Chapter 10
Forecasting Numeric Data Using Regression Models

In the previous Chaps. 7, 8, and 9, we covered classification methods that use mathematical formalism to address everyday life prediction problems. In this Chapter, we will focus on specific model-based statistical methods providing forecasting and classification functionality. Specifically, we will (1) demonstrate the predictive power of multiple linear regression; (2) show the foundation of regression trees and model trees; and (3) examine two complementary case-studies (Baseball Players and Heart Attack).

It may be helpful to first review Chap. 5 (Linear Algebra/Matrix Manipulations) and Chap. 7 (Introduction to Machine Learning).

10.1 Understanding Regression

Regression is a measurement of relationship between a *dependent variable* (value to be predicted) and a group of *independent variables* (predictors similar to features, discussed in Chap. 7). We assume the relationship between our dependent variable and independent variables follows a predefined model, e.g., an affine or hyper-linear model.

10.1.1 Simple Linear Regression

First recall the material presented in Chap. 5 (***Linear Algebra & Matrix Computing***).

The simplest case of regression modeling involves a single predictor.

$$y = a + bx.$$

I. D. Dinov, *Data Science and Predictive Analytics*,
https://doi.org/10.1007/978-3-319-72347-1_10

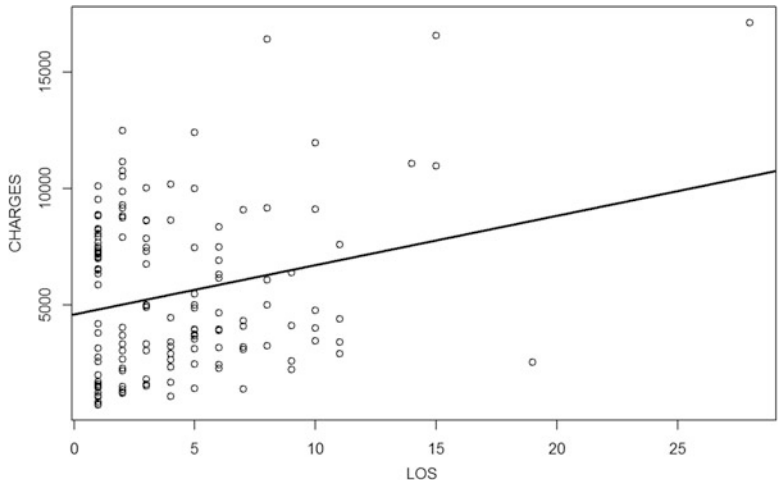

Fig. 10.1 Scatterplot and a linear model of length of stay (LOS) vs. hospital charges for the heart attack data

This formula should appear familiar by now. In this slope-intercept analytical expression, *a* is our intercept while *b* is the slope. That is an equation form of the simple linear regression model. If we know *a* and *b*, for any given *x* (input) we can *predict y* (output) via the above formula. If we plot *x* and *y* in a 2D coordinate system, the model is graphically represented as a straight line.

However, this is the ideal case. When we plot using real world data, the pattern may be harder to recognize. Let's look at the scatter plot (see Chap. 3) and simple linear regression line of two variables "hospital charges" or `CHARGES` (independent variable) and `length of stay` in the hospital or `LOS` (predictor). The data is available online, `CaseStudy12_AdultsHeartAttack_Data`. We removed two observations that have missing data using the command `heart_attack<-heart_attack [complete.cases(heart_attack),]`.

```
heart_attack<-
read.csv("https://umich.instructure.com/files/1644953/download?download_frd
=1", stringsAsFactors = F) heart_attack$CHARGES<-as.numeric
(heart_attack$CHARGES)

## Warning: NAs introduced by coercion

heart_attack<-heart_attack[complete.cases(heart_attack), ]

fit1<-lm(CHARGES~LOS, data=heart_attack)
par(cex=.8)
plot(heart_attack$LOS, heart_attack$CHARGES, xlab="LOS", ylab = "CHARGES")
abline(fit1, lwd=2)
```

It seems to be common sense that the longer you stay in the hospital, the higher the medical costs will be. However, on the scatter plot, we have only a bunch of dots showing some sign of an increasing pattern (Fig. 10.1).

The estimated expression for this regression line is:

$$\hat{y} = 4582.70 + 212.29 \times x,$$

or equivalently

$$CHARGES = 4582.70 + 212.29 \times LOS.$$

It is simple to make predictions with this regression line. Assume we have a patient that spent 10 days in hospital, then we have `LOS=10`. The predicted charge is likely to be \$4582.70 + \$212.29 × 10 = \$6705.6. Plugging `x` into the expression equation automatically gives us an estimated value of the outcome `y`. This Chapter of the Probability and statistics EBook provides an introduction to linear modeling (http://wiki.socr.umich.edu/index.php/EBook#Chapter_X:_Correlation_and_Regression).

10.2 Ordinary Least Squares Estimation

How did we get the estimated expression? The most common estimating method in statistics is *ordinary least squares* (OLS). OLS estimators are obtained by minimizing the sum of the squared errors – that is the sum of squared vertical distances between each point on the scatter plot and its predicted value on the regression line (Fig. 10.2).

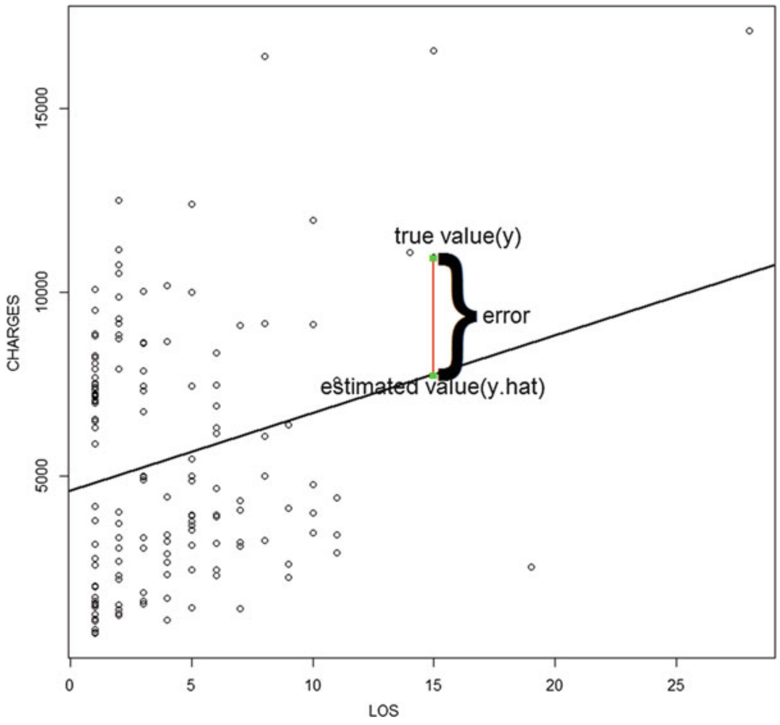

Fig. 10.2 Graphical representation of the residuals representing the difference between observed and predicted values

OLS is minimizing the following formula:

$$\sum_{i=1}^{n}\left(y_i-\hat{y}_i\right)^2=\sum_{i=1}^{n}\left(y_i-(a+b\times x_i)\right)^2=\sum_{i=1}^{n}e_i^2.$$

Some simple mathematical operations to minimize the sum square error yield the following solution for the slope parameter b:

$$b=\frac{\sum\left(x_i-\bar{x}\right)\left(y_i-\bar{y}\right)}{\sum\left(x_i-\bar{x}\right)^2}.$$

While the intercept a is given by:

$$a=\bar{y}-b\bar{x}.$$

Recall what we learned in Chap. 3, where the variance was obtained by averaging the sum of squares $\left(var(x)=\frac{1}{n}\sum_{i=1}^{n}\left(x_i-\mu\right)^2\right)$. When we use $\bar{x}$ to estimate the mean of x, we have the following formula for the sample variance: $var(x)=\frac{1}{n-1}\sum_{i=1}^{n}\left(x_i-\bar{x}\right)^2$. We can see that this is $\frac{1}{n-1}$ times the denominator of b. Similar to the variance, the covariance of x and y measures the average sum of the x deviance times the y deviance.

$$Cov(x,y)=\frac{1}{n}\sum_{i=1}^{n}\left(x_i-\mu_x\right)\left(y_i-\mu_y\right).$$

If we use sample averages ($\bar{x}$, $\bar{y}$), we have: $Cov(x,y)=\frac{1}{n-1}\sum_{i=1}^{n}\left(x_i-\bar{x}\right)\left(y_i-\bar{y}\right)$. This is $\frac{1}{n-1}$ times the numerator of b.

Combining the above, we get an estimate of the slope coefficient (effect-size of LOS on Charge):

$$b=\frac{Cov(x,y)}{var(x)}.$$

Let's examine these closed-form analytical expressions using the heart attack data.

```
b<-cov(heart_attack$LOS, heart_attack$CHARGES)/var(heart_attack$LOS); b
## [1] 212.2869
a<-mean(heart_attack$CHARGES)-b*mean(heart_attack$LOS); a
## [1] 4582.7
```

We can see that these estimates are exactly the same result as the previously reported.

10.2.1 Model Assumptions

Regression modeling has the following five key assumptions:

- Linear relationship,
- Multivariate normality,
- No or little multicollinearity,
- No auto-correlation, independence,
- Homoscedasticity.

10.2.2 Correlations

The SOCR Interactive Scatterplot Game (requires Java enabled browser) provides a dynamic interface demonstrating linear models, trends, correlations, slopes, and residuals.

Using the covariance, we can calculate the correlation, which indicates how closely the relationship between two variables follows a straight line.

$$\rho_{x,y} = Corr(x, y) = \frac{Cov(x, y)}{\sigma_x \sigma_y} = \frac{Cov(x, y)}{\sqrt{Var(x)Var(y)}}.$$

In R, correlation is given by cor() while square root of variance, or standard deviation, is given by sd().

```
r<-cov(heart_attack$LOS, heart_attack$CHARGES)/(sd(heart_attack$LOS)*
sd(heart_attack$CHARGES))
r
## [1] 0.2449743
cor(heart_attack$LOS, heart_attack$CHARGES)
## [1] 0.2449743
```

The same outputs are obtained by the manual and the automated correlation calculations. This correlation is a positive number that is relatively small. We can say there is a weak positive linear association between these two variables. If we have a negative correlation estimate, it suggests a negative linear association. We have a weak association when $0.1 \leq Cor < 0.3$, a moderate association for $0.3 \leq Cor < 0.5$, and a strong association for $0.5 \leq Cor \leq 1.0$. If the correlation is below 0.1 then it suggests little to no linear relation between the variables.

10.2.3 Multiple Linear Regression

In practice, most interesting problems involve multiple predictors and one dependent variable, which requires estimating a multiple linear model. That is:

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k + \epsilon,$$

or equivalently

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k + \epsilon.$$

We usually use the second notation method in statistics. This equation shows the linear relationship between *k* predictors and a dependent variable. In total we have *k* + *1* coefficients to estimate.

The matrix notation for corresponding to the above equation is:

$$Y = X\beta + \epsilon,$$

where

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \ldots \\ y_n \end{pmatrix},$$

$$X = \begin{pmatrix} 1 & x_{11} & x_{21} & \ldots & x_{k1} \\ 1 & x_{12} & x_{22} & \ldots & x_{k2} \\ . & . & . & . & . \\ 1 & x_{1n} & x_{2n} & \ldots & x_{kn} \end{pmatrix},$$

$$\beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \ldots \\ \beta_k \end{pmatrix},$$

and

$$\epsilon = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \ldots \\ \epsilon_n \end{pmatrix}$$

is the error term.

Similar to simple linear regression, our goal is to minimize sum of squared errors. Solving for β, we get:

$$\hat{\beta} = \left(X^T X\right)^{-1} X^T Y.$$

This is the matrix form solution, where X^{-1} is the inverse matrix of X and X^T is the transpose matrix.

Let’s write a simple R function reg (x,y), that implements this matrix formula.

```
reg<-function(y, x){
  x<-as.matrix(x)
  x<-cbind(Intercept=1, x)
  solve(t(x)%*%x)%*%t(x)%*%y
}
```

The method `solve()` is used to compute the matrix inverse and `%*%` is matrix multiplication.

Next, we will apply this function to our heart attack dataset. To begin, let’s check if the simple linear regression output is the same as we calculated earlier.

```
reg(y=heart_attack$CHARGES, x=heart_attack$LOS)

##                 [,1]
## Intercept 4582.6997
##            212.2869
```

As the slope and intercept and consistent with our previous estimates, we can continue and include additional variables as predictors. For instance, we can just add age into the model.

```
str(heart_attack)

## 'data.frame':    148 obs. of  8 variables:
##  $ Patient  : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ DIAGNOSIS: int  41041 41041 41091 41081 41091 41091 41091 41091 41041
41041 ...
##  $ SEX      : chr  "F" "F" "F" "F" ...
##  $ DRG      : int  122 122 122 122 122 121 121 121 121 123 ...
##  $ DIED     : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ CHARGES  : num  4752 3941 3657 1481 1681 ...
##  $ LOS      : int  10 6 5 2 1 9 15 15 2 1 ...
##  $ AGE      : int  79 34 76 80 55 84 84 70 76 65 ...

reg(y=heart_attack$CHARGES, x=heart_attack[, c(7, 8)])

##                  [,1]
## Intercept 7280.55493
## LOS        259.67361
## AGE        -43.67677
```

10.3 Case Study 1: Baseball Players

10.3.1 Step 1: Collecting Data

We utilize the MLB data "01a_data.txt". The dataset contains 1034 records of heights and weights for some current and recent Major League Baseball (MLB) Players. These data were obtained from different resources (e.g., IBM Many Eyes).

This dataset includes the folloing variables:

- **Name**: MLB Player Name,
- **Team**: The Baseball team the player was a member of at the time the data was acquired,
- **Position**: Player field position,
- **Height**: Player height in inch,
- **Weight**: Player weight in pounds, and
- **Age**: Player age at time of record.

10.3.2 Step 2: Exploring and Preparing the Data

Let's load this dataset first. We use `as.is=T` to make non-numerical vectors into characters. Also, we delete the `Name` variable because we don't need players' names in this case study.

```
mlb<- read.table('https://umich.instructure.com/files/330381/download?downlo
ad_frd=1', as.is=T, header=T)
str(mlb)

## 'data.frame':    1034 obs. of  6 variables:
##  $ Name    : chr  "Adam_Donachie" "Paul_Bako" "Ramon_Hernandez"
"Kevin_Millar" ...
##  $ Team    : chr  "BAL" "BAL" "BAL" "BAL" ...
##  $ Position: chr  "Catcher" "Catcher" "Catcher" "First_Baseman" ...
##  $ Height  : int  74 74 72 72 73 69 69 71 76 71 ...
##  $ Weight  : int  180 215 210 210 188 176 209 200 231 180 ...
##  $ Age     : num  23 34.7 30.8 35.4 35.7 ...

mlb<-mlb[, -1]
```

By looking at the `srt()` output, we notice that the variable `TEAM` and `Position` are misspecified as characters. To fix this, we can use the function `as.factor()` to convert numerical or character vectors to factors.

```
mlb$Team<-as.factor(mlb$Team)
mlb$Position<-as.factor(mlb$Position)
```

The data is good to go. Let's explore it using some summary statistics and plots (Fig. 10.3).

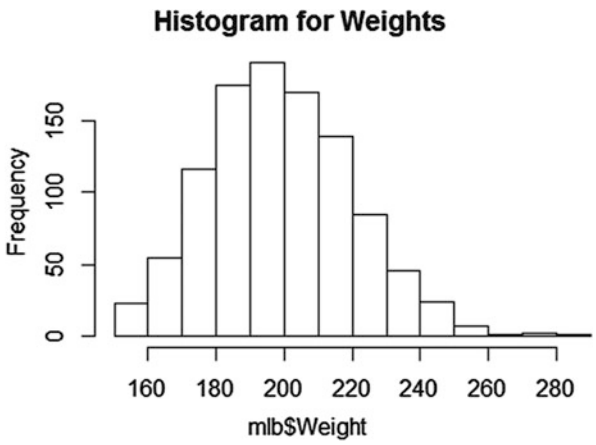

Fig. 10.3 Frequency histogram of the MLB player's weights

```
summary(mlb$Weight)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   150.0   187.0   200.0   201.7   215.0   290.0
hist(mlb$Weight, main = "Histogram for Weights")
```

The above plot illustrates our dependent variable `Weight`. As we learned in Chap. 3, this distribution appears somewhat right-skewed (Fig. 10.3).

Applying `GGpairs` to obtain a compact dataset summary we can mark heavy weight and light weight players (according to *light* < *median* < *heavy*) by different colors in the plot on Fig. 10.4

```
require(GGally)
mlb_binary = mlb

mlb_binary$bi_weight =
as.factor(ifelse(mlb_binary$Weight>median(mlb_binary$Weight),1,0))
g_weight <- ggpairs(data=mlb_binary[-1], title="MLB Light/Heavy Weights",
           mapping=ggplot2::aes(colour = bi_weight),
           lower=list(combo=wrap("facethist",binwidth=1)))
g_weight
```

Next, we may also mark player positions by different colors in the plot (Fig. 10.5).

```
g_position <- ggpairs(data=mlb[-1], title="MLB by Position",
              mapping=ggplot2::aes(colour = Position),
              lower=list(combo=wrap("facethist",binwidth=1)))
g_position
```

What about potential predictors?

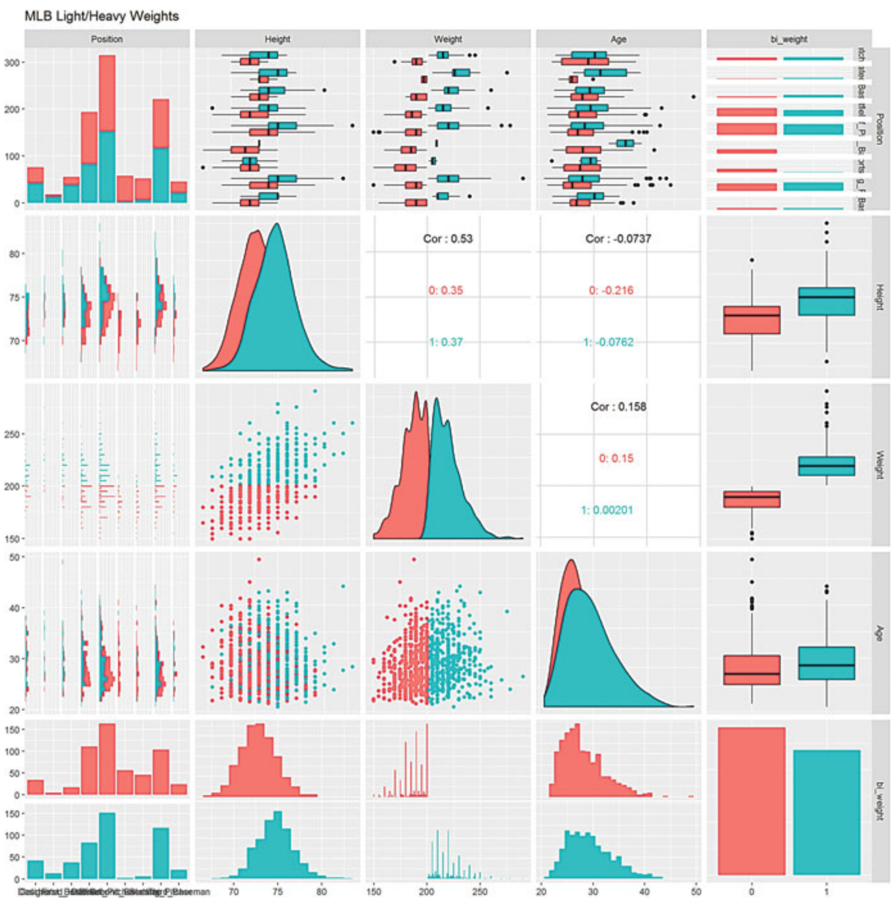

Fig. 10.4 Pair plots of the MLB data by player's light (red) or heavy (blue) weights

```
table(mlb$Team)
## ANA ARZ ATL BAL BOS CHC CIN CLE COL CWS DET FLA HOU  KC  LA MIN MLW NYM
##  35  28  37  35  36  36  36  35  35  33  37  32  34  35  33  33  35  38
## NYY OAK PHI PIT  SD SEA  SF STL  TB TEX TOR WAS
##  32  37  36  35  33  34  34  32  33  35  34  36

table(mlb$Position)

##
##          Catcher Designated_Hitter     First_Baseman        Outfielder
##               76                18                55               194
##    Relief_Pitcher    Second_Baseman         Shortstop  Starting_Pitcher
##              315                58                52               221
##     Third_Baseman
##               45

summary(mlb$Height)
```

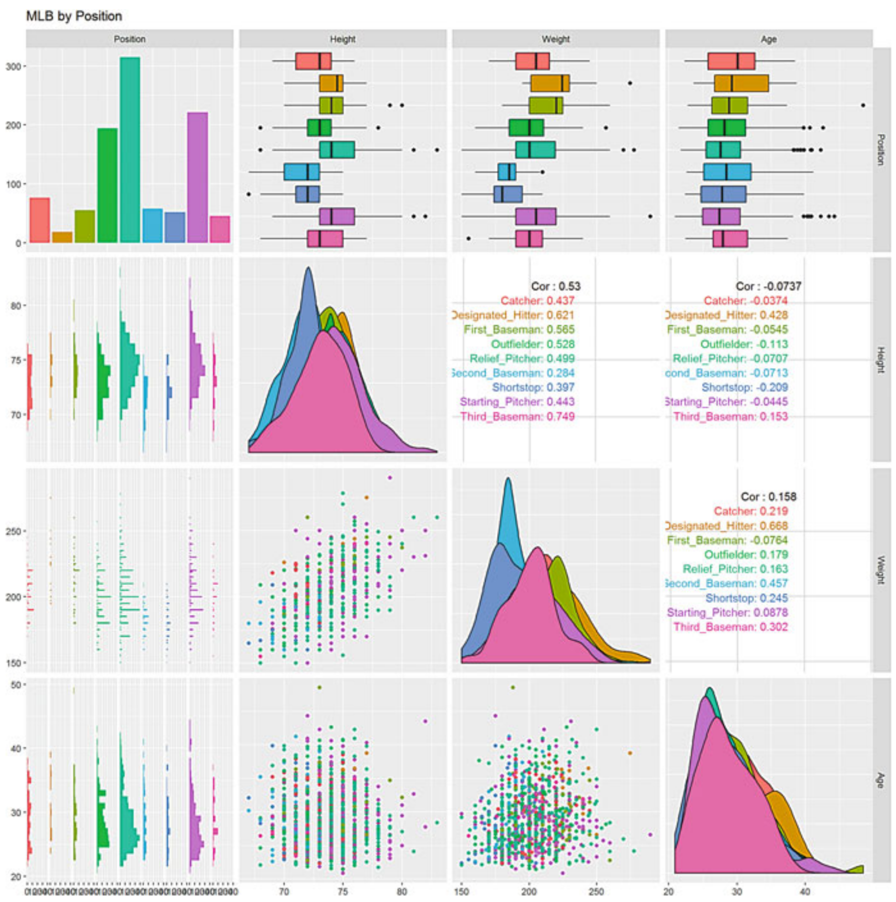

Fig. 10.5 Pair plots of the MLB data by position type

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    67.0    72.0    74.0    73.7    75.0    83.0

summary(mlb$Age)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   20.90   25.44   27.92   28.74   31.23   48.52
```

In this case, we have two numerical predictors, two categorical predictors and 1,034 observations. Let's see how R treats different classes of variables.

10.3.3 *Exploring Relationships Among Features: The Correlation Matrix*

Before fitting a model, let's examine the independence of our potential predictors and the dependent variable. Multiple linear regression assumes that predictors are all independent of each other. Is this assumption valid? As we mentioned earlier, the `cor()` function can answer this question in pairwise manner. Note that we only look at numerical variables.

```
cor(mlb[c("Weight", "Height", "Age")])

##            Weight      Height         Age
## Weight 1.0000000  0.53031802  0.15784706
## Height 0.5303180  1.00000000 -0.07367013
## Age    0.1578471 -0.07367013  1.00000000
```

Observe that $cor(y, x) = cor(x, y)$ and $cov(x, x) = 1$. Also, our `Height` variable is weakly related to the players' age in a negative manner. This looks very good and wouldn't cause any multicollinearity problem. If two of our predictors are highly correlated, they both provide almost the same information, which could imply multicollinearity. A common practice is to delete one of them in the model or use dimensionality reduction methods.

10.3.4 *Visualizing Relationships Among Features: The Scatterplot Matrix*

To visualize pairwise correlations, we could use scatterplot or `pairs()` plot (Fig. 10.6).

```
pairs(mlb[c("Weight", "Height", "Age")])
```

You might get a sense of the data, but it is difficult to see any linear pattern. We can make a more sophisticated graph using `pairs.panels()` in the `psych` package (Fig. 10.7).

```
# install.packages("psych")
library(psych)
pairs.panels(mlb[, c("Weight", "Height", "Age")])
```

This plot provides much more information about the three variables. Above the diagonal, we have our correlation coefficients in numerical form. On the diagonal, there are histograms of variables. Below the diagonal, visual information is presented to help us understand the trend. This specific graph shows that height and weight are positively and strongly correlated. Also, the relationships between age and height, as well as, age and weight are very weak, see the horizontal red line in the panel below the main diagonal graphs, which indicates weak relationships (Fig. 10.7).

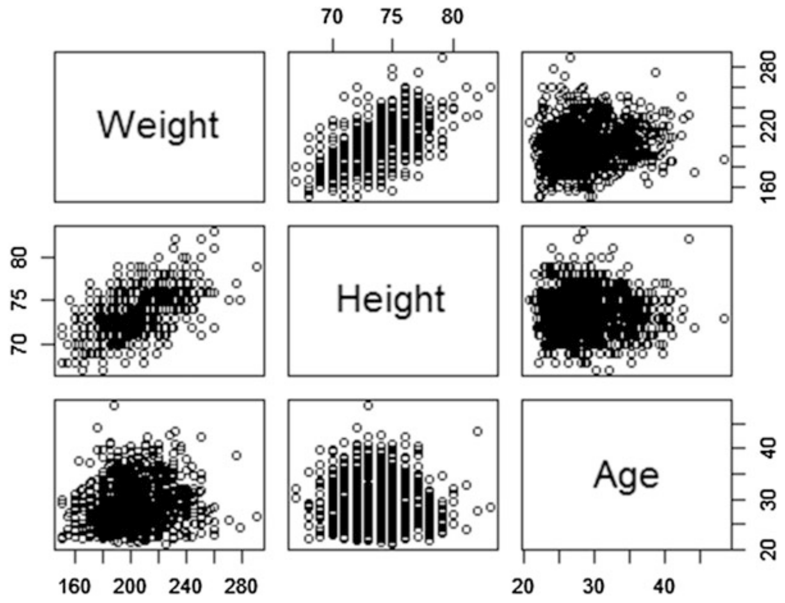

Fig. 10.6 MLB players weights, heights and ages

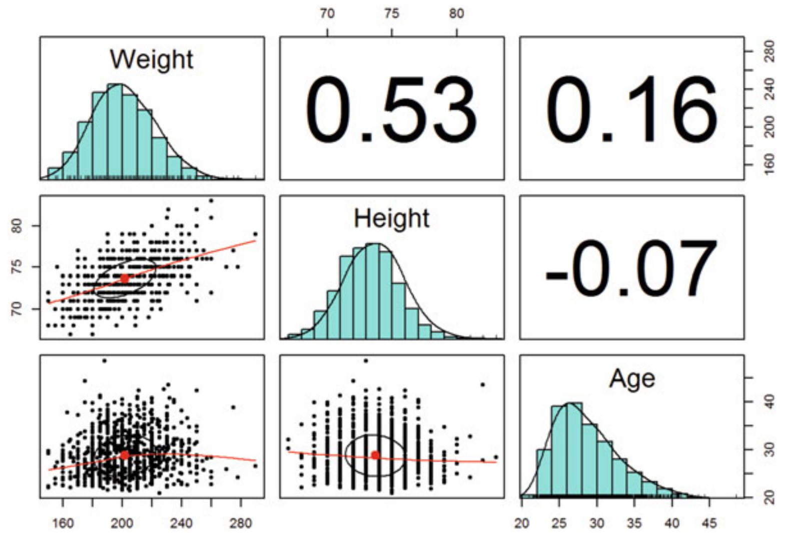

Fig. 10.7 A more detailed pairs plot of MLB players weights, heights and ages

10.3.5 Step 3: Training a Model on the Data

The function we are going to use now is `lm()`. No additional package is needed when using this function.

The `lm()` function has the following components:

m<-lm(dv ~ iv, data=mydata)

- dv: dependent variable
- iv: independent variables. Just like `OneR()` in Chap. 9, if we use `.` as `iv`, then all of the variables, except the dependent variable (*dv*), are included as predictors.
- data: specifies the data containing both dependent viable and independent variables.

```
fit<-lm(Weight~., data=mlb)
fit

##
## Call:
## lm(formula = Weight ~ ., data = mlb)
##
## Coefficients:
##               (Intercept)                    TeamARZ
##                 -164.9995                     7.1881
##                   TeamATL                    TeamBAL
##                   -1.5631                    -5.3128
##                   TeamBOS                    TeamCHC
##                   -0.2838                     0.4026
##                   TeamCIN                    TeamCLE
##                    2.1051                    -1.3160
##                   TeamCOL                    TeamCWS
##                   -3.7836                     4.2944
##                   TeamDET                    TeamFLA
##                    2.3024                     2.6985
##                   TeamHOU                     TeamKC
##                   -0.6808                    -4.7664
##                    TeamLA                    TeamMIN
##                    2.8598                     2.1269
##                   TeamMLW                    TeamNYM
##                    4.2897                    -1.9736
##                   TeamNYY                    TeamOAK
##                    1.7483                    -0.5464
##                   TeamPHI                    TeamPIT
##                   -6.8486                     4.3023
##                    TeamSD                    TeamSEA
##                    2.6133                    -0.9147
##                    TeamSF                    TeamSTL
##                    0.8411                    -1.1341
##                    TeamTB                    TeamTEX
##                   -2.6616                    -0.7695
##                   TeamTOR                    TeamWAS
##                    1.3943                    -1.7555
## PositionDesignated_Hitter      PositionFirst_Baseman
##                    8.9037                     2.4237
##        PositionOutfielder     PositionRelief_Pitcher
##                   -6.2636                    -7.7695
##    PositionSecond_Baseman         PositionShortstop
##                  -13.0843                  -16.9562
##  PositionStarting_Pitcher      PositionThird_Baseman
##                   -7.3599                   -4.6035
##                    Height                       Age
##                    4.7175                    0.8906
```

As we can see from the output, factors are included in the model by creating several indicators, one for each factor level. For each numerical variable, a corresponding model coefficient is estimated.

10.3.6 Step 4: Evaluating Model Performance

As we did in previous case-studies, let's examine the model performance (Figs. 10.8 and 10.9).

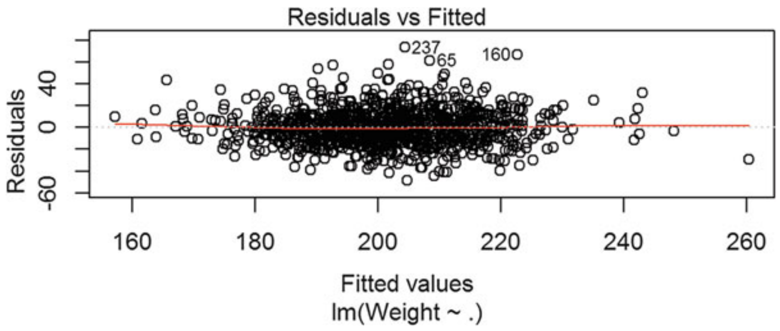

Fig. 10.8 Scatterplot of the residuals vs. model fitted values

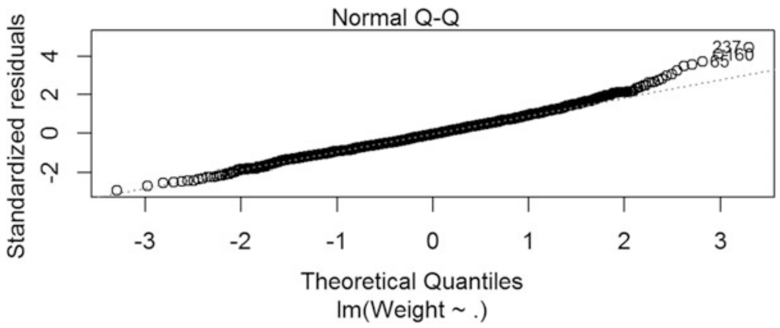

Fig. 10.9 QQ-normal plot of the residuals suggesting a linear model may explain the players' weight

```
summary(fit)

##
## Call:
## lm(formula = Weight ~ ., data = mlb)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -48.692 -10.909  -0.778   9.858  73.649
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 -164.9995    19.3828  -8.513  < 2e-16 ***
## TeamARZ                        7.1881     4.2590   1.688 0.091777 .
## TeamATL                       -1.5631     3.9757  -0.393 0.694278
## TeamBAL                       -5.3128     4.0193  -1.322 0.186533
## TeamBOS                       -0.2838     4.0034  -0.071 0.943492
## TeamCHC                        0.4026     3.9949   0.101 0.919749
## TeamCIN                        2.1051     3.9934   0.527 0.598211
## TeamCLE                       -1.3160     4.0356  -0.326 0.744423
## TeamCOL                       -3.7836     4.0287  -0.939 0.347881
## TeamCWS                        4.2944     4.1022   1.047 0.295413
## TeamDET                        2.3024     3.9725   0.580 0.562326
## TeamFLA                        2.6985     4.1336   0.653 0.514028
## TeamHOU                       -0.6808     4.0634  -0.168 0.866976
## TeamKC                        -4.7664     4.0242  -1.184 0.236525
## TeamLA                         2.8598     4.0817   0.701 0.483686
## TeamMIN                        2.1269     4.0947   0.519 0.603579
## TeamMLW                        4.2897     4.0243   1.066 0.286706
## TeamNYM                       -1.9736     3.9493  -0.500 0.617370
## TeamNYY                        1.7483     4.1234   0.424 0.671655
## TeamOAK                       -0.5464     3.9672  -0.138 0.890474
## TeamPHI                       -6.8486     3.9949  -1.714 0.086778 .
## TeamPIT                        4.3023     4.0210   1.070 0.284890
## TeamSD                         2.6133     4.0915   0.639 0.523148
## TeamSEA                       -0.9147     4.0516  -0.226 0.821436
## TeamSF                         0.8411     4.0520   0.208 0.835593
## TeamSTL                       -1.1341     4.1193  -0.275 0.783132
## TeamTB                        -2.6616     4.0944  -0.650 0.515798
## TeamTEX                       -0.7695     4.0283  -0.191 0.848556
## TeamTOR                        1.3943     4.0681   0.343 0.731871
## TeamWAS                       -1.7555     4.0038  -0.438 0.661142
## PositionDesignated_Hitter      8.9037     4.4533   1.999 0.045842 *
## PositionFirst_Baseman          2.4237     3.0058   0.806 0.420236
## PositionOutfielder            -6.2636     2.2784  -2.749 0.006084 **
## PositionRelief_Pitcher        -7.7695     2.1959  -3.538 0.000421 ***
## PositionSecond_Baseman       -13.0843     2.9638  -4.415 1.12e-05 ***
## PositionShortstop            -16.9562     3.0406  -5.577 3.16e-08 ***
## PositionStarting_Pitcher      -7.3599     2.2976  -3.203 0.001402 **
## PositionThird_Baseman         -4.6035     3.1689  -1.453 0.146613
## Height                         4.7175     0.2563  18.405  < 2e-16 ***
## Age                            0.8906     0.1259   7.075 2.82e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.78 on 994 degrees of freedom
## Multiple R-squared:  0.3858, Adjusted R-squared:  0.3617
## F-statistic: 16.01 on 39 and 994 DF,  p-value: < 2.2e-16

plot(fit, which = 1:2)
```

The model summary shows us how well the model fits the data.

Residuals This tells us about the residuals. If we have extremely large or extremely small residuals for some observations compared to the rest of residuals, either they are outliers due to reporting error or the model fits data poorly. We have 73.649 as our maximum and −48.692 as our minimum. The residuals could be characterized by examining their range and by viewing the residual diagnostic plots.

Coefficients In this section of the output, we look at the very right column that has symbols like stars or dots showing if that variable is significant and should be included in the model. However, if no symbol is included next to a variable, then it means this estimated covariate coefficient in the linear model covariance could be trivial. Another thing we can look at is the `Pr(>|t|)` column. A number close to zero in this column indicates the row variable is significant, otherwise it could be removed from the model.

In this example, some of the teams and positions are significant and some are not. Both `Age` and `Height` are significant.

R-squared What percent in `y` is explained by the included predictors? Here, we have 38.58%, which indicates the model is not bad but could be improved. Usually a well-fitted linear regression would have R-squared over 70%.

The **diagnostic plots** also help us understand the model quality.

Residual vs. Fitted This is the main residual diagnostic plot, Fig. 10.8. We can see that the residuals of observations indexed 65, 160 and 237 are relatively far apart from the rest. They may represent potential influential points or outliers.

Normal Q-Q This plot examines the normality assumption of the model, Fig. 10.9. The scattered dots represent the matched quantiles of the data and the normal distribution. If the Q-Q plot closely resembles a line bisecting the first quadrant in the plane, the normality assumption is valid. In our case, it is relatively close to the line. So, we can say that our model is valid in terms of normality.

10.4 Step 5: Improving Model Performance

We can employ the `step` function to perform forward or backward selection of important features/predictors. It works for both `lm` and `glm` models. In most cases, backward-selection is preferable because it tends to retain much larger models. On the other hand, there are various criteria to evaluate a model. The common model evaluation metrics include *AIC*, *BIC*, Adjusted R^2, etc. In Chap. 14, we will present more details about prediction evaluation and assessment of classificaiton. Let's compare the backward and forward model selection approaches. The `step` function argument `direction` allows this control (default is `both`, which will select the better result from either backward or forward selection).

```
step(fit,direction = "backward")

## Start:  AIC=5871.04
## Weight ~ Team + Position + Height + Age
##
##             Df Sum of Sq    RSS    AIC
## - Team      29      9468 289262 5847.4
## <none>                   279793 5871.0
## - Age        1     14090 293883 5919.8
## - Position   8     20301 300095 5927.5
## - Height     1     95356 375149 6172.3
##
## Step:  AIC=5847.45
## Weight ~ Position + Height + Age
##
##             Df Sum of Sq    RSS    AIC
## <none>                   289262 5847.4
## - Age        1     14616 303877 5896.4
## - Position   8     20406 309668 5901.9
## - Height     1    100435 389697 6153.6

##
## Call:
## lm(formula = Weight ~ Position + Height + Age, data = mlb)
##
## Coefficients:
##                (Intercept)  PositionDesignated_Hitter
##                  -168.0474                     8.6968
##        PositionFirst_Baseman         PositionOutfielder
##                     2.7780                    -6.0457
##       PositionRelief_Pitcher      PositionSecond_Baseman
##                    -7.7782                   -13.0267
##           PositionShortstop   PositionStarting_Pitcher
##                   -16.4821                    -7.3961
##        PositionThird_Baseman                     Height
##                    -4.1361                     4.7639
##                         Age
##                     0.8771

step(fit,direction = "forward")

## Start:  AIC=5871.04
## Weight ~ Team + Position + Height + Age

##
## Call:
## lm(formula = Weight ~ Team + Position + Height + Age, data = mlb)
##
## Coefficients:
##                (Intercept)                    TeamARZ
##                  -164.9995                     7.1881
##                    TeamATL                    TeamBAL
##                    -1.5631                    -5.3128
##                    TeamBOS                    TeamCHC
##                    -0.2838                     0.4026
##                    TeamCIN                    TeamCLE
##                     2.1051                    -1.3160
##                    TeamCOL                    TeamCWS
##                     -3.7836                     4.2944
##                     TeamDET                    TeamFLA
##                      2.3024                     2.6985
##                     TeamHOU                     TeamKC
##                     -0.6808                    -4.7664
##                      TeamLA                    TeamMIN
##                      2.8598                     2.1269
##                     TeamMLW                    TeamNYM
##                      4.2897                    -1.9736
##                     TeamNYY                    TeamOAK
##                      1.7483                    -0.5464
##                     TeamPHI                    TeamPIT
##                     -6.8486                     4.3023
##                      TeamSD                    TeamSEA
##                      2.6133                    -0.9147
##                      TeamSF                    TeamSTL
##                      0.8411                    -1.1341
##                      TeamTB                    TeamTEX
##                     -2.6616                    -0.7695
##                     TeamTOR                    TeamWAS
##                      1.3943                    -1.7555
## PositionDesignated_Hitter      PositionFirst_Baseman
##                      8.9037                     2.4237
##          PositionOutfielder     PositionRelief_Pitcher
##                     -6.2636                    -7.7695
##     PositionSecond_Baseman          PositionShortstop
##                    -13.0843                   -16.9562
##   PositionStarting_Pitcher      PositionThird_Baseman
##                     -7.3599                    -4.6035
##                      Height                        Age
##                      4.7175                     0.8906
```

```
step(fit,direction = "both")
```

```
## Start:  AIC=5871.04
## Weight ~ Team + Position + Height + Age
##
##            Df Sum of Sq    RSS    AIC
## - Team     29      9468 289262 5847.4
## <none>                  279793 5871.0
## - Age       1     14090 293883 5919.8
## - Position  8     20301 300095 5927.5
## - Height    1     95356 375149 6172.3
##
## Step:  AIC=5847.45
## Weight ~ Position + Height + Age
##
##            Df Sum of Sq    RSS    AIC
## <none>                  289262 5847.4
## + Team     29      9468 279793 5871.0
## - Age       1     14616 303877 5896.4
## - Position  8     20406 309668 5901.9
## - Height    1    100435 389697 6153.6

##
## Call:
## lm(formula = Weight ~ Position + Height + Age, data = mlb)
```

```
##
## Coefficients:
##               (Intercept)  PositionDesignated_Hitter
##                 -168.0474                     8.6968
##     PositionFirst_Baseman          PositionOutfielder
##                    2.7780                    -6.0457
##    PositionRelief_Pitcher      PositionSecond_Baseman
##                   -7.7782                   -13.0267
##         PositionShortstop    PositionStarting_Pitcher
##                  -16.4821                    -7.3961
##     PositionThird_Baseman                      Height
##                   -4.1361                     4.7639
##                       Age
##                    0.8771
```

We can observe that forward retains the whole model. The better feature selection model uses `backward` step-wise selection.

Both backward and forward are greedy algorithms and neither guarantees an optimal model result. The optimal feature selection requires exploring every possible combination of the predictors, which is practically not feasible, due to computational complexity, $\binom{n}{k}$ combinations.

Alternatively, we can choose models based on various **information criteria**.

```
step(fit,k=2)

## Start:  AIC=5871.04
## Weight ~ Team + Position + Height + Age
##
##            Df Sum of Sq    RSS    AIC
## - Team      29      9468 289262 5847.4
## <none>                   279793 5871.0
## - Age        1     14090 293883 5919.8
## - Position   8     20301 300095 5927.5
## - Height     1     95356 375149 6172.3
##
## Step:  AIC=5847.45
## Weight ~ Position + Height + Age
##
##            Df Sum of Sq    RSS    AIC
## <none>                   289262 5847.4
## - Age        1     14616 303877 5896.4
## - Position   8     20406 309668 5901.9
## - Height     1    100435 389697 6153.6

##
## Call:
## lm(formula = Weight ~ Position + Height + Age, data = mlb)
##
## Coefficients:
##               (Intercept)  PositionDesignated_Hitter
##                 -168.0474                     8.6968
##     PositionFirst_Baseman          PositionOutfielder
##                    2.7780                     -6.0457
##    PositionRelief_Pitcher     PositionSecond_Baseman
##                   -7.7782                    -13.0267
##         PositionShortstop   PositionStarting_Pitcher
##                  -16.4821                    -7.3961
##     PositionThird_Baseman                     Height
##                   -4.1361                     4.7639
##                       Age
##                    0.8771
```

```
step(fit,k=log(nrow(mlb)))
```

```
## Start:  AIC=6068.69
## Weight ~ Team + Position + Height + Age
##
##             Df Sum of Sq    RSS    AIC
## - Team      29      9468 289262 5901.8
## <none>                   279793 6068.7
## - Position   8     20301 300095 6085.6
## - Age        1     14090 293883 6112.5
## - Height     1     95356 375149 6365.0
##
## Step:  AIC=5901.8
## Weight ~ Position + Height + Age
##
##             Df Sum of Sq    RSS    AIC
## <none>                   289262 5901.8
## - Position   8     20406 309668 5916.8
## - Age        1     14616 303877 5945.8
## - Height     1    100435 389697 6203.0

##
## Call:
## lm(formula = Weight ~ Position + Height + Age, data = mlb)
##
## Coefficients:
##               (Intercept)  PositionDesignated_Hitter
##                 -168.0474                     8.6968
##     PositionFirst_Baseman          PositionOutfielder
##                    2.7780                     -6.0457
##    PositionRelief_Pitcher     PositionSecond_Baseman
##                   -7.7782                    -13.0267
##         PositionShortstop   PositionStarting_Pitcher
##                  -16.4821                    -7.3961
##     PositionThird_Baseman                     Height
##                   -4.1361                     4.7639
##                       Age
##                    0.8771
```

$k = 2$ yields the AIC criterion, and $k = \log(n)$ refers to BIC. Let's try to evaluate the model performance again (Figs. 10.10 and 10.11).

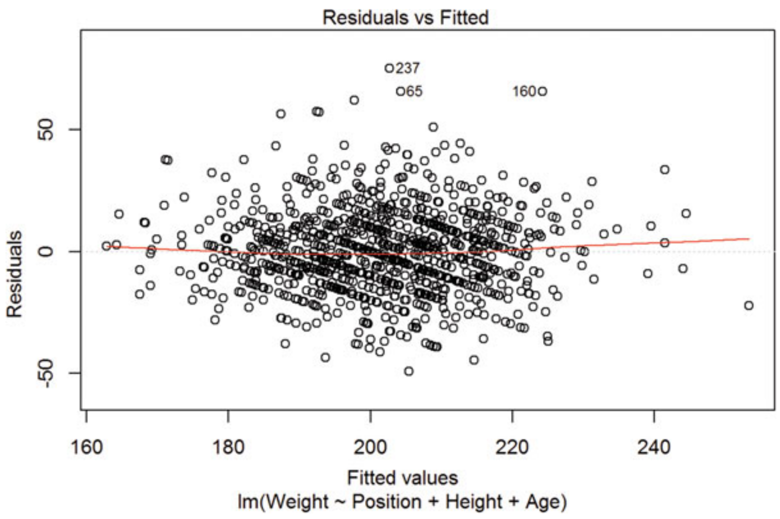

Fig. 10.10 Residuals vs. fitted values scatterplot

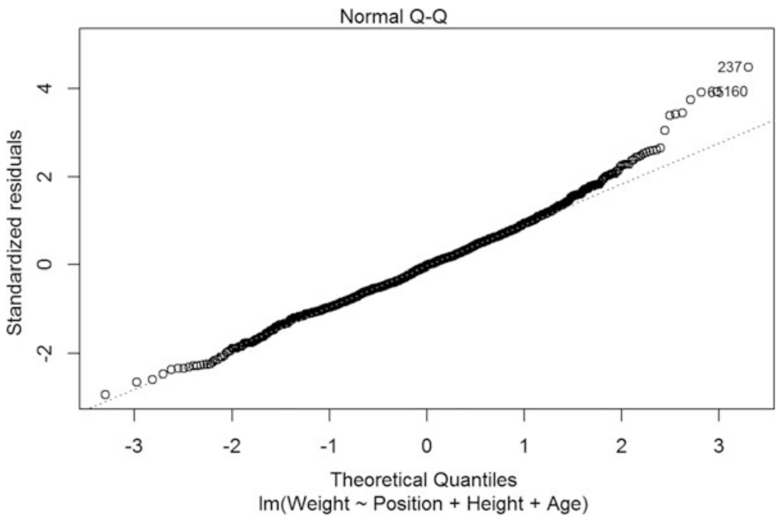

Fig. 10.11 QQ normal probability plot of the model residuals

```
fit2 = step(fit,k=2,direction = "backward")
## Start:  AIC=5871.04
## Weight ~ Team + Position + Height + Age
##
##            Df Sum of Sq    RSS    AIC
## - Team     29      9468 289262 5847.4
## <none>                  279793 5871.0
## - Age       1     14090 293883 5919.8
## - Position  8     20301 300095 5927.5
## - Height    1     95356 375149 6172.3
##
## Step:  AIC=5847.45
## Weight ~ Position + Height + Age
##
##            Df Sum of Sq    RSS    AIC
## <none>                  289262 5847.4
## - Age       1     14616 303877 5896.4
## - Position  8     20406 309668 5901.9
## - Height    1    100435 389697 6153.6
summary(fit2)
##
## Call:
## lm(formula = Weight ~ Position + Height + Age, data = mlb)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.427 -10.855  -0.344  10.110  75.301
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -168.0474    19.0351  -8.828  < 2e-16 ***
## PositionDesignated_Hitter    8.6968     4.4258   1.965 0.049679 *
## PositionFirst_Baseman        2.7780     2.9942   0.928 0.353741
## PositionOutfielder          -6.0457     2.2778  -2.654 0.008072 **
## PositionRelief_Pitcher      -7.7782     2.1913  -3.550 0.000403 ***
## PositionSecond_Baseman     -13.0267     2.9531  -4.411 1.14e-05 ***
## PositionShortstop          -16.4821     3.0372  -5.427 7.16e-08 ***
## PositionStarting_Pitcher    -7.3961     2.2959  -3.221 0.001316 **
## PositionThird_Baseman       -4.1361     3.1656  -1.307 0.191647
## Height                       4.7639     0.2528  18.847  < 2e-16 ***
## Age                          0.8771     0.1220   7.190 1.25e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.82 on 1023 degrees of freedom
## Multiple R-squared:  0.365,  Adjusted R-squared:  0.3588
## F-statistic: 58.81 on 10 and 1023 DF,  p-value: < 2.2e-16
plot(fit2, which = 1:2)
```

Sometimes, we prefer a simpler model even if there is slight loss in performance. In this case, we have a simpler model and $R^2 = 0.365$. The whole model is still very significant. Some potential influential points or outliers that are relatively far from other residuals observations are shown on Fig. 10.12.

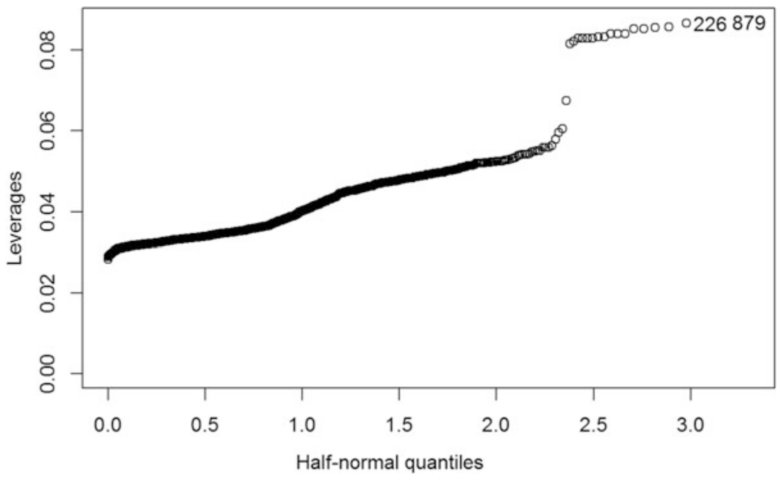

Fig. 10.12 A half-normal probability plot suggesting important factors or interactions by estimating the impact of a given main effect, or interaction, and its rank relative to other main effects and interactions computed via least squares estimation. The horizontal and vertical axes represent the (n-1) theoretical order statistic medians from a half-normal distribution and the ordered absolute value of the estimated effects for the main factors and available interactions, respectively

```
# Half-normal plot for leverages
# install.packages("faraway")
library(faraway)

halfnorm(lm.influence(fit)$hat, nlab = 2, ylab="Leverages")
```

```
mlb[c(226,879),]
##     Team          Position Height Weight   Age
## 226  NYY Designated_Hitter     75    230 36.14
## 879   SD Designated_Hitter     73    200 25.60
summary(mlb)
##       Team                   Position       Height         Weight
##   NYM    : 38   Relief_Pitcher   :315   Min.   :67.0   Min.   :150.0
##   ATL    : 37   Starting_Pitcher:221   1st Qu.:72.0   1st Qu.:187.0
##   DET    : 37   Outfielder       :194   Median :74.0   Median :200.0
##   OAK    : 37   Catcher          : 76   Mean   :73.7   Mean   :201.7
##   BOS    : 36   Second_Baseman   : 58   3rd Qu.:75.0   3rd Qu.:215.0
##   CHC    : 36   First_Baseman    : 55   Max.   :83.0   Max.   :290.0
##   (Other):813   (Other)          :115
##        Age
##   Min.   :20.90
##   1st Qu.:25.44
##   Median :27.93
##   Mean   :28.74
##   3rd Qu.:31.23
##   Max.   :48.52
```

A deeper discussion of variable selection, controlling the false discovery rate, is provided in Chaps. 17 and 18.

10.4.1 Model Specification: Adding Non-linear Relationships

In linear regression, the relationship between independent and dependent variables is assumed to be linear. However, this might not be the case. The relationship between age and weight could be quadratic, since middle-aged people might gain weight dramatically.

```
mlb$age2<-(mlb$Age)^2
fit2<-lm(Weight~., data=mlb)
summary(fit2)

## Call:
## lm(formula = Weight ~ ., data = mlb)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.068 -10.775  -1.021   9.922  74.693
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -209.07068   27.49529  -7.604 6.65e-14 ***
## TeamARZ                       7.41943    4.25154   1.745 0.081274 .
## TeamATL                      -1.43167    3.96793  -0.361 0.718318
## TeamBAL                      -5.38735    4.01119  -1.343 0.179552
## TeamBOS                      -0.06614    3.99633  -0.017 0.986799
## TeamCHC                       0.14541    3.98833   0.036 0.970923
## TeamCIN                       2.24022    3.98571   0.562 0.574201
## TeamCLE                      -1.07546    4.02870  -0.267 0.789563
## TeamCOL                      -3.87254    4.02069  -0.963 0.335705
## TeamCWS                       4.20933    4.09393   1.028 0.304111
## TeamDET                       2.66990    3.96769   0.673 0.501160
## TeamFLA                       3.14627    4.12989   0.762 0.446343
## TeamHOU                      -0.77230    4.05526  -0.190 0.849000
## TeamKC                       -4.90984    4.01648  -1.222 0.221837
## TeamLA                        3.13554    4.07514   0.769 0.441820
## TeamMIN                       2.09951    4.08631   0.514 0.607512
## TeamMLW                       4.16183    4.01646   1.036 0.300363
## TeamNYM                      -1.25057    3.95424  -0.316 0.751870
## TeamNYY                       1.67825    4.11502   0.408 0.683482
## TeamOAK                      -0.68235    3.95951  -0.172 0.863212
## TeamPHI                      -6.85071    3.98672  -1.718 0.086039 .
## TeamPIT                       4.12683    4.01348   1.028 0.304086
## TeamSD                        2.59525    4.08310   0.636 0.525179
## TeamSEA                      -0.67316    4.04471  -0.166 0.867853
## TeamSF                        1.06038    4.04481   0.262 0.793255
## TeamSTL                      -1.38669    4.11234  -0.337 0.736037
## TeamTB                       -2.44396    4.08716  -0.598 0.550003
## TeamTEX                      -0.68740    4.02023  -0.171 0.864270
## TeamTOR                       1.24439    4.06029   0.306 0.759306
## TeamWAS                      -1.87599    3.99594  -0.469 0.638835
## PositionDesignated_Hitter     8.94440    4.44417   2.013 0.044425 *
## PositionFirst_Baseman         2.55100    3.00014   0.850 0.395368
## PositionOutfielder           -6.25702    2.27372  -2.752 0.006033 **
## PositionRelief_Pitcher       -7.68904    2.19166  -3.508 0.000471 ***
## PositionSecond_Baseman      -13.01400    2.95787  -4.400 1.20e-05 ***
## PositionShortstop           -16.82243    3.03494  -5.543 3.81e-08 ***
## PositionStarting_Pitcher     -7.08215    2.29615  -3.084 0.002096 **
## PositionThird_Baseman       -4.66452    3.16249  -1.475 0.140542
## Height                       4.71888    0.25578  18.449  < 2e-16 ***
## Age                          3.82295    1.30621   2.927 0.003503 **
## age2                        -0.04791    0.02124  -2.255 0.024327 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.74 on 993 degrees of freedom
## Multiple R-squared:  0.3889, Adjusted R-squared:  0.3643
## F-statistic:  15.8 on 40 and 993 DF,  p-value: < 2.2e-16
```

This actually brought up the overall R^2 up to 0.3889.

10.4.2 Transformation: Converting a Numeric Variable to a Binary Indicator

As discussed earlier, middle-aged people might have a different pattern in weight increase compared to younger people. The overall pattern could be not cumulative, but could rather represent alternative lines for the young and the middle-aged people. We assume 30 is the age threshold separating young from middle-age players. Players over 30 may have a steeper line for weight increase than those under 30. Here, we use the `ifelse()` function that we mentioned in Chap. 8 to create the indicator of this Age threshold.

```
mlb$age30<-ifelse(mlb$Age>=30, 1, 0)
fit3<-lm(Weight~Team+Position+Age+age30+Height, data=mlb)
summary(fit3)

##
## Call:
## lm(formula = Weight ~ Team + Position + Age + age30 + Height,
##     data = mlb)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -48.313 -11.166  -0.916  10.044  73.630
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -159.8884    19.8862  -8.040 2.54e-15 ***
## TeamARZ                      7.4096     4.2627   1.738 0.082483 .
## TeamATL                     -1.4379     3.9765  -0.362 0.717727
## TeamBAL                     -5.3393     4.0187  -1.329 0.184284
## TeamBOS                     -0.1985     4.0034  -0.050 0.960470
## TeamCHC                      0.4669     3.9947   0.117 0.906976
## TeamCIN                      2.2124     3.9939   0.554 0.579741
## TeamCLE                     -1.1624     4.0371  -0.288 0.773464
## TeamCOL                     -3.6842     4.0290  -0.914 0.360717
## TeamCWS                      4.1920     4.1025   1.022 0.307113
## TeamDET                      2.4708     3.9746   0.622 0.534314
## TeamFLA                      2.8563     4.1352   0.691 0.489903
## TeamHOU                      -0.4964     4.0659  -0.122 0.902846
## TeamKC                       -4.7138     4.0238  -1.171 0.241692
## TeamLA                        2.9194     4.0814   0.715 0.474586
## TeamMIN                       2.2885     4.0965   0.559 0.576528
## TeamMLW                       4.4749     4.0269   1.111 0.266731
## TeamNYM                      -1.8173     3.9510  -0.460 0.645659
## TeamNYY                       1.7074     4.1229   0.414 0.678867
## TeamOAK                      -0.3388     3.9707  -0.085 0.932012
## TeamPHI                      -6.6192     3.9993  -1.655 0.098220 .
## TeamPIT                       4.6716     4.0332   1.158 0.247029
## TeamSD                        2.8600     4.0965   0.698 0.485243
## TeamSEA                      -1.0121     4.0518  -0.250 0.802809
## TeamSF                        1.0244     4.0545   0.253 0.800587
## TeamSTL                      -1.1094     4.1187  -0.269 0.787703
## TeamTB                       -2.4485     4.0980  -0.597 0.550312
## TeamTEX                      -0.6112     4.0300  -0.152 0.879485
## TeamTOR                       1.3959     4.0674   0.343 0.731532
## TeamWAS                      -1.4189     4.0139  -0.354 0.723784
## PositionDesignated_Hitter     9.2378     4.4621   2.070 0.038683 *
## PositionFirst_Baseman         2.6074     3.0096   0.866 0.386501
## PositionOutfielder           -6.0408     2.2863  -2.642 0.008367 **
## PositionRelief_Pitcher       -7.5100     2.2072  -3.403 0.000694 ***
## PositionSecond_Baseman      -12.8870     2.9683  -4.342 1.56e-05 ***
## PositionShortstop           -16.8912     3.0406  -5.555 3.56e-08 ***
## PositionStarting_Pitcher     -7.0825     2.3099  -3.066 0.002227 **
## PositionThird_Baseman        -4.4307     3.1719  -1.397 0.162773
## Age                           0.6904     0.2153   3.207 0.001386 **
## age30                         2.2636     1.9749   1.146 0.251992
## Height                        4.7113     0.2563  18.380  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.77 on 993 degrees of freedom
## Multiple R-squared:  0.3866, Adjusted R-squared:  0.3619
## F-statistic: 15.65 on 40 and 993 DF,  p-value: < 2.2e-16
```

This model performs worse than the quadratic model in terms of R^2. Moreover, `age30` is not significant. So, we will stick with the earlier quadratic model.

10.4.3 Model Specification: Adding Interaction Effects

So far, each feature's individual effect was considered in our models. It is possible that features act in pairs to affect the independent variable. Let's examine that deeper.

Interactions are combined effects of two or more features. If we are not sure whether two features interact term we could test by adding an interaction term into the model. If the interaction term is significant, it confirms that there may be non-trivial interaction between the features.

```
fit4<-lm(Weight~Team+Height+Age*Position+age2, data=mlb)
summary(fit4)

## Call:
## lm(formula = Weight ~ Team + Height + Age * Position + age2,
##     data = mlb)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -48.761 -11.049  -0.761   9.911  75.533
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   -199.15403   29.87269  -6.667 4.35e-11 ***
## TeamARZ                          8.10376    4.26339   1.901   0.0576 .
## TeamATL                         -0.81743    3.97899  -0.205   0.8373
## TeamBAL                         -4.64820    4.03972  -1.151   0.2502
## TeamBOS                          0.37698    4.00743   0.094   0.9251
## TeamCHC                          0.33104    3.99507   0.083   0.9340
## TeamCIN                          2.56023    3.99603   0.641   0.5219
## TeamCLE                         -0.66254    4.03154  -0.164   0.8695
## TeamCOL                         -3.72098    4.03759  -0.922   0.3570
## TeamCWS                          4.63266    4.10884   1.127   0.2598
## TeamDET                          3.21380    3.98231   0.807   0.4199
## TeamFLA                          3.56432    4.14902   0.859   0.3905
## TeamHOU                         -0.38733    4.07249  -0.095   0.9242
## TeamKC                          -4.66678    4.02384  -1.160   0.2464
## TeamLA                           3.51766    4.09400   0.859   0.3904
## TeamMIN                          2.31585    4.10502   0.564   0.5728
## TeamMLW                          4.34793    4.02501   1.080   0.2803
## TeamNYM                         -0.28505    3.98537  -0.072   0.9430
## TeamNYY                          1.87847    4.12774   0.455   0.6491
## TeamOAK                         -0.23791    3.97729  -0.060   0.9523
## TeamPHI                         -6.25671    3.99545  -1.566   0.1177
## TeamPIT                          4.18719    4.01944   1.042   0.2978
## TeamSD                           2.97028    4.08838   0.727   0.4677
## TeamSEA                         -0.07220    4.05922  -0.018   0.9858
## TeamSF                           1.35981    4.07771   0.333   0.7388
## TeamSTL                         -1.23460    4.11960  -0.300   0.7645
## TeamTB                          -1.90885    4.09592  -0.466   0.6413
## TeamTEX                         -0.31570    4.03146  -0.078   0.9376
## TeamTOR                          1.73976    4.08565   0.426   0.6703
## TeamWAS                         -1.43933    4.00274  -0.360   0.7192
## Height                           4.70632    0.25646  18.351  < 2e-16 ***
## Age                              3.32733    1.37088   2.427   0.0154 *
## PositionDesignated_Hitter      -44.82216   30.68202  -1.461   0.1444
## PositionFirst_Baseman           23.51389   20.23553   1.162   0.2455
## PositionOutfielder             -13.33140   15.92500  -0.837   0.4027
## PositionRelief_Pitcher         -16.51308   15.01240  -1.100   0.2716
## PositionSecond_Baseman         -26.56932   20.18773  -1.316   0.1884
## PositionShortstop              -27.89454   20.72123  -1.346   0.1786
## PositionStarting_Pitcher        -2.44578   15.36376  -0.159   0.8736
## PositionThird_Baseman          -10.20102   23.26121  -0.439   0.6611
## age2                            -0.04201    0.02170  -1.936   0.0531 .
## Age:PositionDesignated_Hitter    1.77289    1.00506   1.764   0.0780 .
## Age:PositionFirst_Baseman       -0.71111    0.67848  -1.048   0.2949
## Age:PositionOutfielder           0.24147    0.53650   0.450   0.6527
## Age:PositionRelief_Pitcher       0.30374    0.50564   0.601   0.5482
## Age:PositionSecond_Baseman       0.46281    0.68281   0.678   0.4981
## Age:PositionShortstop            0.38257    0.70998   0.539   0.5901
## Age:PositionStarting_Pitcher    -0.17104    0.51976  -0.329   0.7422
## Age:PositionThird_Baseman        0.18968    0.79561   0.238   0.8116
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.73 on 985 degrees of freedom
## Multiple R-squared:  0.3945, Adjusted R-squared:  0.365
## F-statistic: 13.37 on 48 and 985 DF,  p-value: < 2.2e-16
```

The results indicates that the overall R^2 improved and some of the interactions are significant at the under 0.1 level.

10.5 Understanding Regression Trees and Model Trees

As we saw in Chap. 9, a decision tree builds by multiple `if-else` logical decisions and can classify observations. We could add regression into decision trees so that a decision tree can make numerical predictions.

10.5.1 Adding Regression to Trees

Numeric prediction trees are built in the same way as classification trees. Data are partitioned first via a divide-and-conquer strategy based on features. Recall that, homogeneity in classification trees may be assessed by measures like the entropy. In prediction, tree homogeneity is measured by statistics such as variance, standard deviation or absolute deviation, from the mean.

A common splitting criterion for regression trees is the **standard deviation reduction (SDR)**.

$$SDR = sd(T) - \sum_{i=1}^{n} \left|\frac{T_i}{T}\right| \times sd(T_i),$$

where $sd(T)$ is the standard deviation for the original data. After the summation of all segments, $\left|\frac{T_i}{T}\right|$ is the proportion of observations in the ith segment compared to the total number of observations and $sd(T_i)$ is the standard deviation for the ith segment. Let's look at one simple example.

$$Original\,data : \{1,2,3,3,4,5,6,6,7,8\},$$
$$Split\ method\,1 : \{1,2,3|3,4,5,6,6,7,8\}, \text{and}$$
$$Split\ method\,2 : \{1,2,3,3,4,5|6,6,7,8\}.$$

In split method 1, $T_1 = \{1, 2, 3\}$, $T_2 = \{3, 4, 5, 6, 6, 7, 8\}$. In split method 2, $T_1 = \{1, 2, 3, 3, 4, 5\}$, $T_2 = \{6, 6, 7, 8\}$.

```
ori<-c(1, 2, 3, 3, 4, 5, 6, 6, 7, 8)
at1<-c(1, 2, 3)
at2<-c(3, 4, 5, 6, 6, 7, 8)
bt1<-c(1, 2, 3, 3, 4, 5)
bt2<-c(6, 6, 7, 8)
sdr_a <- sd(ori)-(length(at1)/length(ori)*sd(at1)+length(at2)/length(ori) *
sd(at2))
sdr_b <- sd(ori)-(length(bt1)/length(ori)*sd(bt1)+length(bt2)/length(ori) *
sd(bt2))
sdr_a

## [1] 0.7702557

sdr_b

## [1] 1.041531
```

`length()` is used in the above R codes to get the number of elements in a specific vector.

Larger SDR indicates greater reduction in standard deviation after splitting. Here, split method 2 yields greater SDR, so the regression tree split will use the second method, which results in more homogeneous sets than the first method.

Now, the tree will be split under `bt1` and `bt2` following the same rules (greater SDR wins). When we cannot split further `bt1` and `bt2` are terminal nodes. The observations classified into `bt1` will be predicted with $mean(bt1) = 3$, and those classified as `bt2` with $mean(bt2) = 6.75$.

10.6 Case Study 2: Baseball Players (Take 2)

10.6.1 Step 2: Exploring and Preparing the Data

We will continue with the MLB dataset, which includes 1,034 observations. Let's try to randomly separate them into training and testing datasets first.

```
set.seed(1234)
train_index <- sample(seq_len(nrow(mlb)), size = 0.75*nrow(mlb))
mlb_train<-mlb[train_index, ]
mlb_test<-mlb[-train_index, ]
```

We used a random 75–25% split to divide the data into training and testing sets.

10.6.2 Step 3: Training a Model On the Data

In R, the function `rpart()`, under the `rpart` package, provides regression tree modeling:

m<-rpart(dv~iv, data=mydata)

- dv: dependent variable
- iv: independent variable
- mydata: training data containing `dv` and `iv`.

We use two numerical features in the MLB data "01a_data.txt" `Age` and `Height` as features.

```
#install.packages("rpart")
library(rpart)

mlb.rpart<-rpart(Weight~Height+Age, data=mlb_train)
mlb.rpart

## n= 775
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 775 323502.600 201.4361
##    2) Height< 73.5 366 112465.500 192.5000
##      4) Height< 70.5 55   9865.382 178.7818 *
##      5) Height>=70.5 311  90419.300 194.9260
##       10) Age< 31.585 234  71123.060 192.8547 *
##       11) Age>=31.585 77  15241.250 201.2208 *
##    3) Height>=73.5 409 155656.400 209.4328
##      6) Height< 76.5 335 118511.700 206.8627
##       12) Age< 28.6 194  75010.250 202.2938
##         24) Height< 74.5 76  20688.040 196.8026 *
##         25) Height>=74.5 118  50554.610 205.8305 *
##       13) Age>=28.6 141  33879.870 213.1489 *
##      7) Height>=76.5 74  24914.660 221.0676
##       14) Age< 25.37 12   3018.000 206.0000 *
##       15) Age>=25.37 62  18644.980 223.9839 *
```

The output contains rich information. `split` indicates the decision criterion; `n` is the number of observations that fall in this segment; `yval` is the predicted value if the test data falls into a segment.

10.6.3 Visualizing Decision Trees

A fancy way of drawing the `rpart` decision tree is by the `rpart.plot()` function under `rpart.plot` package (Fig. 10.13).

```
# install.packages("rpart.plot")
library(rpart.plot)
rpart.plot(mlb.rpart, digits=3)
```

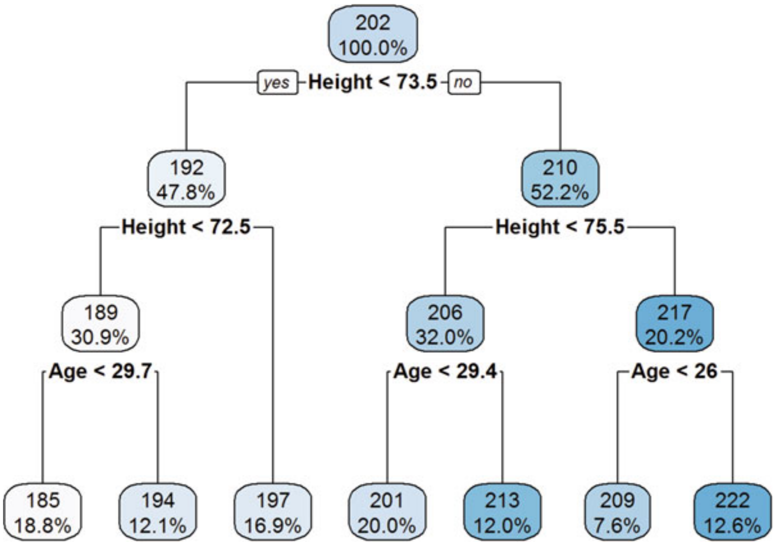

Fig. 10.13 MLB decision tree partitioning

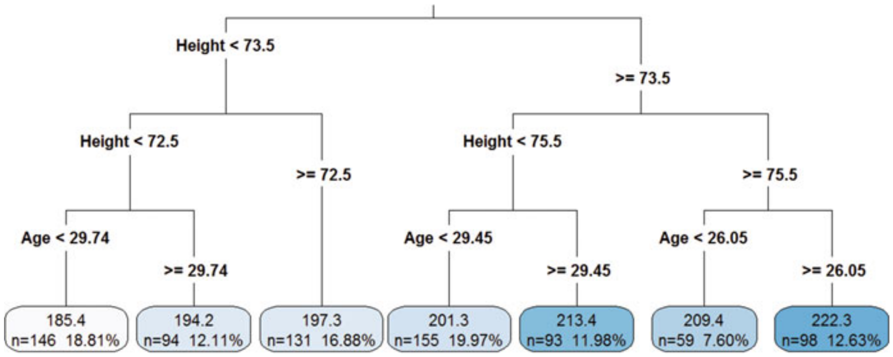

Fig. 10.14 Expanding the decision tree by specifying significant digits, drawing separate split labels for the left and right directions, displaying the number and percentage of observations in the node, and positioning the leaf nodes at the bottom of the graph

A more detailed graph can be obtained by specifying more options in the function call (Fig. 10.14).

```
rpart.plot(mlb.rpart, digits = 4, fallen.leaves = T, type=3, extra=101)
```

We may also use a more elaborate tree plot from package `rattle` to observe the order and rules of splits (Fig. 10.15).

```
library(rattle)
fancyRpartPlot(mlb.rpart, cex = 0.8)
```

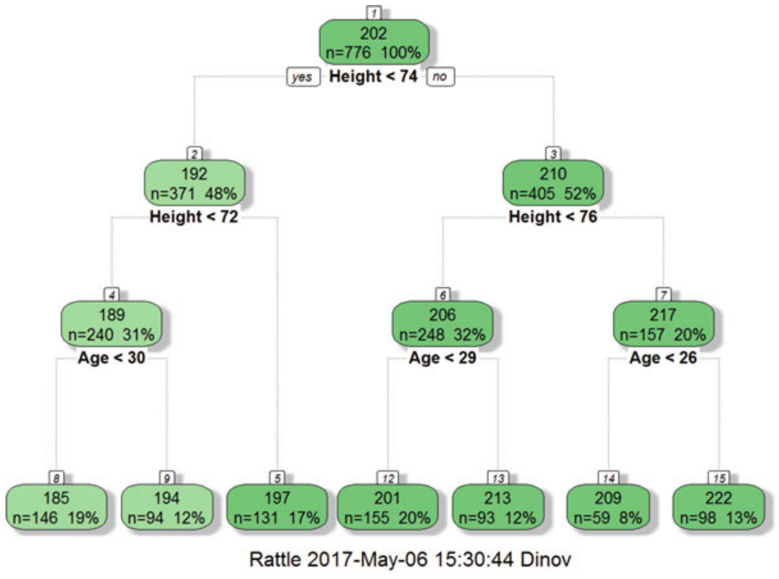

Fig. 10.15 An alternative plot of the MLB decision tree

10.6.4 Step 4: Evaluating Model Performance

Let's make predictions with the regression tree model using the `predict()` command.

```
mlb.p<-predict(mlb.rpart, mlb_test)
summary(mlb.p)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   185.4   194.2   201.3   201.8   213.4   222.3

summary(mlb_test$Weight)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   150.0   190.0   200.0   202.4   215.0   260.0
```

Comparing the five-number statistics for the predicted and true `Weight`, we can see that the model cannot precisely identify extreme cases such as the maximum. However, within the IQR, the predictions are relatively accurate. Correlation could be used to measure the correspondence of two equal length numeric variables. Let's use `cor()` to examine the prediction accuracy.

```
cor(mlb.p, mlb_test$Weight)
## [1] 0.4940257
```

The predicted values are moderately correlated with the true values.

10.6.5 Measuring Performance with Mean Absolute Error

To measure the distance between predicted value and the true value, we can use a measurement called mean absolute error (MAE) defined by the formula:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} \mid pred_i - obs_i \mid ,$$

where the $pred_i$ is the *i*th predicted value and obs_i is the *i*th observed value. Let's make a corresponding MAE function in R and evaluate our model performance.

```
MAE<-function(obs, pred){
  mean(abs(obs-pred))
}
MAE(mlb_test$Weight, mlb.p)

## [1] 14.97519
```

This implies that on average, the difference between the predicted value and the observed value is 14.975. Considering that the `Weight` variable in our test dataset ranges from 150 to 260, the model is reasonable.

What if we used a more the most primitive method for prediction – the test data **mean**?

```
mean(mlb_test$Weight)

## [1] 202.3643

MAE(mlb_test$Weight, 202.3643)

## [1] 17.11207
```

This shows that the regression decision tree is better than using the mean to predict every observation in the test dataset. However, it is not dramatically better. There might be room for improvement.

10.6.6 Step 5: Improving Model Performance

To improve the performance of our decision tree, we are going to use a model tree instead of a regression tree. We can use the `M5P()` function, under the package `RWeka`, which implements the `M5` algorithm. This function uses similar syntax as `rpart()`.

m<-M5P(dv~iv, data=mydata)

```
#install.packages("RWeka")

# Sometimes RWeka installations may be off a bit, see:
# http://stackoverflow.com/questions/41878226/using-rweka-m5p-in-rstudio-yie
lds-java-lang-noclassdeffounderror-no-uib-cipr-ma

Sys.getenv("WEKA_HOME") # where does it point to? Maybe some obscure path?

## [1] ""
# if yes, correct the variable:
Sys.setenv(WEKA_HOME="C:\\MY\\PATH\\WEKA_WPM")
library(RWeka)
# WPM("list-packages", "installed")

mlb.m5<-M5P(Weight~Height+Age, data=mlb_train)
mlb.m5

## M5 pruned model tree:
## (using smoothed linear models)
## LM1 (776/82.097%)
##
## LM num: 1
## Weight =
##  4.9957 * Height
##  + 1.0629 * Age
##  - 197.0898
##
## Number of Rules : 1
```

Instead of using segment averages to predict the player's weight, this model uses a linear regression (`LM1`) as the terminal node. In some datasets with more variables, `M5P` could yield different linear models at each terminal node.

```
summary(mlb.m5)

## === Summary ===
##
## Correlation coefficient                  0.571
## Mean absolute error                     13.3503
## Root mean squared error                 17.1622
## Relative absolute error                 80.3144 %
## Root relative squared error             82.0969 %
## Total Number of Instances              776

mlb.p.m5<-predict(mlb.m5, mlb_test)
summary(mlb.p.m5)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   166.1   193.5   201.7   202.0   209.7   247.8

cor(mlb.p.m5, mlb_test$Weight)

## [1] 0.5500171

MAE(mlb_test$Weight, mlb.p.m5)

## [1] 14.07716
```

`summary(mlb.m5)` reports some rough diagnostic statistics. We can see that the correlation and MAE for this model are better than the previous `rpart()` model.

10.7 Practice Problem: Heart Attack Data

Let's go back to the heart attack dataset (CaseStudy12_ AdultsHeartAttack_Data. csv) and practice this approach.

```
heart_attack<-read.csv("https://umich.instructure.com/files/1644953/download
?download_frd=1", stringsAsFactors = F)
str(heart_attack)
```

```
## 'data.frame':    150 obs. of  8 variables:
##  $ Patient  : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ DIAGNOSIS: int  41041 41041 41091 41081 41091 41091 41091 41091 41041
41041 ...
##  $ SEX      : chr  "F" "F" "F" "F" ...
##  $ DRG      : int  122 122 122 122 122 121 121 121 121 123 ...
##  $ DIED     : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ CHARGES  : chr  "4752" "3941" "3657" "1481" ...
##  $ LOS      : int  10 6 5 2 1 9 15 15 2 1 ...
##  $ AGE      : int  79 34 76 80 55 84 84 70 76 65 ...
```

First, we need to convert the `CHARGES` (independent variable) to numerical form. NA's are created, so let's keep only the complete cases as mentioned in the beginning of this Chapter. Also, let's create a gender variable as an indicator for female patients using `ifelse()` and delete the previous `SEX` column.

```
heart_attack$CHARGES<-as.numeric(heart_attack$CHARGES)

heart_attack<-heart_attack[complete.cases(heart_attack), ]
heart_attack$gender<-ifelse(heart_attack$SEX=="F", 1, 0)
heart_attack<-heart_attack[, -3]
```

Next, we can build a model tree using `M5P()` with all the features in the model. As usual, we need to separate the `heart_attack` data into training and test datasets (e.g., use the 75–25% random split).

Using the model to predict `CHARGES` in the test dataset, we can obtain the following correlation and MAE.

```
## [1] 0.5616003

## [1] 3193.502
```

We can see that the predicted values and observed values are strongly correlated. In terms of MAE, it may seem very large at first glance.

```
range(ha_test$CHARGES)

## [1]   701 17137

# 17137-701
# 3193.502/16436
```

However, the test data itself has a wide range, and the MAE is within 20% of the range. With only 148 observations, the model represents a fairly good prediction of the expected hospital stay charges. Try to reproduce these results and also test the same techniques to other data from the list of our Case-Studies.

10.8 Assignments: 10. Forecasting Numeric Data Using Regression Models

Use the Quality of Life data (Case06_QoL_Symptom_ChronicIllness) to fit several different Multiple Linear Regression models predicting clinically relevant outcomes, e.g., `Chronic Disease Score`.

- Summarize and visualize data using `summary`, `str`, `pairs.panels`, `ggplot`.
- Report correlation for numeric data and try to visualize it (e.g., heatmap, pairs plot, etc.)
- Examine potential dependences of the predictors and the dependent response variables.
- Fit a couple of Multiple Linear Regression models, report the results, and explain the summary, residuals, effect-size coefficients, and the coefficient of determination, R^2.
- Draw model diagnostic plots, at least QQ plot, residuals plot and leverage plot (half norm plot).
- Report results in terms of the model.
- Predict outcomes for new data.
- Try to improve the model performance using `step` function based on AIC and BIC.
- Fit a regression tree model and compare with OLS model.
- Try to use `RWeka::M5P` to improve the model.

References

Fahrmeir, L, Kneib, T, Lang, S, Marx, B. (2013) *Regression: Models, Methods and Applications*, Springer Science & Business Media, ISBN 3642343333, 9783642343339.

Hyndman, RJ, Athanasopoulos, G. (2014) *Forecasting: principles and practice*, OTexts, ISBN 0987507109, 9780987507105.

Zhao, Y. (2012) *R and Data Mining: Examples and Case Studies*, Academic Press, ISBN 012397271X, 9780123972712.

http://wiki.socr.umich.edu/index.php/EBook#Chapter_X:_Correlation_and_Regression.