chapter 8

Regression analysis I

Instructor: Li, Han

Contents

- The framework of OLS regression
- Fit the data using linear model
- Fit the data using quadratic model
- Analyze the output results
- Check the model assumptions

Regression analysis

Regression analysis lives at the heart of statistics. It is a broad term for a set of methodologies used to analyze the functional relationship between a response variable and one or more explanatory(predictor) variables.

In general, regression analysis can

- identify the relevant explanatory variables
- describe the form of the relationships involved
- predict the response for given explanatory variables.

A data example

An exercise physiologist might use regression analysis to develop an equation for predicting the expected number of calories a person will burn while exercising on a treadmill.

response variable:

the number of calories burned

explanatory variables:

- duration of exercise (minutes)
- percentage of time spent at their target heart rate
- average speed (mph)
- age (years), gender, and body mass index (BMI)

Questions:

- What's the relationship between exercise duration and calories burned? Is it linear or curvilinear?
- How does effort (the percentage of time at the target heart rate, the average walking speed) factor in?
- How many calories can a 30-year-old man with a BMI of 28.7 expect to burn if he walks for 45 minutes at an average speed of 4 miles per hour and stays within his target heart rate 80 percent of the time?

Effective regression analysis is an interactive, holistic process with many steps, and it involves many skills.

Given the data, we repeat the following steps until we are satisfied with the model/results:

- propose a model
- fit the model
- check the model assumptions

Different regression models

Regression methods include the followings:

- simple/multiple linear
- polynomial
- logistic
- poisson
- time series

OLS-based regression

Here we focus on regression methods called ordinary least squares (OLS) regression, including simple/multiple linear regression and polynomial regression.

In OLS regression, a quantitative dependent variable is predicted from a weighted sum of predictor variables, where the weights are parameters estimated from the data.

Linear model

$$Y_i = \beta_o + \beta_1 X_{1i} + \ldots + \beta_k X_{ki} + \varepsilon_i, \quad i = 1, \ldots, n.$$

Model assumptions:

- Normality —For fixed values of the independent variables, the dependent variable is normally distributed.
- Independence —The Yi values are independent of each other.
- Linearity
- Homoscedasticity —The variance of Yi values doesn't vary with the levels of the independent variables.

OLS selects model parameters (intercept and slopes) that minimize the difference between actual response values and those predicted by the model. Specifically, model parameters are selected to minimize the sum of squared residuals.

$$\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 = (Y_i - \hat{\beta}_o - \hat{\beta}_1 X_{1i} - \ldots - \hat{\beta}_k X_{ki})^2 = \sum_{i=1}^{n} \varepsilon_i^2 .$$

Fitting regression line with lm()

In R, the basic function for fitting a linear model is lm(). The format is

myfit <- lm(formula, data)

- formula: the model, ususally it is $y \sim x_1 + \ldots + x_k$.
- data: the data frame for regression analysis, includes y, $x_1, \ldots, x_k$.
- myfit: a list containing various information about the regression results.

Table 8.2 Symbols commonly used in R formulas

Symbol	Usage
~	Separates response variables on the left from the explanatory variables on the right. For example, a prediction of `y` from `x`, `z`, and `w` would be coded `y ~ x + z + w`.
+	Separates predictor variables.
:	Denotes an interaction between predictor variables. A prediction of `y` from `x`, `z`, and the interaction between `x` and `z` would be coded `y ~ x + z + x:z`.
*	A shortcut for denoting all possible interactions. The code `y ~ x * z * w` expands to `y ~ x + z + w + x:z + x:w + z:w + x:z:w`.
^	Denotes interactions up to a specified degree. The code `y ~ (x + z + w)^2` expands to `y ~ x + z + w + x:z + x:w + z:w`.
.	A place holder for all other variables in the data frame except the dependent variable. For example, if a data frame contained the variables `x, y, z, and w`, then the code `y ~ .` would expand to `y ~ x + z + w`.

`-`	A minus sign removes a variable from the equation. For example, `y ~ (x + z + w)^2 - x:w` expands to `y ~ x + z + w + x:z + z:w`.
`-1`	Suppresses the intercept. For example, the formula `y ~ x -1` fits a regression of `y` on `x`, and forces the line through the origin at `x=0`.
`I()`	Elements within the parentheses are interpreted arithmetically. For example, `y ~ x + (z + w)^2` would expand to `y ~ x + z + w + z:w`. In contrast, the code `y ~ x + I((z + w)^2)` would expand to `y ~ x + h`, where `h` is a new variable created by squaring the sum of `z` and `w`.
function	Mathematical functions can be used in formulas. For example, `log(y) ~ x + z + w` would predict `log(y)` from `x`, `z`, and `w`.

Table 8.3 Other functions that are useful when fitting linear models

Function	Action
`summary()`	Displays detailed results for the fitted model
`coefficients()`	Lists the model parameters (intercept and slopes) for the fitted model
`confint()`	Provides confidence intervals for the model parameters (95 percent by default)
`fitted()`	Lists the predicted values in a fitted model
`residuals()`	Lists the residual values in a fitted model
`anova()`	Generates an ANOVA table for a fitted model, or an ANOVA table comparing two or more fitted models
`vcov()`	Lists the covariance matrix for model parameters
`AIC()`	Prints Akaike's Information Criterion
`plot()`	Generates diagnostic plots for evaluating the fit of a model
`predict()`	Uses a fitted model to predict response values for a new dataset

A data example

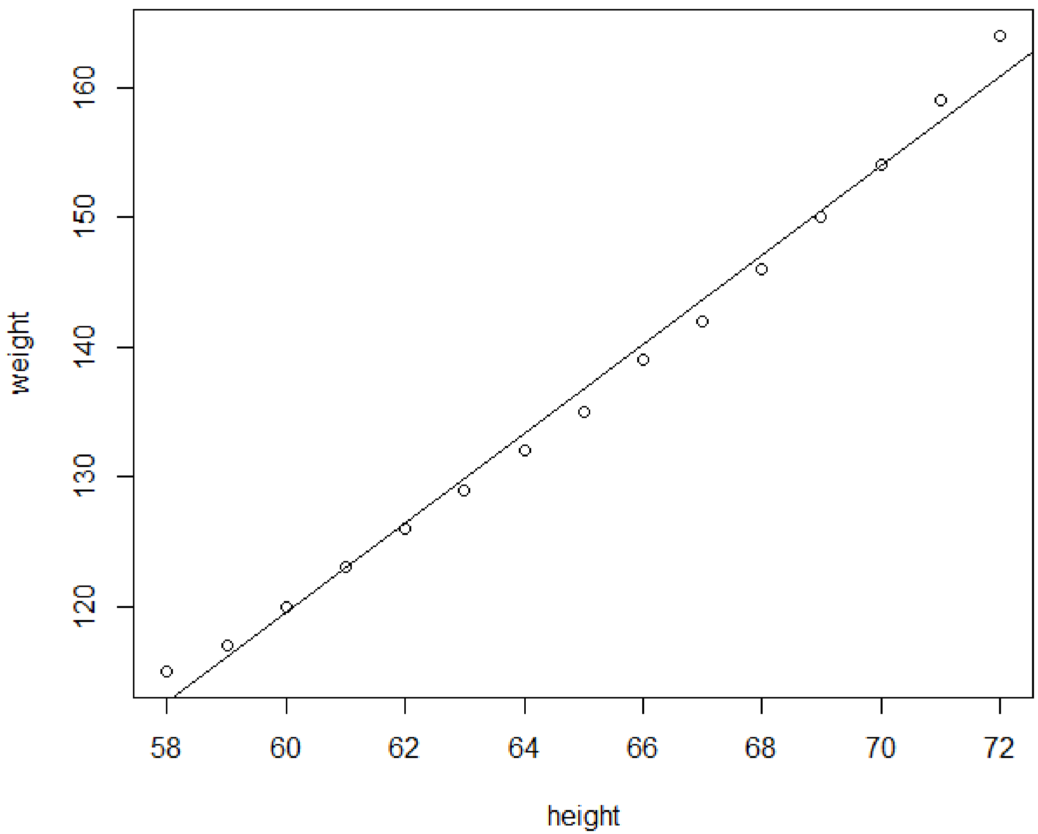
The dataset women in the base installation provides the height and weight for a set of 15 women ages 30 to 39. We want to predict weight from height. Having an equation

for predicting weight from height can help us to identify overweight or underweight individuals.

Example 1 (simple regression)

```
women
fit <- lm(weight ~ height, data=women)
summary(fit)
coefficients(fit)
confint(fit)
fitted(fit)
residuals(fit)

plot(women)
lines(women$height, fitted(fit))
```



weight
120
130
140
150
160
58
60
62
64
66
68
70
72
height

Polynomial regression

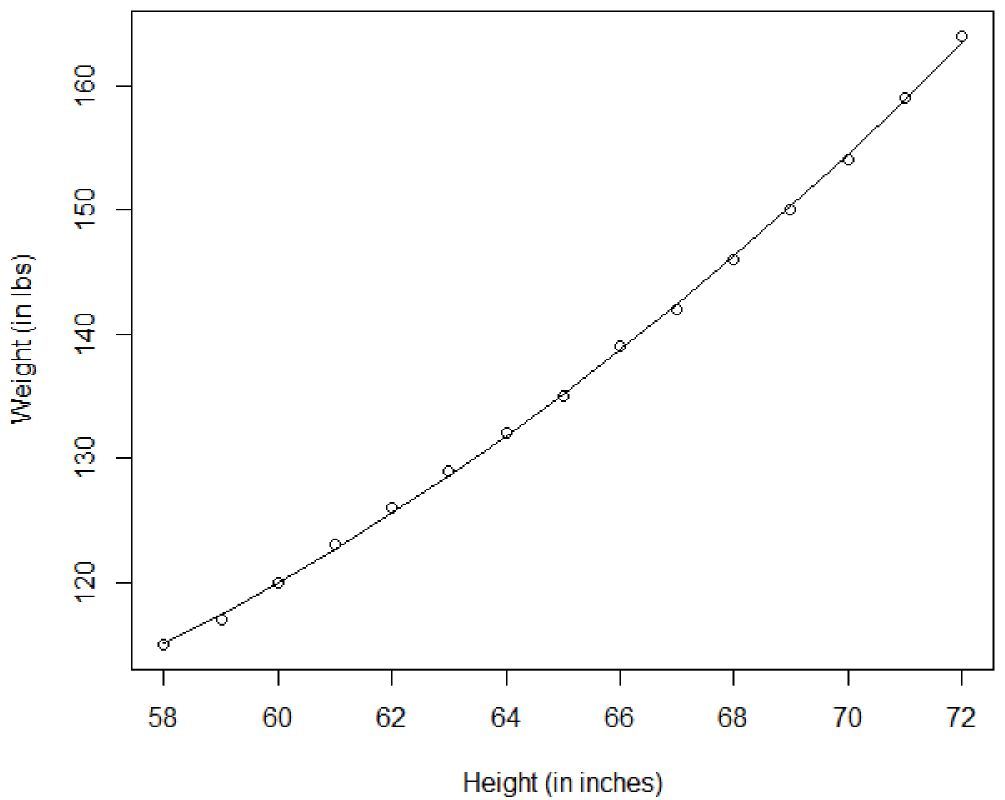
The above plot suggests that you might be able to improve your prediction using a regression with a quadratic term (that is, X^2).

You can fit a quadratic equation using the statement

```
fit2 <- lm(weight ~ height + I(height^2), data=women)
```

Example 2 (polynomial regression)

```
fit2 <- lm(weight ~ height + I(height^2), data=women)
summary(fit2)
plot(women, xlab="Height (in inches)", ylab="Weight (in lbs)")
lines(women$height, fitted(fit2))
```



Weight (in lbs)
120
130
140
150
160
58
60
62
64
66
68
70
72
Height (in inches)

linear relationship

Linear versus nonlinear models

Note that this polynomial equation still fits under the rubric of linear regression. It's linear because the equation involves a weighted sum of predictor variables (height and height-squared in this case). Even a model such as

$$\hat{Y}_i = \hat{\beta}_0 \times \log X_1 + \hat{\beta}_2 \times \sin X_2$$

would be considered a linear model (linear in terms of the parameters) and fit with the formula

$$Y \sim \log(X1) + \sin(X2)$$

In contrast, here's an example of a truly nonlinear model:

$$Y_i = \hat{\beta}_0 + \hat{\beta}_1 e^{\frac{X}{\beta_2}}$$

Nonlinear models of this form can be fit with the `nls()` function.

Multiple linear regression

When there's more than one predictor variable, simple linear regression becomes multiple linear regression, and the analysis becomes more involved. We'll use the state.x77 dataset in the base package for this example.

- response variable: a state's murder rate
- explanatory variables: population, illiteracy rate, average income, and frost levels (mean number of days below freezing).

Because the lm() function requires a data frame (and the state.x77 dataset is a matrix), we need to subset the original data and transform it to a data frame.

Example 3 (data transformation)

```
state.x77
class(state.x77)
states <- as.data.frame(state.x77[,c("Murder", "Population",
"Illiteracy", "Income", "Frost")])
states
class(states)
dim(states)
```

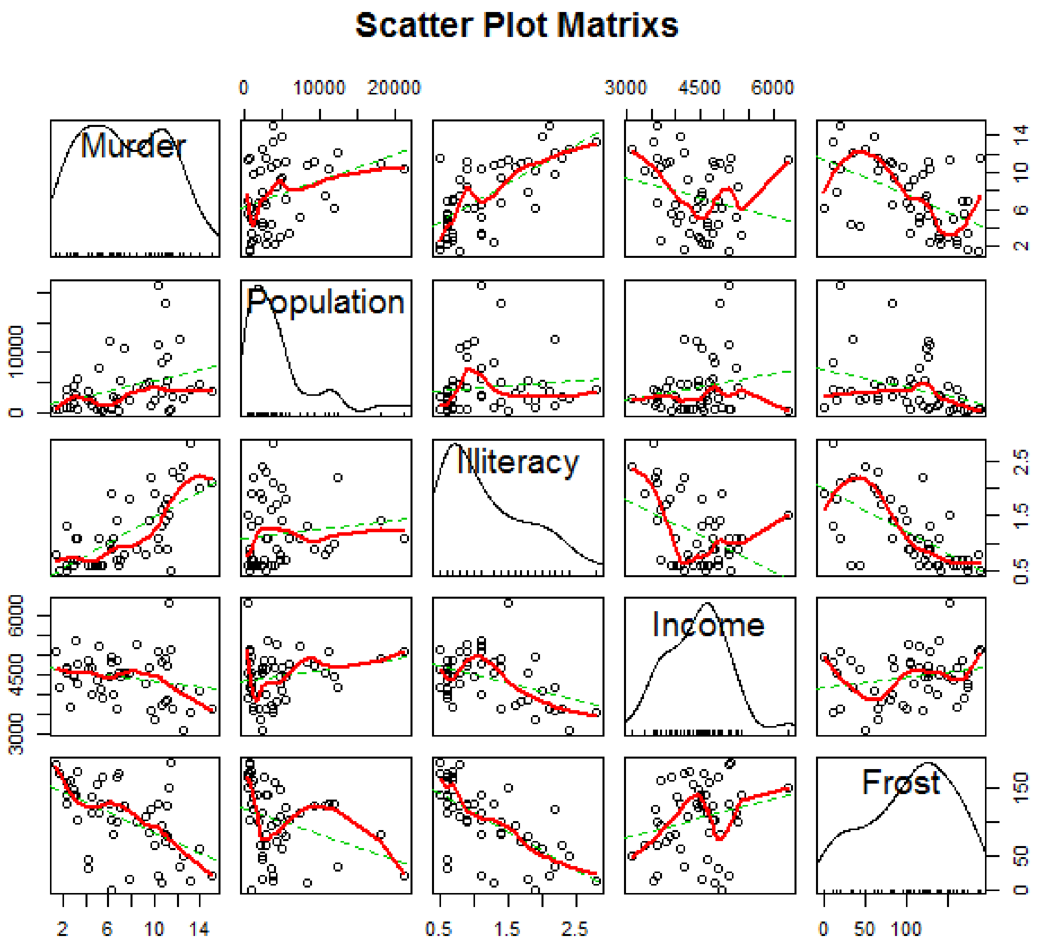
A good first step in multiple regression is to examine the relationships among the variables two at a time.

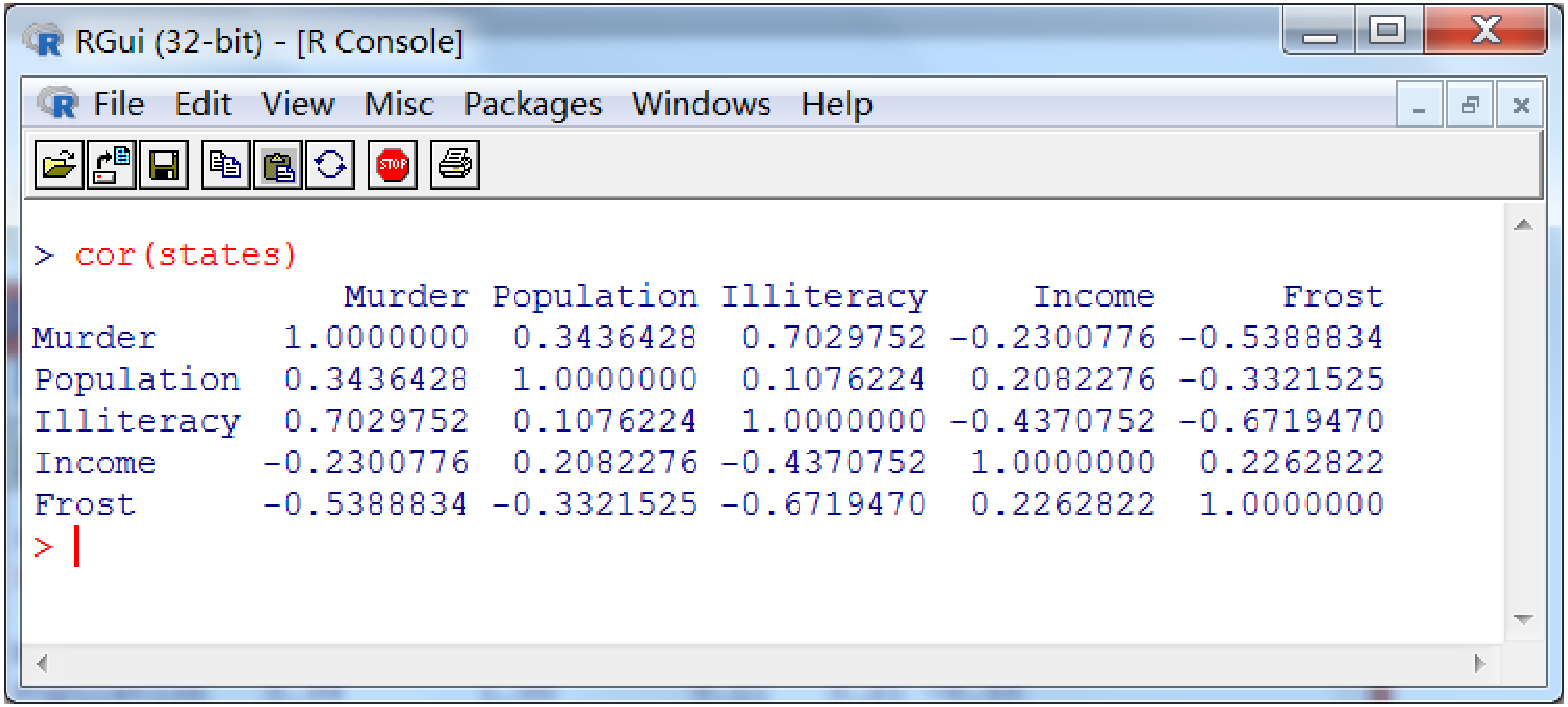
- scatterplot
- correlation matrix

Example 4 (multiple regression)

```
library(car)
scatterplotMatrix(states, spread=FALSE, lty=2, main="Scatter Plot Matrixs")
options(digits=2)
cor(states)
```

Scatterplot






```
> cor(states)
               Murder Population Illiteracy     Income      Frost
Murder      1.0000000  0.3436428  0.7029752 -0.2300776 -0.5388834
Population  0.3436428  1.0000000  0.1076224  0.2082276 -0.3321525
Illiteracy  0.7029752  0.1076224  1.0000000 -0.4370752 -0.6719470
Income     -0.2300776  0.2082276 -0.4370752  1.0000000  0.2262822
Frost      -0.5388834 -0.3321525 -0.6719470  0.2262822  1.0000000
> 
```

After we check that “murder rate” has medium or strong linear relationship with “population” and “ illiteracy”, we continue to do the linear regression.

Example 5 (linear regression)

```
fit &lt;- lm(Murder ~ Population + Illiteracy + Income + Frost,
data=states)
summary(fit)
coefficients(fit)
confint(fit)
residuals(fit)
```

RGui (32-bit) - [R Console]

File Edit View Misc Packages Windows Help

```
> fit <- lm(Murder ~ Population + Illiteracy + Income + Frost, data=states)
> summary(fit)

Call:
lm(formula = Murder ~ Population + Illiteracy + Income + Frost,
    data = states)

Residuals:
    Min      1Q  Median      3Q     Max
-4.7960 -1.6495 -0.0811  1.4815  7.6210

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 1.235e+00  3.866e+00   0.319   0.7510
Population  2.237e-04  9.052e-05   2.471   0.0173 *
Illiteracy  4.143e+00  8.744e-01   4.738 2.19e-05 ***
Income      6.442e-05  6.837e-04   0.094   0.9253
Frost       5.813e-04  1.005e-02   0.058   0.9541
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.535 on 45 degrees of freedom
Multiple R-squared:  0.567,     Adjusted R-squared:  0.5285
F-statistic: 14.73 on 4 and 45 DF,  p-value: 9.133e-08
```

RGui (32-bit) - [R Console]

File Edit View Misc Packages Windows Help

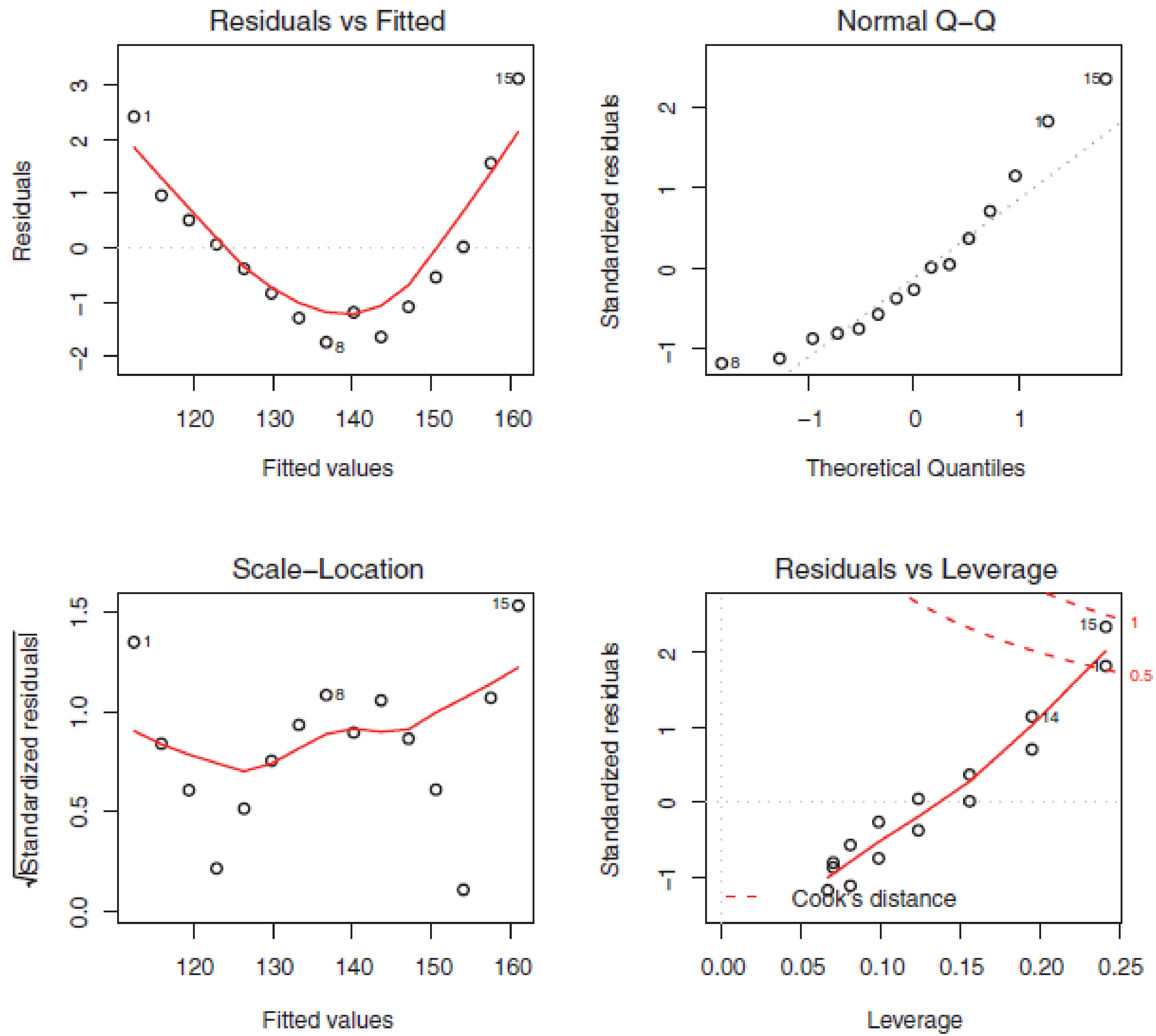
```
> coefficients(fit)
 (Intercept)   Population   Illiteracy       Income        Frost
1.2345634112 0.0002236754 4.1428365903 0.0000644247 0.0005813055
> confint(fit)
                    2.5 %       97.5 %
(Intercept) -6.552191e+00 9.0213182149
Population   4.136397e-05 0.0004059867
Illiteracy   2.381799e+00 5.9038743192
Income      -1.312611e-03 0.0014414600
Frost       -1.966781e-02 0.0208304170
> residuals(fit)
       Alabama         Alaska        Arizona       Arkansas     California
    4.11179210     3.27433977    -1.68700264     0.26668056    -0.57424792
      Colorado    Connecticut       Delaware        Florida        Georgia
    1.68594493    -3.81042204     0.73768277     1.91178879     2.97838044
        Hawaii          Idaho       Illinois        Indiana           Iowa
   -3.41984294     1.05927673     2.42954793     1.41893921    -2.02545720
        Kansas       Kentucky      Louisiana          Maine       Maryland
   -0.09731294     1.68494109    -0.72117551    -2.00277259     2.21479548
 Massachusetts       Michigan      Minnesota    Mississippi       Missouri
   -4.15834611     3.72023253    -2.69149081     0.57035176     3.34806321
       Montana       Nebraska         Nevada  New Hampshire     New Jersey
    0.74271628    -1.53684814     7.62104160    -1.39312273    -2.63613735
    New Mexico       New York North Carolina   North Dakota           Ohio
```

Regression diagnostics

summary() function outputs the model parameters and other statistics. But it does not tells us if the model we fit is appropriate.

Our confidence in inference of regression parameters depends on the degree to which the data have met the statistical assumptions of the OLS model.

```
> fit <- lm(weight ~ height, data=women)
> par(mfrow=c(2,2))
> plot(fit)
```



Residuals vs Fitted
Residuals
Fitted values
Normal Q−Q
Standardized residuals
Theoretical Quantiles
Scale−Location
√|Standardized residuals|
Fitted values
Residuals vs Leverage
Standardized residuals
Cook's distance
Leverage

car package

Table 8.4 Useful functions for regression diagnostics (`car` package)

Function	Purpose
`qqPlot()`	Quantile comparisons plot
`durbinWatsonTest()`	Durbin–Watson test for autocorrelated errors
`crPlots()`	Component plus residual plots
`ncvTest()`	Score test for nonconstant error variance
`spreadLevelPlot()`	Spread-level plot
`outlierTest()`	Bonferroni outlier test
`avPlots()`	Added variable plots
`influencePlot()`	Regression influence plot
`scatterplot()`	Enhanced scatter plot
`scatterplotMatrix()`	Enhanced scatter plot matrix
`vif()`	Variance inflation factors

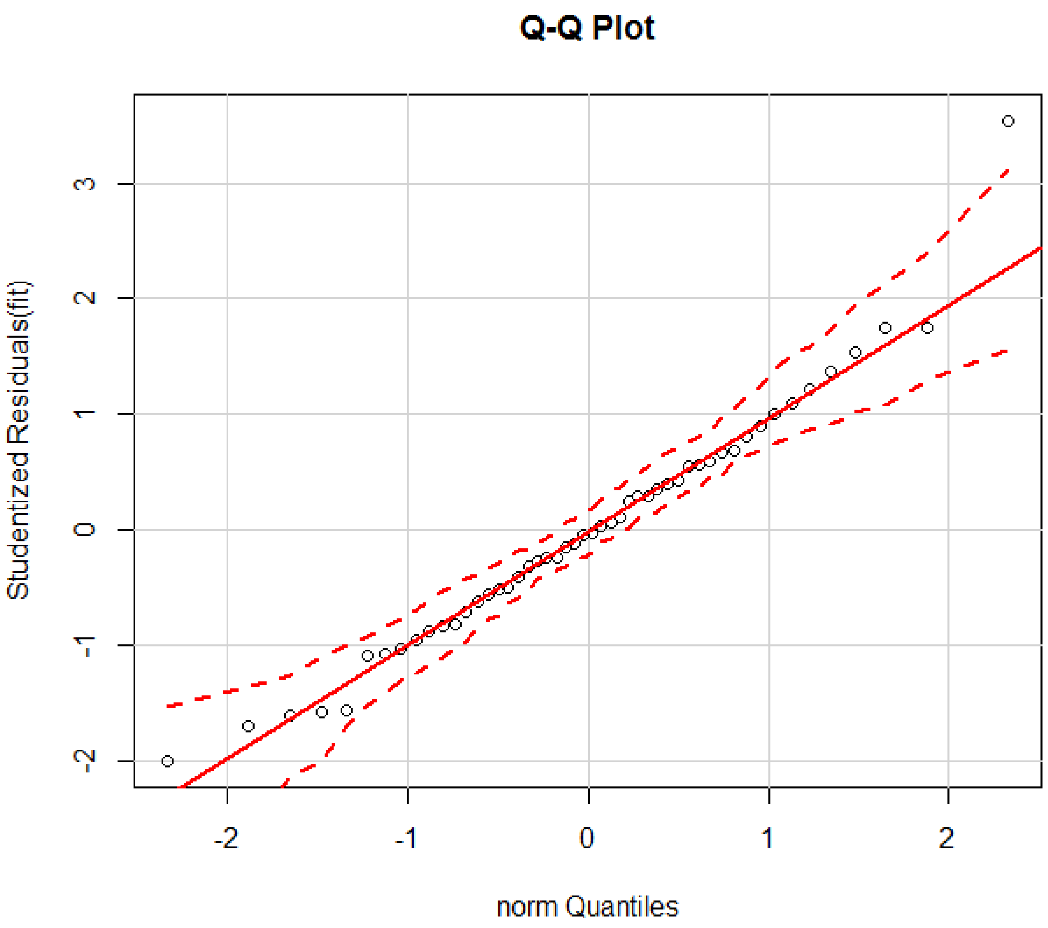
Normality

The qqPlot() function provides an accurate method of assessing the normality assumption.

Example 6 (checking normality)

```
library(car)
fit <- lm(Murder ~ Population + Illiteracy + Income + Frost, data=states)
qqPlot(fit, distribution="norm", labels=row.names(states),
simulate=TRUE, main="Q-Q Plot")
```

QQ plot



Independence of errors

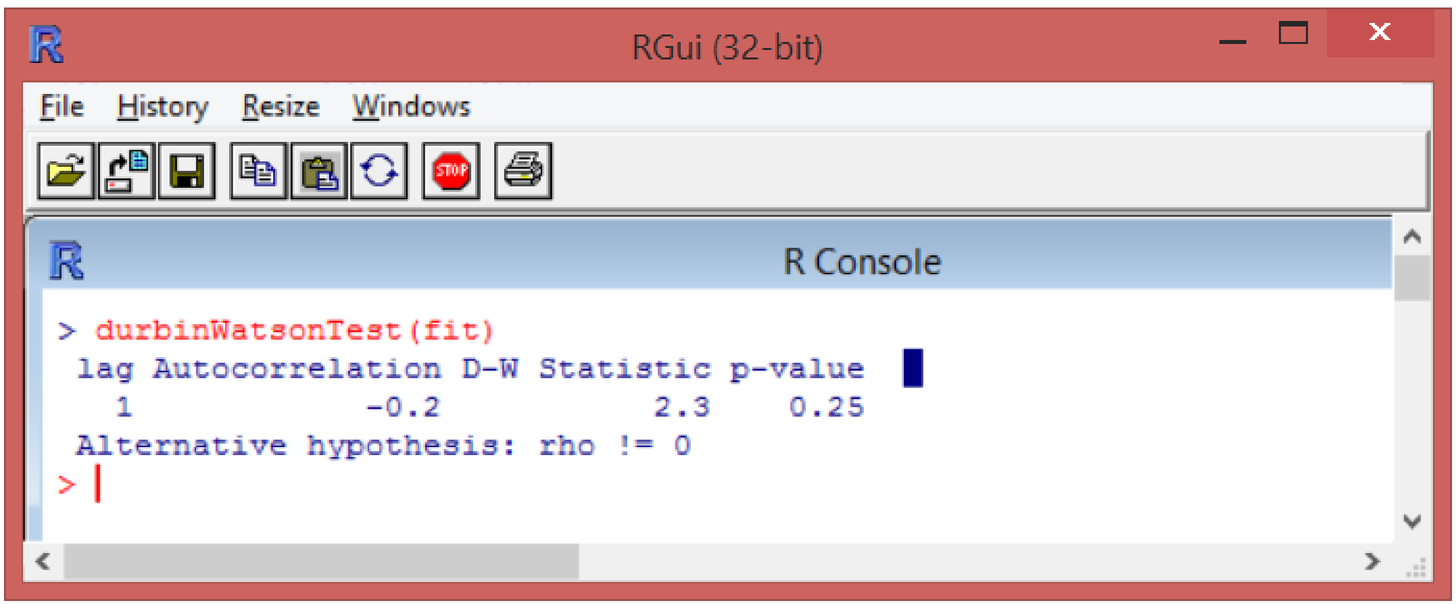
Time series data will often display autocorrelation. Observations collected closer in time will be more correlated with each other than with observations distant in time. We can apply the Durbin–Watson test to the multiple regression problem.

$$DW = \frac{(\varepsilon_2 - \varepsilon_1)^2 + \ldots + (\varepsilon_n - \varepsilon_{n-1})^2}{\sum_{i=1}^{n} \varepsilon_i^2} \approx 2(1 - r)$$

Example 6 (checking independence)

durbinWatsonTest(fit)

Output:



Homoscedasticity

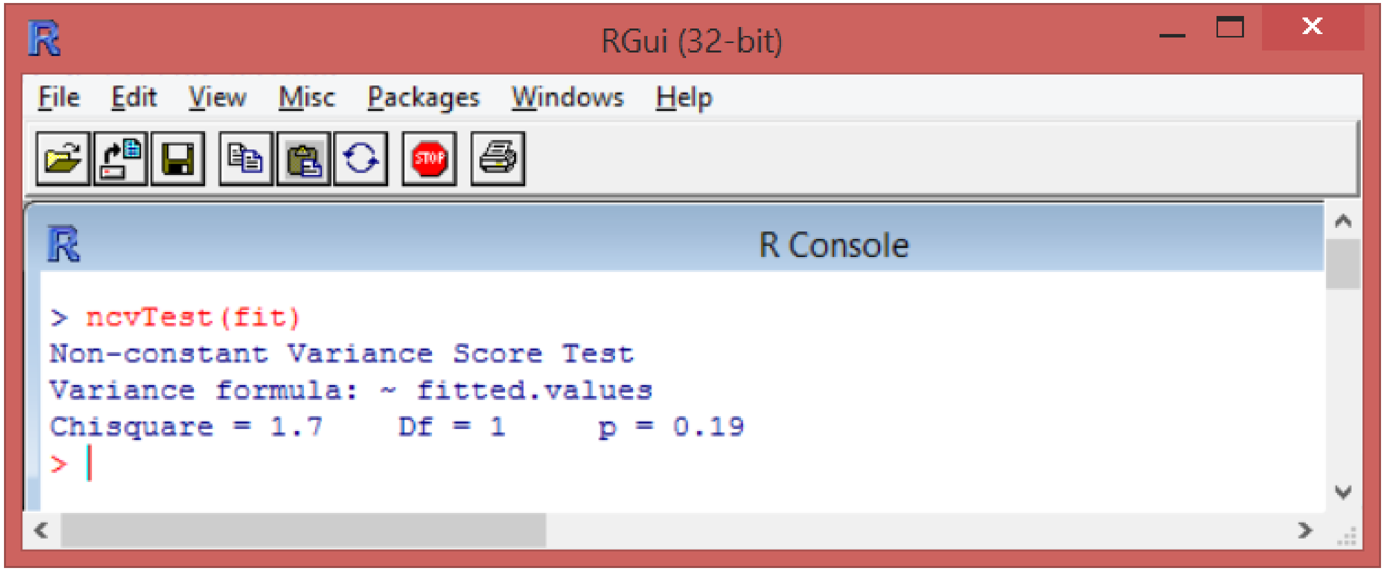
The car package also provides two useful functions for identifying non-constant error variance.

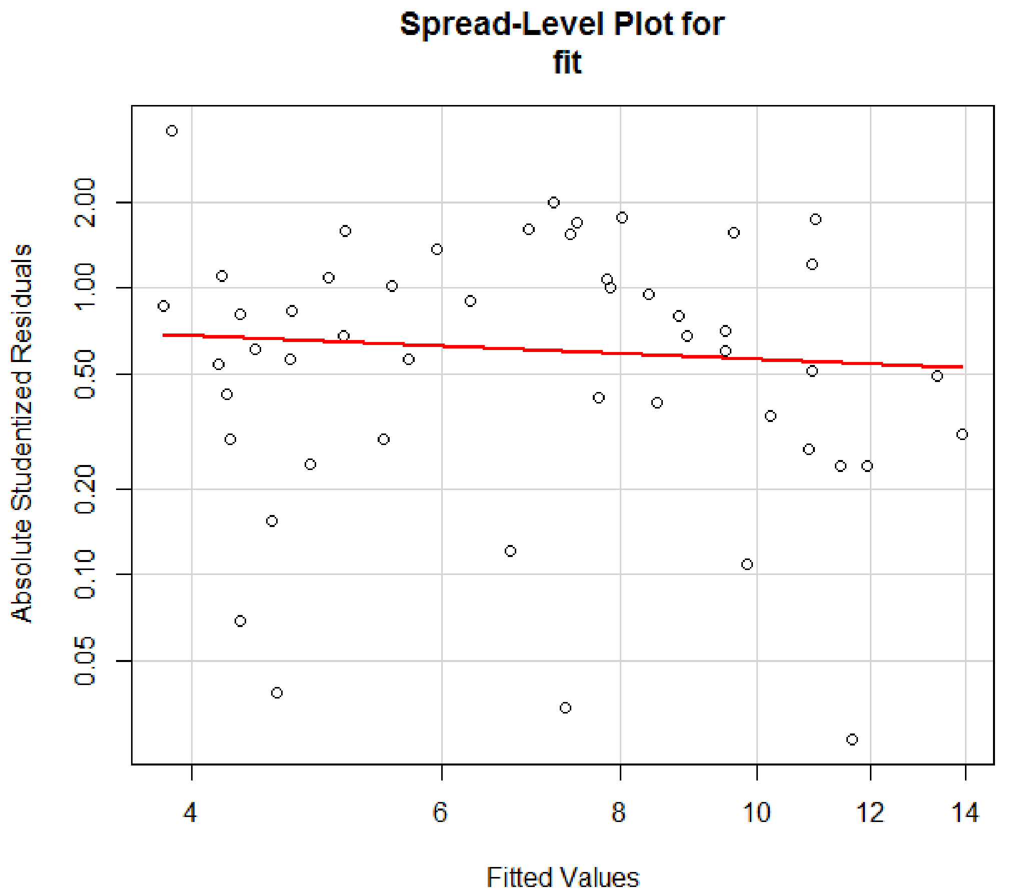
- ncvTest()
- spreadLevelPlot()

Example 7 (checking homoscedasticity)

ncvTest(fit)

spreadLevelPlot(fit)




Spread-Level Plot for
fit
Absolute Studentized Residuals
0.05
0.10
0.20
0.50
1.00
2.00
4
6
8
10
12
14
Fitted Values

Summary

In this session, we have learned

- How to fit linear/polynomial/multiple regression model.
- Check the regression results by using summary() function.
- How to test the model assumptions: normality, independence and homoscedasticity.