship between sunlight intensity (watts per square meter) and algal growth factor for a certain species. Negative growth factors indicate that total alga biomass is decreasing at low levels of sunlight. It is known that the two variables can be related by an exponential law of the form

© 2022 Montogue Quiz

$$f(I) = -ae^{-bI} + c$$

where I is sunlight intensity, $f(I)$ is growth factor, and a, b, and c are curve-fitting constants. We shall use nonlinear regression to fit the following data to the exponential law above.

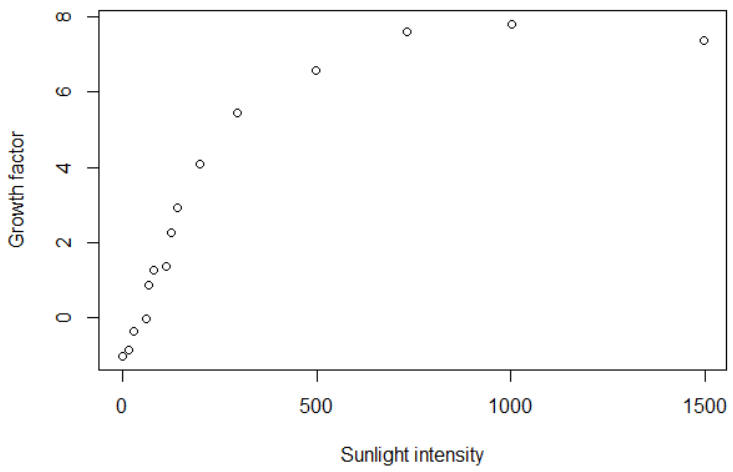
Data point	Sunlight intensity	Growth factor
1	0	-1.01
2	17.6	-0.855
3	30.9	-0.335
4	62.6	-0.023
5	66.7	0.862
6	80.1	1.278
7	111.8	1.381
8	125	2.266
9	142.8	2.943
10	201.2	4.087
11	295.7	5.437
12	499.1	6.576
13	734.3	7.61
14	1001.6	7.809
15	1495.8	7.375

These data are listed in the *algae.txt* file, which can be downloaded from our Google Drive. To read the data, save the file in R's working directory and use the **`read.table`** function:

```
> alg <- read.table("algae.txt",header = TRUE)
```

Nonlinear regression can be conducted with the **`nls`** command. The greatest difficulty in implementing a nonlinear regression fit is to provide the program with a reasonable vector of starting coefficient values. The most straightforward approach is to first use some arbitrary coefficient values for the equation we are fitting and check if the ensuing curve fits the numerical data accurately. For the present example, let us first plot the sunlight intensity-growth factor data:

```
> plot(GrowthFactor~SunlightIntensity, data=alg,
ylab="Sunlight intensity", xlab="Growth factor")
```



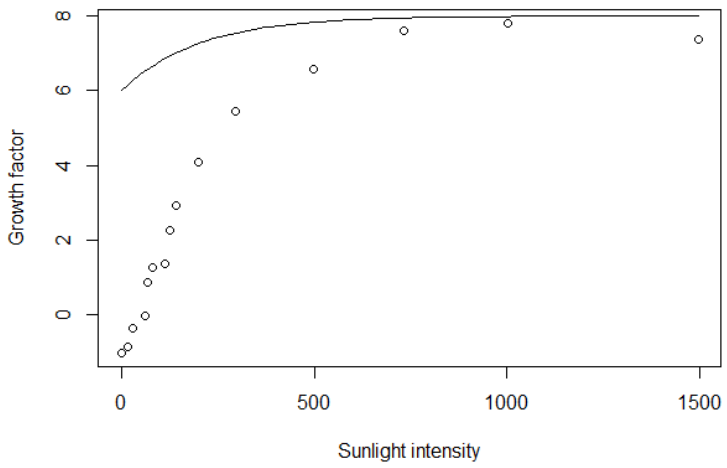
Clearly, the growth factor at first increases rapidly with sunlight intensity, but then levels off at growth factor ≈ 8.0. This suggests that constant *c* in the exponential law is probably equal to 8; we can guess values for a and b. Let us first define a function for the equation to be fitted:

```
> algFunction <- function(x,a,b,c) {-a*exp(-b*x)+c}
```

Next, let us plot **`algFunction`** with a set to, say, 2, and b set to, say, 0.005, and c set to 8:

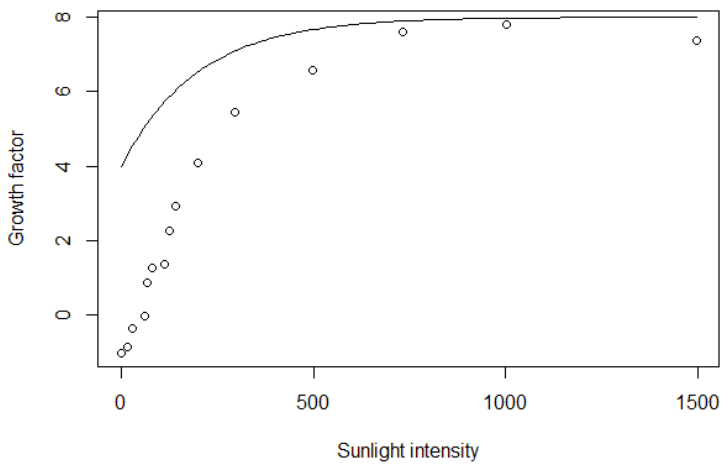
```
> curve(algFunction(x,a=2,b=0.005,c=8),add=TRUE)
```


© 2022 Montogue Quiz



The curve fits the data rather poorly. Let's try another value of a:

```
> curve(algFunction(x,a=4,b=0.005,c=8),add=TRUE)
```



The fit has improved a little. Maybe a = 4, b = 0.005, c = 8 is a good enough starting set of values for **`nls`**. Let's see:

```
> algModel <- nls(GrowthFactor~ -a*exp(-
b*SunlightIntensity)+c, data=alg,
start=list(a=4,b=0.005,c=8),trace=TRUE)
188.4210    (1.11e+01): par = (4 0.005 8)
10.34760    (2.56e+00): par = (9.240918 0.002914785 7.792817)
2.103913    (5.79e-01): par = (8.94942 0.004218594 7.472712)
1.575938    (1.04e-03): par = (9.289437 0.004227015 7.793225)
1.575937    (9.88e-05): par = (9.289472 0.004225843 7.793622)
1.575937    (1.02e-05): par = (9.289468 0.004225964 7.793568)
1.575937    (1.06e-06): par = (9.289469 0.004225951 7.793573)
```

Indeed, **`nls`** has managed to obtain a set of regressed coefficients by improving our initial guess. Convergence was obtained after 6 iterations, and the coefficients obtained are $a \approx 9.289$, $b \approx 0.00423$, and $c \approx 7.794$. The equation that describes the growth factor of the algae as a function of sunlight intensity is then

$$f(I) = -9.289e^{-0.00423I} + 7.794$$

We can summarize statistics for the model just as we would with a linear regression procedure:

```
> summary(algModel)
Formula: GrowthFactor ~ -a * exp(-b * SunlightIntensity) +
c
Parameters:
   Estimate Std. Error t value Pr(>|t|)
a 9.2894686  0.2810344   33.05 3.73e-13 ***
b 0.0042260  0.0003536   11.95 5.05e-08 ***
c 7.7935732  0.2345885   33.22 3.51e-13 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1
‘ ’ 1
Residual standard error: 0.3624 on 12 degrees of freedom
Number of iterations to convergence: 6
Achieved convergence tolerance: 1.055e-06
```


© 2022 Montogue Quiz

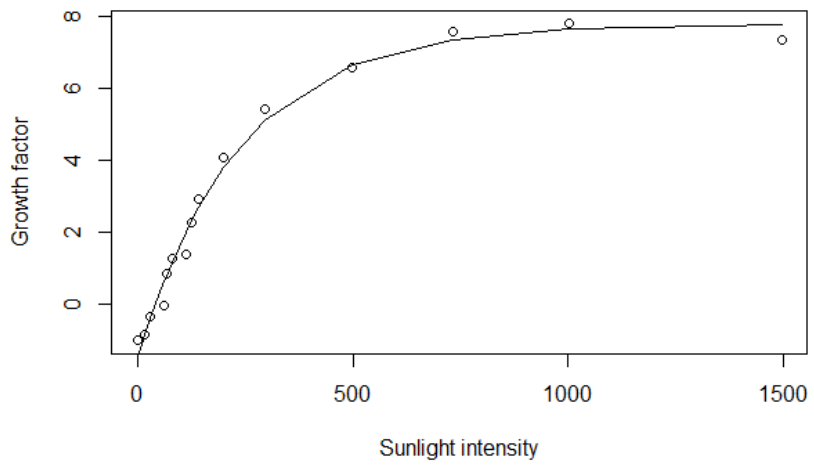
Guesswork is normally sufficient to establish an initial set of coefficients for **nls** to improve upon, but there are other methods. One alternative is use of self-starter functions, which are discussed in Ritz and Streibig (2008).

We can generate a vector of 10 evenly spaced values in the sunlight intensity data and use **predict** to compute the corresponding responses (growth factors):

```
> sunData <- with(alg,
seq(min(SunlightIntensity),max(SunlightIntensity),length.ou
t=10))
> predict(algModel, data.frame(SunlightIntensity =
sunData))
-1.495893  3.191400  5.513572  6.664019  7.233973  7.516338
7.656227  7.725531  7.759865  7.776875
```

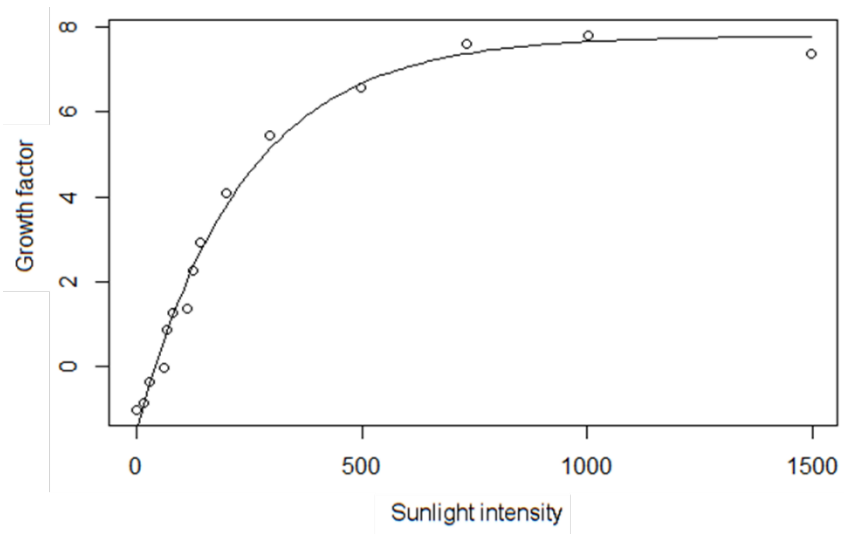
We can superpose the data plot with a plot of fitted values using the command **lines**(), which connects the fitted values for each sunlight intensity with line segments (linear interpolation):

```
> plot(GrowthFactor~SunlightIntensity, data=alg, ylab="
Growth factor", xlab=" Sunlight intensity")
> lines(alg$SunlightIntensity,fitted(algModel))
```



Notice that **lines** affords a rather crude approximation to the nonlinear fit. One way to obtain a smoother regression curve is to first generate a sequence of, say, 100 equally spaced sunlight intensity values (using the function **seq**()) between the minimum and maximum values in the dataset **alg** (using the functions **min**() and **max**()). These 100 sunlight intensity values are stored in the vector **sunValues**. Using the function **with**() with the first argument **alg** avoids having to make explicit reference to the data frame **alg** every time we consider a variable inside the data frame (we need to use **alg** twice), thereby shortening the R statement.

```
> sunValues <- with(alg,
seq(min(SunlightIntensity),max(SunlightIntensity),length.ou
t=100))
> plot(GrowthFactor~SunlightIntensity, data=alg, ylab="
Growth factor", xlab="Sunlight intensity")
> lines(sunValues,
predict(algModel,newdata=data.frame(SunlightIntensity=sunVa
lues)))
```



© 2022 Montogue Quiz

Note that if we can assume that the joint distribution of the parameter estimates in a nonlinear regression follows a normal distribution, then any smooth (i.e., differentiable) transformation of the parameter estimates will likewise be approximately normally distributed. This conclusion enables us to establish approximate normality for derived parameters that stem from the regressed equation, a technique known as the *delta method*. For example, in the relation between sunlight intensity and growth factor we've been exploring, the product ab serves as a measure of an algae culture's sensitivity to enhanced sunlight levels. Of course, one way to estimate ab is to simply recover the coefficients obtained via **`nls()`** and compute the product a times b. However, this pedestrian approach does not yield a standard error or a confidence interval for the newly formed ab parameter. One R function that does just that is **`deltaMethod()`** in the package **`car`**:

```
> deltaMethod(algModel, "a*b")
      Estimate      SE      2.5 %      97.5 %
a * b 0.0392568 0.0034454 0.0325040  0.046
```

That is, ab equals about 0.0393, with a standard error of 0.00345 and a 95% CI (0.0325; 0.0460).

REFERENCES

- HEIBERGER, RM and HOLLAND, B (2015). *Statistical Analysis and Data Display*. 2nd edition. Berlin/Heidelberg: Springer.
- JAMES, G, WITTEN, D, HASTIE, T and TIBSHIRANI, R (2013). *An Introduction to Statistical Learning*. Berlin/Heidelberg: Springer.
- RHINEHART, RR (2016). *Nonlinear Regression Modeling for Engineering Applications*. Hoboken: John Wiley and Sons. ISBN: 978-1-118-59796-5.
- RITZ, C and STREIBIG, JC (2008). *Nonlinear Regression with R*. Berlin/Heidelberg: Springer.

Visit www.montoguequiz.com for more free R tutorials, statistics problem sets, and all things science and engineering!

© 2022 Montogue Quiz