DSCI353-353m-453: Class 11b Linear Regression Overview Part 1

Profs: R. H. French, L. S. Bruckman, P. Leu, K. Davis, S. Cirlos

TAs: W. Oltjen, K. Hernandez, M. Li, M. Li, D. Colvin

10 November, 2022

Contents

11.2.1.1 Class Readings, Assignments, Syllabus Topics

11.2.1.1.1 Reading, Lab Exercises, SemProjects

- Readings:
  - For today: ISLR 2nd Ed. 1, 2.1, 2.2
  - For next class: OIS9
- Laboratory Exercises:
  - LE5 : Due Today Thursday Nov. 10th
  - LE6 : Given Out today or tomorrow
  - LE6 : Due Tuesday Nov. 22nd
- Office Hours: (Class Canvas Calendar for Zoom Link)
  - Mondays @ 4:30 PM to 5:30 PM
  - Wednesdays @ 4:30 PM to 5:30 PM
  - **Office Hours are on Zoom, and recorded**
- Semester Projects
  - DSCI 451 Students **Biweekly Update 6 Due November 18th**
  - DSCI 451 Students
    - Next **Report Out #3 is Due Friday November 25th**
  - All DSCI 351/351M/451 Students:
    - Peer Grading of Report Out #3 is Due Tuesday Nov. 29th
- Exams
  - Final: Monday 12/13/2021, 12:00PM - 3:00PM, Nord 356 or remote

11.2.1.1.2 Textbooks Introduction to R and Data Science

- For R, Coding, Inferential Statistics
  - Peng: R Programming for Data Science
  - Peng: Exploratory Data Analysis with R

Textbooks for this class

- OIS = Diez, Barr, Çetinkaya-Runde: Open Intro Stat v4
- R4DS = Wickham, Grolemund: R for Data Science

Textbooks for DSCI353/353M/453, And in your Repo now

- ISLR = James, Witten, Hastie, Tibshirani: Intro to Statistical Learning with R
- ESL = Trevor Hastie, Tibshirani, Friedman: Elements of Statistical Learning
- DLwR = Chollet, Allaire: Deep Learning with R

Magazine Articles about Deep Learning

- DL1 to DL6 are "Deep Learning" articles in 3-readings/2-articles/

11.2.1.1.3 Syllabus

11.2.1.2 Linear Regression Overview

- Here we'll look at three important topics in data science
  - Fitting and interpreting linear models
  - Evaluating model assumptions
  - Selecting among competing models

Because regression analysis plays

- such a central role in modern statistics,
  - we'll cover it in detail.
- First, we'll look at how to fit and interpret regression models.

Day:Date	Foundation	Practicum	Reading	Due
w01a:Tu:8/30/22	ODS Tool Chain	R, Rstudio, Git		
w01b:Th:9/1/22	Setup ODS Tool Chain	Bash, Git, Slack, Agile	PRP4-33	LE1
w02a:Tu:9/6/22	Bash-Git-Knuth-Lit.Prog.	RIntroR	PRP35-64	
w02b:Th:9/8/22	What is Data Science	OIS:Intro2R	OIS1,2	
w02Pr:Fr:9/9/22			PRP65-93	**451 Update1**
w03a:Tu:9/13/22	Data Intro	Data Analytic Style	PRP94-116	LE2 **LE1 Due**
w03b:Th:9/15/22	Rand. Var. Normal Dist.	Git, Rmds, Loops	OIS4	
w04a:Tu:9/20/22	Tidy Check Explore	Tidy GapMinder	EDA1-31	
w04b:Th:9/22/22	Inference, DSCI Process	Other Distrib. 7 ways	R4DS1-3	LE3 **LE2 Due**
w04Pr:Fr:9/23/22			EDA32-58	**451 Update2**
w05a:Tu:9/27/22	OIS4 Rand. Var.	EDA of PET Degr.	OIS5	
w05b:Th:9/29/22	OIS5 Found. of Infer.	Multivar Corr. Plot	R4DS4-6	
w05Pr:Fr:9/30/22				**451 RepOut1**
w06a:Tu:10/4/22	Pred., Algorithm, Model		R4DS7-8	
w06b:Th:10/6/22	Summ. Stats & Vis.	Anscombe's Quartets	R4DS9-16	LE4 **LE3 Due**
w06Pr:Fr:10/7/22				**451 Update3**
w07a:Tu:10/11/22	Midterm Rev. Tidy Data	Correl Plots Summ Stats	OIS6.1-2	**PeerRv1 Due**
w07b:Th:10/13/22	HypoTest, Infer. Recap	Penguin EDA, Sampling		
w08a:Tu:10/18/22	**MIDTERM**	**EXAM**		
w08b:Th:10/20/22	Programming & Coding	Code Packaging		**LE4 Due**
w08Pr:Fr:10/21/22				**451 Update4**
Tu:10/24,25	**CWRU**	**FALL BREAK**	R4DS17-21	
w09b:Th:10/27/22	Cat. Inf. 1 & 2 propor.	Indep. Test,2-way tables	OIS6.3-4	LE5
w09Pr:Fr:10/28/22				**451 RepOut2**
w10a:Tu:11/1/22	Goodness of Fit, χ^2 test	t-tests 1&2 means	OIS7.1-4	
w10b:Th:11/3/22	Num. Infer, Cont. Tables	Stat. Power		
w10Pr:Fr:11/4/22				**451 Update5**
w11a:Tu:11/8/22	Sample & Effect Size	Stat. Power GGmap	OIS8	**PeerRv2 Due**
w11b:Th:11/10/22	Inf. 4 Regr, Test & Train	Curse of Dimen.	ISLR1,2.1,2	LE6 **LE5 Due**
w12a:Tu:11/15/22	Lin. Regr. Part 1	Residuals	OIS9	
w12b:Th:11/17/22	Lin. Regr. Part 2	Regr. Diagnostics		
w12Pr:Fr:11/18/22				**451 Update6**
w13a:Tu:11/22/22	Mult. Lin. Regr.	Var. & Mod. Selec.,	ISLR3.1	LE7 **LE6 due**
w13b:Th:11/24/22	Log. Regr.	GIS Trends	ISLR3.2	
w13Pr:Fr:11/25/22				**451 RepOut3**
w14a:Tu:11/23/22	Classificat., Sup. Lrning	Caret, Broom 4 modeling	ISLR4.1-3	
Th,Fr:11/24,25	**THANKSGIVING**	**Vacation**		
w15a:Tu:11/29/22		Clustering		**PeerRv3 Due**
w15b:Th:12/1/22	Big Data Analytics	Dist. Comp., Hadoop		
w15SPr:Fr:12/2/22		Read Article by	Mirletz,2015	
w16a:Tu:12/6/22	Final Exam Review			
w15b:Th:12/8/22				**LE7 due**
Friday 12/12	**SemProj**	**Final Report**		**SemProj4 due**
Monday 12/19	**FINAL EXAM**	**12:00-3:00pm**	Nord 356	or remote

Figure 1: DSCI351-351M-451 Syllabus

- Next, we'll review a set of techniques
  - for identifying potential problems with these models
  - and how to deal with them.
- Third, we'll explore the issue of variable selection.
  - Of all the potential predictor variables available,
  - how do you decide which ones to include in your final model?
- Fourth, we'll address the question of generalizability.
  - How well will your model work
  - when you apply it in the real world?
- Finally, we'll consider relative importance
  - of all the predictors in your model,
  - which one is the most important,
  - the second most important,
  - and the least important?

Effective regression analysis is

- an interactive, holistic process with many steps,
  - and it involves more than a little skill.
- Rather than break it up into multiple parts,
  - We'll learn this as an integrated process
  - So as to understand the multi-faceted nature
  - of regression modeling

11.2.1.2.1 The many faces of regression

- The term regression can be confusing
  - because there are so many specialized varieties
    - See table 8.1
  - In addition, R has powerful and comprehensive
    - packages for fitting regression models,
    - and this abundance of options can be confusing as well.
  - For example, in 2005, Vito Ricci
    - created a list of more than 205 functions in R
    - that are used to generate regression analyses
    - Regression RefCard Vito Ricci

We'll focus on regression methods

- that fall under the rubric of
  - ordinary least squares (OLS) regression,
- including simple linear regression,
- polynomial regression, and
- multiple linear regression.

OLS regression is the most common variety of statistical analysis today.

Other types of regression models

- including logistic regression
  - which is really a bi-classification method
- will be covered separately

11.2.1.2.2 Scenarios for using OLS regression

- In OLS regression,
  - a quantitative dependent variable (the **response**)

Table 8.1 Varieties of regression analysis

Type of regression	Typical use
Simple linear	Predicting a quantitative response variable from a quantitative explanatory variable.
Polynomial	Predicting a quantitative response variable from a quantitative explanatory variable, where the relationship is modeled as an *n*th order polynomial.
Multiple linear	Predicting a quantitative response variable from two or more explanatory variables.
Multilevel	Predicting a response variable from data that have a hierarchical structure (for example, students within classrooms within schools). Also called *hierarchical*, *nested*, or *mixed* models.
Multivariate	Predicting more than one response variable from one or more explanatory variables.
Logistic	Predicting a categorical response variable from one or more explanatory variables.
Poisson	Predicting a response variable representing counts from one or more explanatory variables.
Cox proportional hazards	Predicting time to an event (death, failure, relapse) from one or more explanatory variables.
Time-series	Modeling time-series data with correlated errors.
Nonlinear	Predicting a quantitative response variable from one or more explanatory variables, where the form of the model is nonlinear.
Nonparametric	Predicting a quantitative response variable from one or more explanatory variables, where the form of the model is derived from the data and not specified a priori.
Robust	Predicting a quantitative response variable from one or more explanatory variables using an approach that's resistant to the effect of influential observations.

Figure 2: Table 8.1

        * is predicted from a weighted sum of predictor variables,
        * where the weights are parameters estimated from the data.
    - Often **predictors** are also called **features**

Let's look at an example, about concrete (fwa, 2006)

- An engineer wants to identify the most important factors
    - related to bridge deterioration
- such as
    - age,
    - traffic volume,
    - bridge design,
    - construction materials and methods,
    - construction quality,
    - and weather conditions)
- and determine the mathematical form of these relationships.

She collects data on each of these variables

- from a representative sample of bridges
- and models the data using OLS regression.

The approach is highly interactive.

- She fits a series of models,
- checks their compliance with underlying statistical assumptions,
- explores any unexpected or aberrant findings,
- and finally chooses the "best" model
    - from among many possible models.

If successful, the results will help her to:

- Focus on important variables,
    - by determining which of the many collected variables
    - are useful in predicting bridge deterioration,
    - along with their relative importance.
- Look for bridges that are likely to be in trouble,
    - by providing an equation that can be used
    - to predict bridge deterioration for new cases
        * (where the values of the predictor variables are known,
        * but the degree of bridge deterioration isn't).
- Take advantage of serendipity,
    - by identifying unusual bridges.
- If she finds that some bridges
    - deteriorate much faster or slower than predicted by the model,
    - a study of these **outliers** may yield important findings
    - that could help her to understand
        * the mechanisms involved in bridge deterioration.

11.2.1.2.3 OLS Regression

- We'll be predicting the response variable
    - from a set of predictor variables using OLS
        * This is also called regressing the response variable
        * on the predictor variables.

OLS regression fits models of the form

- where n is the number of observations

- and k is the number of predictor variables.

We can represent this model by

$$\hat{Y}_i = \beta_0 + \beta_{1i}X_{1i} + ... + \beta_{ki}X_{ki}, i = 1...n$$

$\hat{Y}_i$ is the predicted value of the dependent variable for observation i (specifically, it's the estimated mean of the Y distribution, conditional on the set of predictor values).

X_{ij} is the *j*th predictor value for the *i*th observation.

$\hat{\beta}_0$ is the intercept (the predicted value of Y when all the predictor variables equal zero).

$\hat{\beta}_j$ is the regression coefficient for the jth predictor (slope representing the change in Y for a unit change in X_j).

Our goal is to select 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:

$$Y_i - \hat{Y}_i = (Y_i - \hat{\beta}_0 + \hat{\beta}_k X_{ki} + ... + \hat{\beta}_1 X_{1i} = \epsilon_i^2)$$

To properly interpret the coefficients of the OLS model,

- you must satisfy a number of statistical assumptions:
- **Normality**
  - For fixed values of the independent variables,
    - the dependent variable is normally distributed.
- **Independence**
  - The Y_i values are independent of each other.
- **Linearity**
  - The dependent variable is linearly related
    - to the independent variables.
- **Homoscedasticity**
  - The variance of the dependent variable
    - doesn't vary with the levels of the independent variables.

If you violate these assumptions,

- your statistical significance tests
  - and confidence intervals
- may not be accurate.

Note that OLS regression also assumes

- that the independent variables

  - are fixed and measured without error,
  - but this assumption is typically relaxed in practice.

Fitting regression models with lm()

- In R, the basic function for fitting a linear model is `lm()`
  - `myfit <- lm(formula, data)`
  - where formula describes the model to be fit
    - and data is the data frame containing the data
    - to be used in fitting the model.

The resulting object (myfit, in this case)

- is a list that contains extensive information about the fitted model.

The formula is typically written as

- `Y ~ X1 + X2 + ... + Xk`
- where the `~` separates the response variable on the left
  - from the predictor variables on the right,
- and the predictor variables are separated by + signs.

Other symbols can be used

- to modify the formula in various ways (see table 8.2).

In addition to lm(), table 8.3 lists several functions

- that are useful
  - when generating a simple or multiple regression analysis.
- Each of these functions
  - is applied to the object returned by lm()
  - in order to generate additional information based on that fitted model.

When the regression model contains

- one dependent variable
  - and one independent variable,
  - the approach is called simple linear regression.
- When there's one predictor variable
  - but powers of the variable are included
  - (for example, X, X^2, X^3),
  - it's called polynomial regression.
- When there's more than one predictor variable,
  - it's called multiple linear regression.

Simple Linear Regression

- We'll start with an example of simple linear regression,
  - then progress to examples of polynomial
    - and multiple linear regression,
  - and end with an example of multiple regression
    - that includes an interaction among the predictors.

```
fit <- lm(weight ~ height, data = women)

summary(fit)
```

```
##
## Call:
## lm(formula = weight ~ height, data = women)
```

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 placeholder 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.

Figure 3: R formula notation

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% 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

Figure 4: functions to use with model objects

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7333 -1.1333 -0.3833  0.7417  3.1167
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -87.51667    5.93694  -14.74 1.71e-09 ***
## height        3.45000    0.09114   37.85 1.09e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.525 on 13 degrees of freedom
## Multiple R-squared:  0.991,  Adjusted R-squared:  0.9903
## F-statistic:  1433 on 1 and 13 DF,  p-value: 1.091e-14
```

```
women$weight
```

```
##  [1] 115 117 120 123 126 129 132 135 139 142 146 150 154 159 164
```

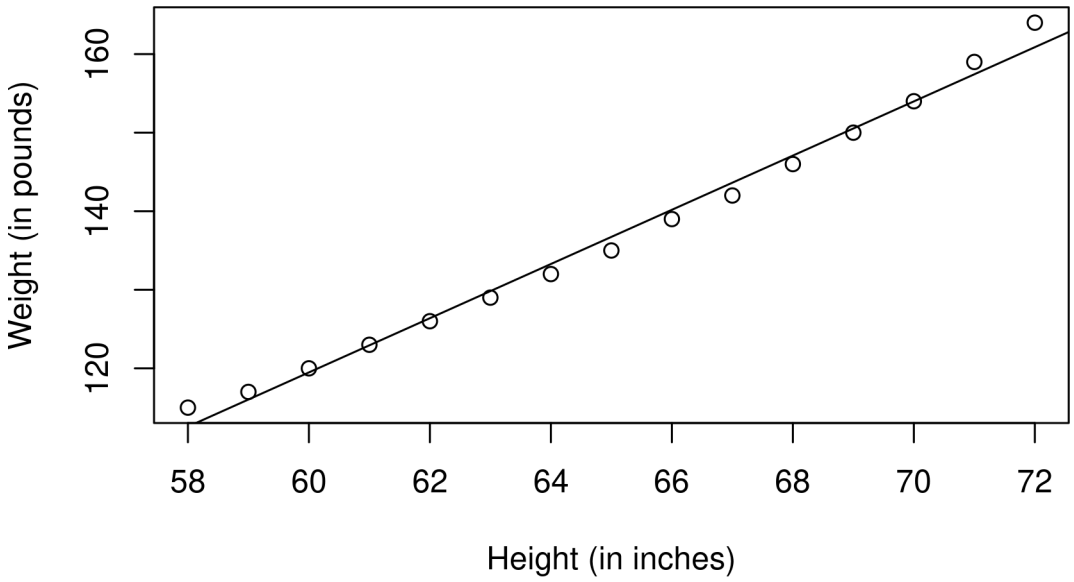
```
fitted(fit)
```

```
##        1        2        3        4        5        6        7        8
## 112.5833 116.0333 119.4833 122.9333 126.3833 129.8333 133.2833 136.7333
##        9       10       11       12       13       14       15
## 140.1833 143.6333 147.0833 150.5333 153.9833 157.4333 160.8833
```

```
residuals(fit)
```

```
##           1           2           3           4           5           6
##  2.41666667  0.96666667  0.51666667  0.06666667 -0.38333333 -0.83333333
##           7           8           9          10          11          12
## -1.28333333 -1.73333333 -1.18333333 -1.63333333 -1.08333333 -0.53333333
##          13          14          15
##  0.01666667  1.56666667  3.11666667
```

```
plot(women$height, women$weight,
     xlab = "Height (in inches)",
     ylab = "Weight (in pounds)")
abline(fit)
```



From the output, we see that the prediction equation is

$$W = -87.52 + 3.45xH$$

Because a height of 0 is impossible,

- you wouldn't try to give a physical interpretation to the intercept.
- It merely becomes an adjustment constant.

From the Pr(>|t|) column,

- you see that the regression coefficient (3.45)
  - is significantly different from zero ($p < 0.001$)
- and indicates that there's an expected increase
  - of 3.45 pounds of weight 0 for every 1 inch increase in height.

The multiple R-squared (0.991)

- indicates that the model accounts for 99.1% of the variance in weights.

The multiple R-squared is

- also the squared correlation
  - between the actual and predicted value
- that is, $R^2 = r_{\hat{y}y}$

The residual standard error (1.53 pounds)

- can be thought of as the average error
  - in predicting weight from height using this model.

The F statistic tests whether

- the predictor variables, taken together,
  - predict the response variable above chance levels.
- Because there's only one predictor variable in simple regression,
  - in this example the F test
  - is equivalent to the t-test for the regression coefficient for height.

11.2.1.3 A quick paced exposure to Linear Regression and Diagnostics

11.2.1.3.1 Simple Linear Regression

- The dataset women in the base installation
  - provides the height and weight
    - for a set of 15 women
    - ages 30 to 39.

Suppose you want to predict weight from height.

Having an equation for predicting weight from height

- can help you to identify overweight or underweight individuals.

The analysis is provided in the following listing,

- and the resulting graph is shown

```
fit <- lm(weight ~ height, data = women)
summary(fit)
```

```
##
## Call:
## lm(formula = weight ~ height, data = women)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7333 -1.1333 -0.3833  0.7417  3.1167
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -87.51667    5.93694  -14.74 1.71e-09 ***
## height        3.45000    0.09114   37.85 1.09e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.525 on 13 degrees of freedom
## Multiple R-squared:  0.991,  Adjusted R-squared:  0.9903
## F-statistic:  1433 on 1 and 13 DF,  p-value: 1.091e-14
```

```
women$weight
```

```
##  [1] 115 117 120 123 126 129 132 135 139 142 146 150 154 159 164
```

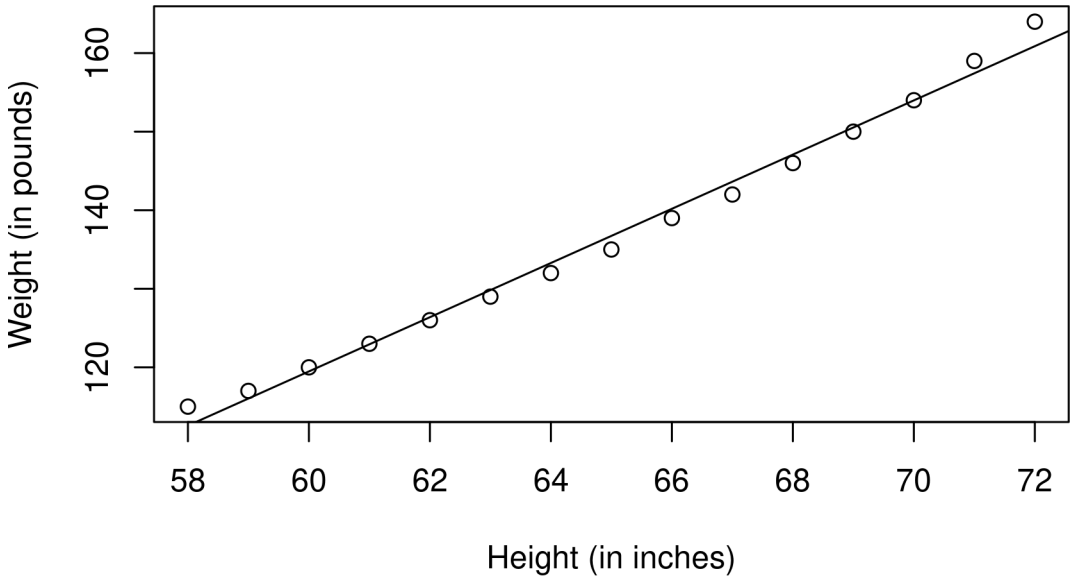
```
fitted(fit)
```

```
##        1        2        3        4        5        6        7        8
## 112.5833 116.0333 119.4833 122.9333 126.3833 129.8333 133.2833 136.7333
##        9       10       11       12       13       14       15
## 140.1833 143.6333 147.0833 150.5333 153.9833 157.4333 160.8833
```

```
residuals(fit)
```

```
##           1           2           3           4           5           6
##  2.41666667  0.96666667  0.51666667  0.06666667 -0.38333333 -0.83333333
##           7           8           9          10          11          12
## -1.28333333 -1.73333333 -1.18333333 -1.63333333 -1.08333333 -0.53333333
##          13          14          15
##  0.01666667  1.56666667  3.11666667
```

```
plot(women$height, women$weight,
     xlab = "Height (in inches)",
     ylab = "Weight (in pounds)")
abline(fit)
```



From the output, you see that the prediction equation is

- $ weight = -87.52 + 3.45 x height $

Because a height of 0 is impossible,

- you wouldn't try to give a physical interpretation
  - to the intercept.
- It merely becomes an adjustment constant.

11.2.1.3.2 Polynomial Regression

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

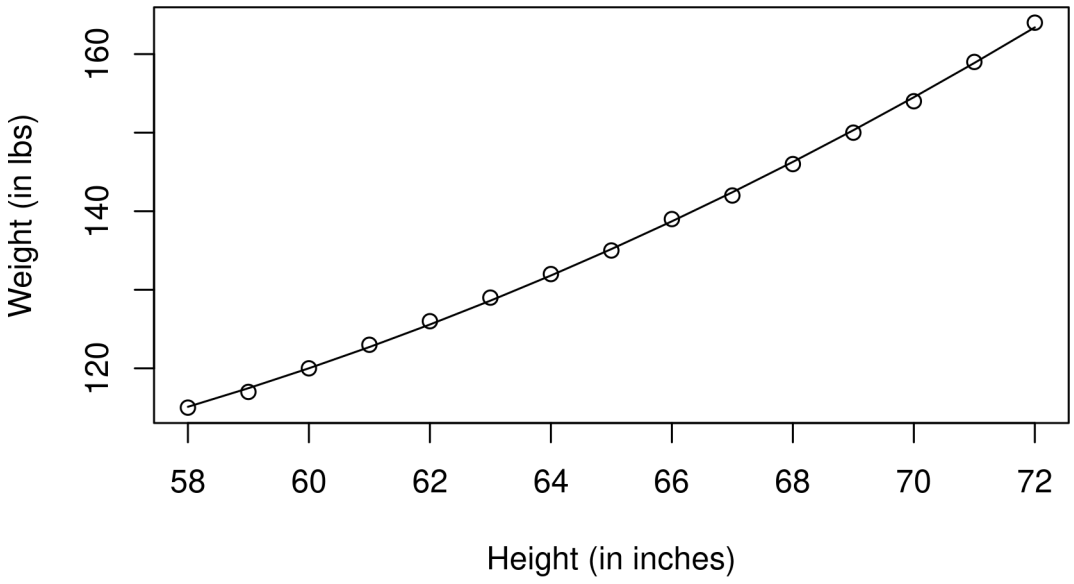
You can fit a quadratic equation using the statement

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

```
##
## Call:
## lm(formula = weight ~ height + I(height^2), data = women)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.50941 -0.29611 -0.00941  0.28615  0.59706
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 261.87818   25.19677  10.393 2.36e-07 ***
## height       -7.34832    0.77769  -9.449 6.58e-07 ***
## I(height^2)   0.08306    0.00598  13.891 9.32e-09 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3841 on 12 degrees of freedom
## Multiple R-squared:  0.9995, Adjusted R-squared:  0.9994
## F-statistic: 1.139e+04 on 2 and 12 DF,  p-value: < 2.2e-16
```

```
plot(women$height, women$weight,
     xlab = "Height (in inches)",
     ylab = "Weight (in lbs)")
lines(women$height, fitted(fit2))
```



11.2.1.3.3 Examining bivariate relationships

- When there's more than one predictor variable,
  - simple linear regression
    - becomes multiple linear regression,
  - and the analysis grows more involved.

Technically, polynomial regression is

- a special case of multiple regression.

Quadratic regression has two predictors (x and X^2), and cubic regression has three predictors (X, X^2, and x^3).

Let's look at a more general example.

We'll use the state.x77 dataset in the base package for this example.

Suppose you want to explore the relationship

- between a state's murder rate
  - and other characteristics of the state,
- including
  - population,
  - illiteracy rate,
  - average income, and
  - frost levels

∗ (mean number of days below freezing).

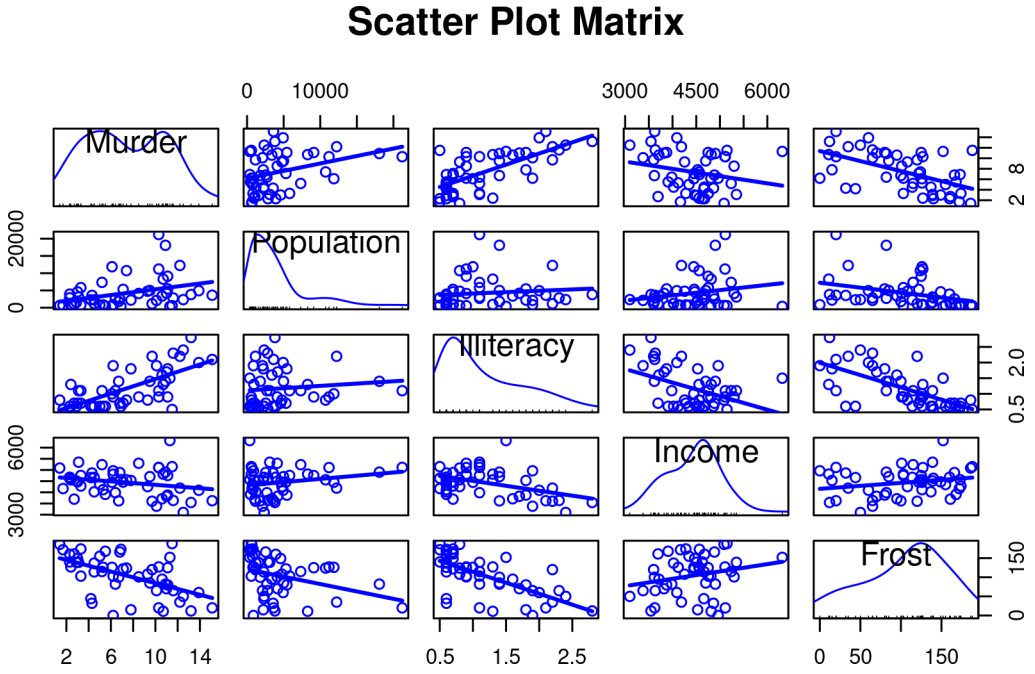
```
states <- as.data.frame(state.x77[, c("Murder", "Population",
                                      "Illiteracy", "Income", "Frost")])
cor(states)
```

```
##                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
```

```
library(car)
```

```
## Loading required package: carData
```

```
scatterplotMatrix(states, smooth = FALSE,
                  main = "Scatter Plot Matrix")
```



11.2.1.3.4 Multiple linear regression

- Now let's fit the multiple regression model with the lm() function

When there's more than one predictor variable,

- the regression coefficients indicate
  - the increase in the dependent variable
  - for a unit change in a predictor variable,
  - holding all other predictor variables constant.

For example, the regression coefficient for Illiteracy

- is 4.14,
- suggesting that an increase of 1% in illiteracy
  - is associated with a 4.14% increase
- in the murder rate,

- controlling for population, income, and temperature

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

11.2.1.3.5 Mutiple linear regression with a significant interaction term

- Some of the most interesting research findings
  - are those involving interactions among predictor variables.

Consider the automobile data in the mtcars data frame.

Let's say that you're interested in

- the impact of automobile weight and horsepower
  - on mileage.

You could fit a regression model that includes both predictors,

- along with their interaction,
- as shown here

```
fit <- lm(mpg ~ hp + wt + hp:wt, data = mtcars)
summary(fit)
```

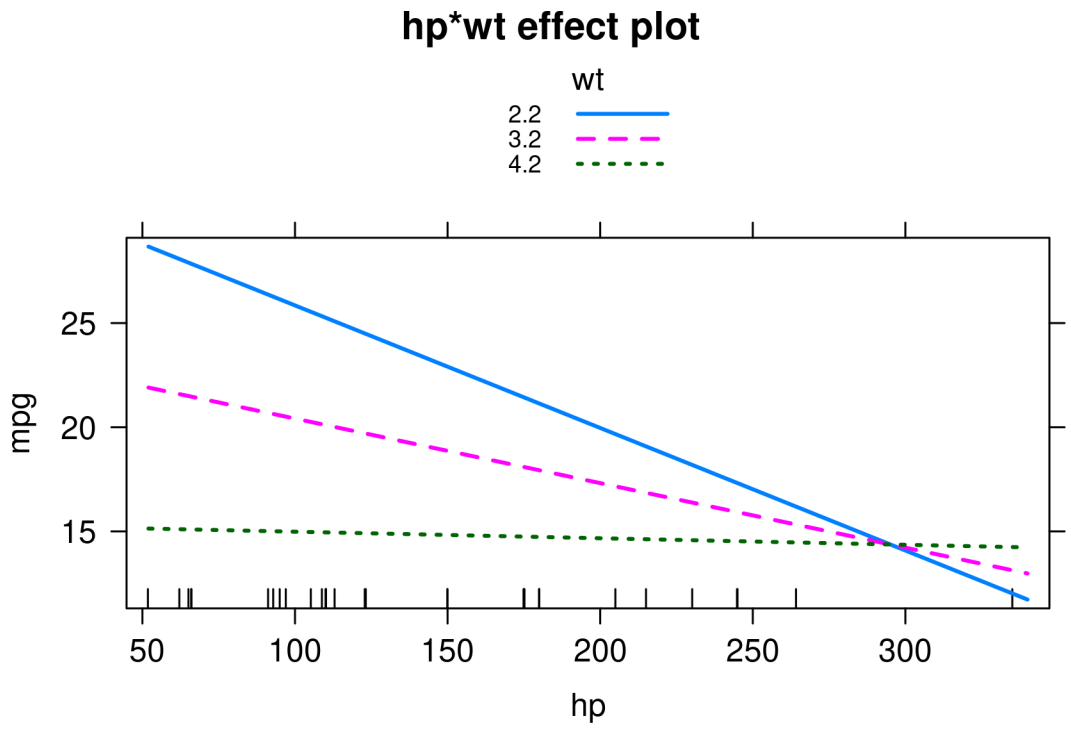
```
##
## Call:
## lm(formula = mpg ~ hp + wt + hp:wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0632 -1.6491 -0.7362  1.4211  4.5513
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 49.80842    3.60516  13.816 5.01e-14 ***
## hp          -0.12010    0.02470  -4.863 4.04e-05 ***
## wt          -8.21662    1.26971  -6.471 5.20e-07 ***
## hp:wt        0.02785    0.00742   3.753 0.000811 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.153 on 28 degrees of freedom
## Multiple R-squared:  0.8848, Adjusted R-squared:  0.8724
## F-statistic: 71.66 on 3 and 28 DF,  p-value: 2.981e-13
```

```
library(effects)
```

```
## lattice theme set by effectsTheme()
## See ?effectsTheme for details.
```

```
plot(effect("hp:wt", fit, , list(wt = c(2.2, 3.2, 4.2))),
     lines = c(1, 2, 3),
     multiline = TRUE)
```



11.2.1.3.6 Simple regression diagnostics

- In the previous section, you used the lm() function
  - to fit an OLS regression model
  - and the summary() function
    - to obtain the model parameters and summary statistics.

Unfortunately, nothing in this printout tells you

- whether the model you've fit is appropriate.

Your confidence in inferences about regression parameters

- depends on the degree to which you've met
  - the statistical assumptions of the OLS model.

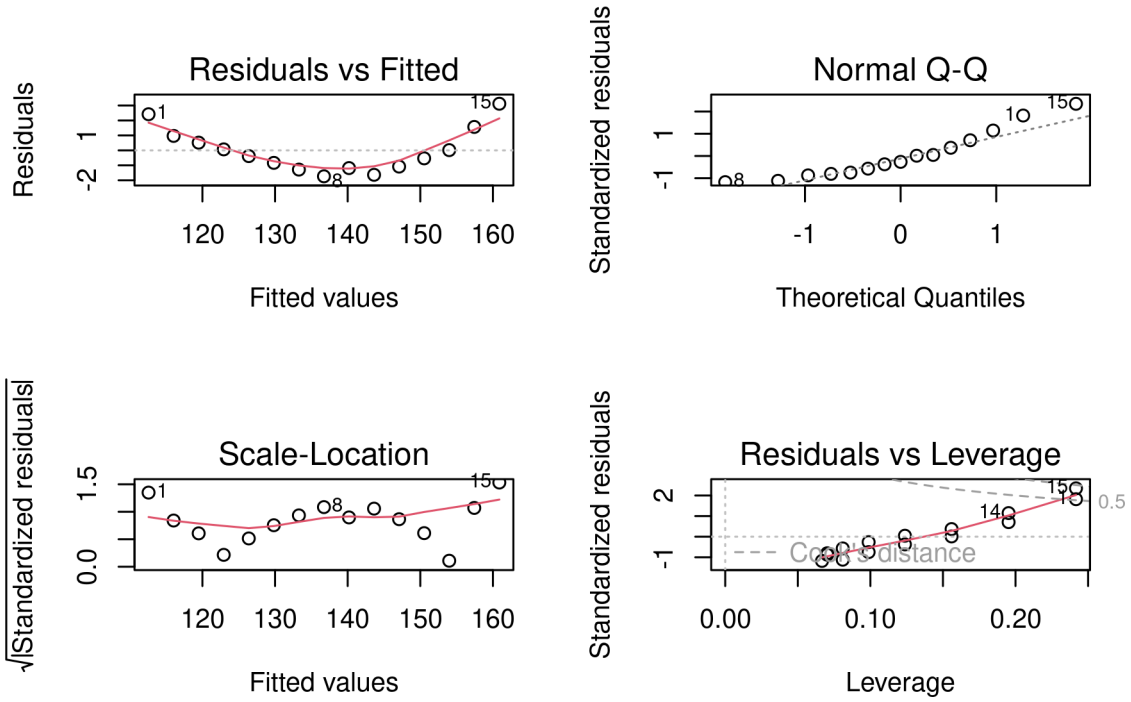
Although the summary() function describes the model,

- it provides no information concerning the degree
  - to which you've satisfied
- the statistical assumptions underlying the model.

Why is important?

- Irregularities in the data
  - or misspecifications of the relationships
  - between the predictors and the response variable
- can lead you to settle on a model that's wildly inaccurate.

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




```
par(mfrow = c(1, 1))
```

To understand these graphs, consider the assumptions of OLS regression:

- Normality—If the dependent variable is normally distributed for a fixed set of predictor values, then the residual values should be normally distributed with a mean of 0. The Normal Q-Q plot (upper right) is a probability plot of the standardized residuals against the values that would be expected under normality. If you've met the normality assumption, the points on this graph should fall on the straight 45-degree line. Because they don't, you've clearly violated the normality assumption.
- Independence—You can't tell if the dependent variable values are independent from these plots. You have to use your understanding of how the data was collected. There's no a priori reason to believe that one woman's weight influences another woman's weight. If you found out that the data were sampled from families, you might have to adjust your assumption of independence.

- Linearity—If the dependent variable is linearly related to the independent variables, there should be no systematic relationship between the residuals and the predicted (that is, fitted) values. In other words, the model should capture all the systematic variance present in the data, leaving nothing but random noise. In the Residuals vs. Fitted graph (upper left), you see clear evidence of a curved relationship, which suggests that you may want to add a quadratic term to the regression.
- Homoscedasticity—If you've met the constant variance assumption, the points in the Scale-Location graph (bottom left) should be a random band around a horizontal line. You seem to meet this assumption.

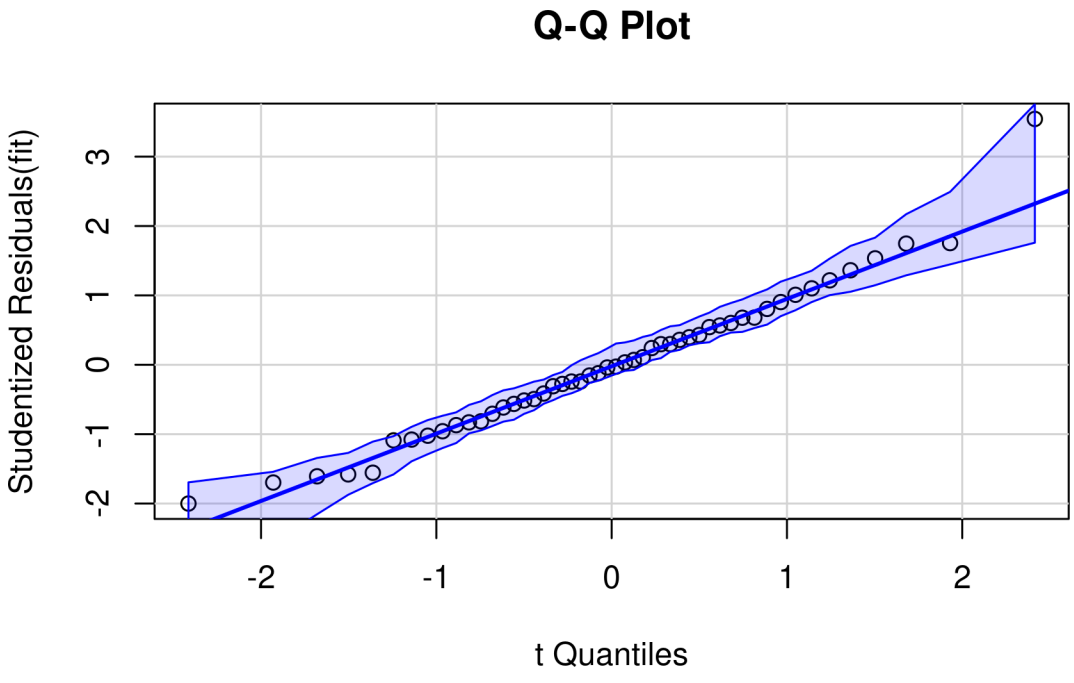
11.2.1.3.7 Assessing normality

- The qqPlot() function provides a more accurate method
  - of assessing the normality assumption
    - than that provided by the plot() function in the base package.

It plots the studentized residuals

- (also called studentized deleted residuals or jackknifed residuals)
  - against a t distribution with n – p – 1 degrees of freedom,
- where n is the sample size and
  - p is the number of regression parameters (including the intercept).

```
library(car)
states <- as.data.frame(state.x77[, c("Murder", "Population",
                                      "Illiteracy", "Income", "Frost")])
fit <-
  lm(Murder ~ Population + Illiteracy + Income + Frost, data = states)
qqPlot(
  fit,
  simulate = TRUE,
  labels = row.names(states),
  id = list(method = "identify"),
  main = "Q-Q Plot"
)
```



11.2.1.3.8 Independence of errors

- the best way to assess whether the dependent variable values
  - (and thus the residuals)
    - are independent is from your knowledge of how the data were collected.

For example, time series data often display autocorrelation

- observations collected closer in time
  - are more correlated with each other
  - than with observations distant in time.

The car package provides a function for the Durbin–Watson test

- to detect such serially correlated errors.

You can apply the Durbin–Watson test to the multiple-regression problem

```
durbinWatsonTest(fit)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1      -0.2006929      2.317691   0.256
##  Alternative hypothesis: rho != 0
```

The nonsignificant p-value (p=0.282)

- suggests a lack of autocorrelation
- and, conversely,
  - an independence of errors.

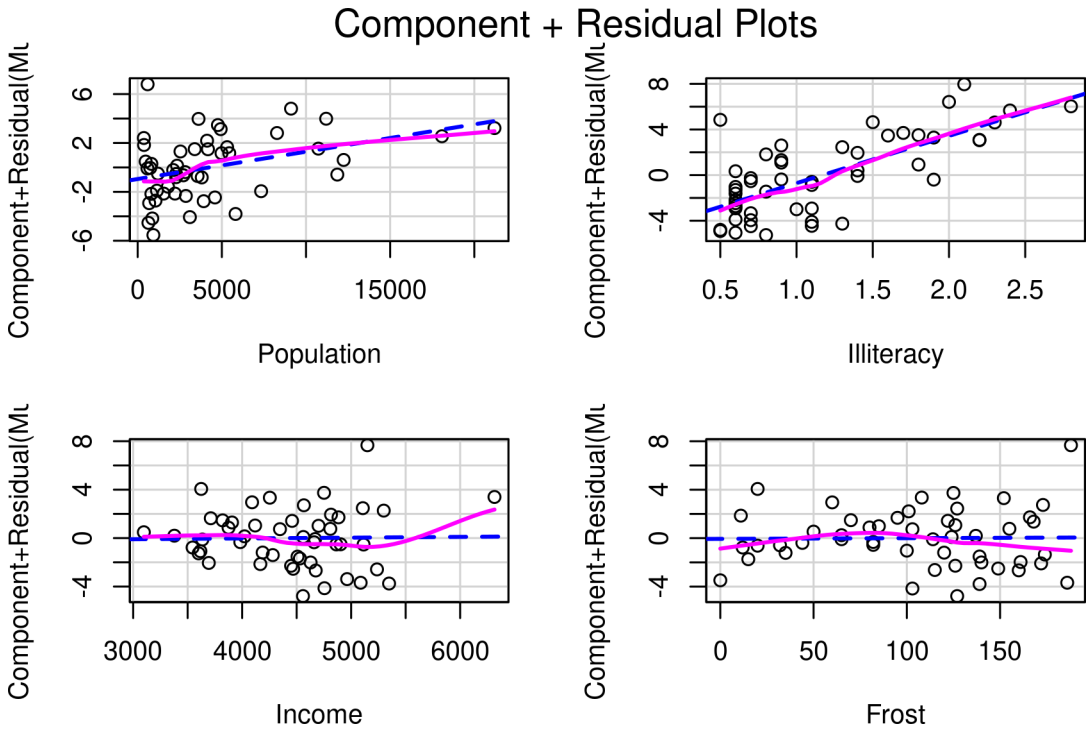
11.2.1.3.9 Assessing linearity

- The component plus residual plots
  - can confirm that you've met the linearity assumption.

The form of the linear model

- seems to be appropriate for this dataset.

```
library(car)
crPlots(fit)
```



11.2.1.3.10 Assessing homoscedasticity

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

The ncvTest() function

- produces a score test of
  - the hypothesis of constant error variance
- against the alternative that
  - the error variance changes
  - with the level of the fitted values.

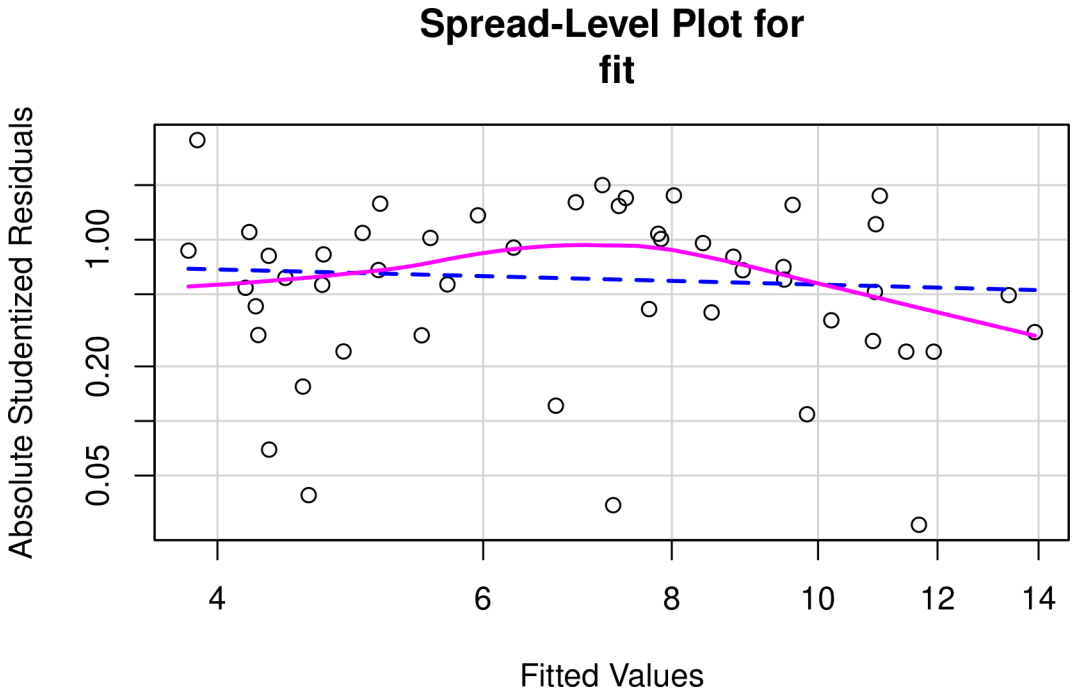
A significant result

- suggests heteroscedasticity (nonconstant error variance).

```
library(car)
ncvTest(fit)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 1.746514, Df = 1, p = 0.18632
```

```
spreadLevelPlot(fit)
```




```
##
## Suggested power transformation:  1.209626
```

The score test is nonsignificant (p = 0.19),

- suggesting that you've met the constant variance assumption.

11.2.1.3.11 Evaluating multi-collinearity

- Before leaving this section on regression diagnostics,
  - let's focus on a problem that's not directly related
    - to statistical assumptions
  - but is important in allowing you
    - to interpret multiple regression results.

Imagine you're conducting a study of grip strength.

- Your independent variables include
  - date of birth (DOB)
  - and age.
- You regress grip strength
  - on DOB and age
  - and find a significant overall F test at p < .001.

But when you look at the individual regression coefficients

- for DOB and age,
- you find that they're both nonsignificant
  - (that is, there's no evidence that either is related to grip strength).

What happened?

```
library(car)
vif(fit)
```

```
## Population Illiteracy     Income      Frost
```

```
##   1.245282   2.165848   1.345822   2.082547
```

```
vif(fit) > 10 # problem?
```

```
## Population Illiteracy     Income      Frost
##      FALSE      FALSE      FALSE      FALSE
```

The problem is that DOB and age

- are perfectly correlated within rounding error.

A regression coefficient

- measures the impact of one predictor variable
  - on the response variable,
- holding all other predictor variables constant.

The problem is called multicollinearity.

- It leads to large confidence intervals for model parameters
- and makes the interpretation of individual coefficients difficult.

Multicollinearity can be detected

- using a statistic called the variance inflation factor (VIF).

For any predictor variable,

- the square root of the VIF
  - indicates the degree to which the confidence interval
- for that variable's regression parameter
  - is expanded relative to a model with uncorrelated predictors
  - (hence the name).

11.2.1.4 Unusual Observations

- A comprehensive regression analysis
  - will also include a screening for unusual observations
    - namely outliers,
    - high-leverage observations,
    - and influential observations.

These are data points that warrant further investigation,

- either because they're different than other observations in some way,
- or because they exert a disproportionate amount of influence
  - on the results.

Let's look at each in turn.

11.2.1.4.1 Assessing outliers

- Outliers are
  - observations not predicted by the model

```
library(car)
outlierTest(fit)
```

```
##        rstudent unadjusted p-value Bonferroni p
## Nevada 3.542929         0.00095088     0.047544
```

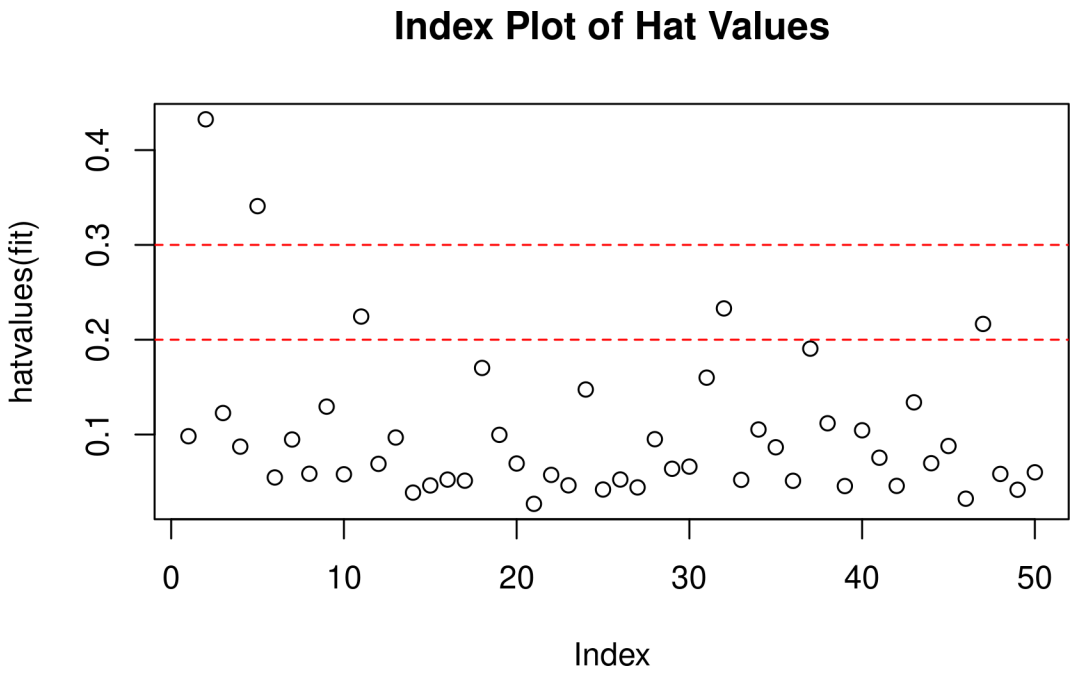
11.2.1.4.2 Identifying high leverage points

- Observations that have high leverage
  - are outliers with regard to the other predictors.

In other words,

- they have an unusual combination of predictor values.

```
hat.plot <- function(fit) {
  p <- length(coefficients(fit))
  n <- length(fitted(fit))
  plot(hatvalues(fit), main = "Index Plot of Hat Values")
  abline(h = c(2, 3) * p / n,
         col = "red",
         lty = 2)
  identify(1:n, hatvalues(fit), names(hatvalues(fit)))
}
hat.plot(fit)
```




```
## integer(0)
```

11.2.1.4.3 Identifying influential observations: Cooks Distance

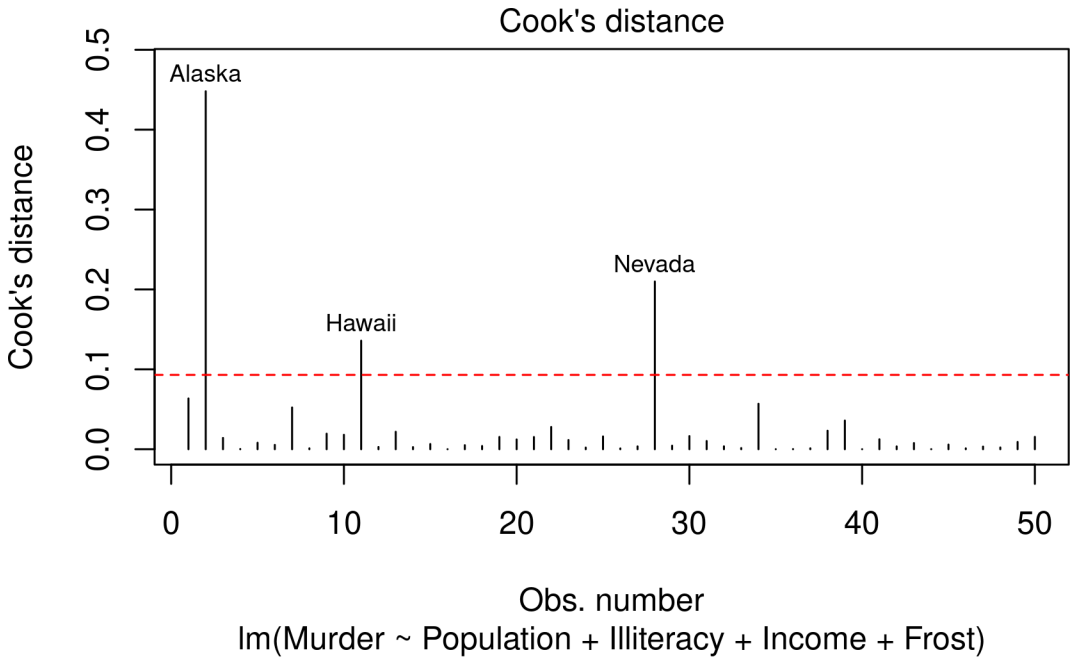
- Influential observations
  - have a disproportionate impact
    - on the values of the model parameters.

There are two methods for identifying influential observations:

- Cook's distance (or D statistic)
- and added variable plots.

```
# Cooks Distance D
# identify D values > 4/(n-k-1)
cutoff <- 4 / (nrow(states) - length(fit$coefficients) - 2)
```

```
plot(fit, which = 4, cook.levels = cutoff)
abline(h = cutoff, lty = 2, col = "red")
```



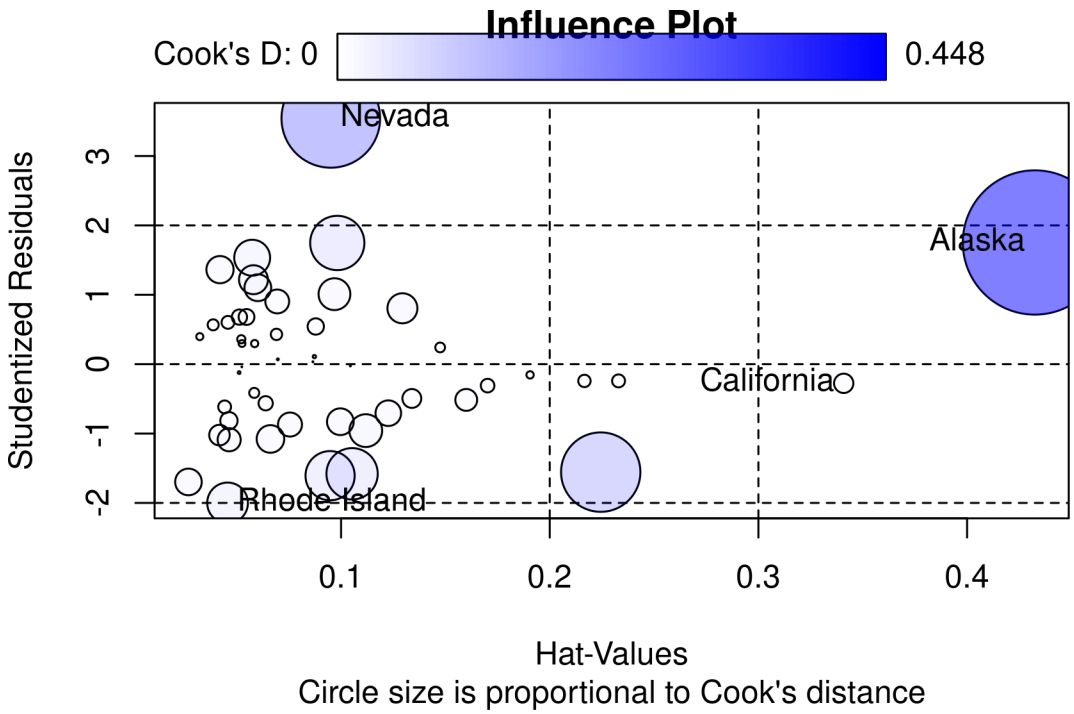
11.2.1.4.4 Added variable plots

```
# add id.method = "identify" to interactively identify points
library(car)
avPlots(fit, ask = FALSE, id.method = "identify")
```

```
library(car)
influencePlot(fit, id = "noteworthy", main = "Influence Plot",
              sub = "Circle size is proportional to Cook's distance")
```

11.2.1.4.5 Influence Plot

```
## Warning in applyDefaults(id, defaults = list(method = "noteworthy", n = 2, :
## unnamed id arguments, will be ignored
```




```
##                 StudRes        Hat       CookD
## Alaska        1.7536917 0.43247319 0.448050997
## California   -0.2761492 0.34087628 0.008052956
## Nevada        3.5429286 0.09508977 0.209915743
## Rhode Island -2.0001631 0.04562377 0.035858963
```

```
library(car)
summary(powerTransform(states$Murder))
```

11.2.1.4.6 Box-Cox Transformation to normality

```
## bcPower Transformation to Normality
##               Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
## states$Murder    0.6055           1       0.0884       1.1227
##
## Likelihood ratio test that transformation parameter is equal to 0
##  (log transformation)
##                            LRT df     pval
## LR test, lambda = (0) 5.665991  1 0.017297
##
## Likelihood ratio test that no transformation is needed
##                            LRT df    pval
## LR test, lambda = (1) 2.122763  1 0.14512
```

```
library(car)
boxTidwell(Murder ~ Population + Illiteracy, data = states)
```

11.2.1.4.7 Box-Tidwell Transformations to linearity

```
##            MLE of lambda Score Statistic (z) Pr(>|z|)
## Population       0.86939             -0.3228   0.7468
```

```
## Illiteracy       1.35812              0.6194   0.5357
##
## iterations =  19
```

```
states <- as.data.frame(state.x77[, c("Murder", "Population",
                                      "Illiteracy", "Income", "Frost")])
fit1 <- lm(Murder ~ Population + Illiteracy + Income + Frost,
           data = states)
fit2 <- lm(Murder ~ Population + Illiteracy, data = states)
anova(fit2, fit1)
```

11.2.1.4.8 Comparing nested models using the anova function

```
## Analysis of Variance Table
##
## Model 1: Murder ~ Population + Illiteracy
## Model 2: Murder ~ Population + Illiteracy + Income + Frost
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     47 289.25
## 2     45 289.17  2  0.078505 0.0061 0.9939
```

```
fit1 <- lm(Murder ~ Population + Illiteracy + Income + Frost,
           data = states)
fit2 <- lm(Murder ~ Population + Illiteracy, data = states)
AIC(fit1, fit2)
```

11.2.1.4.9 Comparing models with the AIC

```
##      df      AIC
## fit1  6 241.6429
## fit2  4 237.6565
```

```
states <- as.data.frame(state.x77[, c("Murder", "Population",
                                      "Illiteracy", "Income", "Frost")])
fit <- lm(Murder ~ Population + Illiteracy + Income + Frost,
          data = states)
step(fit, direction = "backward")
```

11.2.1.4.10 Backward stepwise selection

```
## Start:  AIC=97.75
## Murder ~ Population + Illiteracy + Income + Frost
##
##              Df Sum of Sq    RSS     AIC
## - Frost       1     0.021 289.19  95.753
## - Income      1     0.057 289.22  95.759
## <none>                    289.17  97.749
## - Population  1    39.238 328.41 102.111
## - Illiteracy  1   144.264 433.43 115.986
##
## Step:  AIC=95.75
## Murder ~ Population + Illiteracy + Income
##
```

```
##              Df Sum of Sq    RSS     AIC
## - Income      1     0.057 289.25  93.763
## <none>                    289.19  95.753
## - Population  1    43.658 332.85 100.783
## - Illiteracy  1   236.196 525.38 123.605
##
## Step:  AIC=93.76
## Murder ~ Population + Illiteracy
##
##              Df Sum of Sq    RSS     AIC
## <none>                    289.25  93.763
## - Population  1    48.517 337.76  99.516
## - Illiteracy  1   299.646 588.89 127.311
```

```
##
## Call:
## lm(formula = Murder ~ Population + Illiteracy, data = states)
##
## Coefficients:
## (Intercept)   Population   Illiteracy
##   1.6515497    0.0002242    4.0807366
```

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

leaps <- regsubsets(Murder ~ Population + Illiteracy + Income +
                      Frost,
                    data = states,
                    nbest = 4)

subsTable <- function(obj, scale) {
  x <- summary(leaps)
  m <- cbind(round(x[[scale]], 3), x$which[, -1])
  colnames(m)[1] <- scale
  m[order(m[, 1]),]
}

subsTable(leaps, scale = "adjr2")
```

11.2.1.4.11 All subsets regression

```
##   adjr2 Population Illiteracy Income Frost
## 1 0.033          0          0      1     0
## 1 0.100          1          0      0     0
## 1 0.276          0          0      0     1
## 2 0.292          1          0      0     1
## 3 0.309          1          0      1     1
## 3 0.476          0          1      1     1
## 2 0.480          0          1      1     0
## 2 0.481          0          1      0     1
## 1 0.484          0          1      0     0
## 4 0.528          1          1      1     1
## 3 0.539          1          1      1     0
```

```
## 3 0.539          1          1      0     1
## 2 0.548          1          1      0     0
```

```
shrinkage <- function(fit, k = 10, seed = 1) {
  require(bootstrap)

  theta.fit <- function(x, y) {
    lsfit(x, y)
  }
  theta.predict <- function(fit, x) {
    cbind(1, x) %*% fit$coef
  }

  x <- fit$model[, 2:ncol(fit$model)]
  y <- fit$model[, 1]

  set.seed(seed)
  results <- crossval(x, y, theta.fit, theta.predict, ngroup = k)
  r2    <- cor(y, fit$fitted.values) ^ 2
  r2cv  <- cor(y, results$cv.fit) ^ 2
  cat("Original R-square =", r2, "\n")
  cat(k, "Fold Cross-Validated R-square =", r2cv, "\n")
}

states <- as.data.frame(state.x77[, c("Murder", "Population",
                                      "Illiteracy", "Income", "Frost")])
fit <-
  lm(Murder ~ Population + Income + Illiteracy + Frost, data = states)
shrinkage(fit)
```

11.2.1.4.12 Function for k-fold cross-validated R-square

```
## Loading required package: bootstrap
```

```
## Original R-square = 0.5669502
## 10 Fold Cross-Validated R-square = 0.3564904
```

```
fit2 <- lm(Murder ~ Population + Illiteracy, data = states)
shrinkage(fit2)
```

```
## Original R-square = 0.5668327
## 10 Fold Cross-Validated R-square = 0.514864
```

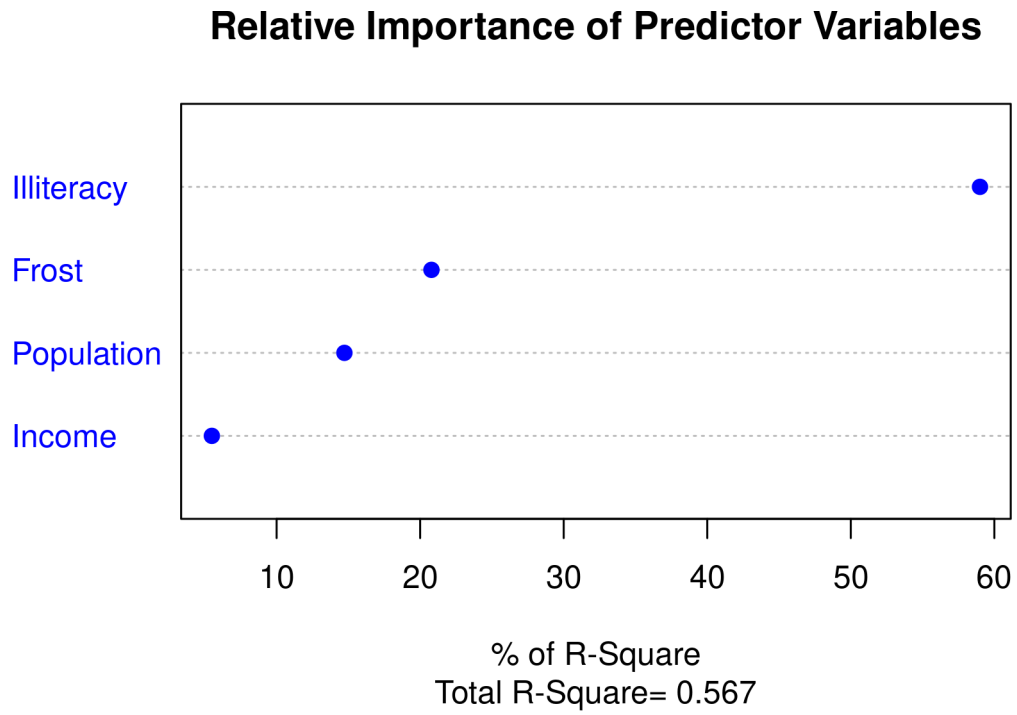
```
relweights <- function(fit, ...) {
  R <- cor(fit$model)
  nvar <- ncol(R)
  rxx <- R[2:nvar, 2:nvar]
  rxy <- R[2:nvar, 1]
  svd <- eigen(rxx)
  evec <- svd$vectors
  ev <- svd$values
  delta <- diag(sqrt(ev))
  lambda <- evec %*% delta %*% t(evec)
  lambdasq <- lambda ^ 2
```

```
  beta <- solve(lambda) %*% rxy
  rsquare <- colSums(beta ^ 2)
  rawwgt <- lambdasq %*% beta ^ 2
  import <- (rawwgt / rsquare) * 100
  import <- as.data.frame(import)
  row.names(import) <- names(fit$model[2:nvar])
  names(import) <- "Weights"
  import <- import[order(import), 1, drop = FALSE]
  dotchart(
    import$Weights,
    labels = row.names(import),
    xlab = "% of R-Square",
    pch = 19,
    main = "Relative Importance of Predictor Variables",
    sub = paste("Total R-Square=", round(rsquare, digits = 3)),
    ...
  )
  return(import)
}
```

11.2.1.4.13 relweights function for calculating relative importance of predictors Applying the relweights function

```
states <- as.data.frame(state.x77[, c("Murder", "Population",
                                      "Illiteracy", "Income", "Frost")])
fit <-
  lm(Murder ~ Population + Illiteracy + Income + Frost, data = states)
relweights(fit, col = "blue")
```

```
## Warning in xtfrm.data.frame(x): cannot xtfrm data frames
```




```
##              Weights
```

```
## Income      5.488962
## Population 14.723401
## Frost      20.787442
## Illiteracy 59.000195
```

11.2.1.5 Links Robert I. Kabacoff, R in Action, 3rd Edition, Manning Publications 2020

Fwa, T., ed. 2006. The Handbook of Highway Engineering, 2nd ed. Boca Raton, FL: CRC Press.