Regression Analysis Using R

Working Paper · August 2016

DOI: 10.13140/RG.2.2.18837.52967

1 author:

Abdaljbbar .B.A. Dawod
University of Khartoum
10 PUBLICATIONS **55** CITATIONS

SEE PROFILE

Regression Analysis Using R

Abdaljbbar B. A. Dawod, King Fahd University of Petrolum and Mineral, KSA

December 20, 2015

Contents

1 introduction

To deal with any prediction problem; the easiest and the common used technique is the regression. Regression analysis is used to model the relationship between a response variable (dependent) and one or more explanatory variables(independent).

Regression analysis helps us to find answers to:

- Prediction of future observations.
- Find the association, the relationship between the variables.
- Identify which variables contribute more towards predicting the future outcomes.

There are diffrent types of regression that can be used regarding to the type of the problem. Therefore, the regression can be linrear regression or nonlinear regression.

2 Linear Regression

linear regression is an approach to model the relationship between a scalar dependent variable ***y*** and one or more explanatory variables (independent variables) denoted by x_i, i=1,2,. . . ,n.

The following are the major assumptions made by standard linear regression models with standard estimation techniques (e.g. ordinary least squares):

- Weak exogeneity.
- Linearity.
- Constant variance.
- Independence.
- Lack of multicollinearity.

The main regression function in R used for modelling linear regression is ***lm()***. Moreover, R includes a wealth tools for more complex modeling such as, ***glm()*** for generalized linear models, ***gam()*** for generalized additive models, ***lme()*** and ***lmer()*** for linear mixed-effects models.

2.1 Simple Linear Regression:

A model deals with one variable, called as independent or predictor variable and one variable called as dependent or response variable is called simple linear regression. In this type of linear regression, it assumes that there exists a linear relation between predictor and response variable of the form,

$$E[Y|X] = \beta_0 + \beta_1 * X + e$$

2.1.1 Estimating a simple linear regression equation

To estimate the equations' parameters, we use the function,

lm(dependent variable ~ Independent variable)

```
data("iris")
model1 = lm(Sepal.Width ~Sepal.Length, data = iris)
model1
```

```
##
## Call:
## lm(formula = Sepal.Width ~ Sepal.Length, data = iris)
##
## Coefficients:
##  (Intercept)  Sepal.Length
##      3.41895      -0.06188
```

Extracting the parameters of the estimated coefficients,

```
coeffs = coefficients(model1); coeffs
```

```
##  (Intercept) Sepal.Length
##    3.4189468   -0.0618848
```

To get more information about the fitted model, ***summary(the model's name)*** can be used to get details about *Residuals*, *Coefficients*, *Residual standard error*, R^2, Adjusted R^2. Moreover, ***summary.anova(the model's name)*** used to get ANOVA table.

```
summary(model1) # same as summary.lm(model1)
```

```
##
## Call:
## lm(formula = Sepal.Width ~ Sepal.Length, data = iris)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.1095 -0.2454 -0.0167  0.2763  1.3338
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.41895    0.25356   13.48   <2e-16 ***
## Sepal.Length -0.06188    0.04297   -1.44    0.152
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4343 on 148 degrees of freedom
## Multiple R-squared:  0.01382,    Adjusted R-squared:  0.007159
## F-statistic: 2.074 on 1 and 148 DF,  p-value: 0.1519
```

```
summary.aov(model1)
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## Sepal.Length   1  0.391  0.3913   2.074  0.152
## Residuals    148 27.916  0.1886
```

2.1.2 Predict with the estimated coefficients,

```r
Lengt = 4.5
width = coeffs[1] + coeffs[2]*Lengt
width
```

```
## (Intercept)
##    3.140465
```

Instead of prediction with the pervious method; an existed function ***predict*** can be used for prediction purposes,

```r
newdata = data.frame(Sepal.Length=4.5)
# apply predict
predict(model1, newdata)
```

```
##        1
## 3.140465
```

2.1.3 Confidence interval and predicted confidence interval

to estimate the confidence interval for our model; we run the below function,

```r
predict(model1, newdata, interval = 'confidence')
```

```
##        fit      lwr      upr
## 1 3.140465 3.006599 3.274331
```

For estimating a predicted confidence interval for the estimated model, we can use ***predict*** instead of ***confidence***.

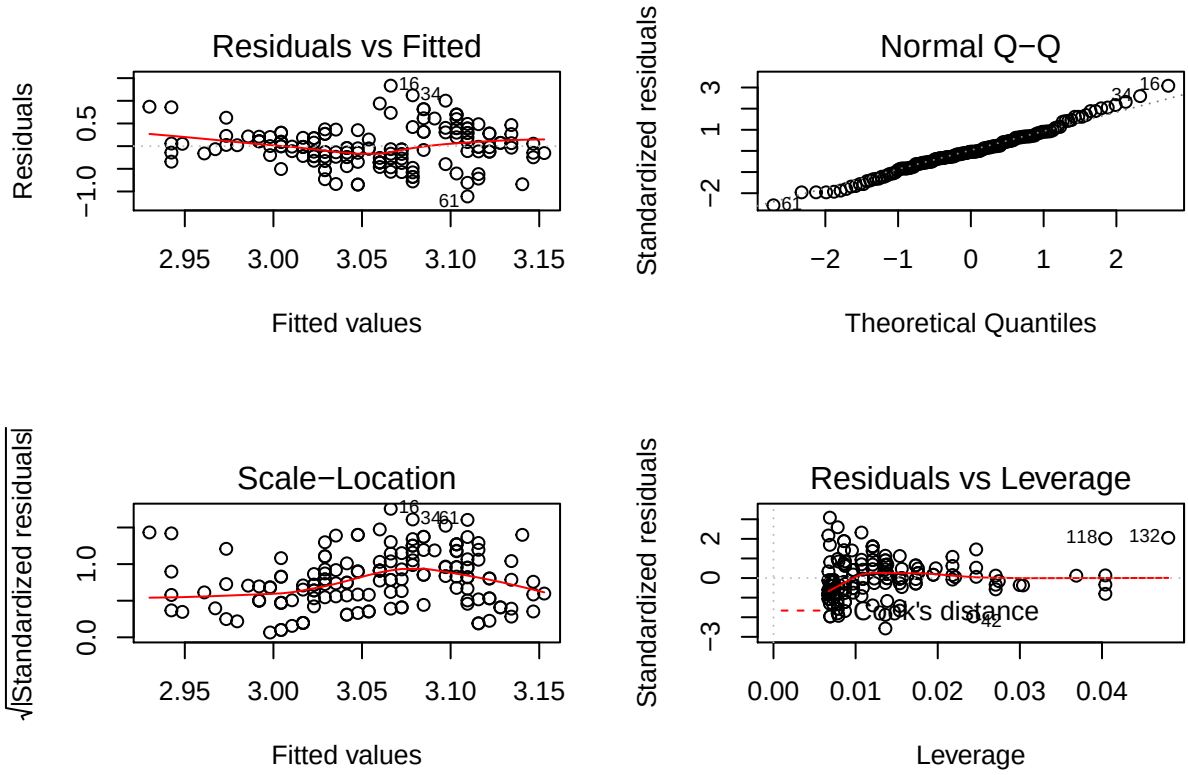
```r
predict(model1, newdata, interval = 'predict')
```

```
##        fit      lwr      upr
## 1 3.140465 2.271852 4.009079
```

2.1.4 Model Diagnosing

For diagnosing the model, the graphical plots can be used. Diagnostic plots provide checks for heteroscedasticity, normality, and influential observerations.

```r
par(mfrow = c(2,2))
plot(model1)
```



- More tests

```
library(car)
qqplot(model1)            # Testing for normality
residualPlots(model1)     # residual plots
ncvTest(model1)           # Testing for heteroskedasticity
outlierTest(model1)       # Outliers -Bonferonni test
vif(model1)               # Testing for multicolinearity, A vif > 4 suggests collinearity
durbinWatsonTest(model1) # Test for Autocorrelated Errors
```

2.2 Multiple Linear Regression:

When the problem contains more than one explanatory variables and one response variable, then it is called multiple linear regression. The general multiple regression can be writen as,

$$E[Y|X] = \alpha + \sum_{i=1} \beta_i X_i + \epsilon$$

In the case of multiple linear regression; we use the same function ***lm()***, but her it takes more than one dependent variable ***lm(dependent variable ~ independent Var1+Var2 +...+Var_n)***.

In our example, we etimated the multiple linear regression model using dataset ***state*** that is avialable in R.

```
data(state) #head(state.x77)
st = as.data.frame(state.x77)
colnames(st)[c(4,6)] = c("LifeExp", "HSGrad" )  # no spaces in variable names
```

```r
model2 = lm(LifeExp ~ Population + Income + Illiteracy + Murder +
                      HSGrad + Frost + Area, data=st)
summary(model2)
```

```
##
## Call:
## lm(formula = LifeExp ~ Population + Income + Illiteracy + Murder +
##     HSGrad + Frost + Area, data = st)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.48895 -0.51232 -0.02747  0.57002  1.49447
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.094e+01  1.748e+00  40.586  < 2e-16 ***
## Population   5.180e-05  2.919e-05   1.775   0.0832 .
## Income      -2.180e-05  2.444e-04  -0.089   0.9293
## Illiteracy   3.382e-02  3.663e-01   0.092   0.9269
## Murder      -3.011e-01  4.662e-02  -6.459 8.68e-08 ***
## HSGrad       4.893e-02  2.332e-02   2.098   0.0420 *
## Frost       -5.735e-03  3.143e-03  -1.825   0.0752 .
## Area        -7.383e-08  1.668e-06  -0.044   0.9649
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7448 on 42 degrees of freedom
## Multiple R-squared:  0.7362, Adjusted R-squared:  0.6922
## F-statistic: 16.74 on 7 and 42 DF,  p-value: 2.534e-10
```

```r
summary.aov(model2) # testing the significance of predictors
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## Population   1  0.409   0.409   0.737  0.39543
## Income       1 11.595  11.595  20.903 4.22e-05 ***
## Illiteracy   1 19.421  19.421  35.012 5.23e-07 ***
## Murder       1 27.429  27.429  49.449 1.31e-08 ***
## HSGrad       1  4.099   4.099   7.389  0.00949 **
## Frost        1  2.049   2.049   3.694  0.06143 .
## Area         1  0.001   0.001   0.002  0.96491
## Residuals   42 23.297   0.555
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

2.2.1 The Minimal Adequate Model

We want to reduce our model where all the remaining predictors are significant. Moreover, Model selection using Akaike's Information Criterion (AIC):

```r
step(model2)
```

```
## Start:  AIC=-22.18
## LifeExp ~ Population + Income + Illiteracy + Murder + HSGrad +
##     Frost + Area
##
##              Df Sum of Sq    RSS     AIC
## - Area        1    0.0011 23.298 -24.182
## - Income      1    0.0044 23.302 -24.175
## - Illiteracy  1    0.0047 23.302 -24.174
## <none>                    23.297 -22.185
## - Population  1    1.7472 25.044 -20.569
## - Frost       1    1.8466 25.144 -20.371
## - HSGrad      1    2.4413 25.738 -19.202
## - Murder      1   23.1411 46.438  10.305
##
## Step:  AIC=-24.18
## LifeExp ~ Population + Income + Illiteracy + Murder + HSGrad +
##     Frost
##
##              Df Sum of Sq    RSS     AIC
## - Illiteracy  1    0.0038 23.302 -26.174
## - Income      1    0.0059 23.304 -26.170
## <none>                    23.298 -24.182
## - Population  1    1.7599 25.058 -22.541
## - Frost       1    2.0488 25.347 -21.968
## - HSGrad      1    2.9804 26.279 -20.163
## - Murder      1   26.2721 49.570  11.569
##
## Step:  AIC=-26.17
## LifeExp ~ Population + Income + Murder + HSGrad + Frost
##
##              Df Sum of Sq    RSS     AIC
## - Income      1     0.006 23.308 -28.161
## <none>                    23.302 -26.174
## - Population  1     1.887 25.189 -24.280
## - Frost       1     3.037 26.339 -22.048
## - HSGrad      1     3.495 26.797 -21.187
## - Murder      1    34.739 58.041  17.456
##
## Step:  AIC=-28.16
## LifeExp ~ Population + Murder + HSGrad + Frost
##
##              Df Sum of Sq    RSS     AIC
## <none>                    23.308 -28.161
## - Population  1     2.064 25.372 -25.920
## - Frost       1     3.122 26.430 -23.877
## - HSGrad      1     5.112 28.420 -20.246
## - Murder      1    34.816 58.124  15.528

##
## Call:
## lm(formula = LifeExp ~ Population + Murder + HSGrad + Frost,
##     data = st)
##
## Coefficients:
```

```
## (Intercept)   Population       Murder       HSGrad        Frost
##   7.103e+01    5.014e-05   -3.001e-01    4.658e-02   -5.943e-03
```

To reach to a model that all its predictors are significats, we can do this by throwing out one predictor at a time. ***Area*** will go first. To do this, we could just recast the model without the ***Area*** variable,

```
model3 = update(model2, .~.-Area)
summary(model3)
```

```
##
## Call:
## lm(formula = LifeExp ~ Population + Income + Illiteracy + Murder +
##     HSGrad + Frost, data = st)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.49047 -0.52533 -0.02546  0.57160  1.50374
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.099e+01  1.387e+00  51.165  < 2e-16 ***
## Population   5.188e-05  2.879e-05   1.802   0.0785 .
## Income      -2.444e-05  2.343e-04  -0.104   0.9174
## Illiteracy   2.846e-02  3.416e-01   0.083   0.9340
## Murder      -3.018e-01  4.334e-02  -6.963 1.45e-08 ***
## HSGrad       4.847e-02  2.067e-02   2.345   0.0237 *
## Frost       -5.776e-03  2.970e-03  -1.945   0.0584 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7361 on 43 degrees of freedom
## Multiple R-squared:  0.7361, Adjusted R-squared:  0.6993
## F-statistic: 19.99 on 6 and 43 DF,  p-value: 5.362e-11
```

The two models can be compared using function ***anova*** as follows,

```
anova(model2, model3)
```

```
## Analysis of Variance Table
##
## Model 1: LifeExp ~ Population + Income + Illiteracy + Murder + HSGrad +
##     Frost + Area
## Model 2: LifeExp ~ Population + Income + Illiteracy + Murder + HSGrad +
##     Frost
##   Res.Df    RSS Df  Sum of Sq     F Pr(>F)
## 1     42 23.297
## 2     43 23.298 -1 -0.0010866 0.002 0.9649
```

What goes next, ***Income*** or ***Illiteracy***? ***Illiteracy***.

```r
model4 = update(model3, .~.-Illiteracy)
summary(model4)
```

```
##
## Call:
## lm(formula = LifeExp ~ Population + Income + Murder + HSGrad +
##     Frost, data = st)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.4892 -0.5122 -0.0329  0.5645  1.5166
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.107e+01  1.029e+00  69.067  < 2e-16 ***
## Population   5.115e-05  2.709e-05   1.888   0.0657 .
## Income      -2.477e-05  2.316e-04  -0.107   0.9153
## Murder      -3.000e-01  3.704e-02  -8.099 2.91e-10 ***
## HSGrad       4.776e-02  1.859e-02   2.569   0.0137 *
## Frost       -5.910e-03  2.468e-03  -2.395   0.0210 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7277 on 44 degrees of freedom
## Multiple R-squared:  0.7361, Adjusted R-squared:  0.7061
## F-statistic: 24.55 on 5 and 44 DF,  p-value: 1.019e-11
```

What goes next, ***Income***.

```r
model5 = update(model4, .~.-Income)
summary(model5)
```

```
##
## Call:
## lm(formula = LifeExp ~ Population + Murder + HSGrad + Frost,
##     data = st)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.47095 -0.53464 -0.03701  0.57621  1.50683
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.103e+01  9.529e-01  74.542  < 2e-16 ***
## Population   5.014e-05  2.512e-05   1.996  0.05201 .
## Murder      -3.001e-01  3.661e-02  -8.199 1.77e-10 ***
## HSGrad       4.658e-02  1.483e-02   3.142  0.00297 **
## Frost       -5.943e-03  2.421e-03  -2.455  0.01802 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7197 on 45 degrees of freedom
## Multiple R-squared:  0.736,  Adjusted R-squared:  0.7126
## F-statistic: 31.37 on 4 and 45 DF,  p-value: 1.696e-12
```

2.2.2 Confidence limits for the estimated coefficients

```
confint(model5)
```

```
##                     2.5 %        97.5 %
## (Intercept)  6.910798e+01 72.9462729104
## Population  -4.543308e-07  0.0001007343
## Murder      -3.738840e-01 -0.2264135705
## HSGrad       1.671901e-02  0.0764454870
## Frost       -1.081918e-02 -0.0010673977
```

To predict the future values using multiple linear regression equation; ***predict*** can be used as it was used in ***section 2.1***

```
newdata = data.frame(Population=3615,LifeExp=70, Murder=10.3, HSGrad=41.3, Frost=20)
 predict(model5, newdata)
```

```
##        1
## 69.92183
```

To estimate the confidence interval and prediction interval for multiple linear regression; ***confidence*** and ***predict*** can be used repectively as input for the function ***predict***

```
predict(model5, newdata, interval="confidence")
```

```
##        fit      lwr      upr
## 1 69.92183 69.45517 70.38849
```

Prediction interval for MLR

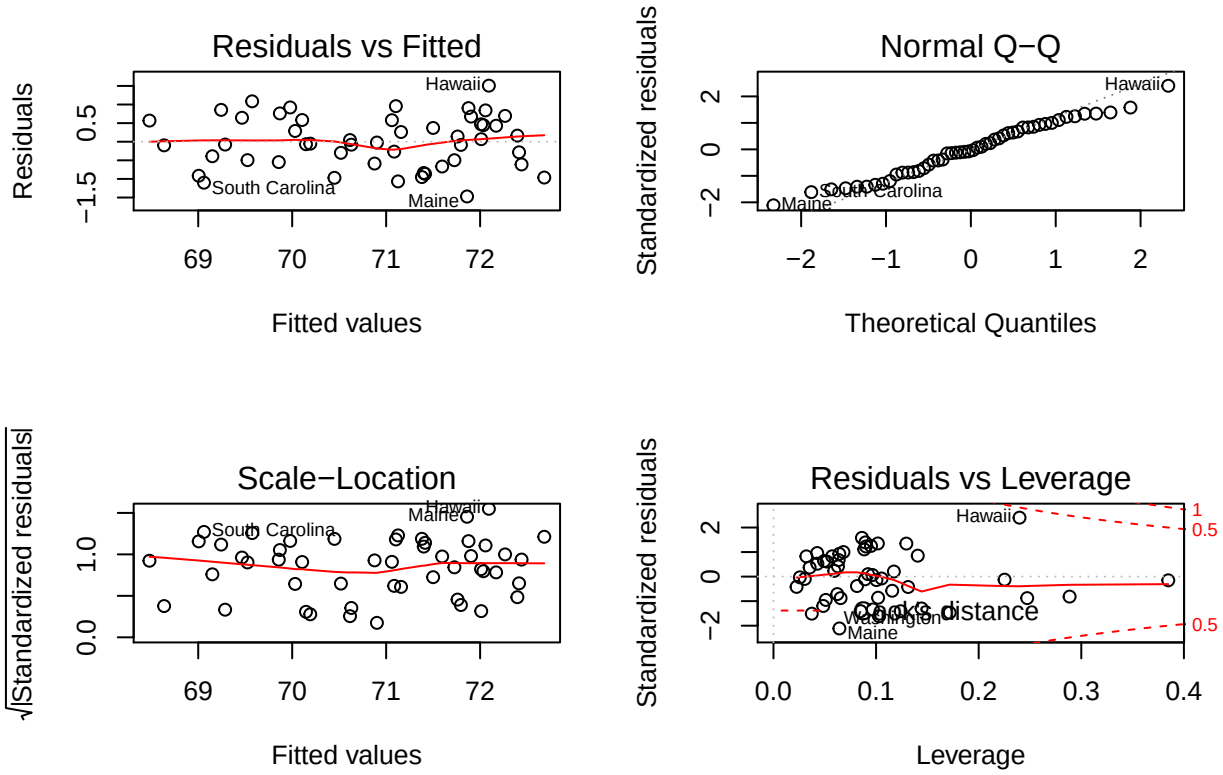
```
predict(model5, newdata, interval="predict")
```

```
##        fit      lwr      upr
## 1 69.92183 68.39903 71.44463
```

2.2.3 Model Diagnostics

The usual diagnostic plots are available...

```
par(mfrow=c(2,2)) # visualize four graphs at once
plot(model5)
```

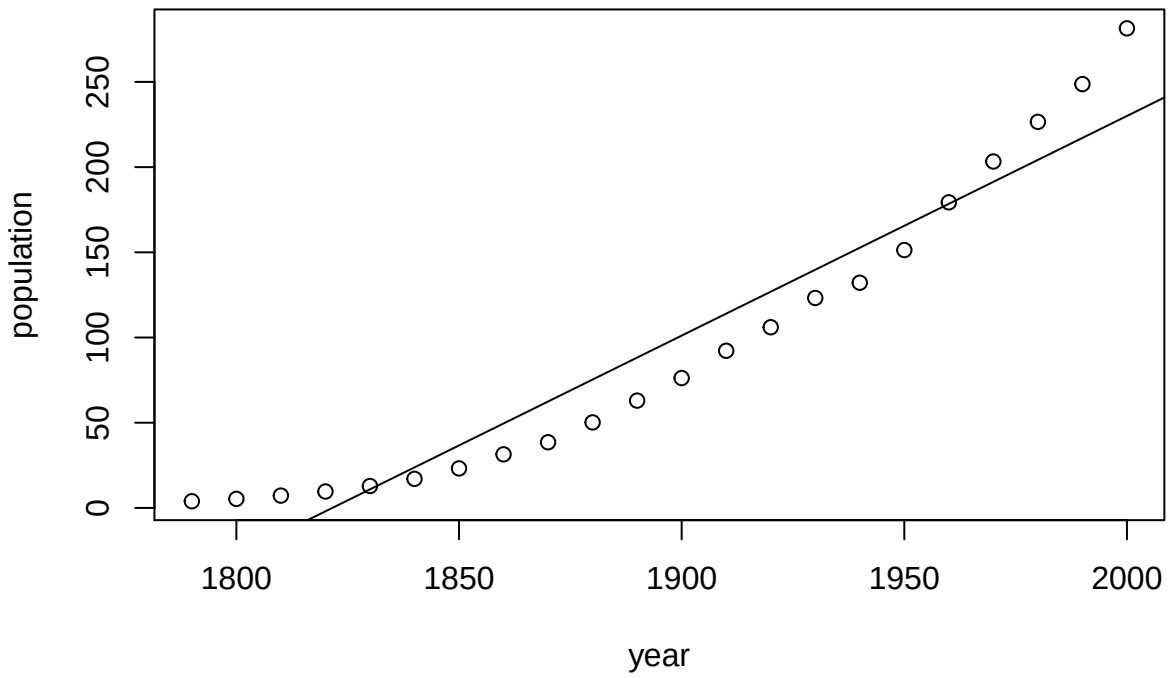


3 Nonlinear Regression

Non-linear regression models are useful when there is a priori knowledge of the theoretical non-linear relationship between two variables.

Example, the data frame ***USPop*** in the car package has decennial US Census population for the United States (in millions), from 1790 through 2000. The data are shown in the figure below,

```
library(car)
plot(population ~ year, data=USPop)
abline(lm(population ~ year, data=USPop))
```

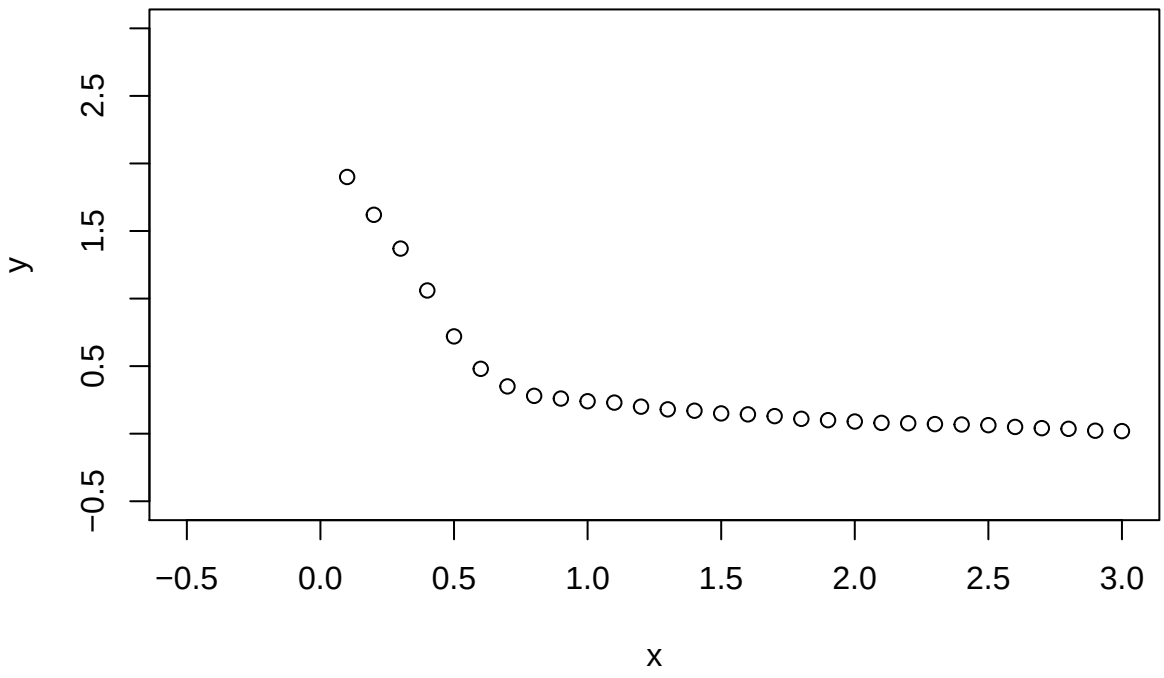


The simple linear regression least-squares line shown on the graph clearly does not match these data: The US population is not growing by the same rate in each decade, as is required by the straight-line model.

3.1 Simple Nonlinear regression

It involves finding some ways to transform one or both of the variables to make the relationship linear.

```
x = c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0,
       1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0,
       2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 3.0)
y = c(1.900, 1.620, 1.370, 1.060, 0.720, 0.480, 0.350,
      0.280, 0.260, 0.240, 0.230, 0.200, 0.180, 0.170, 0.150,
      0.143, 0.130, 0.110, 0.100, 0.090, 0.080, 0.077, 0.071,
      0.068, 0.063, 0.050, 0.041, 0.036,0.022, 0.019)
plot(x,y, xlim=c(-.5,3), ylim=c(-.5,3))
```




```
library(robust)
```

```
## Warning: package 'robust' was built under R version 3.2.3
```

```
## Loading required package: fit.models
```

```
## Warning: package 'fit.models' was built under R version 3.2.3
```

```
## Loading required package: lattice
## Loading required package: MASS
## Loading required package: robustbase
```

```
## Warning: package 'robustbase' was built under R version 3.2.3
```

```
## Loading required package: rrcov
```

```
## Warning: package 'rrcov' was built under R version 3.2.3
```

```
## Scalable Robust Estimators with High Breakdown Point (version 1.3-8)
```

- Apply a common linear (OLS) regression; all 'looks' well, but....

i. Robust linear regression.

```
linear.rob = lmRob(y ~ x)
summary(linear.rob)
```

```
##
## Call:
## lmRob(formula = y ~ x)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.023533 -0.010905  0.009337  0.030720  1.573370
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.337989   0.020334   16.62 4.88e-16 ***
## x           -0.113587   0.009893  -11.48 4.19e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02312 on 28 degrees of freedom
## Multiple R-Squared: 0.5532
##
## Test for Bias:
##             statistic p-value
## M-estimate     2.2626  0.3226
## LS-estimate    0.3297  0.8480
```

ii. A quadratic regression.

```
Linear.quad = lm(y ~ x^2 + x)
summary(Linear.quad)
```

```
##
## Call:
## lm(formula = y ~ x^2 + x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.38232 -0.25982 -0.06613  0.17756  0.94027
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.00222    0.12229   8.195 6.39e-09 ***
## x           -0.42487    0.06888  -6.168 1.17e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3266 on 28 degrees of freedom
## Multiple R-squared:  0.576,  Adjusted R-squared:  0.5609
## F-statistic: 38.04 on 1 and 28 DF,  p-value: 1.166e-06
```

iii. A cubic regression.

```r
Linear.cub = lm(y ~ x^3 + x^2 + x)
summary(Linear.cub)
```

```
##
## Call:
## lm(formula = y ~ x^3 + x^2 + x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.38232 -0.25982 -0.06613  0.17756  0.94027
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.00222    0.12229   8.195 6.39e-09 ***
## x           -0.42487    0.06888  -6.168 1.17e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3266 on 28 degrees of freedom
## Multiple R-squared:  0.576,  Adjusted R-squared:  0.5609
## F-statistic: 38.04 on 1 and 28 DF,  p-value: 1.166e-06
```

3.2 REAL Non-Linear Regression

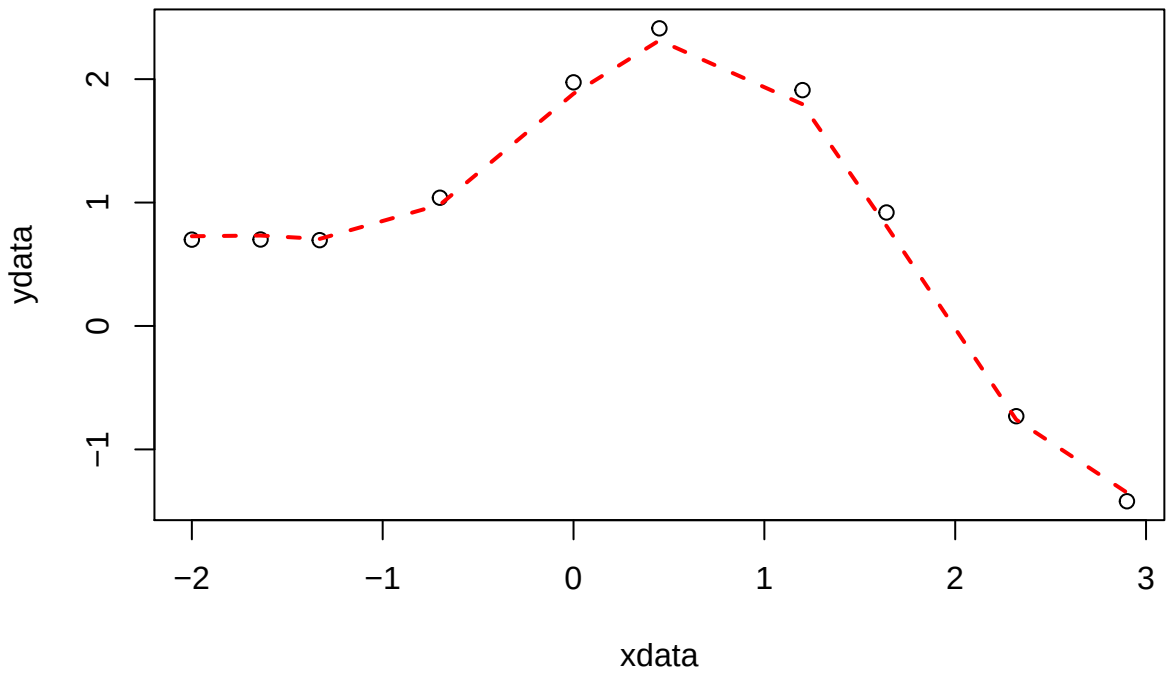
Example:

```r
library(nls2)
```

```
## Loading required package: proto
```

```r
xdata = c(-2,-1.64,-1.33,-0.7,0,0.45,1.2,1.64,2.32,2.9)
ydata = c(0.699369,0.700462,0.695354,1.03905,1.97389,
          2.41143,1.91091,0.919576,-0.730975,-1.42001)
plot(xdata,ydata)
p1 = 1; p2 = 0.2 # some starting values
fit = nls(ydata ~ p1*cos(p2*xdata) + p2*sin(p1*xdata), start=list(p1=p1,p2=p2))

#Draw the fit on the plot by getting the prediction from
#the fit at 200 x-coordinates across the range of xdata
lines(xdata, fitted(fit), lty = 2, col = "red", lwd = 2)
```




```
summary(fit)  # summarise
```

```
##
## Formula: ydata ~ p1 * cos(p2 * xdata) + p2 * sin(p1 * xdata)
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## p1 1.881851   0.027430   68.61 2.27e-12 ***
## p2 0.700230   0.009153   76.51 9.50e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08202 on 8 degrees of freedom
##
## Number of iterations to convergence: 7
## Achieved convergence tolerance: 2.189e-06
```

4 Logistic Regression

Logistic regression is a method for fitting a regression curve, $\boldsymbol{y = f(x)}$, when $\boldsymbol{y}$ is a categorical variable. The typical use of this model is predicting $\boldsymbol{y}$ given a set of predictors $\boldsymbol{x}$. The predictors can be continuous, categorical or a mix of both.

The categorical variable y, in general, can assume different values. In the simplest case scenario y is binary meaning that it can assume either the value 1 or 0.

The logistic regression equation is:

$$E[Y|X_1, x_2, ..., x_n] = \frac{1}{1 + e^{-(\alpha + \sum_{i=1} \beta_i X_i)}}$$

- Generalized Linear Models

Generalized linear models are fit using the glm( ) function. The form of the glm function is,

glm(formula, family=familytype(link=linkfunction), data=)

Family	Default Link Function
binomial	(link = "logit")
gaussian	(link = "identity")
Gamma	(link = "inverse")
inverse.gaussian	(link = "1/mu^2")
poisson	(link = "log")
quasi	(link = "identity", variance = "constant")
quasibinomial	(link = "logit")
quasipoisson	(link = "log")

We use the ***glm()*** function, include the variables in the usual way, and specify a binomial error distribution, as follows:

```
model1 = glm(formula= vs ~ wt + disp, data=mtcars, family=binomial)
model2 = glm(vs ~ mpg, data=mtcars, family=binomial(link="logit"))

summary(model1)
```

```
##
## Call:
## glm(formula = vs ~ wt + disp, family = binomial, data = mtcars)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.67506  -0.28444  -0.08401   0.57281   2.08234
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.60859    2.43903   0.660    0.510
## wt           1.62635    1.49068   1.091    0.275
## disp        -0.03443    0.01536  -2.241    0.025 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 43.86  on 31  degrees of freedom
## Residual deviance: 21.40  on 29  degrees of freedom
## AIC: 27.4
##
## Number of Fisher Scoring iterations: 6
```

```r
exp(coef(model1))
```

```
## (Intercept)          wt        disp
##   4.9957752   5.0852960   0.9661524
```

```r
summary(model2)
```

```
##
## Call:
## glm(formula = vs ~ mpg, family = binomial(link = "logit"), data = mtcars)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2127  -0.5121  -0.2276   0.6402   1.6980
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -8.8331     3.1623  -2.793  0.00522 **
## mpg           0.4304     0.1584   2.717  0.00659 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 43.860  on 31  degrees of freedom
## Residual deviance: 25.533  on 30  degrees of freedom
## AIC: 29.533
##
## Number of Fisher Scoring iterations: 6
```

```r
exp(model2$coefficients)
```

```
##  (Intercept)          mpg
## 0.0001458295 1.5378933421
```

We see from the estimates of the coefficients that weight influences vs positively, while displacement has a slightly negative effect.

Our goal here is to calculate a predicted probability of a V engine, for specific values of the predictors: a weight of 2100 lbs and engine displacement of 180 cubic inches.

```r
newdata = data.frame(wt = 2.1, disp = 180)
predict(model1, newdata, type="response")
```

```
##         1
## 0.2361081
```

Now we can run the ***anova()*** function on the model to analyze the table of deviance

```r
anova(model1, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: vs
##
## Terms added sequentially (first to last)
##
##
##      Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                    31     43.860
## wt    1  12.4934        30     31.367 0.0004084 ***
## disp  1   9.9663        29     21.400 0.0015943 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

5 Refrences

1. M. Maathuis, 2012, Regression, Lecture notes for the Spring 2012 course, ETH Zurich.
2. cookbook for r, http://www.cookbook-r.com/Statistical_analysis/Logistic_regression
3. Crawley, M.J., 2013, The R Book, Wiley, Chichester, 1051 p.
4. Long, J. Scott, 1997, Regression Models for Categorical and Limited Dependent Variables. Thousand Oaks, CA: Sage Publications.
5. Baty, F., Ritz, C., Charles, S., Brutsche, M., Flandrois, J. P., & Delignette-Muller, M. L. (2014). A toolbox for nonlinear regression in R: the package nlstools. at the Journal of Statistical Software.
6. Meier C. Introduction to R Code for Linear, Non-linear, and Linear Mixed Effects Models, Department of Geological Sciences, University of Colorado at Boulder
7. Turner, H., & Firth, D. (2007). Generalized nonlinear models in R: An overview of the gnm package.