Regression and Prediction

Trevor Barnes

9/2/2020

Perhaps the most common goal in statistics is to answer the question: Is the variable X (or more likely, $X_1, ..., X_p$) associated with a variable Y, and, if so, what is the relationship and can we use it to predict Y?

Nowhere is the nexus between statistics and data science stronger than in the realm of prediction—specifically the prediction of an outcome (target) variable based on the values of other "predictor" variables. Another important connection is in the area of anomaly detection, where regression diagnostics originally intended for data analysis and improving the regression model can be used to detect unusual records. The antecedents of correlation and linear regression date back over a century.

Simple Linear Regression

Simple linear regression models the relationship between the magnitude of one variable and that of a second—for example, as X increases, Y also increases. Or as X increases, Y decreases. Correlation is another way to measure how two variables are related. The difference is that while correlation measures the strength of an association between two variables, regression quantifies the nature of the relationship.

Table 1: **KEY TERMS FOR SIMPLE LINEAR REGRESSION**

Term	Definition	Synonym
Response	The variable we are trying to predict.	dependent variable, Y-variable, target, outcome
Independent Variable	The variable used to predict the response.	independent variable, X-variable, feature, attribute
Record	The vector of predictor and outcome values for a specific individual or case.	row, case, instance, example
Intercept	The intercept of the regression line—that is, the predicted value when $X = 0$.	b_0, β_0
Regression Coefficient	The slope of the regression line.	slope, b_1, β_1, parameter estimates, weights
Fitted Values	The estimates $\hat{Y}_i$ obtained from the regression line.	predicted values
Residuals	The difference between the observed values and the fitted values.	errors
Least Squares	The method of fitting a regression by minimizing the sum of squared residuals.	ordinary least squares

The Regression Equation

Simple linear regression estimates exactly how much Y will change when X changes by a certain amount. With the correlation coefficient, the variables X and Y are interchangable. With regression, we are trying to predict the Y variable from X using a linear relationship (i.e., a line):

$Y = b_0 + b_1 X$

We read this as "Y equals b_1 times X, plus a constant b_0." The symbol b_0 is known as the ***intercept*** (or constant), and the symbol b_1 as the ***slope*** for X. Both appear in `R` output as ***coefficients***, though in general use the term ***coefficient*** is often reserved for b_1. The Y variable is known as the ***response*** or ***dependent*** variable since it depends on X. The X variable is known as the predictor or independent variable. The machine learning community tends to use other terms, calling Y the ***target*** and X a ***feature*** vector.

The `lm` function in `R` can be used to fit a linear regression.

```
model <- lm (PEFR ~ Exposure, data=lung)
```

`lm` standards for ***linear model*** and the ~ symbol denotes that `PEFR` is predicted by `Exposure`.

Printing the `model` object produces the following output:

```
model
```

```
##
## Call:
## lm(formula = PEFR ~ Exposure, data = lung)
##
## Coefficients:
## (Intercept)     Exposure
##     424.583       -4.185
```

The intercept, or b_0, is 424.583 and can be interpreted as the predicted PEFR for a worker with zero years exposure. The regression coefficient, or b_1, can be interpreted as follows: for each additional year that a worker is exposed to cotton dust, the worker's `PEFR` measurement is reduced by –4.185.

Fitted Values and Residuals

Important concepts in regression analysis are the ***fitted*** values and ***residuals***. In general, the data doesn't fall exactly on a line, so the regression equation should include an explicit error term e_i:

$Y_i = b_0 + b_1 X_i + e_i$

The ***fitted*** values, also referred to as the ***predicted*** values, are typically denoted by $\hat{Y}_i$ (Y-hat). These are given by:

$\hat{Y}_i = \hat{b}_0 + \hat{b}_1 X_i$

The notation $\hat{b}_0$ and $\hat{b}_1$ indicates that the coefficients are estimated versus known.

We compute the residuals $\hat{e}_i$ by subtracting the ***predicted*** values from the original data:

$\hat{e}_i = Y_i - \hat{Y}_i$

In `R`, we can obtain the fitted values and residuals using the functions `predict` and `residuals`:

```
fitted <- predict(model)
resid <- residuals(model)
```

Least Squares

In practice, the regression line is the estimate that minimizes the sum of squared residual values, also called the residual sum of squares or RSS:

$$RSS = \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

which in turn equals:

$$= \sum_{i=1}^{n}(Y_i - \hat{b}_0 - \hat{b}_1 X_i)^2$$

The estimates $\hat{b}_0$ and $\hat{b}_1$ are the values that minimize RSS.

The method of minimizing the sum of the squared residuals is termed ***least squares regression***, or ***ordinary least squares*** (OLS) regression. It is often attributed to Carl Friedrich Gauss, the German mathmetician, but was first published by the French mathmetician Adrien-Marie Legendre in 1805. Least squares regression leads to a simple formula to compute the coefficients:

$$\hat{b}_1 = \frac{\sum_{i=1}^{n}(Y_i - \bar{Y})(X_i - \bar{X})}{\sum_{i=1}^{n}(X_i - \bar{X})^2} \quad \hat{b}_0 = \bar{Y} - \hat{b}_1 \bar{X}$$

Historically, computational convenience is one reason for the widespread use of least squares in regression. With the advent of big data, computational speed is still an important factor. Least squares, like the mean, are sensitive to outliers, although this tends to be a signicant problem only in small or moderate-sized problems.

Prediction versus Explanation (Profiling)

Historically, a primary use of regression was to illuminate a supposed linear relationship between predictor variables and an outcome variable. The goal has been to understand a relationship and explain it using the data that the regression was fit to. In this case, the primary focus is on the estimated slope of the regression equation, $\hat{b}$. Economists want to know the relationship between consumer spending and GDP growth. Public health officials might want to understand whether a public information campaign is effective in promoting safe sex practices. In such cases, the focus is not on predicting individual cases, but rather on understanding the overall relationship.

With the advent of big data, regression is widely used to form a model to predict individual outcomes for new data, rather than explain data in hand (i.e., a predictive model). In this instance, the main items of interest are the fitted values $\hat{Y}$. In marketing, regression can be used to predict the change in revenue in response to the size of an ad campaign. Universities use regression to predict students' GPA based on their SAT scores.

A regression model that fits the data well is set up such that changes in X lead to changes in Y. However, by itself, the regression equation does not prove the direction of causation. Conclusions about causation must come from a broader context of understanding about the relationship. For example, a regression equation might show a definite relationship between number of clicks on a web ad and number of conversions. It is our knowledge of the marketing process, not the regression equation, that leads us to the conclusion that clicks on the ad lead to sales, and not vice versa.

Key Ideas
The regression equation models the relationship between a response variable Y and a predictor variable X as a line.
A regression model yields fitted values and residuals—predictions of the response and the errors of the predictions.
Regression models are typically fit by the method of least squares.
Regression is used both for prediction and explanation.

Multiple Linear Regression

When there are multiple predictors, the equation is simply extended to accommodate them:

$Y = b_0 + b_1X_1 + b_2X_2 + ... + b_pX_p + e$

Instead of a line, we now have a linear model—the relationship between each coefficient and its variable (feature) is linear.

Table 3: **KEY TERMS FOR MULTIPLE LINEAR REGRESSION**

Term	Definition	Synonym
Root Mean Squared Error	The square root of the average squared error of the regression (this is the most widely used metric to compare regression models).	RMSE
Residual Standard Error	The same as the root mean squared error, but adjusted for degrees of freedom.	RSE
R-Squared	The proportion of variance explained by the model, from 0 to 1.	coefficient of determination, R^2
t-statistic	The coefficient for a predictor, divided by the standard error of the coefficient, giving a metric to compare the importance of variables in the model.	
Weighted Regression	Regression with the records having different weights.	

All of the other concepts in simple linear regression, such as fitting by least squares and the definition of fitted values and residuals, extend to the multiple linear regression setting. For example, the fitted values are given by:

$\hat{Y}_i = \hat{b}_0 + \hat{b}_1X_{1,i} + \hat{b}_2X_{2,i} + ... + \hat{b}_pX_{p,i}$

Example: King County Housing Data

An example of using regression is in estimating the value of houses. County assessors must estimate the value of a house for the purposes of assessing taxes. Real estate consumers and professionals consult popular

websites such as Zillow to ascertain a fair price. Here are a few rows of housing data from King County (Seattle), Washington, from the house `data.frame`:

```
head(house[, c("AdjSalePrice", "SqFtTotLiving", "SqFtLot", "Bathrooms", "Bedrooms", "BldgGrade")])
```

```
##   AdjSalePrice SqFtTotLiving SqFtLot Bathrooms Bedrooms BldgGrade
## 1       300805          2400    9373      3.00        6         7
## 2      1076162          3764   20156      3.75        4        10
## 3       761805          2060   26036      1.75        4         8
## 4       442065          3200    8618      3.75        5         7
## 5       297065          1720    8620      1.75        4         7
## 6       411781           930    1012      1.50        2         8
```

The goal is to predict the sales price from the other variables. The lm handles the multiple regression case simply by including more terms on the righthand side of the equation; the argument na.action=na.omit causes the model to drop records that have missing values:

```
house_lm <- lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms + Bedrooms + BldgGrade,
               data = house, na.action = na.omit)

house_lm
```

```
##
## Call:
## lm(formula = AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
##     Bedrooms + BldgGrade, data = house, na.action = na.omit)
##
## Coefficients:
##   (Intercept)  SqFtTotLiving        SqFtLot      Bathrooms       Bedrooms
##    -5.219e+05      2.288e+02     -6.051e-02     -1.944e+04     -4.778e+04
##     BldgGrade
##     1.061e+05
```

The interpretation of the coefficients is as with simple linear regression: the predicted value $\hat{Y}$ changes by the coefficient $b_j$for each unit change in X_j assuming all the other variables, X_k for $k \neq j$, remain the same. For example, adding an extra finished square foot to a house increases the estimated value by roughly $229; adding 1,000 finished square feet implies the value will increase by $228,800.

Assessing the Model

The most important performance metric from a data science perspective is ***root mean squared error***, or ***RMSE***. RMSE is the square root of the average squared error in the predicted $\hat{y}_i$ values:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

This measures the overall accuracy of the model, and is a basis for comparing it to other models (including models fit using machine learning techniques). Similar to RMSE is the ***residual standard error***, or ***RSE***. In this case we have p predictors, and the RSE is given by:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n-p-1}}$$

The only difference is that the denominator is the degrees of freedom, as opposed to number of records. In practice, for linear regression, the difference between RMSE and RSE is very small, particularly for big data applications.

The summary function in R computes RSE as well as other metrics for a regression model:

```
summary(house_lm)
```

```
##
## Call:
## lm(formula = AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
##     Bedrooms + BldgGrade, data = house, na.action = na.omit)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1199508  -118879   -20982    87414  9472982
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -5.219e+05  1.565e+04 -33.349  < 2e-16 ***
## SqFtTotLiving  2.288e+02  3.898e+00  58.699  < 2e-16 ***
## SqFtLot       -6.051e-02  6.118e-02  -0.989    0.323
## Bathrooms     -1.944e+04  3.625e+03  -5.362 8.32e-08 ***
## Bedrooms      -4.778e+04  2.489e+03 -19.194  < 2e-16 ***
## BldgGrade      1.061e+05  2.396e+03  44.287  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 261200 on 22683 degrees of freedom
## Multiple R-squared:  0.5407, Adjusted R-squared:  0.5406
## F-statistic:  5340 on 5 and 22683 DF,  p-value: < 2.2e-16
```

Another useful metric that you will see in software output is the ***coefficient of determination***, also called the ***R-squared*** statistic or R^2. R-squared ranges from 0 to 1 and measures the proportion of variation in the data that is accounted for in the model. It is useful mainly in explanatory uses of regression where you want to assess how well the model fits the data. The formula for R^2 is:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

The denominator is proportional to the variance of Y. The output from R also reports an ***adjusted R-squared***, which adjusts for the degrees of freedom; seldom is this significantly different in multiple regression.

Along with the estimated coefficients, R reports the standard error of the coefficients (SE) and a ***t-statistic***:

$$t_b = \frac{\hat{b}}{SE(\hat{b})}$$

The t-statistic—and its mirror image, the p-value—measures the extent to which a coefficient is "statistically significant"—that is, outside the range of what a random chance arrangement of predictor and target variable might produce. The higher the t-statistic (and the lower the p-value), the more significant the predictor. Since parsimony is a valuable model feature, it is useful to have a tool like this to guide choice of variables to include as predictors.

Cross-Validation

Classic statistical regression metrics (R2, F-statistics, and p-values) are all "in- sample" metrics—they are applied to the same data that was used to fit the model. Intuitively, you can see that it would make a lot

of sense to set aside some of the original data, not use it to fit the model, and then apply the model to the set-aside (holdout) data to see how well it does. Normally, you would use a majority of the data to fit the model, and use a smaller portion to test the model.

This idea of "out-of-sample" validation is not new, but it did not really take hold until larger data sets became more prevalent; with a small data set, analysts typically want to use all the data and fit the best possible model.

Using a holdout sample, though, leaves you subject to some uncertainty that arises simply from variability in the small holdout sample. How different would the assessment be if you selected a different holdout sample?

Cross-validation extends the idea of a holdout sample to multiple sequential holdout samples. The algorithm for basic ***k-fold cross-validation*** is as follows:

1. Set aside $1/k$ of the data as a holdout sample.
2. Train the model on the remaining data.
3. Apply (score) the model to the $1/k$ holdout, and record needed model assessment metrics.
4. Restore the first $1/k$ of the data, and set aside the next $1/k$ (excluding any records that got picked the first time).
5. Repeat steps 2 and 3.
6. Repeat until each record has been used in the holdout portion.
7. Average or otherwise combine themodel assessment metrics.

The division of the data into the training sample and the holdout sample is also called a ***fold***.

Model Selection and Stepwise Regression

In some problems, many variables could be used as predictors in a regression. For example, to predict house value, additional variables such as the basement size or year built could be used. In R, these are easy to add to the regression equation:

```
house_full <- lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms + Bedrooms + BldgGrade + PropertyTy
                    SqFtFinBasement + YrBuilt + YrRenovated + NewConstruction,
                 data=house,
                 na.action=na.omit)
```

Adding more variables, however, does not necessarily mean we have a better model. Statisticians use the principle of Occam's razor to guide the choice of a model: all things being equal, a simpler model should be used in preference to a more complicated model.

Including additional variables always reduces RMSE and increases R^2. Hence, these are not appropriate to help guide the model choice. In the 1970s, Hirotugu Akaike, the eminent Japanese statistician, developed a metric called AIC (Akaike's Information Criteria) that penalizes adding terms to a model. In the case of regression, AIC has the form: $AIC = 2P + nlog(RSS/n)$ where p is the number of variables and n is the number of records. The goal is to find the model that minimizes AIC; models with k more extra variables are penalized by $2k$.

How do we find the model that minimizes AIC? One approach is to search through all possible models, called all ***subset regression***. This is computationally expensive and is not feasible for problems with large data and many variables. An attractive alternative is to use ***stepwise regression***, which successively adds and drops predictors to find a model that lowers AIC. The MASS package by Venebles and Ripley offers a stepwise regression function called `stepAIC`:

```
step <- stepAIC(house_full, direction = "both")
```

```
## Start:  AIC=563193.9
## AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms + Bedrooms +
##     BldgGrade + PropertyType + NbrLivingUnits + SqFtFinBasement +
##     YrBuilt + YrRenovated + NewConstruction
##
##                    Df  Sum of Sq        RSS    AIC
## - NbrLivingUnits    1 6.4936e+09 1.3662e+15 563192
## - NewConstruction   1 1.0188e+10 1.3662e+15 563192
## - YrRenovated       1 2.4839e+10 1.3662e+15 563192
## - SqFtLot           1 1.0643e+11 1.3663e+15 563194
## <none>                           1.3662e+15 563194
## - SqFtFinBasement   1 1.3984e+11 1.3664e+15 563194
## - PropertyType      2 4.4129e+12 1.3706e+15 563263
## - Bathrooms         1 7.6340e+12 1.3739e+15 563318
## - Bedrooms          1 2.8246e+13 1.3945e+15 563656
## - YrBuilt           1 1.2905e+14 1.4953e+15 565240
## - SqFtTotLiving     1 1.3265e+14 1.4989e+15 565294
## - BldgGrade         1 1.9059e+14 1.5568e+15 566155
##
## Step:  AIC=563192.1
## AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms + Bedrooms +
##     BldgGrade + PropertyType + SqFtFinBasement + YrBuilt + YrRenovated +
##     NewConstruction
##
##                    Df  Sum of Sq        RSS    AIC
## - NewConstruction   1 1.0394e+10 1.3662e+15 563190
## - YrRenovated       1 2.5399e+10 1.3663e+15 563190
## - SqFtLot           1 1.0717e+11 1.3663e+15 563192
## <none>                           1.3662e+15 563192
## - SqFtFinBasement   1 1.3781e+11 1.3664e+15 563192
## + NbrLivingUnits    1 6.4936e+09 1.3662e+15 563194
## - PropertyType      2 4.4221e+12 1.3706e+15 563261
## - Bathrooms         1 7.7519e+12 1.3740e+15 563318
## - Bedrooms          1 2.8308e+13 1.3945e+15 563655
## - YrBuilt           1 1.3011e+14 1.4963e+15 565254
## - SqFtTotLiving     1 1.3288e+14 1.4991e+15 565296
## - BldgGrade         1 1.9186e+14 1.5581e+15 566172
##
## Step:  AIC=563190.2
## AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms + Bedrooms +
##     BldgGrade + PropertyType + SqFtFinBasement + YrBuilt + YrRenovated
##
##                    Df  Sum of Sq        RSS    AIC
## - YrRenovated       1 2.5655e+10 1.3663e+15 563189
## - SqFtLot           1 1.1468e+11 1.3664e+15 563190
## <none>                           1.3662e+15 563190
## - SqFtFinBasement   1 1.4477e+11 1.3664e+15 563191
## + NewConstruction   1 1.0394e+10 1.3662e+15 563192
## + NbrLivingUnits    1 6.7000e+09 1.3662e+15 563192
## - PropertyType      2 4.5236e+12 1.3708e+15 563261
## - Bathrooms         1 7.7505e+12 1.3740e+15 563317
```

```
## - Bedrooms         1 2.8304e+13 1.3945e+15 563653
## - SqFtTotLiving    1 1.3391e+14 1.5001e+15 565310
## - YrBuilt          1 1.3757e+14 1.5038e+15 565365
## - BldgGrade        1 1.9253e+14 1.5588e+15 566179
##
## Step:  AIC=563188.7
## AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms + Bedrooms +
##     BldgGrade + PropertyType + SqFtFinBasement + YrBuilt
##
##                   Df  Sum of Sq        RSS    AIC
## - SqFtLot          1 1.1399e+11 1.3664e+15 563189
## <none>                          1.3663e+15 563189
## - SqFtFinBasement  1 1.4938e+11 1.3664e+15 563189
## + YrRenovated      1 2.5655e+10 1.3662e+15 563190
## + NewConstruction  1 1.0651e+10 1.3663e+15 563190
## + NbrLivingUnits   1 7.2737e+09 1.3663e+15 563191
## - PropertyType     2 4.5012e+12 1.3708e+15 563259
## - Bathrooms        1 7.7815e+12 1.3740e+15 563316
## - Bedrooms         1 2.8286e+13 1.3945e+15 563652
## - SqFtTotLiving    1 1.3389e+14 1.5002e+15 565308
## - YrBuilt          1 1.5088e+14 1.5171e+15 565563
## - BldgGrade        1 1.9253e+14 1.5588e+15 566178
##
## Step:  AIC=563188.5
## AdjSalePrice ~ SqFtTotLiving + Bathrooms + Bedrooms + BldgGrade +
##     PropertyType + SqFtFinBasement + YrBuilt
##
##                   Df  Sum of Sq        RSS    AIC
## <none>                          1.3664e+15 563189
## + SqFtLot          1 1.1399e+11 1.3663e+15 563189
## - SqFtFinBasement  1 1.4058e+11 1.3665e+15 563189
## + YrRenovated      1 2.4965e+10 1.3664e+15 563190
## + NewConstruction  1 1.8206e+10 1.3664e+15 563190
## + NbrLivingUnits   1 8.1478e+09 1.3664e+15 563190
## - PropertyType     2 4.4353e+12 1.3708e+15 563258
## - Bathrooms        1 7.7135e+12 1.3741e+15 563314
## - Bedrooms         1 2.8588e+13 1.3950e+15 563656
## - SqFtTotLiving    1 1.3750e+14 1.5039e+15 565362
## - YrBuilt          1 1.5077e+14 1.5171e+15 565561
## - BldgGrade        1 1.9243e+14 1.5588e+15 566176
```

```
step
```

```
##
## Call:
## lm(formula = AdjSalePrice ~ SqFtTotLiving + Bathrooms + Bedrooms +
##     BldgGrade + PropertyType + SqFtFinBasement + YrBuilt, data = house,
##     na.action = na.omit)
##
## Coefficients:
##               (Intercept)              SqFtTotLiving
##                 6.178e+06                  1.993e+02
##                 Bathrooms                  Bedrooms
##                 4.240e+04                 -5.197e+04
```

```
##                 BldgGrade  PropertyTypeSingle Family
##                 1.372e+05                  2.285e+04
##     PropertyTypeTownhouse            SqFtFinBasement
##                 8.438e+04                  7.032e+00
##                   YrBuilt
##                -3.565e+03
```

The function chose a model in which several variables were dropped from house_full: `SqFtLot`, `NbrLivingUnits`, `YrRenovated`, and `NewConstruction`.

Simpler yet are ***forward selection*** and ***backward selection***. In forward selection, you start with no predictors and add them one-by-one, at each step adding the predictor that has the largest contribution to R^2, stopping when the contribution is no longer statistically significant. In backward selection, or ***backward elimination***, you start with the full model and take away predictors that are not statistically significant until you are left with a model in which all predictors are statistically significant.

Penalized regression is similar in spirit to AIC. Instead of explicitly searching through a discrete set of models, the model-fitting equation incorporates a constraint that penalizes the model for too many variables (parameters). Rather than eliminating predictor variables entirely—as with stepwise, forward, and backward selection—penalized regression applies the penalty by reducing coefficients, in some cases to near zero. Common penalized regression methods are ***ridge regression*** and ***lasso regression***.

Stepwise regression and all subset regression are ***in-sample*** methods to assess and tune models. This means the model selection is possibly subject to overfitting and may not perform as well when applied to new data. One common approach to avoid this is to use cross-validation to validate the models. In linear regression, overfitting is typically not a major issue, due to the simple (linear) global structure imposed on the data. For more sophisticated types of models, particularly iterative procedures that respond to local data structure, cross- validation is a very important tool.

Weighted Regression

Weighted regression is used by statisticians for a variety of purposes; in particular, it is important for analysis of complex surveys. Data scientists may find weighted regression useful in two cases:

- Inverse-variance weighting when different observations have been measured with different precision.
- Analysis of data in an aggregated form such that the weight variable encodes how many original observations each row in the aggregated data represents.

For example, with the housing data, older sales are less reliable than more recent sales. Using the DocumentDate to determine the year of the sale, we can compute a Weight as the number of years since 2005 (the beginning of the data).

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
house$Year = year(house$DocumentDate)
house$Weight = house$Year - 2005
```

We can compute a weighted regression with the lm function using the weight argument.

```
house_wt <- lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms + Bedrooms + BldgGrade,
               data = house,
               weight = Weight)

round(cbind(house_lm = house_lm$coefficients,
            house_wt = house_wt$coefficients),
      digits = 3)
```

```
##                  house_lm    house_wt
## (Intercept)   -521924.722 -584265.244
## SqFtTotLiving     228.832     245.017
## SqFtLot            -0.061      -0.292
## Bathrooms      -19438.099  -26079.171
## Bedrooms       -47781.153  -53625.404
## BldgGrade      106117.210  115259.026
```

The coefficents in the weighted regression are slightly different from the original regression.

Key Ideas
Multiple linear regression models the relationship between a response variable Y and multiple predictor variables $X_1, ..., X_p$.
The most important metrics to evaluate a model are root mean squared error (RMSE) and R-squared (R^2).
The standard error of the coefficients can be used to measure the reliability of a variable's contribution to a model.
Stepwise regression is a way to automatically determine which variables should be included in the model.
Weighted regression is used to give certain records more or less weight in fitting the equation.

Prediction using Regression

The primary purpose of regression in data science is prediction. This is useful to keep in mind, since regression, being an old and established statistical method, comes with baggage that is more relevant to its traditional explanatory modeling role than to prediction.

Table 5: **KEY TERMS FOR PREDICTION USING REGRESSION**

Term	Definition
Prediction Interval	An uncertainty interval around an individual predicted value.
Extrapolation	Extension of a model beyond the range of the data used to fit it.

The Dangers of Extrapolation

Regression models should not be used to extrapolate beyond the range of the data. The model is valid only for predictor values for which the data has sufficient values (even in the case that sufficient data is available, there could be other problems. As an extreme case, suppose `model_lm` is used to predict the value of a 5,000-square- foot empty lot. In such a case, all the predictors related to the building would have a value of 0 and the regression equation would yield an absurd prediction of –521,900 + 5,000 × –.0605 = –$522,202. Why did this happen? The data contains only parcels with buildings—there are no records corresponding to vacant land. Consequently, the model has no information to tell it how to predict the sales price for vacant land.

Confidence and Prediction Intervals

Much of statistics involves understanding and measuring variability (uncertainty). The t-statistics and p-values reported in regression output deal with this in a formal way, which is sometimes useful for variable selection. More useful metrics are confidence intervals, which are uncertainty intervals placed around regression coefficients and predictions. An easy way to understand this is via the bootstrap. The most common regression confidence intervals encountered in software output are those for regression parameters (coefficients). Here is a bootstrap algorithm for generating confidence intervals for regression parameters (coefficients) for a data set with P predictors and n records (rows):

1. Consider each row (including out come variable) as a single "ticket" and place all the n tickets in a box.
2. Draw a ticket at random, record the values, and replace it in the box.
3. Repeat step 2 n times; you now have one bootstrap resample.
4. Fit a regression to the bootstrap sample, and record the estimated coefficients.
5. Repeat steps 2 through 4, say,1,000 times.
6. You now have 1,000 bootstrap values for each coefficient; find the appropriate percentiles for each one (e.g., 5th and 95th for a 90% confidence interval).

You can use the Boot function in `R` to generate actual bootstrap confidence intervals for the coefficients, or you can simply use the formula-based intervals that are a routine `R` output. The conceptual meaning and interpretation are the same, and not of central importance to data scientists, because they concern the regression coefficients. Of greater interest to data scientists are intervals around predicted y values ($\hat{Y}_i$). The uncertainty around $\hat{Y}_i$ comes from two sources:

- Uncertainty about what the relevant predictor variables and their coefficients are (see the preceding bootstrap algorithm)
- Additional error inherent in individual data points

The individual data point error can be thought of as follows: even if we knew for certain what the regression equation was (e.g., if we had a huge number of records to fit it), the ***actual outcome*** values for a given set of predictor values will vary.

Key Ideas
Extrapolation beyond the range of the data can lead to error.
Confidence intervals quantify uncertainty around regression coefficients.
Prediction intervals quantify uncertainty in individual predictions.
Most software, R included, will produce prediction and confidence intervals in default or specified output, using formulas.
The bootstrap can also be used; the interpretation and idea are the same.

Factor Variables in Regression

Factor variables, also termed ***categorical*** variables, take on a limited number of discrete values. For example, a loan purpose can be "debt consolidation," "wedding," "car," and so on. The binary (yes/no) variable, also called an ***indicator*** variable, is a special case of a factor variable. Regression requires numerical inputs, so factor variables need to be recoded to use in the model. The most common approach is to convert a variable into a set of binary ***dummy*** variables.

Term	Definition	Synonym
Dummy Variables	Binary 0–1 variables derived by recoding factor data for use in regression and other models.	
Reference Coding	The most common type of coding used by statisticians, in which one level of a factor is used as a reference and other factors are compared to that level.	treatment coding
One Hot Encoder	A common type of coding used in the machine learning community in which all factors levels are retained. While useful for certain machine learning algorithms, this approach is not appropriate for multiple linear regression.	
Deviation Coding	A type of coding that compares each level against the overall mean as opposed to the reference level.	sum contrasts

Dummy Variable Representation

In the King County housing data, there is a factor variable for the property type; a small subset of six records is shown below.

```
head(house[, 'PropertyType'])
```

```
## [1] "Multiplex"     "Single Family" "Single Family" "Single Family"
## [5] "Single Family" "Townhouse"
```

There are three possible values: `Multiplex`, `Single Family`, and `Townhouse`. To use this factor variable, we need to convert it to a set of binary variables. We do this by creating a binary variable for each possible value of the factor variable. To do this in `R`, we use the `model.matrix` function:

```
prop_type_dummies <- model.matrix(~PropertyType -1, data=house)
head(prop_type_dummies)
```

```
##   PropertyTypeMultiplex PropertyTypeSingle Family PropertyTypeTownhouse
## 1                     1                         0                     0
```

```
## 2                    0                    1                    0
## 3                    0                    1                    0
## 4                    0                    1                    0
## 5                    0                    1                    0
## 6                    0                    0                    1
```

The function `model.matrix` converts a data frame into a matrix suitable to a linear model. The factor variable `PropertyType`, which has three distinct levels, is represented as a matrix with three columns. In the machine learning community, this representation is referred to as one hot encoding (see "One Hot Encoder"). In certain machine learning algorithms, such as nearest neighbors and tree models, one hot encoding is the standard way to represent factor variables (for example, see "Tree Models").

In the regression setting, a factor variable with P distinct levels is usually represented by a matrix with only P˘1 columns. This is because a regression model typically includes an intercept term. With an intercept, once you have defined the values for P˘1 binaries, the value for the Pth is known and could be considered redundant. Adding the Pth column will cause a multicollinearity error (see "Multicollinearity").

The default representation in R is to use the first factor level as a reference and interpret the remaining levels relative to that factor.

```
lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms + Bedrooms + BldgGrade + PropertyType,
   data=house)
```

```
##
## Call:
## lm(formula = AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
##     Bedrooms + BldgGrade + PropertyType, data = house)
##
## Coefficients:
##               (Intercept)              SqFtTotLiving
##                -4.469e+05                 2.234e+02
##                   SqFtLot                  Bathrooms
##                -7.041e-02                -1.597e+04
##                  Bedrooms                  BldgGrade
##                -5.090e+04                 1.094e+05
## PropertyTypeSingle Family      PropertyTypeTownhouse
##                -8.469e+04                -1.151e+05
```

The output from the R regression shows two coefficients corresponding to `PropertyType`: `PropertyTypeSingle Family` and `PropertyTypeTownhouse`. There is no coefficient of `Multiplex` since it is implicitly defined when `PropertyTypeSingle Family` == 0 and `PropertyTypeTownhouse` == 0. The coefficients are interpreted as relative to `Multiplex`, so a home that is `Single Family` is worth almost $85,000 less, and a home that is `Townhouse` is worth over $150,000 less.

Factor Variables with Many Levels

Some factor variables can produce a huge number of binary dummies—zip codes are a factor variable and there are 43,000 zip codes in the US. In such cases, it is useful to explore the data, and the relationships between predictor variables and the outcome, to determine whether useful information is contained in the categories. If so, you must further decide whether it is useful to retain all factors, or whether the levels should be consolidated.

In King County, there are 82 zip codes with a house sale:

```
table(house$ZipCode)
```

```
##
##  9800 89118 98001 98002 98003 98004 98005 98006 98007 98008 98010 98011 98014
##     1     1   358   180   241   293   133   460   112   291    56   163    85
## 98019 98022 98023 98024 98027 98028 98029 98030 98031 98032 98033 98034 98038
##   242   188   455    31   366   252   475   263   308   121   517   575   788
## 98039 98040 98042 98043 98045 98047 98050 98051 98052 98053 98055 98056 98057
##    47   244   641     1   222    48     7    32   614   499   332   402     4
## 98058 98059 98065 98068 98070 98072 98074 98075 98077 98092 98102 98103 98105
##   420   513   430     1    89   245   502   388   204   289   106   671   313
## 98106 98107 98108 98109 98112 98113 98115 98116 98117 98118 98119 98122 98125
##   361   296   155   149   357     1   620   364   619   492   260   380   409
## 98126 98133 98136 98144 98146 98148 98155 98166 98168 98177 98178 98188 98198
##   473   465   310   332   287    40   358   193   332   216   266   101   225
## 98199 98224 98288 98354
##   393     3     4     9
```

`ZipCode` is an important variable, since it is a proxy for the effect of location on the value of a house. Including all levels requires 81 coefficients corresponding to 81 degrees of freedom. The original model `house_lm` has only 5 degress of freedom; see "Assessing the Model". Moreover, several zip codes have only one sale. In some problems, you can consolidate a zip code using the first two or three digits, corresponding to a submetropolitan geographic region. For King County, almost all of the sales occur in 980xx or 981xx, so this doesn't help.

An alternative approach is to group the zip codes according to another variable, such as sale price. Even better is to form zip code groups using the residuals from an initial model. The following `dplyr` code consolidates the 82 zip codes into five groups based on the median of the residual from the `house_lm` regression:

```
zip_groups <- house %>%
  mutate(resid = residuals(house_lm)) %>%
  group_by(ZipCode) %>%
  summarize(med_resid = median(resid),
            cnt = n()) %>%
  arrange(med_resid) %>%
  mutate(cum_cnt = cumsum(cnt),
         ZipGroup = ntile(cum_cnt, 5))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
house <- house %>%
  left_join(select(zip_groups, ZipCode, ZipGroup), by='ZipCode')
```

The median residual is computed for each zip and the `ntile` function is used to split the zip codes, sorted by the median, into five groups. See "Confounding Variables" for an example of how this is used as a term in a regression improving upon the original fit.

The concept of using the residuals to help guide the regression fitting is a fundamental step in the modeling process; see "Testing the Assumptions: Regression Diagnostics".

Ordered Factor Variables

Some factor variables reflect levels of a factor; these are termed ordered factor variables or ***ordered categorical variables***. For example, the loan grade could be A, B, C, and so on—each grade carries more risk than the prior grade. Ordered factor variables can typically be converted to numerical values and used as is. For example, the variable `BldgGrade` is an ordered factor variable. While the grades have specific meaning, the numeric value is ordered from low to high, corresponding to higher-grade homes. With the regression model `house_lm`, fit in "Multiple Linear Regression", `BldgGrade` was treated as a numeric variable.

Treating ordered factors as a numeric variable preserves the information contained in the ordering that would be lost if it were converted to a factor.

Key Ideas
Factor variables need to be converted into numeric variables for use in a regression.
The most common method to encode a factor variable with P distinct values is to represent them using P-1 dummy variables.
A factor variable with many levels, even in very big data sets, may need to be consolidated into a variable with fewer levels.
Some factors have levels that are ordered and can be represented as a single numeric variable.

Interpreting the Regression Equation

In data science, the most important use of regression is to predict some dependent (outcome) variable. In some cases, however, gaining insight from the equation itself to understand the nature of the relationship between the predictors and the outcome can be of value. This section provides guidance on examining the regression equation and interpreting it.

Term	Definition	Synonym
Correlated variables	When the predictor variables are highly correlated, it is difficult to interpret the individual coefficients.	
Multicollinearity	When the predictor variables have perfect, or near-perfect, correlation, the regression can be unstable or impossible to compute.	collinearity
Confounding variables	An important predictor that, when omitted, leads to spurious relationships in a regression equation.	
Main Effects	The relationship between a predictor and the outcome variable, independent from other variables.	
Interactions	An interdependent relationship between two or more predictors and the response.	

Correlated Predictors

In multiple regression, the predictor variables are often correlated with each other. As an example, examine the regression coefficients for the model `step_lm`:

```
step_lm <- step
step_lm$coefficients
```

```
##               (Intercept)             SqFtTotLiving                 Bathrooms
##              6.177658e+06              1.992747e+02              4.240306e+04
##                  Bedrooms                 BldgGrade PropertyTypeSingle Family
##             -5.197477e+04              1.371811e+05              2.285488e+04
##     PropertyTypeTownhouse           SqFtFinBasement                   YrBuilt
##              8.437893e+04              7.032179e+00             -3.564935e+03
```

The coefficient for `Bedrooms` is negative! This implies that adding a bedroom to a house will reduce its value. How can this be? This is because the predictor variables are correlated: larger houses tend to have more bedrooms, and it is the size that drives house value, not the number of bedrooms. Consider two homes of the exact same size: it is reasonable to expect that a home with more, but smaller, bedrooms would be considered less desirable.

Having correlated predictors can make it difficult to interpret the sign and value of regression coefficients (and can inflate the standard error of the estimates). The variables for bedrooms, house size, and number of bathrooms are all correlated. This is illustrated by the following example, which fits another regression removing the variables `SqFtTotLiving`, `SqFtFinBasement`, and `Bathrooms` from the equation:

```
update(step_lm, . ~ . -SqFtTotLiving - SqFtFinBasement - Bathrooms)
```

```
##
## Call:
## lm(formula = AdjSalePrice ~ Bedrooms + BldgGrade + PropertyType +
##     YrBuilt, data = house, na.action = na.omit)
##
## Coefficients:
##               (Intercept)                   Bedrooms
##                   4913025                      27136
##                 BldgGrade  PropertyTypeSingle Family
##                    249019                     -19932
##     PropertyTypeTownhouse                    YrBuilt
##                    -47424                      -3211
```

The update function can be used to add or remove variables from a model. Now the coefficient for bedrooms is positive—in line with what we would expect (though it is really acting as a proxy for house size, now that those variables have been removed).

Correlated variables are only one issue with interpreting regression coefficients. In `house_lm`, there is no variable to account for the location of the home, and the model is mixing together very different types of regions. Location may be a ***confounding*** variable.

Multicollinearity

An extreme case of correlated variables produces multicollinearity—a condition in which there is redundance among the predictor variables. Perfect multicollinearity occurs when one predictor variable can be expressed as a linear combination of others. Multicollinearity occurs when:

- A variable is included multiple times by error.
- P dummiesm instead of $P - 1$ dummies, are created from a factor variable.
- Two variables are nearly perfectly correlated with one another.

Multicollinearity in regression must be addressed—variables should be removed until the multicollinearity is gone. A regression does not have a well-defined solution in the presence of perfect multicollinearity. Many software packages, including `R`, automatically handle certain types of multicolliearity. For example, if `SqFtTotLiving` is included twice in the regression of the house data, the results are the same as for the `house_lm` model. In the case of nonperfect multicollinearity, the software may obtain a solution but the results may be unstable.

Confounding Variables

With correlated variables, the problem is one of commission: including different variables that have a similar predictive relationship with the response. With ***confounding variables***, the problem is one of omission: an important variable is not included in the regression equation. Naive interpretation of the equation coefficients can lead to invalid conclusions.

Take, for example, the King County regression equation `house_lm` from "Example: King County Housing Data". The regression coefficients of `SqFtLot`, `Bathrooms`, and `Bedrooms` are all negative. The original regression model does not contain a variable to represent location—a very important predictor of house price. To model location, include a variable `ZipGroup` that categorizes the zip code into one of five groups, from least expensive (1) to most expensive (5).

```
lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms + Bedrooms + BldgGrade + PropertyType + ZipGroup,
   data=house,
   na.action=na.omit)
```

```
##
## Call:
## lm(formula = AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
##     Bedrooms + BldgGrade + PropertyType + ZipGroup, data = house,
##     na.action = na.omit)
##
## Coefficients:
##                (Intercept)              SqFtTotLiving
##                  -7.997e+05                  2.105e+02
##                     SqFtLot                  Bathrooms
##                   4.244e-01                  7.406e+03
##                    Bedrooms                  BldgGrade
##                  -4.186e+04                  1.031e+05
## PropertyTypeSingle Family      PropertyTypeTownhouse
##                   1.453e+04                 -8.634e+04
##                    ZipGroup
##                   8.105e+04
```

`ZipGroup` is clearly an important variable: a home in the most expensive zip code group is estimated to have a higher sales price by almost $340,000. The coefficients of `SqFtLot` and `Bathrooms` are now positive and adding a bathroom increases the sale price by $7,500.

The coefficient for `Bedrooms` is still negative. While this is unintuitive, this is a well-known phenomenon in real estate. For homes of the same livable area and number of bathrooms, having more, and therefore smaller, bedrooms is associated with less valuable homes.

Interaction and Main Effects

Statisticians like to distinguish between ***main effects***, or independent variables, and the ***interactions*** between the main effects. Main effects are what are often referred to as the ***predictor variables*** in the regression equation. An implicit assumption when only main effects are used in a model is that the relationship between a predictor variable and the response is independent of the other predictor variables. This is often not the case.

For example, the model fit to the King County Housing Data in "Confounding Variables" includes several variables as main effects, including `ZipCode`. Location in real estate is everything, and it is natural to presume that the relationship between, say, house size and the sale price depends on location. A big house built in a low-rent district is not going to retain the same value as a big house built in an expensive area. You include interactions between variables in `R` using the `*` operator. For the King County data, the following fits an interaction between `SqFtTotLiving` and `ZipGroup`:

```
lm(AdjSalePrice ~ SqFtTotLiving*ZipGroup + SqFtLot + Bathrooms + Bedrooms + BldgGrade + PropertyType,
   data=house,
   na.action=na.omit)
```

```
##
## Call:
## lm(formula = AdjSalePrice ~ SqFtTotLiving * ZipGroup + SqFtLot +
##     Bathrooms + Bedrooms + BldgGrade + PropertyType, data = house,
##     na.action = na.omit)
##
## Coefficients:
##                (Intercept)              SqFtTotLiving
##                 -4.236e+05                  3.226e+01
##                   ZipGroup                    SqFtLot
##                 -3.923e+04                  6.030e-01
##                  Bathrooms                   Bedrooms
##                 -6.493e+03                 -4.466e+04
##                  BldgGrade  PropertyTypeSingle Family
##                  1.101e+05                  8.056e+03
##     PropertyTypeTownhouse     SqFtTotLiving:ZipGroup
##                 -6.670e+04                  5.583e+01
```

The resulting model has four new terms: `SqFtTotLiving:ZipGroup2`, `SqFtTotLiving:ZipGroup3`, and so on.

Location and house size appear to have a strong interaction. For a home in the lowest `ZipGroup`, the slope is the same as the slope for the main effect `SqFtTotLiving`, which is \$177 per square foot (this is because `R` uses reference coding for factor variables). For a home in the highest `ZipGroup`, the slope is the sum of the main effect plus `SqFtTotLiving:ZipGroup5`, or \$177 + \$230 = \$447 per square foot. In other words, adding a square foot in the most expensive zip code group boosts the predicted sale price by a factor of almost 2.7, compared to the boost in the least expensive zip code group.

Key Ideas |
Because of correlation between predictors, care must be taken in the interpretation of the coefficients in multiple linear regression. |
Multicollinearity can cause numerical instability in fitting the regression equation. |
A confounding variable is an important predictor that is omitted from a model and can lead to a regression equation with spurious relationships. |
An interaction term between two variables is needed if the relationship between the variables and the response is interdependent. |

Testing the Assumptions: Regression Diagnostics

In explanatory modeling (i.e., in a research context), various steps, in addition to the metrics mentioned previously, are taken to assess how well the model fits the data. Most are based on analysis of the residuals, which can test the assumptions underlying the model. These steps do not directly address predictive accuracy, but they can provide useful insight in a predictive setting.

Table 10: **KEY TERMS FOR REGRESSION DIAGNOSTICS**

Term	Definition	Synonym
Standardized Residuals	Residuals divided by the standard error of the residuals.	
Outliers	Records (or outcome values) that are distant from the rest of the data (or the predicted outcome).	
Influential Value	A value or record whose presence or absence makes a big difference in the regression equation.	
Leverage	The degree of influence that a single record has on a regression equation.	hat-value
Non-Normal Residuals	Non-normally distributed residuals can invalidate some technical requirements of regression, but are usually not a concern in data science.	
Heteroskedasticity	When some ranges of the outcome experience residuals with higher variance (may indicate a predictor missing from the equation).	
Partial Residual Plots	A diagnostic plot to illuminate the relationship between the outcome variable and a single predictor.	added variables plot

Generally speaking, an extreme value, also called an ***outlier***, is one that is distant from most of the other observations. Just as outliers need to be handled for estimates of location and variability, outliers can cause problems with regression models. In regression, an outlier is a record whose actual y value is distant from the predicted value. You can detect outliers by examining the ***standardized residual***, which is the residual divided by the standard error of the residuals.

There is no statistical theory that separates outliers from nonoutliers. Rather, there are (arbitrary) rules of thumb for how distant from the bulk of the data an observation needs to be in order to be called an outlier. For example, with the boxplot, outliers are those data points that are too far above or below the box boundaries (see "Percentiles and Boxplots"), where "too far" = "more than 1.5 times the inter-quartile range." In regression, the standardized residual is the metric that is typically used to determine whether a record is classified as an outlier. Standardized residuals can be interpreted as "the number of standard errors away from the regression line."

Let's fit a regression to the King County house sales data for all sales in zip code 98105:

```
house_98105 <- house[house$ZipCode == 98105,]
lm_98105 <- lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms + Bedrooms + BldgGrade,
               data=house_98105)
```

We extract the standardized residuals using the rstandard function and obtain the index of the smallest residual using the order function:

```
sresid <- rstandard(lm_98105)
idx <- order(sresid)
sresid[idx[1]]
```

```
##     20431
## -4.326732
```

The biggest overestimate from the model is more than four standard errors above the regression line, corresponding to an overestimate of $757,753. The original data record corresponding to this outlier is as follows:

```
house_98105[idx[1], c('AdjSalePrice', 'SqFtTotLiving', 'SqFtLot', 'Bathrooms', 'Bedrooms', 'BldgGrade')]
```

```
##       AdjSalePrice SqFtTotLiving SqFtLot Bathrooms Bedrooms BldgGrade
## 20431       119748          2900    7276         3        6         7
```

Influential Values

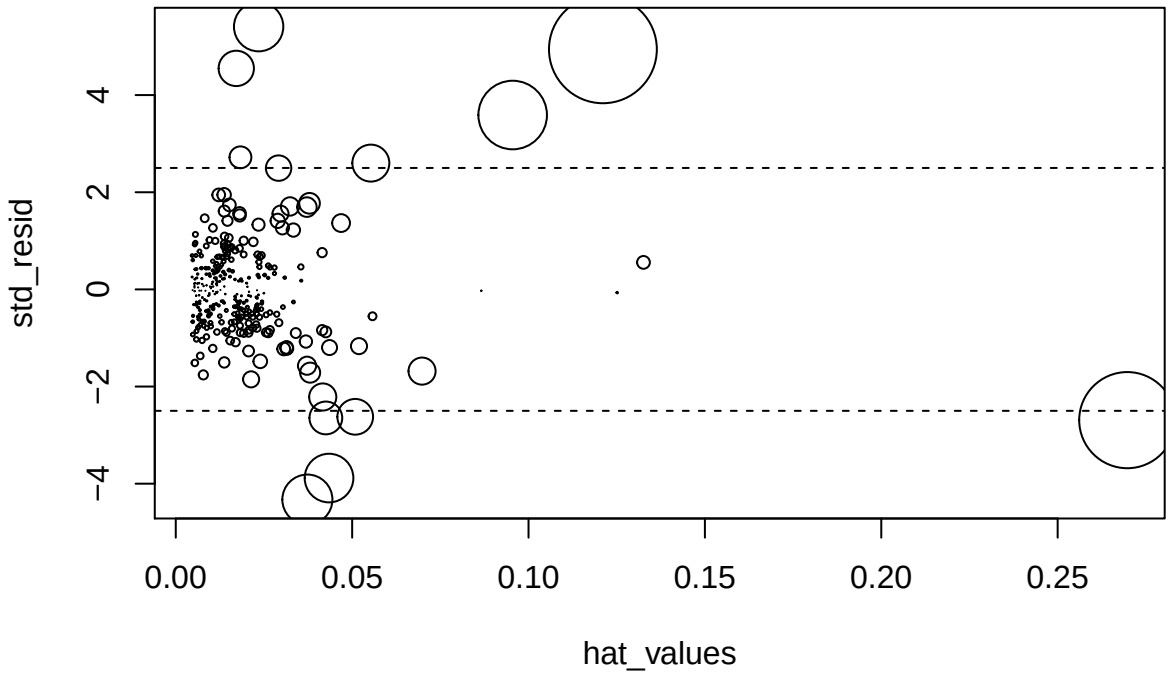
A value whose absence would significantly change the regression equation is termed an ***influential observation***. In regression, such a value need not be associated with a large residual. The solid line corresponds to the regression with all the data, while the dashed line corresponds to the regression with the point in the upper-right removed. Clearly, that data value has a huge influence on the regression even though it is not associated with a large outlier (from the full regression). This data value is considered to have high ***leverage*** on the regression.

In addition to standardized residuals, statisticians have developed several metrics to determine the influence of a single record on a regression. A common measure of leverage is the hat-value; values above $2(P+1)/n$ indicate a high-leverage data value.

Another metric is ***Cook's distance***, which defines influence as a combination of leverage and residual size. A rule of thumb is that an observation has high influence if Cook's distance exceeds $4/(n-P-1)$.

An ***influence plot*** or ***bubble plot*** combines standardized residuals, the hat-value, and Cook's distance in a single plot.

```
std_resid <- rstandard(lm_98105)
cooks_D <- cooks.distance(lm_98105)
hat_values <- hatvalues(lm_98105)
plot(hat_values, std_resid, cex=10*sqrt(cooks_D))
abline(h=c(-2.5, 2.5), lty=2)
```



There are apparently several data points that exhibit large influence in the regression. Cook's distance can be computed using the function cooks.distance, and you can use hatvalues to compute the diagnostics. The hat values are plotted on the x-axis, the residuals are plotted on the y-axis, and the size of the points is related to the value of Cook's distance.

For purposes of fitting a regression that reliably predicts future data, identifying influential observations is only useful in smaller data sets. For regressions involving many records, it is unlikely that any one observation will carry sufficient weight to cause extreme influence on the fitted equation (although the regression may still have big outliers). For purposes of anomaly detection, though, identifying influential observations can be very useful.

Heteroskedasticity, Non-Normality and Correlated Errors

Statisticians pay considerable attention to the distribution of the residuals. It turns out that ordinary least squares (see "Least Squares") are unbiased, and in some cases the "optimal" estimator, under a wide range of distributional assumptions. This means that in most problems, data scientists do not need to be too concerned with the distribution of the residuals.

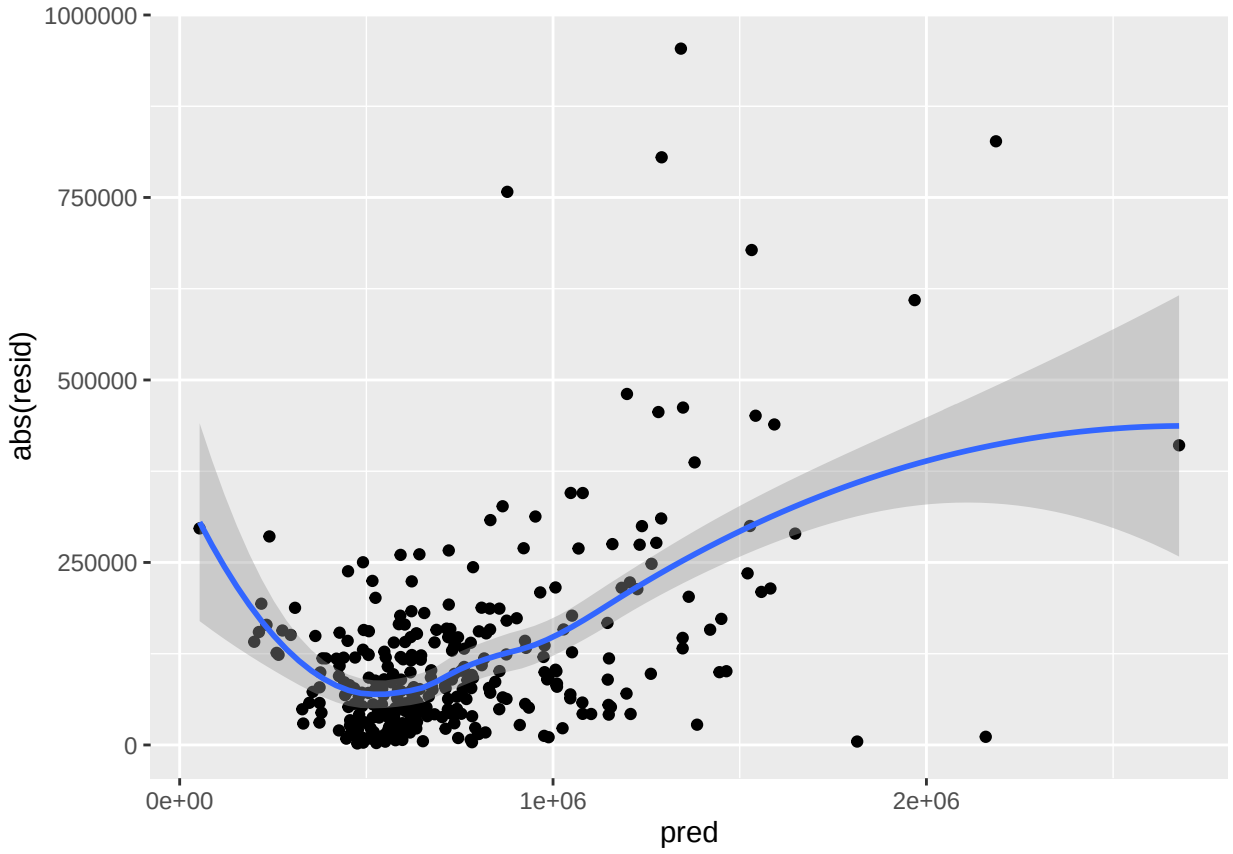
The distribution of the residuals is relevant mainly for the validity of formal statistical inference (hypothesis tests and p-values), which is of minimal importance to data scientists concerned mainly with predictive accuracy. For formal inference to be fully valid, the residuals are assumed to be normally distributed, have the same variance, and be independent. One area where this may be of concern to data scientists is the standard calculation of confidence intervals for predicted values, which are based upon the assumptions about the residuals.

Heteroskedasticity is the lack of constant residual variance across the range of the predicted values. In other words, errors are greater for some portions of the range than for others. The ggplot2 package has some convenient tools to analyze residuals.

The following code plots the absolute residuals versus the predicted values for the lm_98105 regression fit in "Outliers".

```
df <- data.frame(
  resid = residuals(lm_98105),
  pred = predict(lm_98105))
ggplot(df, aes(pred, abs(resid))) +
  geom_point() +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Evidently, the variance of the residuals tends to increase for higher-valued homes, but is also large for lower-valued homes. This plot indicates that lm_98105 has heteroskedastic errors.

Statisticians may also check the assumption that the errors are independent. This is particularly true for data that is collected over time. The ***Durbin-Watson*** statistic can be used to detect if there is significant autocorrelation in a regression involving time series data.

Partial Residual Plots and Nonlinearity

Partial residual plots are a way to visualize how well the estimated fit explains the relationship between a predictor and the outcome. Along with detection of outliers, this is probably the most important diagnostic for data scientists. The basic idea of a partial residual plot is to isolate the relationship between a predictor variable and the response, ***taking into account all of the other predictor variables***. A partial residual might be thought of as a "synthetic outcome" value, combining the prediction based on a single predictor with the actual residual from the full regression equation. A partial residual for predictor X_i is the ordinary residual plus the regression term associated with X_i:

$$PartialResidual = Residual + \hat{b}_i X_i$$

where $\hat{b}_i$ is the estimated regression coefficient. The predict function in R has an option to return the individual regression terms $\hat{b}X_i$:

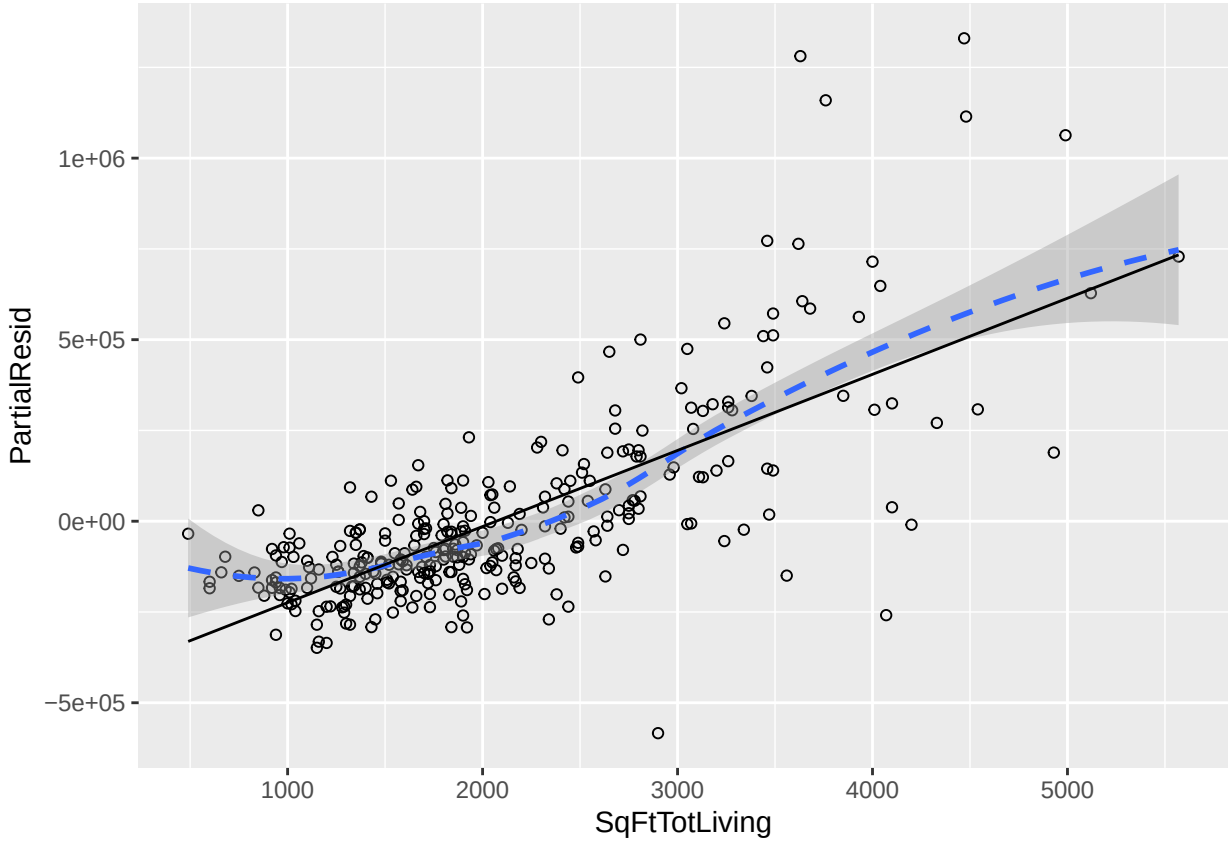
```
terms <- predict(lm_98105, type='terms')
partial_resid <- resid(lm_98105) + terms
```

The partial residual plot displays the X_i on the x-axis and the partial residuals on the y-axis. Using ggplot2 makes it easy to superpose a smooth of the partial residuals.

```
df <- data.frame(SqFtTotLiving = house_98105[, 'SqFtTotLiving'],
                 Terms = terms[, 'SqFtTotLiving'],
                 PartialResid = partial_resid[, 'SqFtTotLiving'])

ggplot(df, aes(SqFtTotLiving, PartialResid)) +
  geom_point(shape=1) + scale_shape(solid = FALSE) +
  geom_smooth(linetype=2) +
  geom_line(aes(SqFtTotLiving, Terms))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



The partial residual is an estimate of the contribution that `SqFtTotLiving` adds to the sales price. The relationship between `SqFtTotLiving` and the sales price is evidently nonlinear. The regression line underestimates the sales price for homes less than 1,000 square feet and overestimates the price for homes between 2,000 and 3,000 square feet. There are too few data points above 4,000 square feet to draw conclusions for those homes.

This nonlinearity makes sense in this case: adding 500 feet in a small home makes a much bigger difference than adding 500 feet in a large home. This suggests that, instead of a simple linear term for SqFtTotLiving, a nonlinear term should be considered.

Key Ideas
While outliers can cause problems for small data sets, the primary interest with outliers is to identify problems with the data, or locate anomalies.
Single records (including regression outliers) can have a big influence on a regression equation with small data, but this effect washes out in big data.
If the regression model is used for formal inference (p-values and the like), then certain assumptions about the distribution of the residuals should be checked. In general, however, the distribution of residuals is not critical in data science.
The partial residuals plot can be used to qualitatively assess the fit for each regression term, possibly leading to alternative model specification.

Polynomial and Spline Regression

Table 12: **KEY TERMS FOR NONLINEAR REGRESSION**

Term	Definition	Synonym
Polynomial Regression	Adds polynomial terms (squares, cubes, etc.) to a regression.	
Spline Regression	Fitting a smooth curve with a series of polynomial segments.	
Knots	Values that separate spline segments.	
Generalized additive models	Spline models with automated selection of knots.	GAM

Polynomial

Polynomial regression involves including polynomial terms to a regression equation. The use of polynomial regression dates back almost to the development of regression itself with a paper by Gergonne in 1815. For example, a quadratic regression between the response Y and the predictor X would take the form:

$$Y = b_0 + b_1 X + b_2 X^2 + e$$

Polynomial regression can be fit in R through the poly function. For example, the following fits a quadratic polynomial for `SqFtTotLiving` with the King County housing data:

```
lm(AdjSalePrice ~ poly(SqFtTotLiving, 2) + SqFtLot + BldgGrade + Bathrooms + Bedrooms,
   data=house_98105)
```

```
##
## Call:
## lm(formula = AdjSalePrice ~ poly(SqFtTotLiving, 2) + SqFtLot +
##     BldgGrade + Bathrooms + Bedrooms, data = house_98105)
##
```

```
## Coefficients:
##              (Intercept)  poly(SqFtTotLiving, 2)1  poly(SqFtTotLiving, 2)2
##                -402530.47               3271519.49                776934.02
##                   SqFtLot                BldgGrade                Bathrooms
##                     32.56                135717.06                 -1435.12
##                  Bedrooms
##                  -9191.94
```

Splines

Polynomial regression only captures a certain amount of curvature in a nonlinear relationship. Adding in higher-order terms, such as a cubic quartic polynomial, often leads to undesirable "wiggliness" in the regression equation. An alternative, and often superior, approach to modeling nonlinear relationships is to use ***splines***. Splines provide a way to smoothly interpolate between fixed points. Splines were originally used by draftsmen to draw a smooth curve, particularly in ship and aircraft building.

The technical definition of a spline is a series of piecewise continuous polynomials. They were first developed during World War II at the US Aberdeen Proving Grounds by I. J. Schoenberg, a Romanian mathematician. The polynomial pieces are smoothly connected at a series of fixed points in a predictor variable, referred to as ***knots***. Formulation of splines is much more complicated than polynomial regression; statistical software usually handles the details of fitting a spline. The `R` package `splines` includes the function bs to create a ***b-spline*** term in a regression model. For example, the following adds a b-spline term to the house regression model:

```
library(splines)
knots <- quantile(house_98105$SqFtTotLiving, p=c(.25, .5, .75))
lm_spline <- lm(AdjSalePrice ~ bs(SqFtTotLiving, knots=knots, degree=3) + SqFtLot + Bathrooms + Bedrooms
                data=house_98105)
```

Two parameters need to be specified: the degree of the polynomial and the location of the knots. In this case, the predictor `SqFtTotLiving` is included in the model using a cubic spline (`degree=3`). By default, bs places knots at the boundaries; in addition, knots were also placed at the lower quartile, the median quartile, and the upper quartile.

In contrast to a linear term, for which the coefficient has a direct meaning, the coefficients for a spline term are not interpretable. Instead, it is more useful to use the visual display to reveal the nature of the spline fit. In contrast to the polynomial model, the spline model more closely matches the smooth, demonstrating the greater flexibility of splines. In this case, the line more closely fits the data. Does this mean the spline regression is a better model? Not necessarily: it doesn't make economic sense that very small homes (less than 1,000 square feet) would have higher value than slightly larger homes. This is possibly an artifact of a confounding variable.

Generalized Additive Models

Suppose you suspect a nonlinear relationship between the response and a predictor variable, either by a priori knowledge or by examining the regression diagnostics. Polynomial terms may not flexible enough to capture the relationship, and spline terms require specifying the knots. ***Generalized additive models***, or ***GAM***, are a technique to automatically fit a spline regression. The `gam` package in `R` can be used to fit a GAM model to the housing data:

```
library(mgcv)
```

```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'

## The following object is masked from 'package:dplyr':
##
##     collapse

## This is mgcv 1.8-31. For overview type 'help("mgcv-package")'.
```

```
lm_gam <- gam(AdjSalePrice ~ s(SqFtTotLiving) + SqFtLot + Bathrooms + Bedrooms + BldgGrade,
              data=house_98105)
```

Key Ideas
Outliers in a regression are records with a large residual.
Multicollinearity can cause numerical instability in fitting the regression equation.
A confounding variable is an important predictor that is omitted from a model and can lead to a regression equation with spurious relationships.
An interaction term between two variables is needed if the effect of one variable depends on the level of the other.
Polynomial regression can fit nonlinear relationships between predictors and the outcome variable.
Splines are series of polynomial segments strung together, joining at knots.
Generalized additive models (GAM) automate the process of specifying the knots in splines.