Lecture 8

Forecasting Numeric Data – Regression Methods

How regression is different from what you have studied so far

As we know, variables can be characterized as either *quantitative* or *qualitative* (or *categorical*). Quantitative variables take on numerical values. Examples include a person's age, height, or income, the value of a house, and the price of a stock. In contrast, qualitative variables take on values in one of *K* different *classes*, or categories. Examples of qualitative variables include a person's gender, the brand of product purchased, or whether a person defaults on a debt. We tend to refer to problems with a quantitative variables as *regression* problems, while those involving a qualitative variables are often referred to as *classification* problems. However, the distinction is not always that crisp. Least squares linear regression, for example, is used with a quantitative variables, whereas logistic regression is typically used with a qualitative (two-class, or *binary*) variables.

So far we have been dealing with qualitative variables. Now, we are turning our attention to quantitative variables and regression.

Regression

Regression is a broad term for a set of methodologies used to predict a response variable (also called a dependent, criterion, or outcome variable) from one or more predictor variables (also called independent or explanatory variables). There are many types of regression:

- **Simple linear:** Predicting a quantitative response variable from a quantitative explanatory variable
- **Polynomial:** Predicting a quantitative response variable from a quantitative explanatory variable, where the relationship is modeled as an nth order polynomial
- **Multiple linear:** Predicting a quantitative response variable from two or more explanatory variables
- **Multivariate:** Predicting more than one response variable from one or more explanatory variables
- **Logistic:** Predicting a categorical response variable from one or more explanatory variables
- **Poisson:** Predicting a response variable representing counts from one or more explanatory variables
- **Time-series**: Modeling time-series data with correlated errors
- **Nonlinear:** Predicting a quantitative response variable from one or more explanatory variables, where the form of the model is nonlinear
- **Nonparametric:** Predicting a quantitative response variable from one or more explanatory variables, where the form of the model is derived from the data and not specified a priori

Correlation and Regression

Medical researchers are interested in questions such as, “Is caffeine related to heart damage?” or “Is there a relationship between a person’s age and his or her blood pressure?” These are only a few of the many questions that can be answered by using the techniques of correlation and regression analysis. **Correlation** is a statistical method used to determine whether a relationship between variables exists. **Regression** is a statistical method used to describe the nature of the relationship between variables, that is, positive or negative, linear or nonlinear.

In order to determine whether two or more variables are related and what the strength of the relationship is researchers use a numerical measure called **correlation coefficient**. There are several types of correlation coefficients. One of the most frequently used is the **Pearson correlation coefficient,** named after statistician Karl Pearson.

The Pearson correlation coefficient computed from the sample data measures the strength and direction of a **linear** relationship between two variables. The symbol for the sample correlation coefficient is *r.* The symbol for the population correlation coefficient is ρ (Greek letter rho).

Pearson correlation coefficient

The formula for the sample correlation coefficient is:

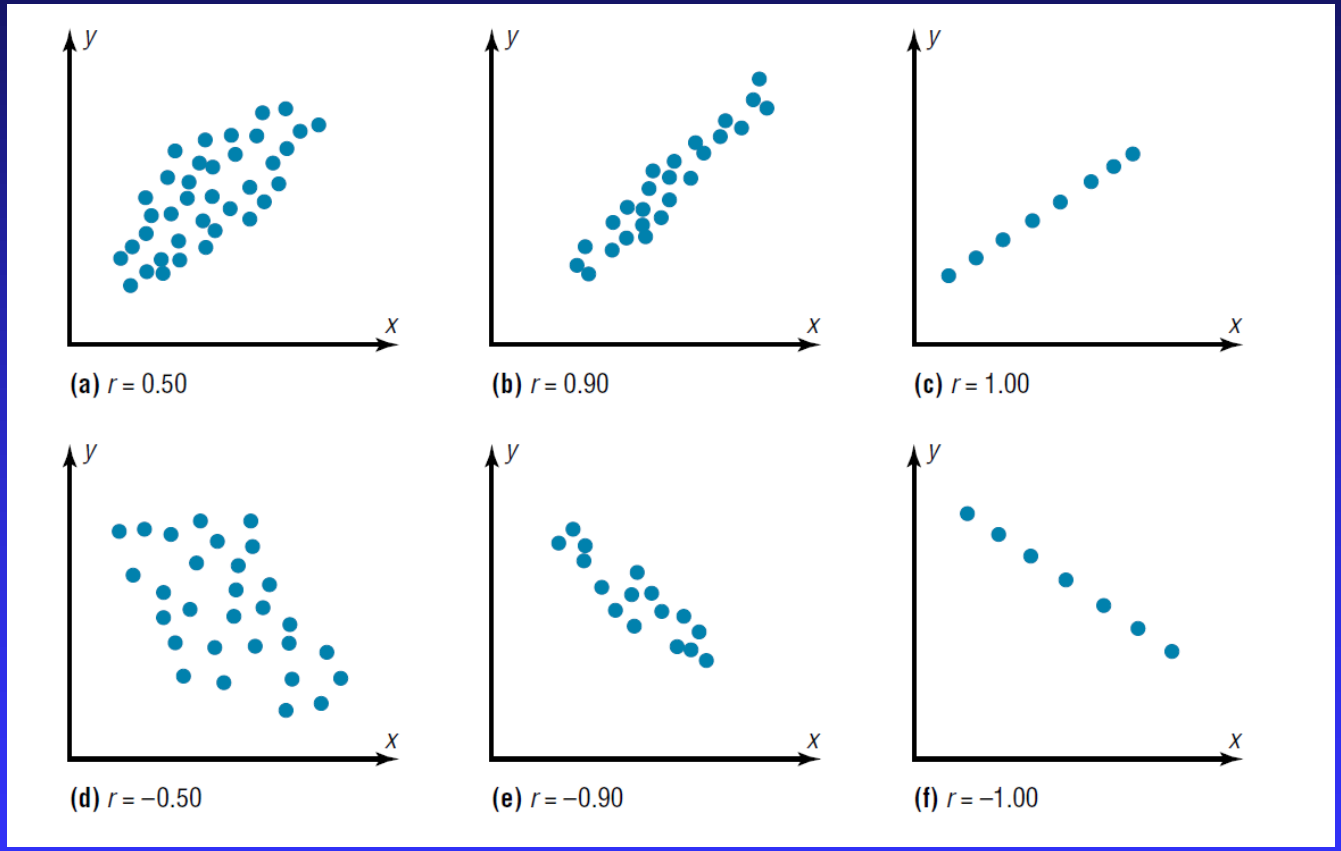
$$r = \frac{\sum(X - \overline{X})(Y - \overline{Y})}{\sqrt{\sum(X - \overline{X})^2}\sqrt{\sum(Y - \overline{Y})^2}}$$

Where $\overline{X}$ and $\overline{Y}$ are the sample means for X and Y, respectively.

The *range of the correlation coefficient* is from -1 to 1. If there is a *strong positive linear relationship* between the variables, the value of *r* will be close to 1. If there is a *strong negative linear relationship* between the variables, the value of *r* will be close to -1. When there is no linear relationship between the variables or only a weak relationship, the value of *r* will be close to 0.

Positive relationship means that y increases as x increases, whereas negative relationship means that y decreases as x increases. Notice that as the value of the correlation coefficient increases from 0 to 1 (parts *a*, *b*, and *c* of the next slide), data values become closer to an increasingly stronger relationship. As the value of the correlation coefficient decreases from 0 to -1 (parts *d*, *e*, and *f* on the next slide), the data values also become closer to a straight line. Again this suggests a stronger relationship.

Pearson correlation coefficient *(cont.)*



Pearson correlation coefficient *(cont.)*

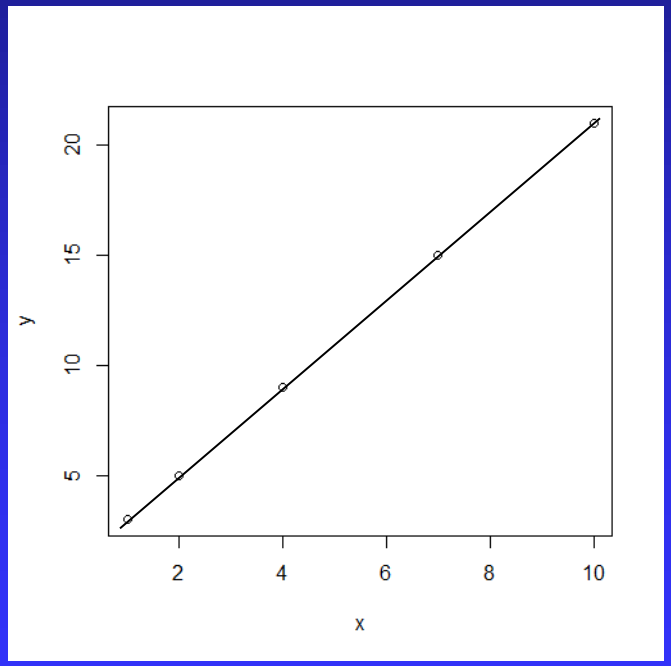
Let's find the correlation coefficient for the following data using the built-in function *cor():*

X	Y
5	2
9	4
5	1
21	10
15	7

```
> x <- c(2,4,1,10,7)
> y <- c(5,9,3,21,15)
> cor(x,y)
[1] 1
```

Pearson correlation coefficient *(cont.)*

It is not surprising that the correlation coefficient is 1 because the relationship between x and y is perfectly linear:



Correlation and Causation

Many students mistakenly belief that a correlation between X and Y implies causation, that is X causes Y or Y causes X. This is not always true:

- *The relationship between the variables may be caused by a third variable.* Suppose, for example, that there is a correlation between the number of deaths due to drowning and the number of cans of soft drink consumed daily during the summer. However, the soft drink is not necessarily responsible for the deaths, since both variables may be related to heat and humidity.

- *There may be a complexity of interrelationships among many variables.* For example, a correlation between students' high school grades and college grades could be based on many other variables involved, such as IQ, hours of study, influence of parents, motivation, age, and instructors.

- *The relationship may be coincidental.* For example, the correlation between the increase in the number of people who are exercising and the increase in the number of people who are committing crimes is coincidental.

Pearson correlation coefficient *(cont.)*

The formula for the Pearson correlation coefficient:

$$r = \frac{\sum(X - \overline{X})(Y - \overline{Y})}{\sqrt{\sum(X - \overline{X})^2}\sqrt{\sum(Y - \overline{Y})^2}}$$

can also be rewritten as:

$$\frac{\mathrm{Cov}(x, y)}{\sigma_x \sigma_y}$$

because the numerator is the covariance between x and y multiplied by (N-1) and the denominator is the product of the two standard deviations:

$$\mathrm{Cov}(x, y) = \frac{\Sigma(X - \overline{X})(Y - \overline{Y})}{n - 1}$$

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{i=n}(X_i - \bar{X})^2}$$

Pearson correlation coefficient *(cont.)*

The covariance between *X* and *Y* is intended to measure the degree to which *X* and *Y* tend to be large at the same time or the degree to which one tends to be large while the other is small. For example, suppose that Cov*(X, Y)* is positive. Then $X > \bar{X}$ and $Y > \bar{Y}$ must occur together and/or $X < \bar{X}$ and $Y < \bar{Y}$ must occur together to a larger extent than $X > \bar{X}$ occurs with $Y < \bar{Y}$ and $X < \bar{X}$ occurs with $Y > \bar{Y}$. Otherwise, the covariance would be negative. Similarly, if Cov*(X, Y)* is negative, then $X > \bar{X}$ and $Y < \bar{Y}$must occur together and/or $X < \bar{X}$ and $Y > \bar{Y}$. must occur together to larger extent than the other two inequalities. If Cov*(X, Y)* = 0, then the extent to which *X* and *Y* are on the same sides of their respective means exactly balances the extent to which they are on opposite sides of their means.

Linear regression

Linear regression assumes that there is a linear relationship between numeric variables. One of the variables, usually denoted by Y is called dependent variable. The other variables are usually denoted by X_1, X_2,… and they are called independent variables. The assumption is that the independent variables affect/determine the value of the dependent variable. Mathematically, Y is function of X_1, X_2,… If there is only one independent variable, the regression is called **simple**. Otherwise(if the number of the independent variables is greater than 1), the regression is called **multiple**.

In simple linear regression, there is only one independent variable x and the relationship between the dependent variable y and x is modeled as straight line:

$$y = \alpha + \beta x$$

β is the slope of the line and α is the intercept. Together, α and β are known as the model *coefficients* or *parameters*. Once we have used training data to produce estimates $\hat{\alpha}$ and $\hat{\beta}$ for the model coefficients, we can predict future values of y by using the following equation:

$$\hat{y} = \hat{\alpha} + \hat{\beta}\, x$$

Where $\hat{y}$ is the predicted value (it could differ from the actual value y)

Linear regression *(cont.)*

Let $\hat{y}_i = \hat{\alpha} + \hat{\beta}\, x_i$ be the prediction for *y* based on the *i*th value of *x*.

Then $e_i = yi - \hat{y}_i$ represents the *i*th *residual*—this is the difference between the *i*th observed value and the *i*th value that is predicted by our linear model. In a sense, it measures the error of prediction for the ith value. We define the *residual sum of squares* (RSS) as:

RSS = $$\sum (y_i - \hat{y}_i)^2 = \sum e_i^2$$

The Ordinary Least Squares (**OLS**) method determines $\hat{\alpha}$ and $\hat{\beta}$ so that to minimize RSS. It is mathematically proved that RSS is minimized for the following values of $\hat{\alpha}$ and $\hat{\beta}$:

$$\hat{\beta} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

$$\hat{\alpha} = \bar{y} - \hat{\beta} x$$

Linear regression *(cont.)*

If we take into account that the variance of x is:

$$\mathrm{Var}(x) = \frac{\sum (x_i - \bar{x})^2}{n}$$

and the covariance between x and y is:

$$\mathrm{Cov}(x, y) = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{n}$$

then the formula for $\hat{\beta}$ can be rewritten as:

$$\hat{\beta} = \frac{\mathrm{Cov}(x, y)}{\mathrm{Var}(x)}$$

Linear regression *(cont.)*

Your textbook provides a dataset from the launches of the space shuttle Challenger. It tries to find whether there is a linear relationship between the Challenger launch temperature and the number of distress events (failure of components called O-rings). First, we need to load the dataset (for your convenience it is also provided in Course Materials Section on Blackboard):

```
> launch <- read.csv("challenger.csv")
```

Let's explore the structure of the dataset:

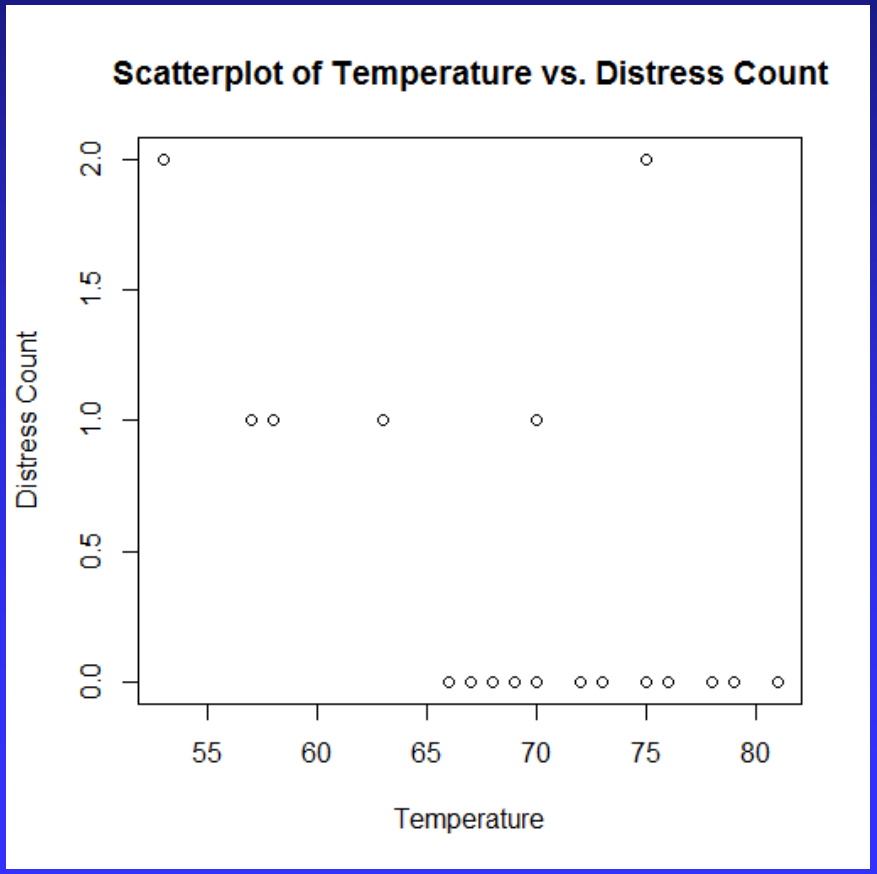
```
> str(launch)
'data.frame':   23 obs. of  4 variables:
 $ distress_ct          : int  0 1 0 0 0 0 0 0 1 1 ...
 $ temperature          : int  66 70 69 68 67 72 73 70 57 63
...
 $ field_check_pressure: int  50 50 50 50 50 50 100 100 200
200 ...
 $ flight_num           : int  1 2 3 4 5 6 7 8 9 10 ...
```

The dataset contains four variables, distress_ct, temperature, field_check_pressure, and flight_num. We will study the first two variables, the distress count and the launch temperature.

Linear regression *(cont.)*

Let's visualize the relationship between temperature and distress:

```
> plot(x = launch$temperature, y = launch$distress_ct,
+ main = "Scatterplot of Temperature vs. Distress Count",
+ xlab = "Temperature",
+ ylab = "Distress Count")
```



Linear regression *(cont.)*

The correlation between he distress count and the launch temperature is approx. 0.5:

```
> cor(launch$temperature, launch$distress_ct)
[1] -0.5111264
```

Instead of using the formulas from the previous slides, you can directly run a linear regression model using the built-in function *lm()*:

```
> launch.lm = lm(distress_ct ~ temperature, data=launch)
> coefficients(launch.lm)
(Intercept) temperature
 3.69841270 -0.04753968
```

In other words, $\hat{\alpha}$ = 3.6984127 and $\hat{\beta}$ = -0.04753968

Multiple linear regression

You can skip the formulas on page 183. The only thing you need to remember about multiple regression is that there are multiple independent variables:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_i x_i + \varepsilon$$

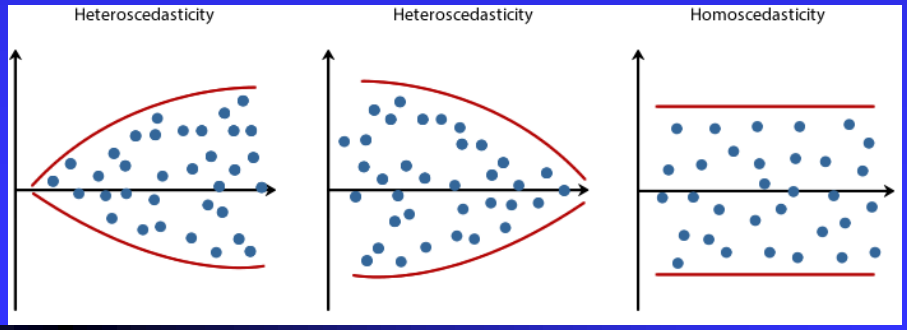
Epsilon is the error and each coefficient β_i measures the relative impact of x_i on y. In other words, *y* changes by the amount β_i for each unit increase in x_i. The intercept β_0 is then the expected value of *y* when the independent variables are all zero. The coefficients β_i can be computed using the built-in functions in R.

When we can use linear regression

There are four principal assumptions which justify the use of linear regression models:

- Residuals, $\varepsilon_i = yi - \alpha - \beta xi$, are **normally** distributed.
- Residuals are **independent**
- **Linearity**: The dependent variable is linearly related to the independent variables.
- The expected value of the residuals is independent of the predictor variable x, i.e. $E(\varepsilon|x) = 0$ for all x.
- **Homoscedasticity** (constant variance of errors): The variance of the dependent variables doesn't vary with the levels of the independent variables. In other words, variances along the line of best fit remain similar as you move along the line. That is, the variance of the residuals is independent of the predictor variable x, $Var(\varepsilon|x)$ is constant for all x

If any of these assumptions is violated, then the predictions might not be accurate. In general, a linear regression model fits better if the assumptions are met.



Example – predicting medical expenses using linear regression

In order for a health insurance company to make money, it needs to collect more in yearly premiums than it spends on medical care to its beneficiaries. As a result, insurance companies invest in models that forecast medical expenses for the insured population. We will use patient data to estimate the average medical care expenses for a given population. These estimates can be used to create actuarial tables that set the price of yearly premiums, depending on the expected treatment costs.

The data set in your textbook was created using demographic statistics from the US Census Bureau. You can download it from the textbook website or directly from the Course Materials Section on Blackboard.

As usual, we will read the dataset into a data frame and will explore its structure (on the next slide). The insurance file contains 1338 observations of 7 variables: age, sex, body mass index (bmi), children, smoker, region, and expenses. One can expect that older people and smokers are at higher risk of medical expenses. We will run a multiple regression model using expenses as dependent variable and all other variables as independent.

Predicting medical expenses *(cont.)*

```
> insurance <- read.csv("insurance.csv", stringsAsFactors = TRUE)
> str(insurance)
'data.frame':   1338 obs. of  7 variables:
 $ age     : int  19 18 28 33 32 31 46 37 37 60 ...
 $ sex     : Factor w/ 2 levels "female","male": 1 2 2 2 2 1 1 1 2 1 ...
 $ bmi     : num  27.9 33.8 33 22.7 28.9 25.7 33.4 27.7 29.8 25.8 ...
 $ children: int  0 1 3 0 0 0 1 3 2 0 ...
 $ smoker  : Factor w/ 2 levels "no","yes": 2 1 1 1 1 1 1 1 1 1 ...
 $ region  : Factor w/ 4 levels "northeast","northwest",..: 4 3 3 2 2 3 3
2 1 2 ...
 $ expenses: num  16885 1726 4449 21984 3867 ...
```

Correlation matrix

The correlation matrix provides a quick overview of how the independent variables are related to the dependent variable and each other using pairwise correlation coefficients. The correlation matrix is created using the *cor()* command. Note, that the *cor()* command can ne used both for single correlation coefficients and for correlation matrixes.

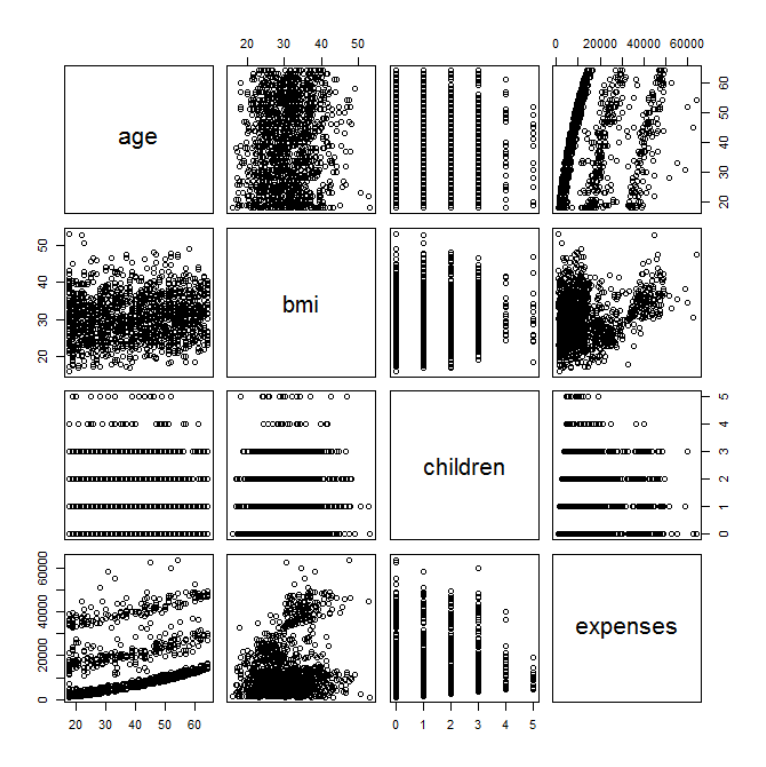
```
> cor(insurance[c("age", "bmi", "children", "expenses")])
                age         bmi   children   expenses
age      1.0000000 0.10934101 0.04246900 0.29900819
bmi      0.1093410 1.00000000 0.01264471 0.19857626
children 0.0424690 0.01264471 1.00000000 0.06799823
expenses 0.2990082 0.19857626 0.06799823 1.00000000
```

The element on the intersection of the i^{th} row with the j^{th} column shows the correlation coefficient between the i^{th} row and the j^{th} column. Every correlation matrix is symmetric and has 1s on its main diagonal (because these are the correlation coefficients for each variable with itself, i.e., there is a perfect correlation between a variable and itself) Here, we have a correlation matrix for rows age, bmi, children and expenses. The other three variables from the dataset are nominal. According to the matrix, the strongest correlation (with a correlation coefficient of 0.299) is between age and expenses.

Scatterplot matrix (SPLOM)

The correlations can be visualized using a **scatterplot matrix** (**SPLOM**), which represents plaits of scatterplots arranged in a grid. The SPLOM makes the correlations easier to spot. A SPLOM can me created by the *pairs()* command:

```
> pairs(insurance[c("age", "bmi", "children", "expenses")])
```



For example, the first scatterplot on the last row is the scatterplot between age and expenses. It shows three lines which are close to straight lines. There are no visual trends in the other plots.

The following command is equivalent to the previous one and creates the same SPLOM:

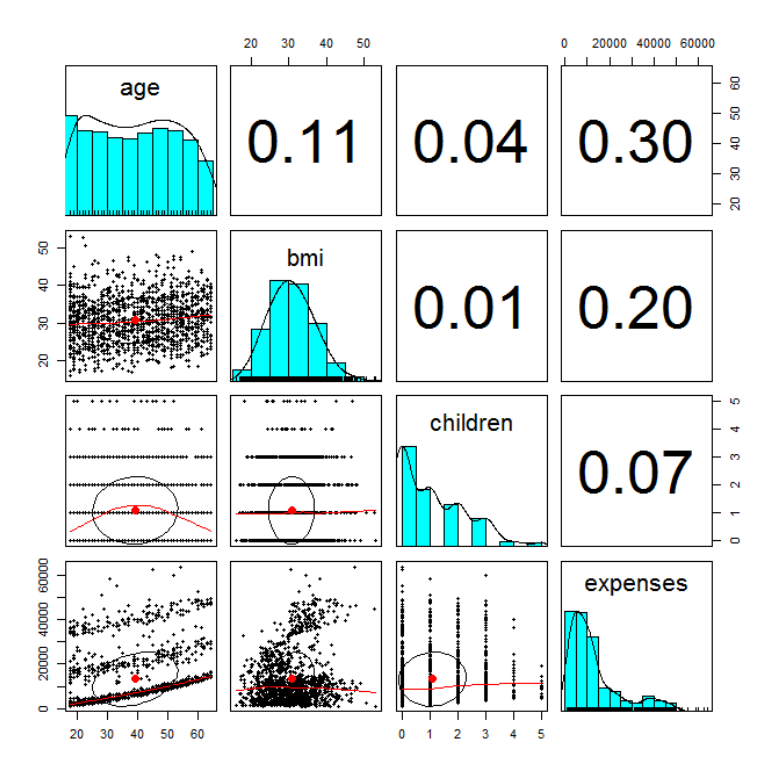
```
> pairs(~age+bmi+children+expenses,
+ data=insurance)
```

~age+bmi+children+expenses is a formula in R. It means “include variables age and bmi and children and expenses”.

Scatterplot matrix *(cont.)*

Additional information about the pairwise scatterplots can be obtained using the *pairs.pannels()* function from the *psych* package:

```
> install.packages("psych")
> library(psych)
> pairs.panels(insurance[c("age", "bmi", "children", "expenses")])
```



The correlation coefficients are shown above the main diagonal. The diagonal shows histograms for the distribution of values for each variable. The scatterplots below the diagonal show the correlation ellipse, the shape of which indicates the correlation strength for each scatterplot. There is a dot at the center of each correlation ellipse that is drawn at the mean values for the x and y axis variables. The more stretched the ellipse is, the stronger the correlation. An almost perfectly round oval, as with bmi and children, indicates a very weak correlation..

Training a model on the data

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

`myfit <- lm(formula, data)`

where *formula* describes the model to be fit and *data* is the data frame containing the data to be used in fitting the model. The resulting object (myfit in this case) is a list that contains extensive information about the fitted model. The formula is typically written as

Y ~ X1 + X2 + … + Xk

where the ~ separates the dependent variable on the left from the independent variables on the right, and the predictor variables are separated by + signs.

In general, a formula in R would look like

x~y+z+v

Here, x is the dependent variable and y, z, and v are independent variables. The formula represents the linear regression model:

$$x = \beta_0 + \beta_1 y + \beta_2 z + \beta_3 v$$

In each formula, such as x~y+z+v, there is no need to specify the regression model's intercept term as it is assumed by default:

Training a model on the data *(cont.)*

There are several other symbols used in formulas shown on the next slide. For example, the following three formulae are all equivalent:

Y ~ X + Z + W + X:Z + X:W + Z:W + X:Z:W

Y ~ X * Z * W

Y ~ (X + Z + W)^3

And their correspond to the following model:

$$Y = \beta_0 + \beta_1 X + \beta_2 Z + \beta_3 W + \beta_4 XZ + \beta_5 XW + \beta_6 ZW + \beta_7 XZW$$

In our example of medical expenses, we can run a regression model using expenses as dependent variable and age, children, bmi, sex, smoker, and region as independent variables:

```
> ins_model <- lm(expenses ~ age + children + bmi + sex +
+ smoker + region, data = insurance)
```

The "." character can be used to specify all the features. As a result, the same command can be rewritten as:

```
> ins_model <- lm(expenses ~ ., data = insurance)
```

Symbols used in formulas

Symbol	Usage
~	Separates response variables on the left from the explanatory variables on the right. For example, a prediction of y from x, z, and w would be coded `y ~ x + z + w`.
+	Separates predictor variables.
:	Denotes an interaction between predictor variables. A prediction of y from x, z, and the interaction between x and z would be coded `y ~ x + z + x:z`.
*	A shortcut for denoting all possible interactions. The code `y ~ x * z * w` expands to `y ~ x + z + w + x:z + x:w + z:w + x:z:w`.
^	Denotes interactions up to a specified degree. The code `y ~ (x + z + w)^2` expands to `y ~ x + z + w + x:z + x:w + z:w`.
.	A place holder for all other variables in the data frame except the dependent variable. For example, if a data frame contained the variables `x, y, z, and w`, then the code `y ~ .` would expand to `y ~ x + z + w`.
-	A minus sign removes a variable from the equation. For example, `y ~ (x + z + w)^2 - x:w` expands to `y ~ x + z + w + x:z + z:w`.
-1	Suppresses the intercept. For example, the formula `y ~ x -1` fits a regression of y on x, and forces the line through the origin at `x=0`.
I()	Elements within the parentheses are interpreted arithmetically. For example, `y ~ x + (z + w)^2` would expand to `y ~ x + z + w + z:w`. In contrast, the code `y ~ x + I((z + w)^2)` would expand to `y ~ x + h`, where h is a new variable created by squaring the sum of z and w.
function	Mathematical functions can be used in formulas. For example, `log(y) ~ x + z + w` would predict `log(y)` from x, z, and w.

Training a model on the data *(cont.)*

We can get a brief look at the correlation coefficients:

```
> ins_model

Call:
lm(formula = expenses ~ ., data = insurance)

Coefficients:
    (Intercept)              age          sexmale              bmi
      -11941.6            256.8           -131.4            339.3
       children        smokeryes  regionnorthwest  regionsoutheast
          475.7          23847.5           -352.8          -1035.6
regionsouthwest
          -959.3
```

This means that we have built the following regression model:

$$expenses = -11941.6 + 256.8 * age - 131.4 * sexmale + 339.3 * bmi + 475.7 * children + 23847.5 * smokeyes - 352.8 * regionwest - 1035.6 * regionsoutheast - 959.3 * regionsouthwest$$

Dummy variables and dummy coding

You may notice that we had 6 variables originally (age, children, bmi, sex, smoker, region). However, our regression equation includes 8 variables (age, children, bmi, sexmale, smokeryes, regionnorthwest, regionsoutheast, regionsouthwest). R has automatically replaced sex with sexmale and region with regionnorthwest, regionsoutheast, and regionsouthwest. This was done in order to convert the nominal/factor variables sex and region into numeric (actually Boolean) variables. For example, sex was coded with one Boolean variable called sexmale. That is, sexmale=1 represents sex=male and sexmale=0 corresponds to sex=female. Similary, region (which has four levels: northeast, northwest, southeast, southwest) is modeled with three dummy variables called regionnorthwest, regionsoutheast, and regionsouthwest. When regionnorthwest=1, regionsoutheast=0, and regionsouthwest=0, then region=northwest. When regionnorthwest=0, regionsoutheast=1, and regionsouthwest=0, then region=southeast. When regionnorthwest=0, regionsoutheast=0, and regionsouthwest=1, then region=southwest. Finally, when regionnorthwest=0, regionsoutheast=0, and regionsouthwest=0, then region=northeast.

Evaluating model performance

To obtain more information about the regression model we can run:

```
>  summary(ins_model)
Call:
lm(formula = expenses ~ ., data = insurance)
Residuals:
     Min       1Q   Median       3Q      Max
-11302.7  -2850.9   -979.6   1383.9  29981.7
Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     -11941.6      987.8 -12.089  < 2e-16 ***
age                256.8       11.9  21.586  < 2e-16 ***
sexmale           -131.3      332.9  -0.395 0.693255
bmi                339.3       28.6  11.864  < 2e-16 ***
children           475.7      137.8   3.452 0.000574 ***
smokeryes        23847.5      413.1  57.723  < 2e-16 ***
regionnorthwest   -352.8      476.3  -0.741 0.458976
regionsoutheast  -1035.6      478.7  -2.163 0.030685 *
regionsouthwest   -959.3      477.9  -2.007 0.044921 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 6062 on 1329 degrees of freedom
Multiple R-squared:  0.7509,    Adjusted R-squared:  0.7494
F-statistic: 500.9 on 8 and 1329 DF,  p-value: < 2.2e-16
```

Evaluating model performance: residuals statistics

```
Residuals:
     Min       1Q   Median       3Q      Max
-11302.7  -2850.9   -979.6   1383.9  29981.7
```

Since a residual is equal to the true value minus the predicted value, the maximum error of 29981.7 suggests that the model under-predicted expenses by nearly $30,000 for at least one observation. This is the maximum absolute residual error.

Ideally, the regression residuals would have a perfect, normal distribution (the ideal lineal regression algorithm is mathematically guaranteed to produce residuals with a mean of zero). These statistics help you identify possible deviations from normality. Hence the sign of the median indicates the skew's direction, and the magnitude of the median indicates the extent. In this case the median is negative, which suggests some skew to the right. If the residuals have a nice, bell-shaped distribution, then the first quartile (1Q) and third quartile (3Q) should have about the same magnitude. In this example, the larger magnitude of 1Q versus 3Q (2850.9 versus 1383.9) also indicates a skew to the right in our data.

Evaluating model performance: coefficients

```
Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     -11941.6      987.8 -12.089  < 2e-16 ***
age                256.8       11.9  21.586  < 2e-16 ***
sexmale           -131.3      332.9  -0.395 0.693255
bmi                339.3       28.6  11.864  < 2e-16 ***
children           475.7      137.8   3.452 0.000574 ***
smokeryes        23847.5      413.1  57.723  < 2e-16 ***
regionnorthwest   -352.8      476.3  -0.741 0.458976
regionsoutheast  -1035.6      478.7  -2.163 0.030685 *
regionsouthwest   -959.3      477.9  -2.007 0.044921 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The column labeled Estimate contains the estimated regression coefficients as calculated by ordinary least squares. Theoretically, if a variable's coefficient is zero then the variable is worthless; it adds nothing to the model. Yet the coefficients shown here are only estimates, and they will never be exactly zero. We therefore ask: Statistically speaking, how likely is it that the true coefficient is zero? That is the purpose of the *t* statistics and the *p*-values, which in the summary are labeled (respectively) t value and Pr(>|t|).

Evaluating model performance: coefficients *(cont.)*

The *p*-value is a probability. It gauges the likelihood that the coefficient is *not* significant, so smaller is better. Big is bad because it indicates a high likelihood of insignificance. In this example, the *p*-value for the *age* coefficient is a less than 2e-16, so *age* is likely significant. The *p*-value for sexmale, however, is 0.693255; this is just over the conventional limits (the standard significance levels are α=0.10, α=0.05, and α=0.01), which suggests that *w* is likely insignificant. Variables with large *p*-values are candidates for elimination.

A handy feature is that R flags the significant variables for quick identification by using triple asterisks, double asterisks, a single asterisk, and a period:

*** *p*-value between 0 and 0.001

** *p*-value between 0.001 and 0.01

* *p*-value between 0.01 and 0.05

. *p*-value between 0.05 and 0.1

(blank) *p*-value between 0.1 and 1.0

Evaluating model performance: R2 (coefficient of determination)

```
Multiple R-squared:  0.7509,    Adjusted R-squared:  0.7494
```

R^2 is a measure of the model's quality. Bigger is better. Mathematically, it is the fraction of the variance of *y* that is explained by the regression model. The remaining variance is not explained by the model, so it must be due to other factors (i.e., unknown variables or sampling variability). In this case, the model explains 0.7509 (75.09%) of the variance of *y*, and the remaining 0.2491 (24.91%) is unexplained. The adjusted R^2 value accounts for the number of variables in our model and so is a more realistic assessment of its effectiveness. The adjusted R-squared value corrects R-squared by penalizing models with a large number of independent variables. It is useful for comparing the performance of models with different numbers of independent variables.

Evaluating model performance: F statistic

F-statistic: 500.9 on 8 and 1329 DF, p-value: < 2.2e-16

The *F* statistic tells you whether the model is significant or insignificant. The model is significant if any of the coefficients are nonzero (i.e., if $\beta_i \neq 0$ for some *i*). It is insignificant if all coefficients are zero ($\beta_1 = \beta_2 = \ldots = \beta_n = 0$).

A *p*-value of less than 0.05 (or 0.01, or 0.10) indicates that the model is likely significant (one or more β_i are nonzero) whereas larger values indicate that the model is likely not significant. Here, the probability is less than 2.2e-16, indicating that our models is significant.

Improving model performance

- **Converting a numeric variable to a binary indicator:** Suppose that you expect the effect of an independent variable to be not cumulative, i. e., the independent variable affects the dependent variable only after the independent variable reaches a specific threshold. For example, you might expect that BMI may have impact on expenditures only if BMI > 30. To model this, we will create a new binary feature insurance$bmi30 which is 1 if BMI > 30 and 0 otherwise.

```
> insurance$bmi30 <- ifelse(insurance$bmi >= 30, 1, 0)
```

- Adding nonlinear effects: There are two types of nonlineraity: A single variable nonlinearity and interaction between variables. A single variable nonlinearity occurs when the dependent variable depends on higher powers of the independent variable. For example, we can create an age squared term and add it to age:

```
> insurance$age2 <- insurance$age^2
lm(expenses ~ age + age2, data = insurance)
```

- Interaction between variables is modeled as mixed terms of higher powers involving two or more variables. For example, in the equation:

$$Y = \beta_0 + \beta_1 X + \beta_2 Y + \beta_3 Z + \beta_4 XY + \beta_5 XZ + \beta_6 YZ + \beta_7 XYZ$$

The mixed terms XY, XZ, YZ, XYZ represent interactions between variables. The following equation models the interaction between bmi30 and smokeryes:

expenses ~ bmi30 + smokeryes + bmi30:smokeryes .

Improving model performance *(cont.)*

Basically, it is equivalent to:

$$expenses = \beta_0 + \beta_1 bmi30 + \beta_2 smokeryes + \beta_3 bmi30 * smokeryes$$

In the formula

expenses ~ bmi30 + smokeryes + bmi30:smokeryes,

the ":" sign represents multiplication. There is a shorthand for the same formula:

expenses ~ bmi30*smokeryes.

Here, "*" means "all possible interactions" between bmi30 and smokeryes. In other words, bmi30*smokeryes unfolds to:

bmi30 + smokeryes + bmi30:smokeryes.

Now, we will train another model using the following nonlinear equation:

expenses ~ age + age2 + children + bmi + sex + bmi30*smoker + region

```
> ins_model2 <- lm(expenses ~ age + age2 + children + bmi +
+ sex + bmi30*smoker + region, data = insurance)
```

The results are:

```
> summary(ins_model2)
```

Improving model performance *(cont.)*

```
Call:
lm(formula = expenses ~ age + age2 + children + bmi + sex + bmi30 *
    smoker + region, data = insurance)
Residuals:
     Min       1Q   Median       3Q      Max
-17297.1  -1656.0  -1262.7   -727.8  24161.6
Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        139.0053  1363.1359   0.102 0.918792
age                -32.6181    59.8250  -0.545 0.585690
age2                 3.7307     0.7463   4.999 6.54e-07 ***
children           678.6017   105.8855   6.409 2.03e-10 ***
bmi                119.7715    34.2796   3.494 0.000492 ***
sexmale           -496.7690   244.3713  -2.033 0.042267 *
bmi30             -997.9355   422.9607  -2.359 0.018449 *
smokeryes        13404.5952   439.9591  30.468  < 2e-16 ***
regionnorthwest   -279.1661   349.2826  -0.799 0.424285
regionsoutheast   -828.0345   351.6484  -2.355 0.018682 *
regionsouthwest  -1222.1619   350.5314  -3.487 0.000505 ***
bmi30:smokeryes 19810.1534   604.6769  32.762  < 2e-16 ***
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 4445 on 1326 degrees of freedom
Multiple R-squared:  0.8664,    Adjusted R-squared:  0.8653
F-statistic: 781.7 on 11 and 1326 DF,  p-value: < 2.2e-16
```

Improving model performance *(cont.)*

The results show that the nonlinear terms *age2* and *bmi30:smokeryes* are statistically significant. The coefficient 19810 for the mixed term *bmi30:smokeryes* shows that obese smokers spend another $19,810. Moreover, the R-squared of the model has improved from 0.75 to 0.87, and the adjusted R-squared value has improved from 0.75 to 0.87. In addition, the new model is explaining 87 percent of the variation in medical costs.

Regression trees

When we use a decision tree to make predications for a continuous target, we refer to the tree as a regression tree. Typically, the value output by the leaf node of a regression tree is the mean of the target values of the instances from the training set that reached that node. This means that the error of a regression tree when making a prediction for the query instance is the difference between the mean of the training instances that reached the leaf node that returns the prediction and the correct value that should have been returned. Then, it makes sense to construct regression trees in a manner that reduces the variance in the target feature values of the set of the training instances at each leaf node in the tree. We can do this by adapting the ID3 algorithm to use a measure of variance rather than a measure of entropy when selecting the best feature. Using variance as a measure of impurity, the impurity at a node can be calculated as:

$$var(T) = \frac{\sum_{i=1}^{n}\left(d_i - \overline{d}\right)^2}{n-1}$$

Where T is the dataset that reached the node, n is the number of instances in T, $\overline{d}$ is the mean of the target value for the dataset T, and t_i iterates across the target value of each instance in T.

Regression trees *(cont.)*

Using variance as our measure of impurity, we can select which feature to split on at a node by selecting the feature that minimizes the weighted variance across the resulting partitions. The weighted variance is computed by summing the variance of the target feature within each partition created by splitting a dataset on the feature multiplied by the fraction of the dataset in each partition:

$$\sum_i \frac{|T_i|}{|T|} \mathrm{var}(\mathrm{Ti})$$

As a result, leaf nodes with small variance in the target feature values across the set of instances at the node are preferred over leaf nodes where the variable in the target feature value across the set of instances at the node is large.

Your textbook describes a feature selection measure called **Standard Deviation Reduction,** based on the following formula:

$$\mathrm{SDR} = sd(T) - \sum_i \frac{|T_i|}{|T|} \times sd(T_i)$$

Regression trees *(cont.)*

It is basically the same because it will select a feature with minimal weighted standard deviation:

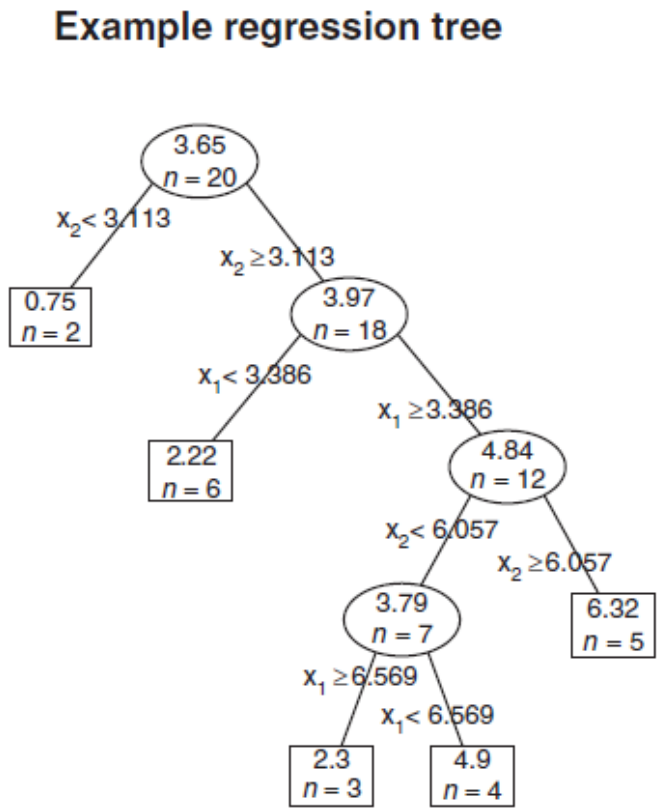
$$\sum_i \frac{|T_i|}{|T|} \times sd(T_i)$$

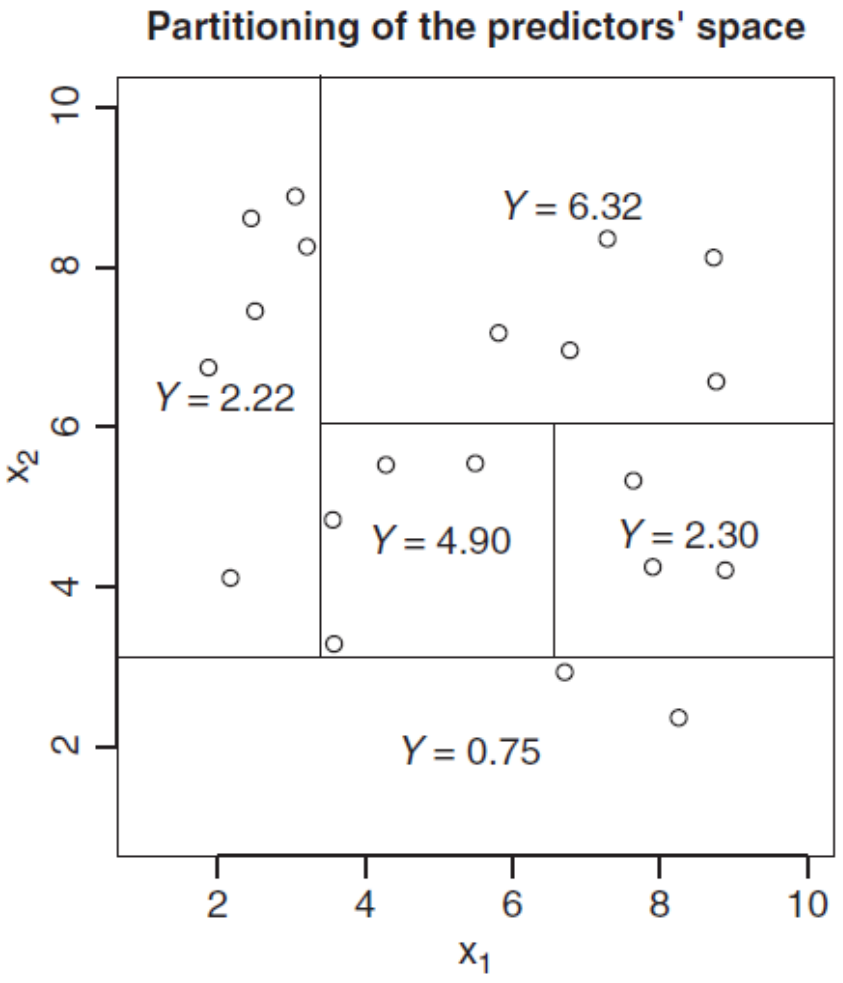
The only difference is that the Standard Deviation Reduction uses standard deviation instead of variance.

In the ID3 algorithm we create a leaf node when there are no instances in the partition being processes, when there are no features left on which to split data, or when we have created a pure partition. An algorithm for regression trees can use the first two cases. When these cases occur, the algorithm will create a leaf node that returns the mean value of the target feature in a data partition.

One advantage of regression trees over regression models is that regression models makes assumptions about how numeric data is distributed that are often violated in real-world data. For example, linear regression models require normality, independence, linearity, and homoscedasticity. This is not the case for regression trees.

Example of a regression tree





Regression trees *(cont.)*

Tests on numerical variables usually take the form $x_i < \alpha$, with $\alpha \in \mathbb{R}$, while tests on nominal variables have the form $x_i \in \{v\,1, v_2, \ldots vm\}$. Any regression tree provides a full mutually exclusive partition of the predictor space into regions with boundaries that are parallel to the predictors' axes due to the form of the tests. The figure on the right side of the previous slide illustrates these ideas with a tree and the respective partitioning. All observations in a partition are predicted with the same constant value, and that is the reason for regression trees sometimes being referred to as piecewise constant models.

Regression trees may overfit the training data which will inevitably lead to poor out-of-the-sample predictive performance. The standard procedure to fight this undesirable effect is to grow an overly large tree and then to use some reliable error estimation procedure to find the "best" sub-tree of this large model. This procedure is known as post-pruning a tree An alternative is to stop tree growth sooner in a process known as pre-pruning, which again needs to be guided by reliable error estimation to know when overfitting is starting to occur.

Advantages and disadvantages of regression trees

Regression trees have several features that make them a very interesting approach to several multiple regression problems. For example, regression trees provide:

(a) automatic variable selection making them highly insensitive to irrelevant variables;

(b) computational efficiency that allows addressing large problems;

(c) handling of unknown variable values;

(d) handling of both numerical and nominal predictor variables;

(e) insensitivity to predictors' scales; and

(f) Interpretable models for most domains.

In spite of all these advantages, regression trees have poor prediction accuracy in several domains because of the piecewise constant approximation they provide.

Estimating the quality of wines with regression trees and model trees

We will use regression trees and model trees to estimate the quality of white wines. You can download the dataset from Blackboard or from the textbook website. The dataset includes examples of white Vinho Verde wines from Portugal.

```
> wine <- read.csv("whitewines.csv")
> str(wine)
'data.frame':   4898 obs. of  12 variables:
 $ fixed.acidity       : num  6.7 5.7 5.9 5.3 6.4 7 7.9 6.6 7 6.5 ...
 $ volatile.acidity    : num  0.62 0.22 0.19 0.47 0.29 0.14 0.12 0.38 0.16 0.37
...
 $ citric.acid         : num  0.24 0.2 0.26 0.1 0.21 0.41 0.49 0.28 0.3 0.33 ...
 $ residual.sugar      : num  1.1 16 7.4 1.3 9.65 0.9 5.2 2.8 2.6 3.9 ...
 $ chlorides           : num  0.039 0.044 0.034 0.036 0.041 0.037 0.049 0.043
0.043 0.027 ...
 $ free.sulfur.dioxide : num  6 41 33 11 36 22 33 17 34 40 ...
 $ total.sulfur.dioxide: num  62 113 123 74 119 95 152 67 90 130 ...
 $ density             : num  0.993 0.999 0.995 0.991 0.993 ...
 $ pH                  : num  3.41 3.22 3.49 3.48 2.99 3.25 3.18 3.21 2.88 3.28
...
 $ sulphates           : num  0.32 0.46 0.42 0.54 0.34 0.43 0.47 0.47 0.47 0.39
...
 $ alcohol             : num  10.4 8.9 10.1 11.2 10.9 ...
 $ quality              : int  5 6 6 4 6 6 6 6 6 7 ...
```

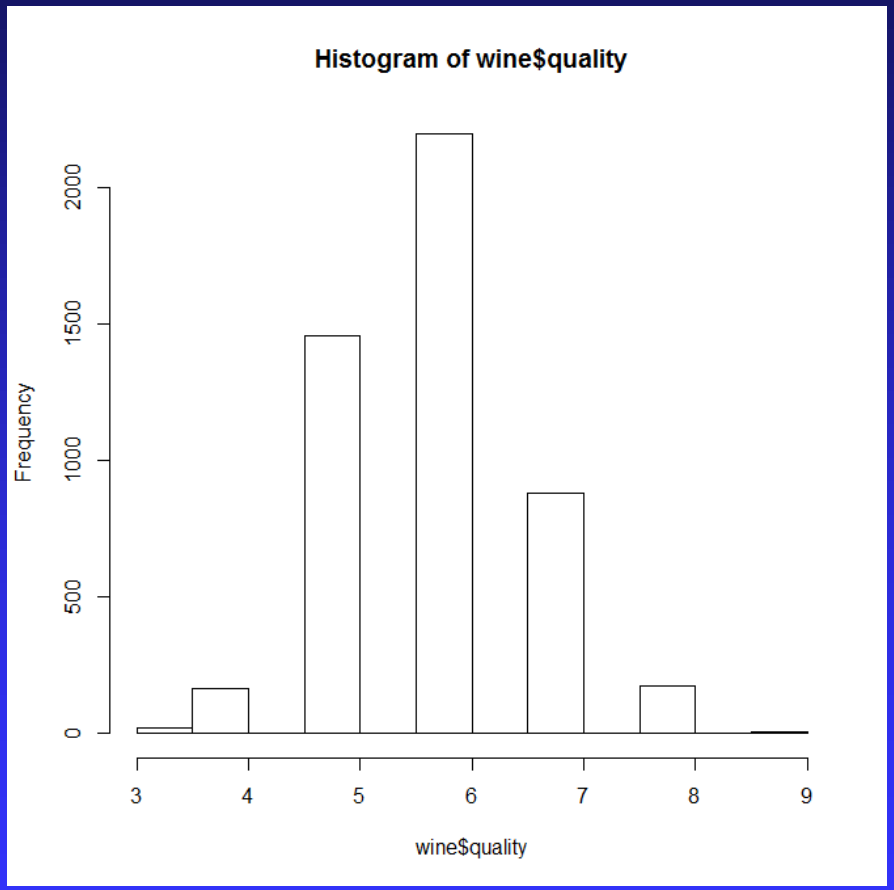
Estimating the quality of white wines

The white wine data includes information on 11 chemical properties of 4,898 wine samples. For each wine, a laboratory analysis measured characteristics such as acidity, sugar content, chlorides, sulfur, alcohol, pH, and density. The samples were then rated, wine$quality, in a blind tasting by panels of no less than three judges on a quality scale ranging from zero (very bad) to 10 (excellent). In the case of judges disagreeing on the rating, the median value was used.

One advantage of regression trees is that they can handle many types of data without preprocessing. In other words, we do not need to normalize or standardize the features.

Let's check the distribution of wine ratings using a simple histogram:

```
> hist(wine$quality)
```



Estimating the quality of white wines

The histogram, shows that ratings have an approximately normal, bell-shaped distribution, centered around a value of six. In other words, most wines are of average quality; few are particularly bad or good.

The next step is select the training and testing datasets. Assuming that the observations are stored in a random order, we can select the first 3750 observations as training set and the remaining 25% of the dataset as a testing set.

```
> wine_train <- wine[1:3750, ]
> wine_test <- wine[3751:4898, ]
```

We will use an implementation of regression and model trees from the rpart (recursive partitioning) package:

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

The rpart function can be used to produce a regression tree. In the function call, we specify quality as the outcome variable and use the dot notation to specify that the outcome variable depends on all the other columns in the wine_train data frame. The resulting regression tree model object is stored into the m.rpart variable:

```
> m.rpart <- rpart(quality ~ ., data = wine_train)
```

Estimating the quality of white wines

```
> m.rpart
n= 3750
node), split, n, deviance, yval
      * denotes terminal node
 1) root 3750 2945.53200 5.870933
   2) alcohol< 10.85 2372 1418.86100 5.604975
     4) volatile.acidity>=0.2275 1611  821.30730 5.432030
       8) volatile.acidity>=0.3025 688  278.97670 5.255814 *
       9) volatile.acidity< 0.3025 923  505.04230 5.563380 *
     5) volatile.acidity< 0.2275 761  447.36400 5.971091 *
   3) alcohol>=10.85 1378 1070.08200 6.328737
     6) free.sulfur.dioxide< 10.5 84   95.55952 5.369048 *
     7) free.sulfur.dioxide>=10.5 1294  892.13600 6.391036
      14) alcohol< 11.76667 629  430.11130 6.173291
        28) volatile.acidity>=0.465 11   10.72727 4.545455 *
        29) volatile.acidity< 0.465 618  389.71680 6.202265 *
      15) alcohol>=11.76667 665  403.99400 6.596992 *
```

Estimating the quality of white wines *(cont.)*

Let's explain the following node:

```
volatile.acidity>=0.3025 688  278.97670 5.255814 *
```

For each node pf the tree the following information is listed:

- The test performed at the node: volatile.acidity>=0.3025

- The number of the examples at that node: 688

- The deviance: 278.97670. It is a measure of goodness of fit of the model. Higher numbers indicate worse fit.

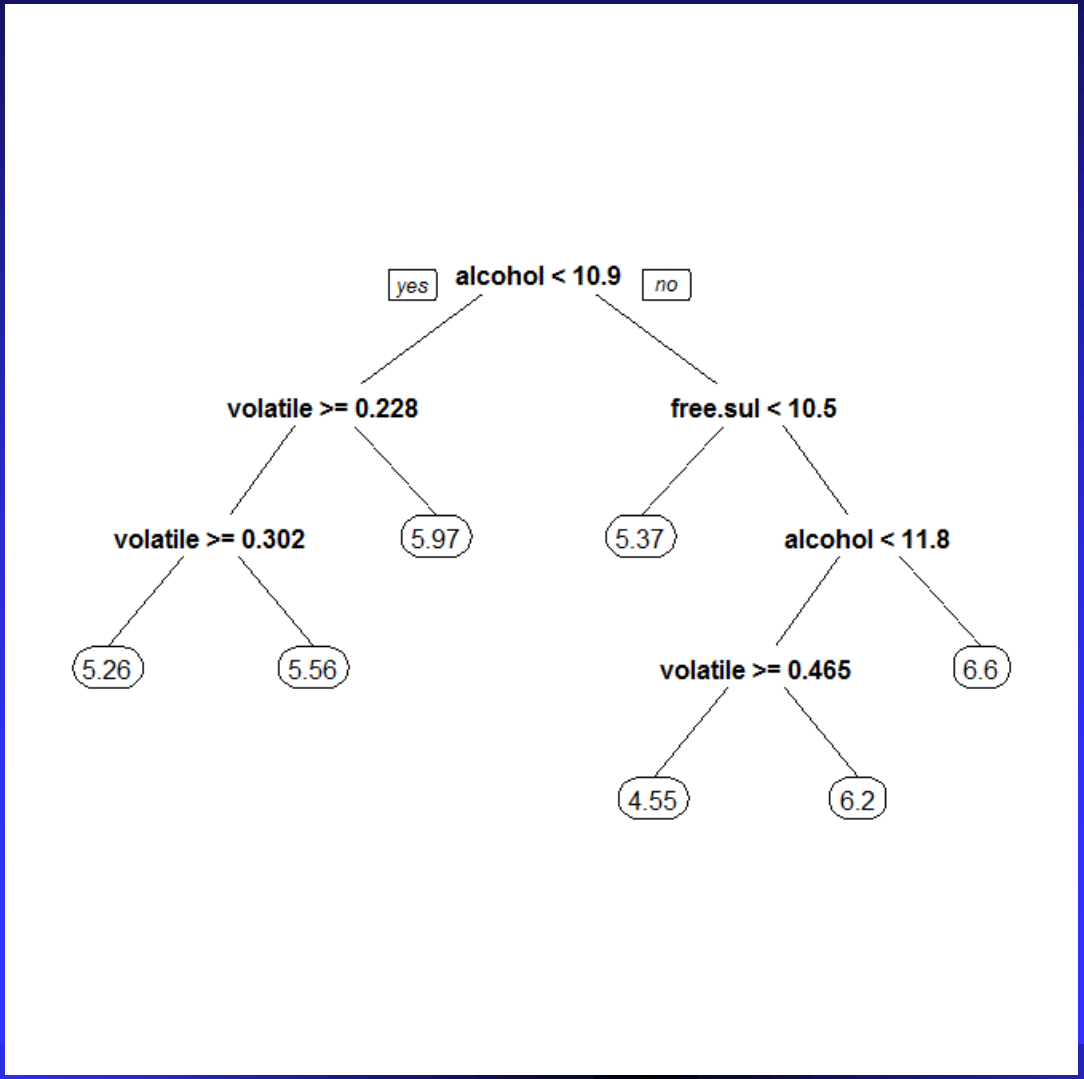
- yval: 5.255814. This is the fitted (the predicted) value for leaf nodes. The leaf nodes are marked by "*". In other words, all examples for which alcohol< 10.85 and volatile.acidity>=0.3025 will have a predicted value of 5.255814 for the dependent variable.

Regression trees can be conveniently visualized using the rpart.plot package.

```
> install.packages("rpart.plot")
> library(rpart.plot)
>  rpart.plot(m.rpart, digits = 3)
```

The regression tree is shown on the next slide.

Estimating the quality of white wines *(cont.)*



Estimating the quality of white wines *(cont.)*

The rpart.plot function takes several parameters.

```
rpart.plot(x=stop("no 'x' arg"),
        type=0, extra=0, under=FALSE, clip.right.labs=TRUE,
        fallen.leaves=FALSE, branch=if(fallen.leaves) 1
        else .2,
        uniform=TRUE,
        digits=2, varlen=-8, faclen=3,
        cex=NULL, tweak=1,
        compress=TRUE, ycompress=uniform,
        snip=FALSE,
...)
```

You can find the meaning of all parameters in the rpart.plot package description. *Type* is the type of the plot. 0 is the default (draw a split label at each split and a node label at each leaf). *Extra* displays extra information at the nodes. The default value is 0 (no extra information). *fallen.leaves* displays the leaves at the bottom of the graph if it is TRUE.

Evaluating model performance

To evaluate the model performance we will make predictions on the test data using the *predict()* function and store the result in a vector *p.rpart*:

```
> p.rpart <- predict(m.rpart, wine_test)
```

Let's compare predicted with actual values:

```
> summary(p.rpart)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.545   5.563   5.971   5.893   6.202   6.597
> summary(wine_test$quality)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  3.000   5.000   6.000   5.901   6.000   9.000
```

It is clear that our model does well in the interval between the first quartile and the second quartile and not so well outside this interval.

Estimating the quality of white wines

Regression trees syntax

using the `rpart()` function in the `rpart` package

Building the model:

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

- `dv` is the dependent variable in the `mydata` data frame to be modeled
- `iv` is an R formula specifying the independent variables in the `mydata` data frame to use in the model
- `data` specifies the data frame in which the `dv` and `iv` variables can be found

The function will return a regression tree model object that can be used to make predictions.

Making predictions:

```
p <- predict(m, test, type = "vector")
```

- `m` is a model trained by the `rpart()` function
- `test` is a data frame containing test data with the same features as the training data used to build the model
- `type` specifies the type of prediction to return, either `"vector"` (for predicted numeric values), `"class"` for predicted classes, or `"prob"` (for predicted class probabilities)

The function will return a vector of predictions depending on the `type` parameter.

Example:

```
wine_model <- rpart(quality ~ alcohol + sulfates,
                    data = wine_train)
wine_predictions <- predict(wine_model, wine_test)
```

Measuring the performance of the regression model

There are several measures that can be used to measure the performance of regression model:

- mean absolute error (MAE): $\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$, where y_i is the actual value and $\hat{y}_i$ is the predicted value.

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

- relative square error: $\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y_i})^2}$, here $\overline{y_i}$ is the mean value of all actual observations

- R-square = 1 – relative-square-error

Measuring the performance of the regression model *(cont.)*

In our case, the mean absolute error is 0.5872652

```
> mean(abs(wine_test$quality-p.rpart))
[1] 0.5872652
```

The root mean square error is 0.008058241:

```
> (mean(wine_test$quality - p.rpart)^2)^0.5
[1] 0.008058241
```

The relative square error is 0.7118109:

```
> mean((wine_test$quality- p.rpart)^2) /
mean((wine_test$quality - mean(wine_test$quality))^2)
[1] 0.7118109
```

One way to improve the performance of the model is to use a model tree instead of regression tree.

The R-square is 1- 0.7118109 = 0.2881891

Model trees

Model trees are motivated by the purpose of overcoming some of the known limitations of regression trees caused by their piecewise constant approximation. By using constants at the leaves, regression trees provide a coarse grained prediction leading to poor accuracy in some domains. Model trees try to overcome this by using regression models on the leaves. The added complexity of the models used in the leaves increases the computational complexity of model trees when compared to regression trees, and also decreases their interpretability. The most common form of model used in leaves is linear regression.

We will use M5′ variant of the original M5 model tree algorithm proposed by J.R. Quinlan in 1992. The M5′ algorithm is available in R via the RWeka package and the M5P() function.

The parameters of M5P function

Model trees syntax

using the `M5P()` function in the `RWeka` package

Building the model:

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

- `dv` is the dependent variable in the `mydata` data frame to be modeled
- `iv` is an R formula specifying the independent variables in the `mydata` data frame to use in the model
- `data` specifies the data frame in which the `dv` and `iv` variables can be found

The function will return a model tree object that can be used to make predictions.

Making predictions:

```
p <- predict(m, test)
```

- `m` is a model trained by the `M5P()` function
- `test` is a data frame containing test data with the same features as the training data used to build the model

The function will return a vector of predicted numeric values.

Example:

```
wine_model <- M5P(quality ~ alcohol + sulfates,
                  data = wine_train)
wine_predictions <- predict(wine_model, wine_test)
```

Model trees *(cont.)*

We need first to install the RWeka package. Since the package is java-based you need to have java preinstalled on your computer:

```
> install.packages("RWeka")
> library(RWeka)
> m.m5p <- M5P(quality ~ ., data = wine_train)
```

This is a quite large tree, which can be visualized using the plot(m.m5p) function from the package partykit:

```
> install.packages("partykit")
> library(partykit)
> plot(m.m5p)
```

You can also examine the tree using:

```
> m.m5p
```

You can see that the tree has 36 leaf nodes and every leaf node represents a linear regression model.

Model trees *(cont.)*

Let's look at the first leaf node:

```
the alcohol <= 10.85 :
|   volatile.acidity <= 0.238 :
|   |   fixed.acidity <= 6.85 : LM1 (406/66.024%)
```

The node will be reached if (alcohol <= 10.85) AND (volatile.acidity <= 0.238) AND (fixed.acidity <= 6.85). 406 is the number of instances at this leaf node. The percentage is the RMSE for the leaf node model on the leaf node training data divided by the RMSE for the ZeroR model (i.e., the global standard deviation of the target values in the training set).

The leaf is associated with the following regression model:

```
LM num: 1
quality =
        0.266 * fixed.acidity
        - 2.3082 * volatile.acidity
        - 0.012 * citric.acid
        + 0.0421 * residual.sugar
        + 0.1126 * chlorides
        + 0 * free.sulfur.dioxide
        - 0.0015 * total.sulfur.dioxide
        - 109.8813 * density
        + 0.035 * pH
        + 1.4122 * sulphates
        - 0.0046 * alcohol
        + 113.1021
```

In other words, if (alcohol <= 10.85) AND (volatile.acidity <= 0.238) AND (fixed.acidity <= 6.85), then the wine quality will be estimated using this regression formula.

Model trees *(cont.)*

To evaluate the model performance we will make predictions on the test data using the *predict()* function and store the result in a vector *p.m5p*:

```
> p.m5p <- predict(m.m5p, wine_test)
```

Then, we can compare predicted and actual values:

```
> summary(p.m5p)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.389   5.430   5.863   5.874   6.305   7.437
> summary(wine_test$quality)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  3.000   5.000   6.000   5.901   6.000   9.000
```

The model tree appears to be predicting a wider range of values than the regression tree. You can also get a summary of different types of errors:

```
> summary(m.m5p)
=== Summary ===

Correlation coefficient                  0.6666
Mean absolute error                      0.5151
Root mean squared error                  0.6614
Relative absolute error                 76.4921 %
Root relative squared error             74.6259 %
Total Number of Instances             3750
```