Lesson 1.1: Terminology of Linear Regression

Introduction to Linear Models

Required Reading

- Chapter 1

Before proceeding with this lesson, make sure you read though Chapter 1 of your textbook, which lays down the groundwork for the entire course. Amongst the many popular textbooks on linear regression that are currently in publication, your book has been chosen for its expository skills. What it lacks, however, is guidance on how to actually go about implementing a linear regression analysis with appropriate software. The Canvas notes and course activities are designed to give you practice with how to perform *computations* for a linear regression analysis, while the textbook provides explanations on how to *interpret* such computations.

After you have finished with reading this first chapter, you will be tasked with going out and finding your own data set upon which you will be performing a complete regression analysis throughout the semester. As you will be needing to start analysis of this data rather soonish, you should place a top priority on completing that task before progressing with the rest of the lessons in this module.

Models

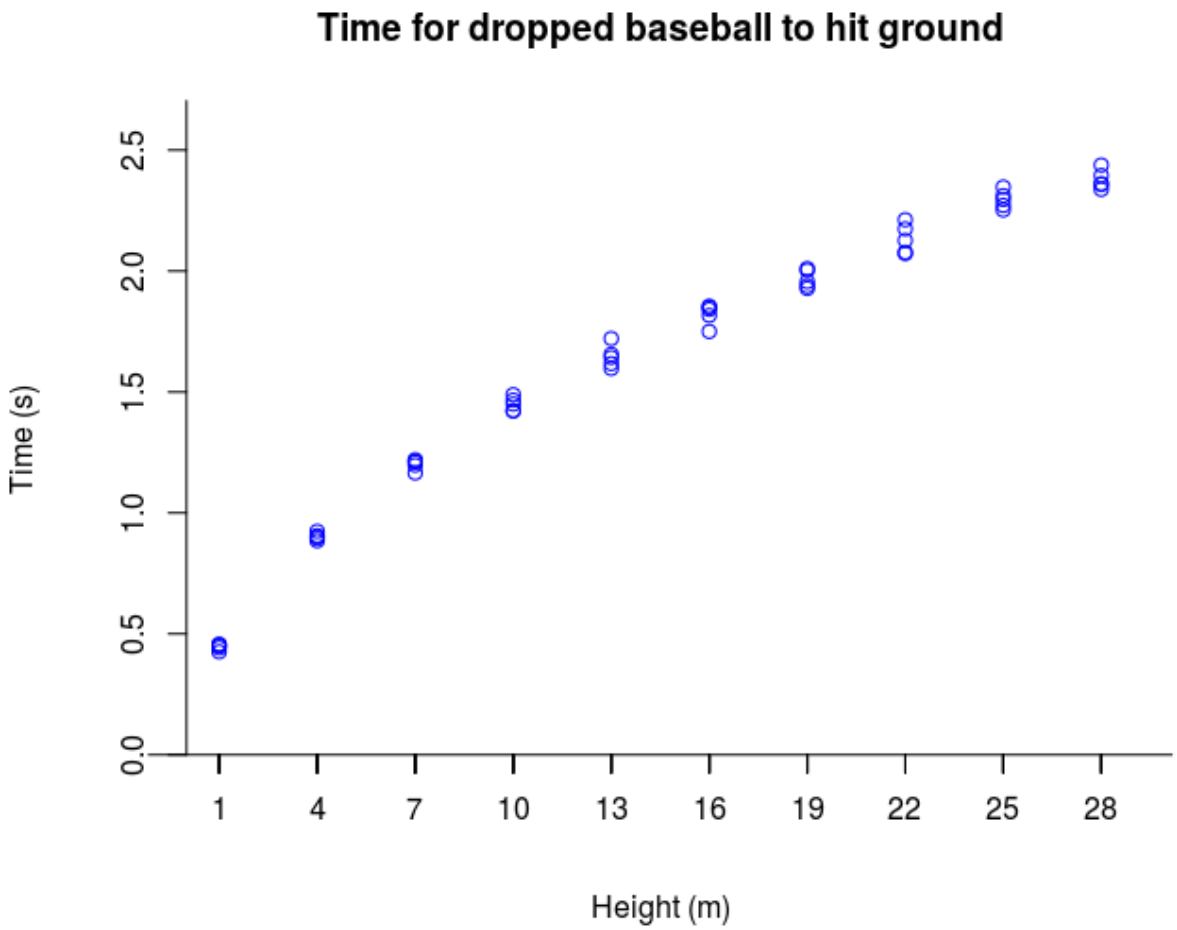
At the heart of what is commonly referred to as *linear regression analysis* is a **linear model**. You should be familiar with many *deterministic* models from various STEM courses you have taken in the past:

- $E=mc^2$, Einstein's famous equation describing the relationship between mass and energy of a particle

- $V=IR$, Ohm's Law which relates voltage, current, and resistance in an electrical circuit
- $F=ma$, Newton's Second Law of Motion relating the force applied to an object to its mass and acceleration

The point of such deterministic models is that they are intended to accurately describe some aspects of reality. This will also be the purpose of the models to be studied in this course, though we will have no lofty ambitions that mysteries of the universe will be unlocked with our investigations. You should note, however, that nearly all of the "laws" that are commonly used in physics and related disciplines really should be referred to as models (albeit, quite excellent models).

Suppose you had no knowledge of Newton's Laws of Motion, and wished to find a relationship between the amount of time it takes an object to fall to the ground, and the height from which that object was dropped. You might conduct an experiment wherein you take a baseball to a tall building, and drop the baseball from varying heights, and record the amount of time it takes for the baseball to hit the ground. Being a thorough and tedious experimenter, you obtain five replicates of your measurements at each of several selected heights, and try to ensure that the heights are accurately measured with each replicate of the experiment. After collecting the data, the points are plotted and illustrated below:



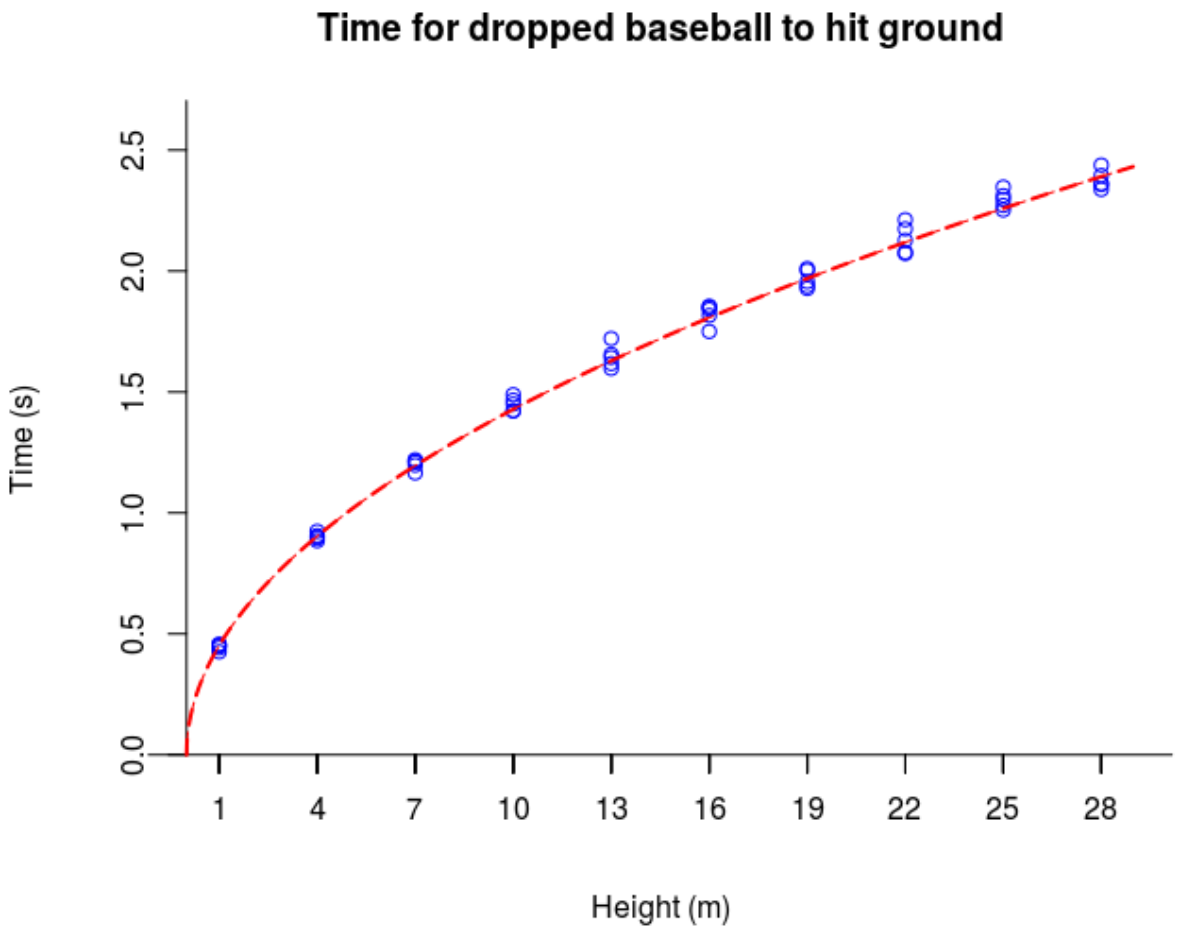
Having a wealth of knowledge concerning mathematical functions, you might suppose that a model of the form

$$t=\beta_0+\beta_1 h_\alpha$$

might describe the relationship between time (t) and height (h). In the terminology of a statistical model, we would call...

- time, t, the **response** variable (or *dependent* variable)
- height, h, an **explanatory** variable (or **predictor**, or *independent variable*)
- the unknown constants β_0, β_1, and α **model coefficients**

Note, however, that the points in the graph do not perfectly fall on any such curve, no matter the choices of coefficients. Rather, the points (h_j,t_j) seem to fall "near" such a curve. Through some means or other, suppose that you have guessed that the functional relationship between t and h takes the form as plotted in the red dashed line below:



With the benefit of hindsight from Newton's Laws, we could say that

$$t=0.452\sqrt{h}$$

which would be a deterministic model. Without that benefit, however, we would start with the **statistical model**

$$t=\beta_0+\beta_1 h_\alpha+\varepsilon$$

where ε is called an **error term**. This quantity is assumed to be a random variable, and is used to account for the deviations of the observed pairs (h_j,t_j) from the deterministic curve (obtained with the benefit of hindsight).

We will have no reason to question Newton's Laws of Motion here, but will take them as established "fact." Nevertheless, if you were to go out and conduct a similar experiment, you would undoubtedly find that your points do not perfectly agree with the well-established Laws of Motion. This *error* would presumably be chalked up to what is commonly called **measurement error**, which is one special instance of something more general called **statistical error**.

- In this situation, the measurement error would likely be due to imprecision and inaccuracies in the measurement of both height and time.

It should be noted that "error" here does not necessarily imply that a mistake was made. It is simply referring to the demonstrable fact that theoretical models cannot perfectly encapsulate observed reality. These deviations of the observed from the theoretical are essentially what is encapsulated by the term "statistical error."

Linear Models

Stepping away from a particular example, a linear model can be expressed as follows:

$$Y=\beta_0+\beta_1X_1+\beta_2X_2+...+\beta_pX_p+\varepsilon$$

Again...

- Y is called the response variable
- $X_1,...,X_p$ are called predictors, or explanatory variables
- ε is called the error term
- the *contsants* $\beta_0,...,\beta_p$ are called the model coefficients

What makes this model linear (as opposed to nonlinear) is the fact that the right-hand side of the above equation connects the predictors solely through *addition* and *scalar multiplication* (multiplication by a constant).

While this form of a relationship between response and predictors may seem quite restrictive, it is actually quite flexible once we realize that Y and/or X_j can be practically anything that we like. The process of tranforming a nonlinear model into a linear one is called *linearization*.

Practice determining if the following models are linear or non-linear by clicking on the correct answers.

Question 1: Is the following model linear or non-linear?

$$Y=\beta_0+\beta_1X_1+\beta_2X_{21}+\beta_3e_{X_2}+\varepsilon$$

Linear
Non-Linear

Question 2: Is the following model linear or non-linear?

$$Y=\beta_0+\beta_1\cos(X_1)+\beta_2\sin(X_1)+\cos(\beta_3X_2)+\varepsilon$$

Linear
Non-Linear

Question 3: Is the following model linear or non-linear?

$$Y=\beta_0+\beta_1 x_{21}+X_{22}\sqrt{}+\beta_3 X_3\sqrt{}+\varepsilon$$

Linear
Non-Linear

Lesson 1.2: Matrix Computations in R

Introduction

Before proceeding further into the course, you will need to familiarize yourself with some of the more basic matrix operations that are found in the R programming language.

If it has been a while since you've done anything with matrices, you should definitely catch up on some basic terminology and operations. One decent and short review of matrices and their operations (addition, multiplication, transposes, inverses, determinants) is found in the document Review of MatricesLinks to an external site.. R code that can be used to replicate some of the calculations in this document will follow.

Creating a Matrix

Consider the matrix A in Example 1.1.1 of the document. There are various routes we can take to create this matrix in R:

- inputting the vector of 12 numbers into the `matrix` function
- creating three row vectors and combining them with the `rbind` function
- creating four column vectors and combining them with the `cbind` function

```
> a <- c(2, -3, 4, 5, 6, 7, 4, 3, 3, 1, 2, 4)
> A <- matrix(a, nrow=3, ncol=4, byrow=TRUE)
> A
     [,1] [,2] [,3] [,4]
[1,]    2   -3    4    5
```

```
[2,]    6    7    4    3
[3,]    3    1    2    4
```

Note the use of the `byrow=TRUE` argument. What matrix would you get if you dropped this argument?

To extract a single element from the matrix A, we can use the extraction operator `[,]` as follows:

```
> A[2,4] # element in 2nd row, 4th column
[1] 3
```

To recover the *dimension* of a matrix, use the `dim` command:

```
> dim(A)
[1] 3 4
```

The nearly universal convention is that the first value in the dimension is the number of rows, and the second is the number of columns, so that we see that A is a matrix of dimension 3×4. Note there are also the functions `nrow` and `ncol` for obtaining just the number of row or columns:

```
> nrow(A)
[1] 3
```

```
> ncol(A)
[1] 4
```

A linear regression analysis begins (computationally, anyhow) with the creation of a *design matrix*, for which the emphasis is on the columns of a matrix. The `cbind` (for *binding columns* of matrices together) will be more useful for us:

```
> col1 <- c(2, 6, 3) # 1st column
> col2 <- c(-3, 7, 1) # 2nd column
> col3 <- c(4, 4, 2) # 3rd column
> col4 <- c(5, 3, 4) # 4th column
> A2 <- cbind(col1, col2, col3, col4)
> A2
     col1 col2 col3 col4
[1,]    2   -3    4    5
[2,]    6    7    4    3
[3,]    3    1    2    4
```

Diagonal Matrices and Diagonal Elements

You can readily create a diagonal matrix with the `diag` function, supplying a vector containing the diagonal entries. For instance, to create the matrix E in Example 1.1.2:

```
> E <- diag(c(2, -3, -6))
> E
     [,1] [,2] [,3]
[1,]    2    0    0
[2,]    0   -3    0
[3,]    0    0   -6
```

This `diag` function can actually serve a couple of purposes:

- to *create* a new diagonal matrix, in conjunction with the assignment operator `<-` (as was done above)
- to *extract* the diagonal entries from a matrix that has already been created

An implementation of this second purpose follows:

```
> D <- cbind(c(1,3), c(2, -4))
> D
     [,1] [,2]
[1,]    1    2
[2,]    3   -4
> diag(D)
[1]  1 -4
```

The above creates the matrix D (still in Example 1.1.2), then obtains the set of diagonal elements (namely, $d_{1,1}=1$ and $d_{2,2}=-4$) from it with the `diag` function.

Matrix Operations in R

Click each item to read more.

Scalar Multiplication

Matrix Addition

Matrix Multiplication

Matrix Transpose

Cross Products

Matrix Inverses and Solving Matrix Equations

For a 2×2 matrix B, the *determinant* is calculated as $b_{11}b_{22}-b_{12}b_{21}$. We will have relatively little use for calculating the *determinant* of a square matrix, but should the need ever arise this operation is implemented with the `det` function:

```
> det(B)
```

```
[1] -7
```

Rather, an important task we need to be able to perform with matrices is calculating the *inverse* of a square matrix (when it exists) and using this to solve matrix equations.

Given a square matrix A, the inverse of A is denoted by A^{-1}, and is the unique matrix that satisfies the equations $A \cdot A^{-1} = I = A^{-1} \cdot A$ where I denotes the identity matrix. Not all square matrices have an inverse:

- a matrix whose inverse exists is said to be *nonsingular*
- if A^{-1} doesn't exist, A is said to be *singular*

The `solve` function in R can be used to calculate the inverse of a matrix.

Consider Example 1.4.1:

```
> A <- matrix(c(2,4,3,0), nrow=2)
> A
     [,1] [,2]
[1,]    2    3
[2,]    4    0
> A_inv <- solve(A)
> A_inv
          [,1]      [,2]
[1,] 0.0000000  0.2500000
```

```
[2,] 0.3333333 -0.1666667
```

The matrix `A_inv` above is A^{-1}, as evidenced by calculating $A \cdot A^{-1}$ or $A^{-1} \cdot A$:

```
> A %*% A_inv
     [,1] [,2]
[1,]    1    0
[2,]    0    1
> A_inv %*% A
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

What happens if we ask R to calculate the inverse of a singular matrix?

```
> B <- matrix(c(2, 4, 1, 2), nrow=2)
> B
     [,1] [,2]
[1,]    2    1
[2,]    4    2
> det(B)
[1] 0
```

The matrix B above is known to be singular as $\det(B)=0$. Trying to calculate B^{-1} (which doesn't exist) in R results in an error:

```
> solve(B)
```

```
Error in solve.default(B): Lapack routine dgesv: system is exactly singular
: U[2,2] = 0
```

The `solve` function also has one more use in solving a particular kind of matrix equation. Consider $AX=B$ where A and B are *known* matrices and X is the matrix we wish to determine. If A^{-1} exists, then we have $AX=B \Rightarrow\Rightarrow\Rightarrow A^{-1}AX=A^{-1}BIX=A^{-1}BX=A^{-1}B$

The command `solve(A, B)` results in the solution to the above equation, namely $A^{-1}B$. There are all kinds of other matrix equations which you might be interested in solving in various applications, but in this course solutions to $AX=B$ will be the primary ones equations we wish to solve.

Consider Exercise 2(a) in Section 1.5.2 of the linked document.

```
> A <- matrix(c(-1,-3,2,1), nrow=2)
> B <- matrix(c(9, -7, 2, 4), nrow=2)
> A
     [,1] [,2]
[1,]   -1    2
[2,]   -3    1
> B
     [,1] [,2]
[1,]    9    2
[2,]   -7    4
> Y <- solve(A,B)
```

```
> Y
     [,1] [,2]
[1,]  4.6 -1.2
[2,]  6.8  0.4
```

Let's verify that $AY=B$:

```
> A %*% Y
     [,1] [,2]
[1,]    9    2
[2,]   -7    4
> B
     [,1] [,2]
[1,]    9    2
[2,]   -7    4
```

Exercises

Brush up on your knowledge of matrix multiplication and transposes. You won't be performing such calculations by hand, but should at least be aware of when an operation is permitted and what the dimensions of the resulting product will be.

Suppose you have the following information about some generic matrices:

- A is of dimension 6×20

- B is of dimension 20×4
- C is of dimension 4×6
- D is of dimension 20×1
- E is of dimension 1×1

Answer the following questions. Click 'Check' to check answers and click 'Next' to advance.

Lesson 1.3: Covariance and Correlation

- **Introduction**
- **Scatter Plots**
- **Covariance and Correlation**

Introduction

Required Reading

- Sections 2.1-2.3

As the material in the second chapter of your textbook is mostly review, little effort will be spent on these Canvas notes in regards to explaining concepts, but rather we will focus on the R commands needed to perform a simple linear

regression and how these computations can be performed using matrix algebra.

You will want to have a session of R open while reading through the Canvas lecture notes and recreate the provided code on your own. Note that you will be presumed to be comfortable with the basics of the R programming language, so little explanation will be provided in regards to R functions and operations that you have previously encountered.

To start off, you should start a new R program, download the data file `computer.repairs.csv` Download computer.repairs.csvand import this file into a data frame:

```
> repairs <- read.csv('computer.repairs.csv')
```

Recall that the `str` and `View` functions should typically be used to examine the *structure* of a data frame and its contents.

```
> str(repairs)
'data.frame':   14 obs. of  2 variables:
 $ Minutes: int  23 29 49 64 74 87 96 97 109 119 ...
 $ Units  : int  1 2 3 4 4 5 6 6 7 8 ...
> View(repairs)
```

We see there are two variables, `Minutes` and `Units`, that are observed on a total of 14 observations. The `View` function provides a nice "spreadsheet-like" display of the data (though this may not be overly useful if there are too many variables or too many observations).

When trying to describe the relationship between two (or even more) variables it is helpful to have both a visual display as well as some summary statistics. Both of these methods of summarizing data are indispensable.

- While data displays like a scatter plot (you will be investigating many more plots throughout the course) are invaluable tools, they also require a rather subjective interpretation on the part of the viewer.
- Numerical summary statistics such as the covariance or correlation (again, there will be many more to be investigated) are valuable in providing an air of "objectivity" to an analysis, but they might also mask important features of a data set that might only be apparent with a visual display.

Use of `attach` Function

For those of you who have taken MA5701 previously, you may recall that the `attach` function was used copiously to extract variables from a data frame.

I must apologize to you if you became overly reliant on this function, as its use is generally frowned upon as emblematic of bad programming practice.

There are a few more acceptable ways to perform operations on variables contained with an R data frame:

- use the `$` extraction operator, such as `mean(repairs$Minutes)`
- whenever a function takes an R formula as its first argument, the second argument will typically be the name of the data frame, such as `plot(Minutes~Units, repairs)`
- you can use the `with` function to create a local environment in which expressions can be evaluated

As an example of this last function, note that

```
> mean(Minutes)
Error in mean(Minutes): object 'Minutes' not found
```

produces an error, whereas

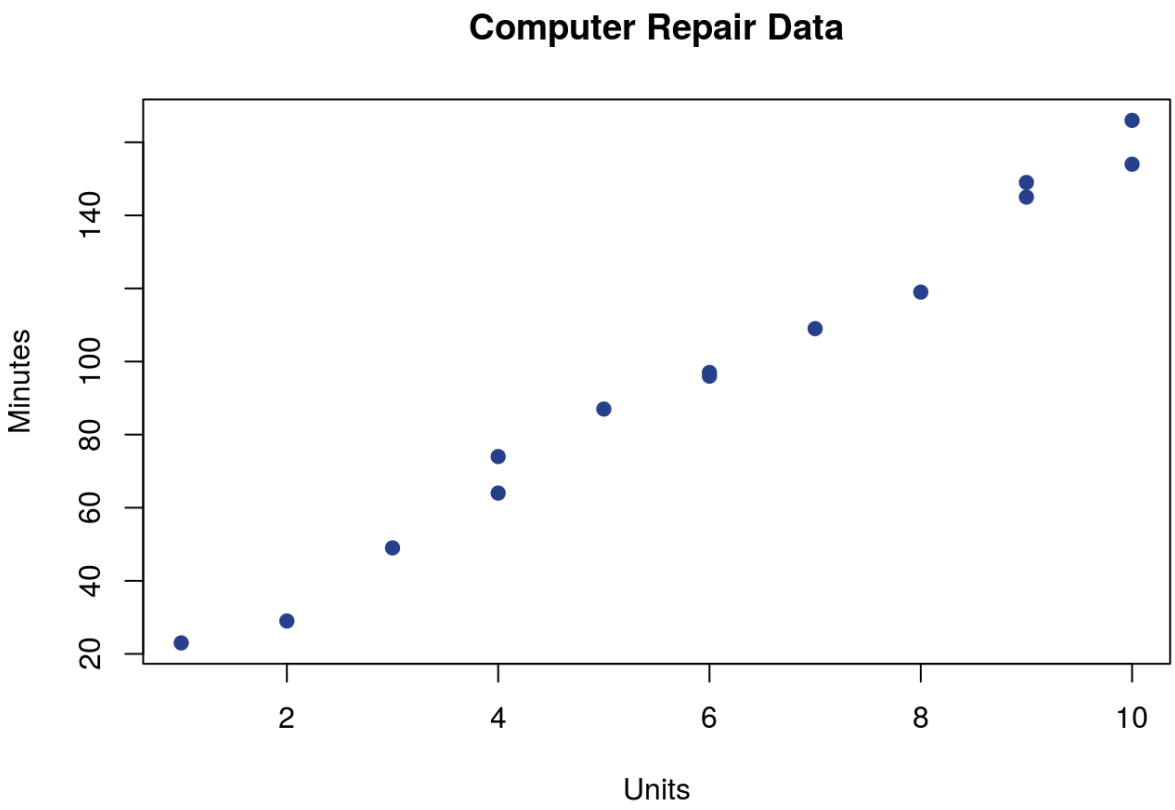
```
> with(repairs, mean(Minutes))
[1] 97.21429
```

does not. It will ultimately be your choice as to whether or not you use the `attach` function, but note that you may have already run into some of the problems associated with using it. Namely, changes to a variable outside of the data frame are not made within the data frame.

Scatter Plots

An appropriate visual display, such as a **scatter plot** of two variables, has the power of making evident at one quick glance whether or not there exists a relationship between two variables, and moreover gives the audience an idea as to how to begin describing the relationship. With the example at hand, we view the amount of time that it takes to finish a service call as being "*dependent*" on the number of units that need to be repaired. This variable should go on the y-axis, with the "*independent*" variable (the number of units in need of repair) lying along the x-axis.

```
> plot(Minutes~Units, repairs, main='Computer Repair Data', pch=19, col='ro
yalblue4')
```



You will not be required to "pretty" your graphics overly much in this course, but should you be so inclined...

- the `points` help page in R provides a list of the various plotting symbols you can choose from; circles are probably the most conventional route, but there are several other shapes from which to choose
    - specify your choice of plotting symbol with the `pch` argument

- the `col` argument can be used to specify the color of the plotting symbol; try out the `colors()` function to get a list of all of the named colors in R
- a more advanced package, called `ggplot2`, is probably the preferred route amongst professional statisticians or data scientists to create customizable plots in R
    - this pakage is part of the *tidyverse*Links to an external site. in R, which is a collection of packages aimed at supplying a consistent "grammar" to data management and displays
    - you are encouraged to look into learning how to effectively use the tidyverse *on your own time* (which may be scarce while enrolled in the MSAS program!), but this course will focus on using the plotting functions from the base package of R

This scatter plot illustrates a fairly strong *linear* relationship between the number of units that need to be repaired and the amount of time spent on a service call. Moreover, we would say there is a *positive* correlation between these two variables:

- the more units that need to be repaired, the more time is required for the service call

Covariance and Correlation

The **sample covariance** between two numerical variables is defined as

$$\text{Cov}(x,y) = \frac{1}{n-1}\sum_{j=1}^{n}(x_j - \bar{x})(y_j - \bar{y})$$

and can be used as a means of measuring the *strength* of a linear relationship between those variables. While this is a useful statistic, its interpretation is ultimately dependent on the units that are attached to the two variables.

Consider the covariance between `Minutes` and `Units`:

```
> with(repairs, cov(Minutes, Units))
[1] 136
```

The units on this quantity are "minute·parts". What if the amount of time were measured in hours instead?

```
> repairs$Hours <- repairs$Minutes / 60
> with(repairs, cov(Hours, Units))
[1] 2.266667
```

This new covariance has units of "hour·parts".

- Note that the covariance between minutes and units is 60 times that of the covariance between hours and units
- In general, $\text{Cov}(ax,by)=ab\cdot\text{Cov}(x,y)$ for any constants a and b, so that a scale change in one (or both) variable(s) correspondingly scales the covariance

To make interpretation easier, and independent of any arbitrary choice of units, the **sample correlation** is much nicer:

$$\text{Cor}(x,y)=\text{Cov}(x,y)s_x s_y=\text{Cov}(x,y)\text{Cov}(x,x)\sqrt{}\cdot\text{Cov}(y,y)\sqrt{}$$

where s_x and s_y are the sample standard deviations of x and y, respectively. The second equation comes from the fact that the covariance of a variable with itself is simply the sample variance:

$$s^2_x=\text{Var}(x)=\frac{1}{n-1}\sum_{j=1}^{n}(x_j-\bar{x})^2=\frac{1}{n-1}\sum_{j=1}^{n}(x_j-\bar{x})(x_j-\bar{x})=\text{Cov}(x,x)$$

```
> with(repairs, cov(Minutes, Minutes)) # covariance of minutes with itself
[1] 2136.027
> var(repairs$Minutes) # same as the sample variance of minutes
[1] 2136.027
```

In rescaling the sample covariance by dividing by the standard deviations of the two variables, the sample correlation is now a *unitless* quantity. Any change of the units on the original variables has no effect on the sample correlation:

```
> with(repairs, cor(Minutes, Units))
[1] 0.9936987
> with(repairs, cor(Hours, Units)) # correlation is unchanged by a change i
n units
[1] 0.9936987
```

The correlation is nice in that it always takes values between -1 and 1, providing us with a *rough* means of gauging the strength of the linear relationship between two variables.

- as $|\text{Cor}(x,y)|$ becomes closer to 1, the points in a scatter plot tend to fall closer to a straight line
- the sign of $\text{Cor}(x,y)$ indicates the *direction* of the correlation (or association) between x and y

A Word of Caution

You must always be wary of relying solely upon the sample correlation to judge the strength of a *linear* relationship between two variables! Your textbook discusses a few examples of data sets where...

- there is a "strong" correlation as measured by $\text{Cor}(x,y)$, while a scatter plot of the data suggests a highly nonlinear relationship

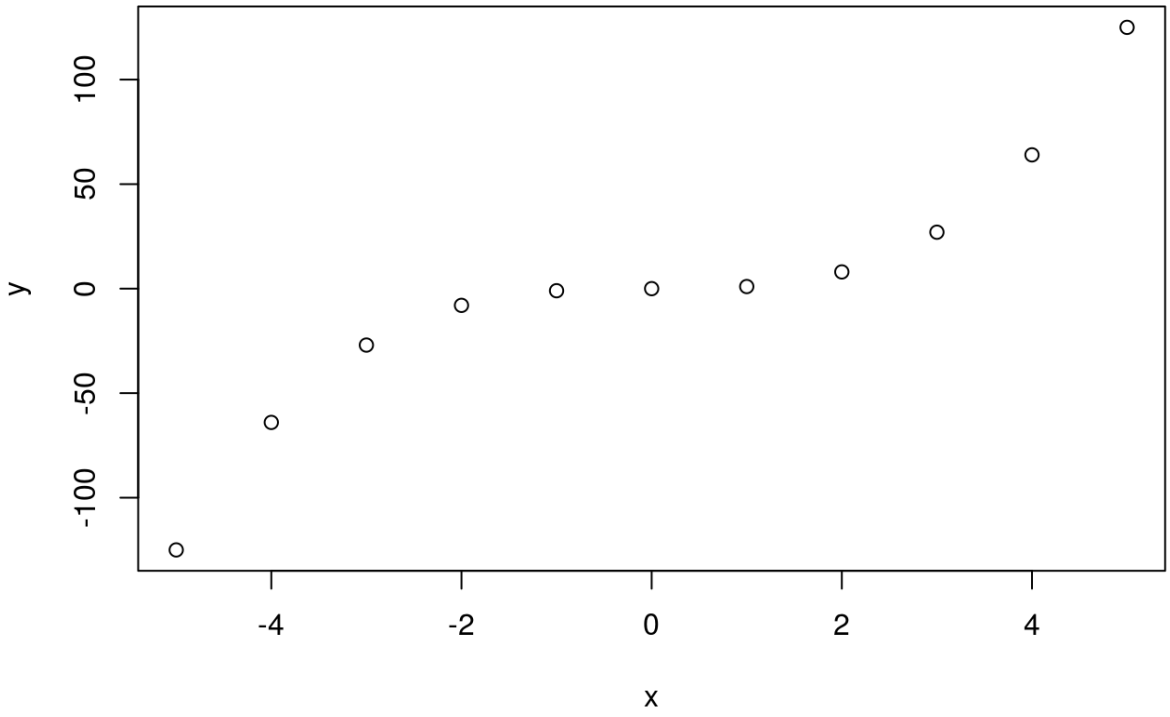
    - three of the datasets in Anscombe's quartet exhibited this behavior
- there was a "weak" correlation as measured by Cor(x,y), while a scatter plot of the data suggests a very strong *functional* (but nonlinear) relationship
    - the data in Table 2.3 and Figure 2.2 were examples of this.

Let's try out a few more toy examples.

```
> x <- -5:5
> y <- x^3
> cor(x, y)
[1] 0.9216491
```

The sample correlation between `x` and `y` is very large (≈0.92), and yet there is a *cubic* (not linear) relationship between these two variables:

```
> plot(y~x)
```



Try this out with a few other toy examples. Create the two variables as indicated, then calculate the sample correlation between the two variables (round your answer to the nearest 0.001). Produce a scatter plot of the variables as well.

Enter your answers in the spaces provide and click 'Check,' then click 'Next' to advance to the next problem.

Lesson 1.3: Covariance and Correlation

- **Introduction**

- **Scatter Plots**
- **Covariance and Correlation**

Introduction

Required Reading

- Sections 2.1-2.3

As the material in the second chapter of your textbook is mostly review, little effort will be spent on these Canvas notes in regards to explaining concepts, but rather we will focus on the R commands needed to perform a simple linear regression and how these computations can be performed using matrix algebra.

You will want to have a session of R open while reading through the Canvas lecture notes and recreate the provided code on your own. Note that you will be presumed to be comfortable with the basics of the R programming language, so little explanation will be provided in regards to R functions and operations that you have previously encountered.

To start off, you should start a new R program, download the data file `computer.repairs.csv` Download computer.repairs.csvand import this file into a data frame:

```
> repairs <- read.csv('computer.repairs.csv')
```

Recall that the `str` and `View` functions should typically be used to examine the *structure* of a data frame and its contents.

```
> str(repairs)
'data.frame':   14 obs. of  2 variables:
 $ Minutes: int  23 29 49 64 74 87 96 97 109 119 ...
 $ Units  : int  1 2 3 4 4 5 6 6 7 8 ...
> View(repairs)
```

We see there are two variables, `Minutes` and `Units`, that are observed on a total of 14 observations. The `View` function provides a nice "spreadsheet-like" display of the data (though this may not be overly useful if there are too many variables or too many observations).

When trying to describe the relationship between two (or even more) variables it is helpful to have both a visual display as well as some summary statistics. Both of these methods of summarizing data are indispensable.

- While data displays like a scatter plot (you will be investigating many more plots throughout the course) are invaluable tools, they also require a rather subjective interpretation on the part of the viewer.
- Numerical summary statistics such as the covariance or correlation (again, there will be many more to be investigated) are valuable in providing an air of "objectivity" to an analysis, but they might also mask important features of a data set that might only be apparent with a visual display.

Use of `attach` Function

For those of you who have taken MA5701 previously, you may recall that the `attach` function was used copiously to extract variables from a data frame.

I must apologize to you if you became overly reliant on this function, as its use is generally frowned upon as emblematic of bad programming practice.

There are a few more acceptable ways to perform operations on variables contained with an R data frame:

- use the `$` extraction operator, such as `mean(repairs$Minutes)`

- whenever a function takes an R formula as its first argument, the second argument will typically be the name of the data frame, such as `plot(Minutes~Units, repairs)`
- you can use the `with` function to create a local environment in which expressions can be evaluated

As an example of this last function, note that

```
> mean(Minutes)
Error in mean(Minutes): object 'Minutes' not found
```

produces an error, whereas

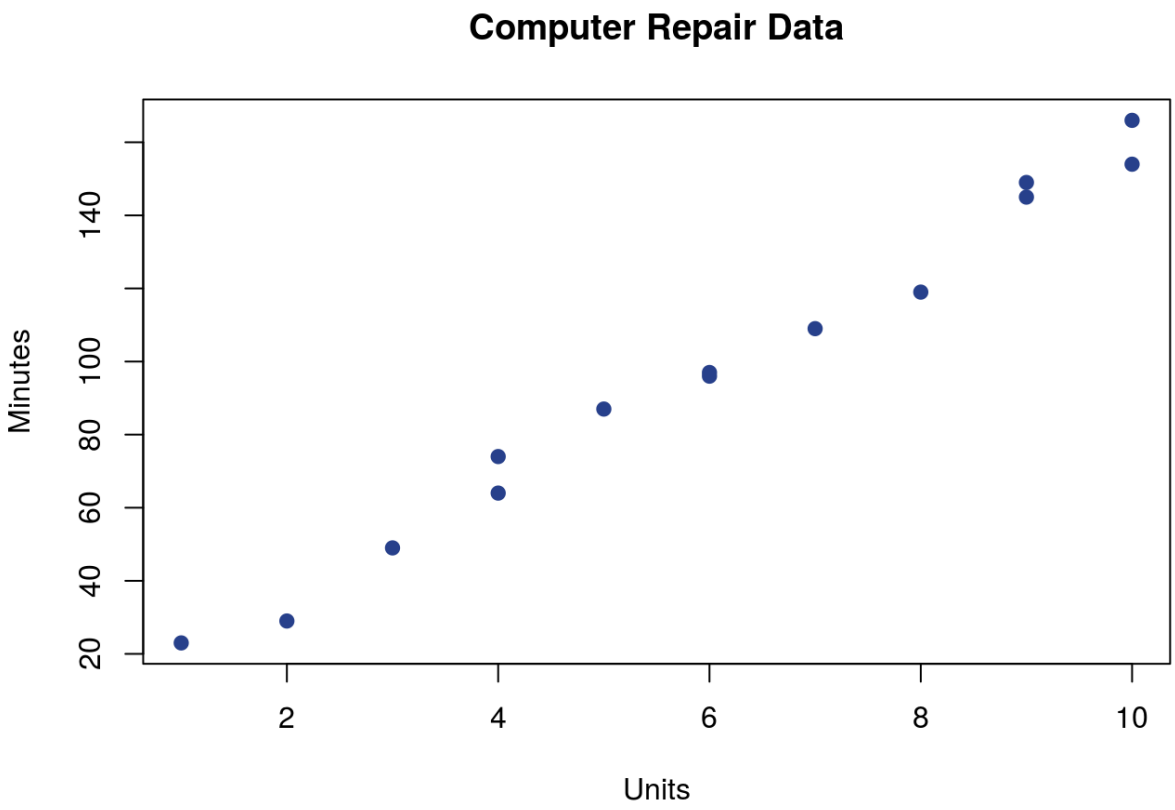
```
> with(repairs, mean(Minutes))
[1] 97.21429
```

does not. It will ultimately be your choice as to whether or not you use the `attach` function, but note that you may have already run into some of the problems associated with using it. Namely, changes to a variable outside of the data frame are not made within the data frame.

Scatter Plots

An appropriate visual display, such as a **scatter plot** of two variables, has the power of making evident at one quick glance whether or not there exists a relationship between two variables, and moreover gives the audience an idea as to how to begin describing the relationship. With the example at hand, we view the amount of time that it takes to finish a service call as being "*dependent*" on the number of units that need to be repaired. This variable should go on the y-axis, with the "*independent*" variable (the number of units in need of repair) lying along the x-axis.

```
> plot(Minutes~Units, repairs, main='Computer Repair Data', pch=19, col='ro
yalblue4')
```



You will not be required to "pretty" your graphics overly much in this course, but should you be so inclined...

- the `points` help page in R provides a list of the various plotting symbols you can choose from; circles are probably the most conventional route, but there are several other shapes from which to choose
    - specify your choice of plotting symbol with the `pch` argument

- the `col` argument can be used to specify the color of the plotting symbol; try out the `colors()` function to get a list of all of the named colors in R
- a more advanced package, called `ggplot2`, is probably the preferred route amongst professional statisticians or data scientists to create customizable plots in R
    - this pakage is part of the *tidyverse*Links to an external site. in R, which is a collection of packages aimed at supplying a consistent "grammar" to data management and displays
    - you are encouraged to look into learning how to effectively use the tidyverse *on your own time* (which may be scarce while enrolled in the MSAS program!), but this course will focus on using the plotting functions from the base package of R

This scatter plot illustrates a fairly strong *linear* relationship between the number of units that need to be repaired and the amount of time spent on a service call. Moreover, we would say there is a *positive* correlation between these two variables:

- the more units that need to be repaired, the more time is required for the service call

Covariance and Correlation

The **sample covariance** between two numerical variables is defined as

$$\text{Cov}(x,y) = \frac{1}{n-1}\sum_{j=1}^{n}(x_j - \bar{x})(y_j - \bar{y})$$

and can be used as a means of measuring the *strength* of a linear relationship between those variables. While this is a useful statistic, its interpretation is ultimately dependent on the units that are attached to the two variables.

Consider the covariance between `Minutes` and `Units`:

```
> with(repairs, cov(Minutes, Units))
[1] 136
```

The units on this quantity are "minute·parts". What if the amount of time were measured in hours instead?

```
> repairs$Hours <- repairs$Minutes / 60
> with(repairs, cov(Hours, Units))
[1] 2.266667
```

This new covariance has units of "hour·parts".

- Note that the covariance between minutes and units is 60 times that of the covariance between hours and units
- In general, $\text{Cov}(ax,by)=ab\cdot\text{Cov}(x,y)$ for any constants a and b, so that a scale change in one (or both) variable(s) correspondingly scales the covariance

To make interpretation easier, and independent of any arbitrary choice of units, the **sample correlation** is much nicer:

$$\text{Cor}(x,y)=\text{Cov}(x,y)s_x s_y=\text{Cov}(x,y)\text{Cov}(x,x)\sqrt{}\cdot\text{Cov}(y,y)\sqrt{}$$

where s_x and s_y are the sample standard deviations of x and y, respectively. The second equation comes from the fact that the covariance of a variable with itself is simply the sample variance:

$$s^2_x=\text{Var}(x)=\frac{1}{n-1}\sum_{j=1}^{n}(x_j-\bar{x})^2=\frac{1}{n-1}\sum_{j=1}^{n}(x_j-\bar{x})(x_j-\bar{x})=\text{Cov}(x,x)$$

```
> with(repairs, cov(Minutes, Minutes)) # covariance of minutes with itself
[1] 2136.027
> var(repairs$Minutes) # same as the sample variance of minutes
[1] 2136.027
```

In rescaling the sample covariance by dividing by the standard deviations of the two variables, the sample correlation is now a *unitless* quantity. Any change of the units on the original variables has no effect on the sample correlation:

```
> with(repairs, cor(Minutes, Units))
[1] 0.9936987
> with(repairs, cor(Hours, Units)) # correlation is unchanged by a change i
n units
[1] 0.9936987
```

The correlation is nice in that it always takes values between -1 and 1, providing us with a *rough* means of gauging the strength of the linear relationship between two variables.

- as $|\text{Cor}(x,y)|$ becomes closer to 1, the points in a scatter plot tend to fall closer to a straight line
- the sign of $\text{Cor}(x,y)$ indicates the *direction* of the correlation (or association) between x and y

A Word of Caution

You must always be wary of relying solely upon the sample correlation to judge the strength of a *linear* relationship between two variables! Your textbook discusses a few examples of data sets where...

- there is a "strong" correlation as measured by $\text{Cor}(x,y)$, while a scatter plot of the data suggests a highly nonlinear relationship

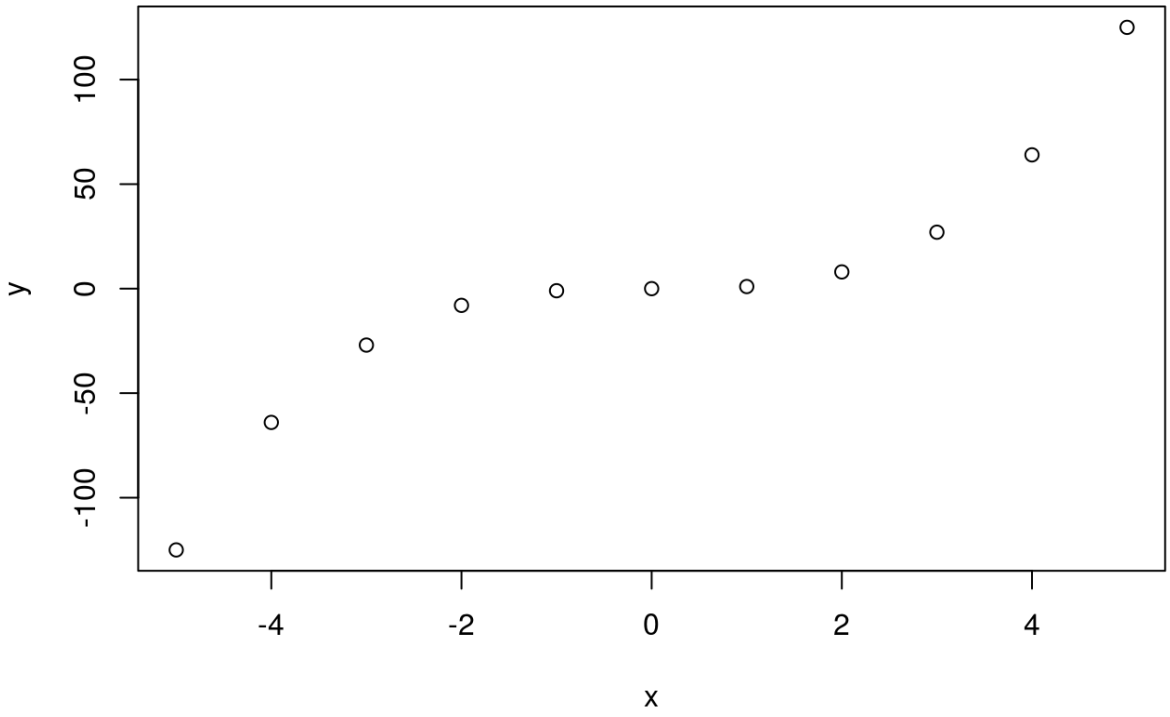
    - three of the datasets in Anscombe's quartet exhibited this behavior
- there was a "weak" correlation as measured by Cor(x,y), while a scatter plot of the data suggests a very strong *functional* (but nonlinear) relationship
    - the data in Table 2.3 and Figure 2.2 were examples of this.

Let's try out a few more toy examples.

```
> x <- -5:5
> y <- x^3
> cor(x, y)
[1] 0.9216491
```

The sample correlation between `x` and `y` is very large (≈0.92), and yet there is a *cubic* (not linear) relationship between these two variables:

```
> plot(y~x)
```



Try this out with a few other toy examples. Create the two variables as indicated, then calculate the sample correlation between the two variables (round your answer to the nearest 0.001). Produce a scatter plot of the variables as well.

Enter your answers in the spaces provide and click 'Check,' then click 'Next' to advance to the next problem.

Lesson 1.3: Covariance and Correlation

- **Introduction**

- **Scatter Plots**
- **Covariance and Correlation**

Introduction

Required Reading

- Sections 2.1-2.3

As the material in the second chapter of your textbook is mostly review, little effort will be spent on these Canvas notes in regards to explaining concepts, but rather we will focus on the R commands needed to perform a simple linear regression and how these computations can be performed using matrix algebra.

You will want to have a session of R open while reading through the Canvas lecture notes and recreate the provided code on your own. Note that you will be presumed to be comfortable with the basics of the R programming language, so little explanation will be provided in regards to R functions and operations that you have previously encountered.

To start off, you should start a new R program, download the data file `computer.repairs.csv` Download computer.repairs.csvand import this file into a data frame:

```
> repairs <- read.csv('computer.repairs.csv')
```

Recall that the `str` and `View` functions should typically be used to examine the *structure* of a data frame and its contents.

```
> str(repairs)
'data.frame':   14 obs. of  2 variables:
 $ Minutes: int  23 29 49 64 74 87 96 97 109 119 ...
 $ Units  : int  1 2 3 4 4 5 6 6 7 8 ...
> View(repairs)
```

We see there are two variables, `Minutes` and `Units`, that are observed on a total of 14 observations. The `View` function provides a nice "spreadsheet-like" display of the data (though this may not be overly useful if there are too many variables or too many observations).

When trying to describe the relationship between two (or even more) variables it is helpful to have both a visual display as well as some summary statistics. Both of these methods of summarizing data are indispensable.

- While data displays like a scatter plot (you will be investigating many more plots throughout the course) are invaluable tools, they also require a rather subjective interpretation on the part of the viewer.
- Numerical summary statistics such as the covariance or correlation (again, there will be many more to be investigated) are valuable in providing an air of "objectivity" to an analysis, but they might also mask important features of a data set that might only be apparent with a visual display.

Use of `attach` Function

For those of you who have taken MA5701 previously, you may recall that the `attach` function was used copiously to extract variables from a data frame.

I must apologize to you if you became overly reliant on this function, as its use is generally frowned upon as emblematic of bad programming practice.

There are a few more acceptable ways to perform operations on variables contained with an R data frame:

- use the `$` extraction operator, such as `mean(repairs$Minutes)`

- whenever a function takes an R formula as its first argument, the second argument will typically be the name of the data frame, such as `plot(Minutes~Units, repairs)`
- you can use the `with` function to create a local environment in which expressions can be evaluated

As an example of this last function, note that

```
> mean(Minutes)
Error in mean(Minutes): object 'Minutes' not found
```

produces an error, whereas

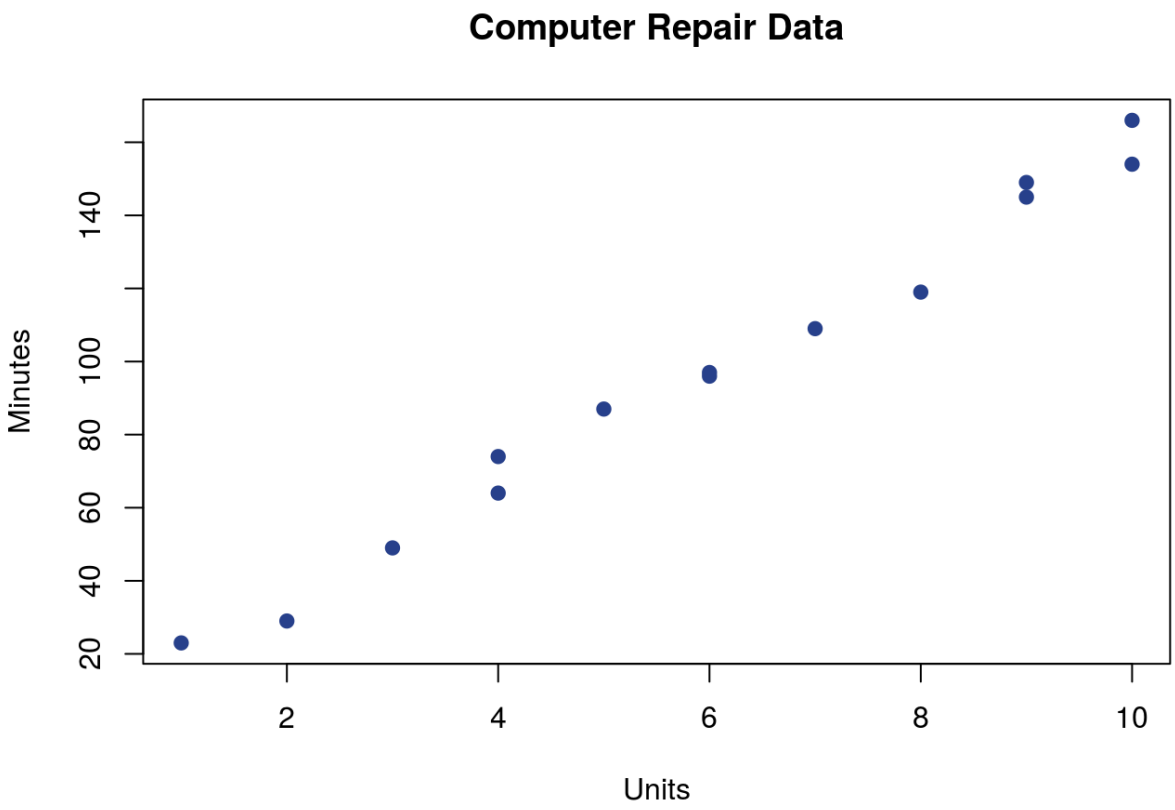
```
> with(repairs, mean(Minutes))
[1] 97.21429
```

does not. It will ultimately be your choice as to whether or not you use the `attach` function, but note that you may have already run into some of the problems associated with using it. Namely, changes to a variable outside of the data frame are not made within the data frame.

Scatter Plots

An appropriate visual display, such as a **scatter plot** of two variables, has the power of making evident at one quick glance whether or not there exists a relationship between two variables, and moreover gives the audience an idea as to how to begin describing the relationship. With the example at hand, we view the amount of time that it takes to finish a service call as being "*dependent*" on the number of units that need to be repaired. This variable should go on the y-axis, with the "*independent*" variable (the number of units in need of repair) lying along the x-axis.

```
> plot(Minutes~Units, repairs, main='Computer Repair Data', pch=19, col='ro
yalblue4')
```



You will not be required to "pretty" your graphics overly much in this course, but should you be so inclined...

- the `points` help page in R provides a list of the various plotting symbols you can choose from; circles are probably the most conventional route, but there are several other shapes from which to choose
    - specify your choice of plotting symbol with the `pch` argument

- the `col` argument can be used to specify the color of the plotting symbol; try out the `colors()` function to get a list of all of the named colors in R
- a more advanced package, called `ggplot2`, is probably the preferred route amongst professional statisticians or data scientists to create customizable plots in R
    - this pakage is part of the *tidyverse*Links to an external site. in R, which is a collection of packages aimed at supplying a consistent "grammar" to data management and displays
    - you are encouraged to look into learning how to effectively use the tidyverse *on your own time* (which may be scarce while enrolled in the MSAS program!), but this course will focus on using the plotting functions from the base package of R

This scatter plot illustrates a fairly strong *linear* relationship between the number of units that need to be repaired and the amount of time spent on a service call. Moreover, we would say there is a *positive* correlation between these two variables:

- the more units that need to be repaired, the more time is required for the service call

Covariance and Correlation

The **sample covariance** between two numerical variables is defined as

$$\text{Cov}(x,y) = \frac{1}{n-1}\sum_{j=1}^{n}(x_j - \bar{x})(y_j - \bar{y})$$

and can be used as a means of measuring the *strength* of a linear relationship between those variables. While this is a useful statistic, its interpretation is ultimately dependent on the units that are attached to the two variables.

Consider the covariance between `Minutes` and `Units`:

```
> with(repairs, cov(Minutes, Units))
[1] 136
```

The units on this quantity are "minute·parts". What if the amount of time were measured in hours instead?

```
> repairs$Hours <- repairs$Minutes / 60
> with(repairs, cov(Hours, Units))
[1] 2.266667
```

This new covariance has units of "hour·parts".

- Note that the covariance between minutes and units is 60 times that of the covariance between hours and units
- In general, $\text{Cov}(ax,by)=ab\cdot\text{Cov}(x,y)$ for any constants a and b, so that a scale change in one (or both) variable(s) correspondingly scales the covariance

To make interpretation easier, and independent of any arbitrary choice of units, the **sample correlation** is much nicer:

$$\text{Cor}(x,y)=\text{Cov}(x,y)s_x s_y=\text{Cov}(x,y)\text{Cov}(x,x)\sqrt{}\cdot\text{Cov}(y,y)\sqrt{}$$

where s_x and s_y are the sample standard deviations of x and y, respectively. The second equation comes from the fact that the covariance of a variable with itself is simply the sample variance:

$$s^2_x=\text{Var}(x)=\frac{1}{n-1}\sum_{j=1}^{n}(x_j-\bar{x})^2=\frac{1}{n-1}\sum_{j=1}^{n}(x_j-\bar{x})(x_j-\bar{x})=\text{Cov}(x,x)$$

```
> with(repairs, cov(Minutes, Minutes)) # covariance of minutes with itself
[1] 2136.027
> var(repairs$Minutes) # same as the sample variance of minutes
[1] 2136.027
```

In rescaling the sample covariance by dividing by the standard deviations of the two variables, the sample correlation is now a *unitless* quantity. Any change of the units on the original variables has no effect on the sample correlation:

```
> with(repairs, cor(Minutes, Units))
[1] 0.9936987
> with(repairs, cor(Hours, Units)) # correlation is unchanged by a change i
n units
[1] 0.9936987
```

The correlation is nice in that it always takes values between -1 and 1, providing us with a *rough* means of gauging the strength of the linear relationship between two variables.

- as $|\text{Cor}(x,y)|$ becomes closer to 1, the points in a scatter plot tend to fall closer to a straight line
- the sign of $\text{Cor}(x,y)$ indicates the *direction* of the correlation (or association) between x and y

A Word of Caution

You must always be wary of relying solely upon the sample correlation to judge the strength of a *linear* relationship between two variables! Your textbook discusses a few examples of data sets where...

- there is a "strong" correlation as measured by $\text{Cor}(x,y)$, while a scatter plot of the data suggests a highly nonlinear relationship

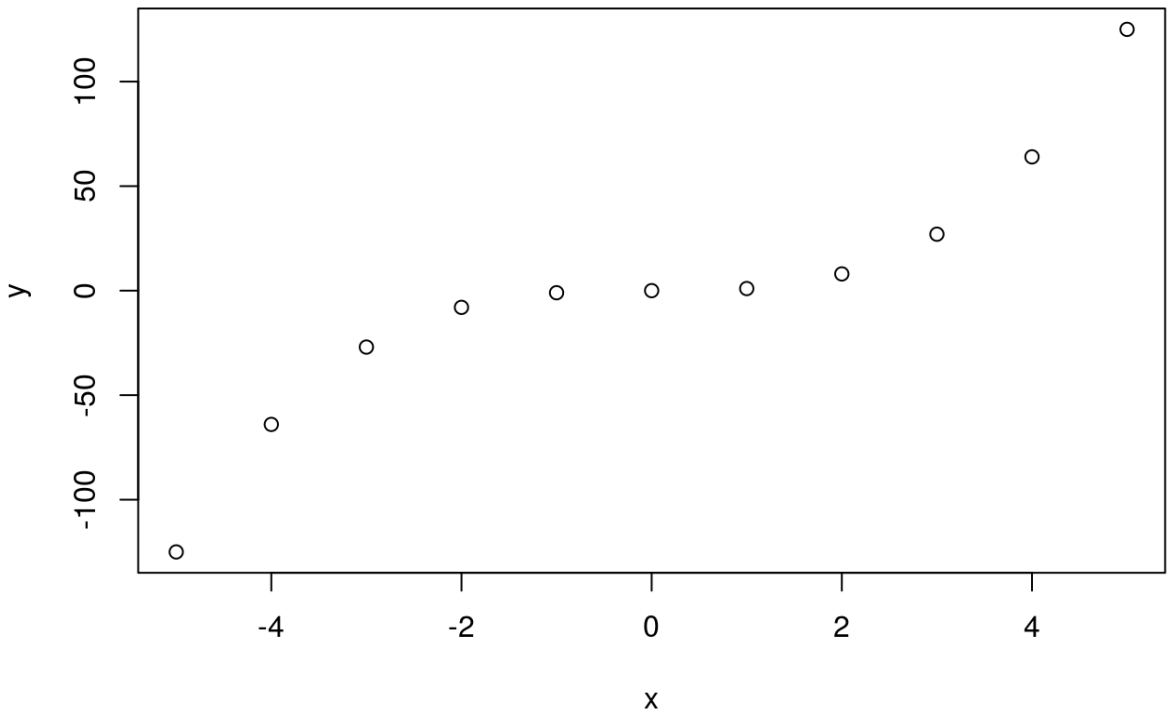
    - three of the datasets in Anscombe's quartet exhibited this behavior
- there was a "weak" correlation as measured by Cor(x,y), while a scatter plot of the data suggests a very strong *functional* (but nonlinear) relationship
    - the data in Table 2.3 and Figure 2.2 were examples of this.

Let's try out a few more toy examples.

```
> x <- -5:5
> y <- x^3
> cor(x, y)
[1] 0.9216491
```

The sample correlation between `x` and `y` is very large (≈0.92), and yet there is a *cubic* (not linear) relationship between these two variables:

```
> plot(y~x)
```



Try this out with a few other toy examples. Create the two variables as indicated, then calculate the sample correlation between the two variables (round your answer to the nearest 0.001). Produce a scatter plot of the variables as well.

Enter your answers in the spaces provide and click 'Check,' then click 'Next' to advance to the next problem.

Lesson 1.3: Covariance and Correlation

- **Introduction**

- **Scatter Plots**
- **Covariance and Correlation**

Introduction

Required Reading

- Sections 2.1-2.3

As the material in the second chapter of your textbook is mostly review, little effort will be spent on these Canvas notes in regards to explaining concepts, but rather we will focus on the R commands needed to perform a simple linear regression and how these computations can be performed using matrix algebra.

You will want to have a session of R open while reading through the Canvas lecture notes and recreate the provided code on your own. Note that you will be presumed to be comfortable with the basics of the R programming language, so little explanation will be provided in regards to R functions and operations that you have previously encountered.

To start off, you should start a new R program, download the data file `computer.repairs.csv` Download computer.repairs.csvand import this file into a data frame:

```
> repairs <- read.csv('computer.repairs.csv')
```

Recall that the `str` and `View` functions should typically be used to examine the *structure* of a data frame and its contents.

```
> str(repairs)
'data.frame':   14 obs. of  2 variables:
 $ Minutes: int  23 29 49 64 74 87 96 97 109 119 ...
 $ Units  : int  1 2 3 4 4 5 6 6 7 8 ...
> View(repairs)
```

We see there are two variables, `Minutes` and `Units`, that are observed on a total of 14 observations. The `View` function provides a nice "spreadsheet-like" display of the data (though this may not be overly useful if there are too many variables or too many observations).

When trying to describe the relationship between two (or even more) variables it is helpful to have both a visual display as well as some summary statistics. Both of these methods of summarizing data are indispensable.

- While data displays like a scatter plot (you will be investigating many more plots throughout the course) are invaluable tools, they also require a rather subjective interpretation on the part of the viewer.
- Numerical summary statistics such as the covariance or correlation (again, there will be many more to be investigated) are valuable in providing an air of "objectivity" to an analysis, but they might also mask important features of a data set that might only be apparent with a visual display.

Use of `attach` Function

For those of you who have taken MA5701 previously, you may recall that the `attach` function was used copiously to extract variables from a data frame.

I must apologize to you if you became overly reliant on this function, as its use is generally frowned upon as emblematic of bad programming practice.

There are a few more acceptable ways to perform operations on variables contained with an R data frame:

- use the `$` extraction operator, such as `mean(repairs$Minutes)`

- whenever a function takes an R formula as its first argument, the second argument will typically be the name of the data frame, such as `plot(Minutes~Units, repairs)`
- you can use the `with` function to create a local environment in which expressions can be evaluated

As an example of this last function, note that

```
> mean(Minutes)
Error in mean(Minutes): object 'Minutes' not found
```

produces an error, whereas

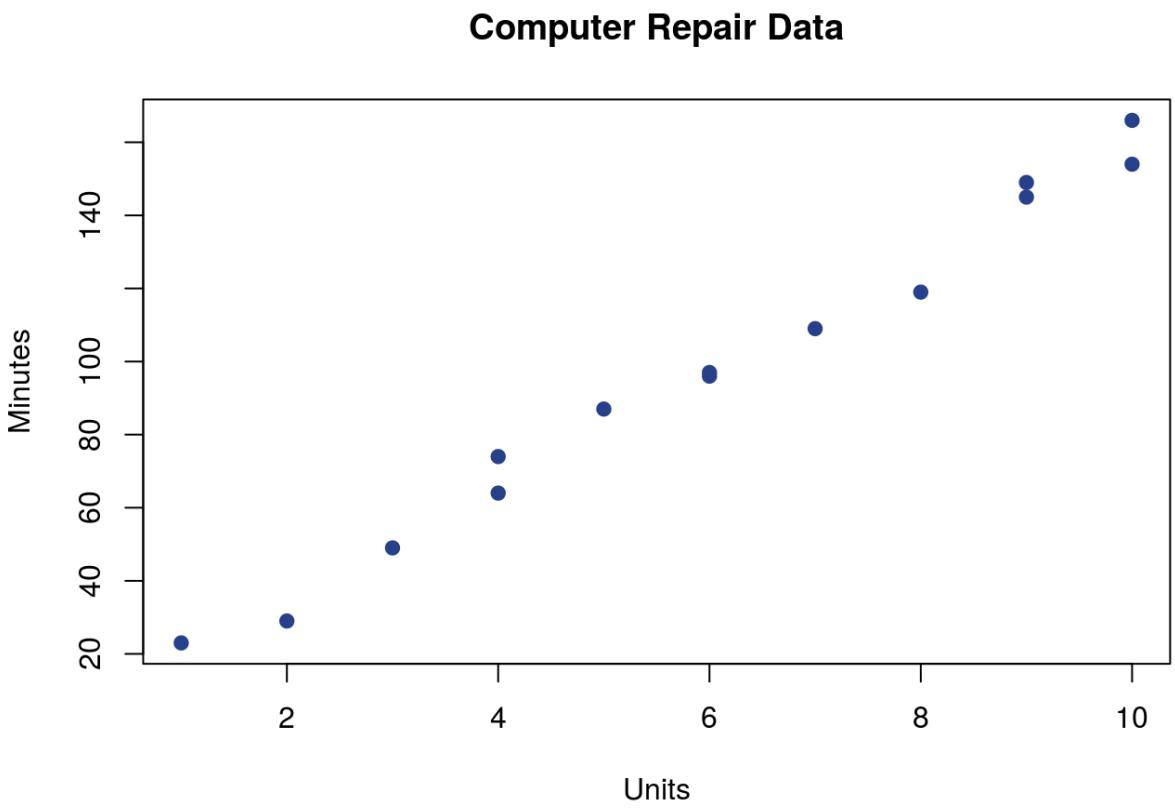
```
> with(repairs, mean(Minutes))
[1] 97.21429
```

does not. It will ultimately be your choice as to whether or not you use the `attach` function, but note that you may have already run into some of the problems associated with using it. Namely, changes to a variable outside of the data frame are not made within the data frame.

Scatter Plots

An appropriate visual display, such as a **scatter plot** of two variables, has the power of making evident at one quick glance whether or not there exists a relationship between two variables, and moreover gives the audience an idea as to how to begin describing the relationship. With the example at hand, we view the amount of time that it takes to finish a service call as being "*dependent*" on the number of units that need to be repaired. This variable should go on the y-axis, with the "*independent*" variable (the number of units in need of repair) lying along the x-axis.

```
> plot(Minutes~Units, repairs, main='Computer Repair Data', pch=19, col='ro
yalblue4')
```



You will not be required to "pretty" your graphics overly much in this course, but should you be so inclined...

- the `points` help page in R provides a list of the various plotting symbols you can choose from; circles are probably the most conventional route, but there are several other shapes from which to choose
    - specify your choice of plotting symbol with the `pch` argument

- the `col` argument can be used to specify the color of the plotting symbol; try out the `colors()` function to get a list of all of the named colors in R
- a more advanced package, called `ggplot2`, is probably the preferred route amongst professional statisticians or data scientists to create customizable plots in R
    - this pakage is part of the *tidyverse*Links to an external site. in R, which is a collection of packages aimed at supplying a consistent "grammar" to data management and displays
    - you are encouraged to look into learning how to effectively use the tidyverse *on your own time* (which may be scarce while enrolled in the MSAS program!), but this course will focus on using the plotting functions from the base package of R

This scatter plot illustrates a fairly strong *linear* relationship between the number of units that need to be repaired and the amount of time spent on a service call. Moreover, we would say there is a *positive* correlation between these two variables:

- the more units that need to be repaired, the more time is required for the service call

Covariance and Correlation

The **sample covariance** between two numerical variables is defined as

$$\text{Cov}(x,y) = \frac{1}{n-1}\sum_{j=1}^{n}(x_j - \bar{x})(y_j - \bar{y})$$

and can be used as a means of measuring the *strength* of a linear relationship between those variables. While this is a useful statistic, its interpretation is ultimately dependent on the units that are attached to the two variables.

Consider the covariance between `Minutes` and `Units`:

```
> with(repairs, cov(Minutes, Units))
[1] 136
```

The units on this quantity are "minute·parts". What if the amount of time were measured in hours instead?

```
> repairs$Hours <- repairs$Minutes / 60
> with(repairs, cov(Hours, Units))
[1] 2.266667
```

This new covariance has units of "hour·parts".

- Note that the covariance between minutes and units is 60 times that of the covariance between hours and units
- In general, $\text{Cov}(ax,by)=ab\cdot\text{Cov}(x,y)$ for any constants a and b, so that a scale change in one (or both) variable(s) correspondingly scales the covariance

To make interpretation easier, and independent of any arbitrary choice of units, the **sample correlation** is much nicer:

$$\text{Cor}(x,y)=\text{Cov}(x,y)s_x s_y=\text{Cov}(x,y)\text{Cov}(x,x)\sqrt{}\cdot\text{Cov}(y,y)\sqrt{}$$

where s_x and s_y are the sample standard deviations of x and y, respectively. The second equation comes from the fact that the covariance of a variable with itself is simply the sample variance:

$$s^2_x=\text{Var}(x)=\frac{1}{n-1}\sum_{j=1}^{n}(x_j-\bar{x})^2=\frac{1}{n-1}\sum_{j=1}^{n}(x_j-\bar{x})(x_j-\bar{x})=\text{Cov}(x,x)$$

```
> with(repairs, cov(Minutes, Minutes)) # covariance of minutes with itself
[1] 2136.027
> var(repairs$Minutes) # same as the sample variance of minutes
[1] 2136.027
```

In rescaling the sample covariance by dividing by the standard deviations of the two variables, the sample correlation is now a *unitless* quantity. Any change of the units on the original variables has no effect on the sample correlation:

```
> with(repairs, cor(Minutes, Units))
[1] 0.9936987
> with(repairs, cor(Hours, Units)) # correlation is unchanged by a change i
n units
[1] 0.9936987
```

The correlation is nice in that it always takes values between -1 and 1, providing us with a *rough* means of gauging the strength of the linear relationship between two variables.

- as $|\text{Cor}(x,y)|$ becomes closer to 1, the points in a scatter plot tend to fall closer to a straight line
- the sign of $\text{Cor}(x,y)$ indicates the *direction* of the correlation (or association) between x and y

A Word of Caution

You must always be wary of relying solely upon the sample correlation to judge the strength of a *linear* relationship between two variables! Your textbook discusses a few examples of data sets where...

- there is a "strong" correlation as measured by $\text{Cor}(x,y)$, while a scatter plot of the data suggests a highly nonlinear relationship

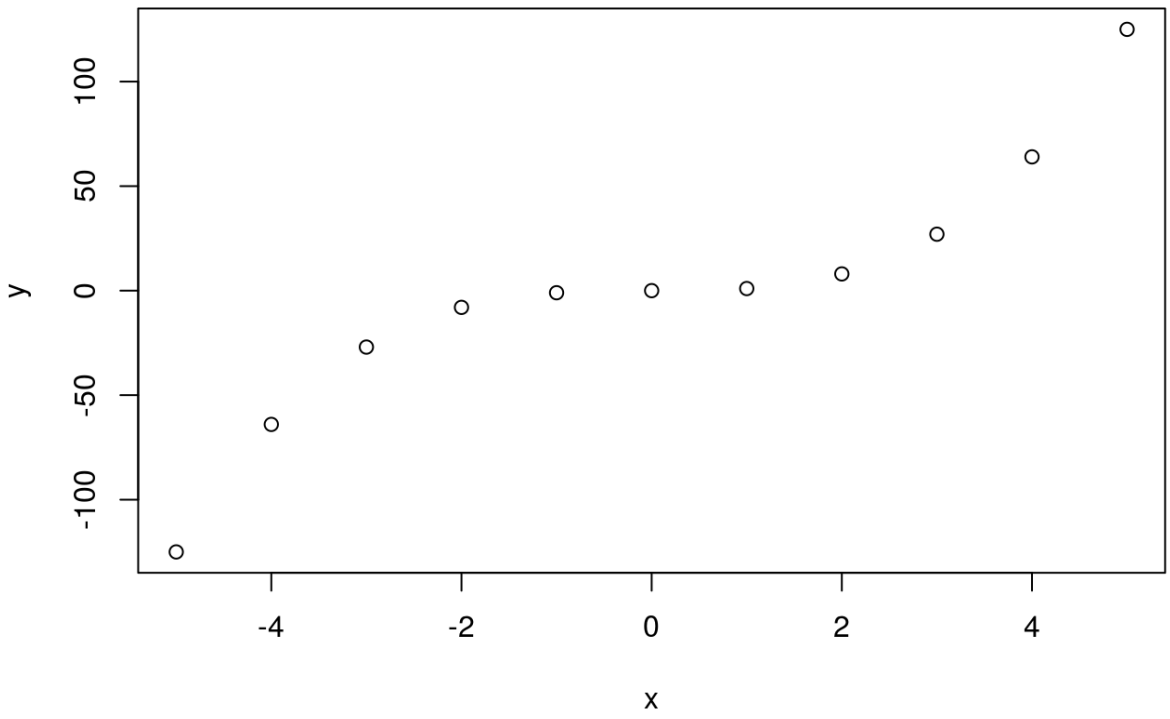
    - three of the datasets in Anscombe's quartet exhibited this behavior
- there was a "weak" correlation as measured by Cor(x,y), while a scatter plot of the data suggests a very strong *functional* (but nonlinear) relationship
    - the data in Table 2.3 and Figure 2.2 were examples of this.

Let's try out a few more toy examples.

```
> x <- -5:5
> y <- x^3
> cor(x, y)
[1] 0.9216491
```

The sample correlation between `x` and `y` is very large (≈0.92), and yet there is a *cubic* (not linear) relationship between these two variables:

```
> plot(y~x)
```



Try this out with a few other toy examples. Create the two variables as indicated, then calculate the sample correlation between the two variables (round your answer to the nearest 0.001). Produce a scatter plot of the variables as well.

Enter your answers in the spaces provide and click 'Check,' then click 'Next' to advance to the next problem.

Lesson 1.4: Simple Linear Regression Model

- **Simple Linear Regression Model**

- **Design Matrix**
- **Derivation of the Normal Equations**
- **Estimating the Regression Coefficients**
- **Plotting the Regression Line**
- **Exercise**

Required Reading

- Sections 2.4-2.5

Simple Linear Regression Model

After having inspected a scatter plot of two variables and decided that a linear relationship might be appropriate, it is desirable to estimate the line that is a best fit for the given data. We do this by supposing that the two variables x and y satisfy the following **simple linear regression model**: $y=\beta_0+\beta_1 x+\varepsilon$

Recall...

- y is called the **response variable**
- x is called an **explanatory variable**
- β_0 is called the **intercept**
- β_1 is called the **slope**
- ε is called an **error term**

We view y as being a random variable that can be picked from a population and observed.

- the explanatory variable could be viewed as being a *fixed constant* which could be chosen by the experimenter, in which case we have what is called a **fixed-effects model**
- alternatively, x might also be thought of as *random*, which would lead to a **random-effects model**

The mathematics that goes on behind the scenes for random-effects models becomes more complicated, but the computations that are used in estimating β_0 or β_1 will be the same. The error term ε is used to model the **statistical error**, which is the deviation of what is actually observed (y) from what is predicted by a model ($\beta_0+\beta_1 x$).

In order to *fit* this simple linear regression model, the experimenter goes and and collects n pairs of observations $(x_1,y_1),\ldots,(x_n,y_n)$ attached to sampling units sampled from some population of interest. With n instances of pairs (x,y), we then have n error terms ε_j in the linear model, which can now be expressed as $y_j=\beta_0+\beta_1 x_j+\varepsilon_j, j=1,2,\ldots,n$

Design Matrix

While there are many relatively painless ways to find point estimates for the coefficients β_0 and β_1 via R or other software packages, it is desirable that these functions not be a [black boxLinks to an external site.](#).

- We will investigate in some depth the *formulas* and *algorithms* that are needed to move from n pairs of numbers (x_j,y_j) to a set of other numbers which can be interpreted via statistical inference
- Unfortunately, these formulas and algorithms will have to remain something of a black box, as the mathematical theory used to develop these formulas is rather deep and time-consuming to learn
    - when feasible, some rough justifications for the various formulas will be provided

The simple linear regression model can be expressed as follows:

$$\boldsymbol{Y}=\boldsymbol{X\beta}+\boldsymbol{\varepsilon}$$

where

$$\boldsymbol{Y}=\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \boldsymbol{\beta}=[\beta_0 \beta_1], \boldsymbol{X}=\begin{bmatrix} 1 & 1 \\ 1 & x_1 \\ x_2 & \vdots \\ x_n \end{bmatrix}, \text{and } \boldsymbol{\varepsilon}=\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

Recall how matrix multiplication and addition works: $\boldsymbol{X\beta}+\boldsymbol{\varepsilon}=\begin{bmatrix} 1 & 1 \\ 1 & x_1 \\ x_2 & \vdots \\ x_n \end{bmatrix}[\beta_0 \beta_1]+\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}=\begin{bmatrix} 1\cdot\beta_0+x_1\cdot\beta_1 \\ 1\cdot\beta_0+x_2\cdot\beta_1 \\ 1\cdot\beta_0+x_n\cdot\beta_n \end{bmatrix}+\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}=\begin{bmatrix} \beta_0+\beta_1 x_1+\varepsilon_1 \\ \beta_0+\beta_1 x_2+\varepsilon_2 \\ \beta_0+\beta_1 x_n+\varepsilon_n \end{bmatrix}=\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$

We call the matrix $\boldsymbol{X}$ the **design matrix** for this model. Note that the design matrix can be created in R relatively simply by using the `cbind` function:

```
> n <- nrow(repairs)
> X <- cbind(rep(1, n), repairs$Units)
```

Try the `View` function on this design matrix.

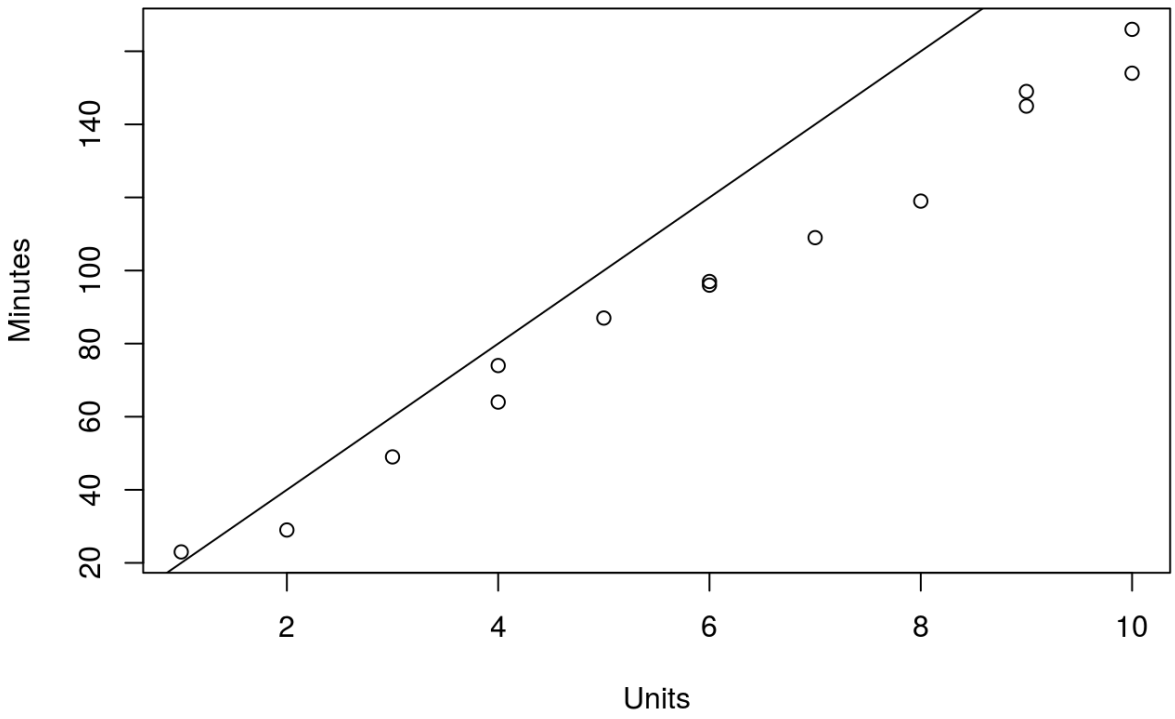
Suppose that we have two numbers $\hat{\beta}_0$ and $\hat{\beta}_1$ which we wish to use as estimates of the model coefficients β_0 and β_1. Then the **residuals**, or **observed errors**, are defined

as $e_j = y_j - \hat{y}_j = y_j - (\hat{\beta}_0 + \hat{\beta}_1 x_j)$, where $\hat{y}_j = \hat{\beta}_0 + \hat{\beta}_1 x_j$ is called the **fitted value** of the response variable.
The goal in choosing the numbers $\hat{\beta}_0$ and $\hat{\beta}_1$ is to minimize the errors in estimating y_j with $\hat{y}_j$.

- one idea might be to choose $\hat{\beta}_0$ and $\hat{\beta}_1$ so as to minimize $\sum_{nj=1} |e_j|$
    - while this seems like a natural idea, the mathematics behind this method are even messier than what will be considered
- instead, let us decide to minimize the **sum of the squared residuals**, $SSE = \sum_j e_{2j}$ (the "E" in SSE stands for error)
- yet another method that could be used is the *orthogonal regression method*, as discussed in Exercise 2.14 of your textbook
    - if you're up to the challenge of a formal mathematical derivation, please feel free to try out that exercise and post your solution to Piazza (a digital photo of a handwritten solution is likely the easiest route)

Define a couple of more matrices to represent the estimated model coefficients and the residuals: $\boldsymbol{\beta} = [\beta_0 \beta_1]$, and $\boldsymbol{e} = \begin{bmatrix} e_1 e_2 : e_n \end{bmatrix} = \begin{bmatrix} y_1 - (\beta_0 + \beta_1 x_1) y_2 - (\beta_0 + \beta_1 x_2) : y_n - (\beta_0 + \beta_1 x_n) \end{bmatrix} = \boldsymbol{Y} - \boldsymbol{X\beta}$ Let's try out a few different values of $\boldsymbol{\beta}$ to see what this *sum of squared errors* (SSE) looks like, and also overlay the associated line on top of the scatter plot:

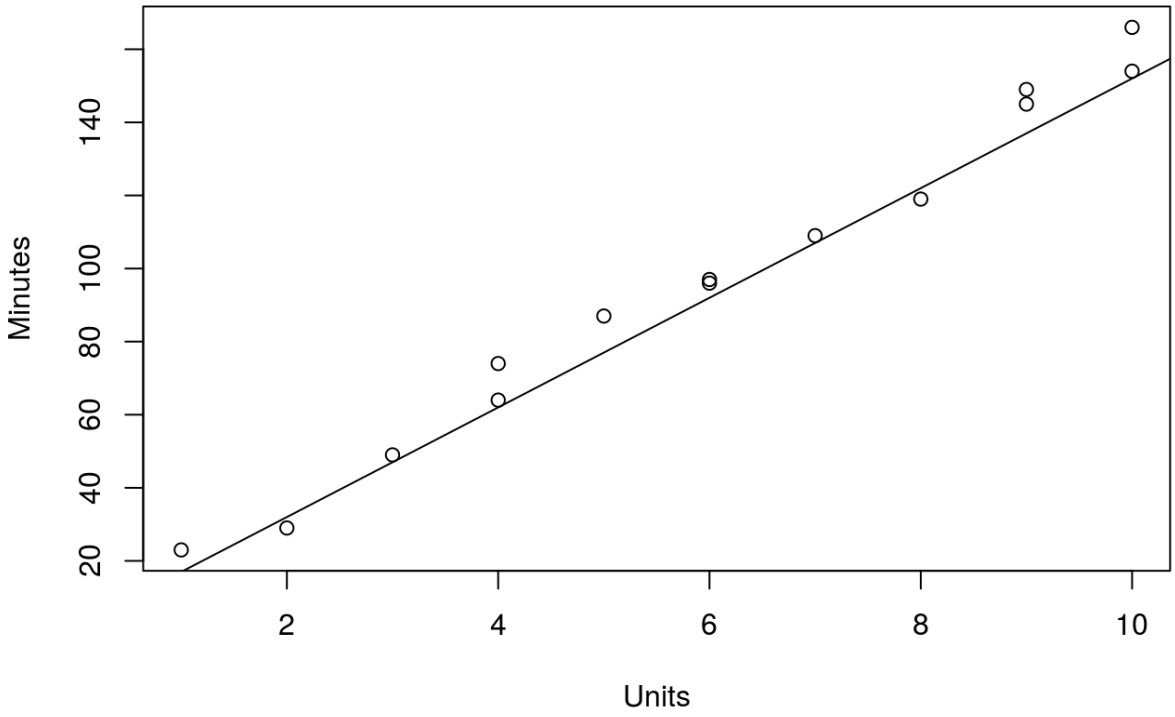
```
> beta.hat <- c(0, 20)
> Y <- repairs$Minutes
> e <- Y - X %*% beta.hat
> crossprod(e) # sum of squared residuals
     [,1]
[1,] 9917
> plot(Minutes~Units, repairs)
> abline(a=0, b=20) # plots a straight line with intercept equal to a and s
lope equal to b
```



Recall the `crossprod` function, when applied to a single vector, is just the sum of the squares of the elements in that vector; this could also be implemented with the command `sum(e^2)`. That is, when we choose $\beta_0=0$ and $\beta_1=20$, the sum of the squared residuals is 9917.

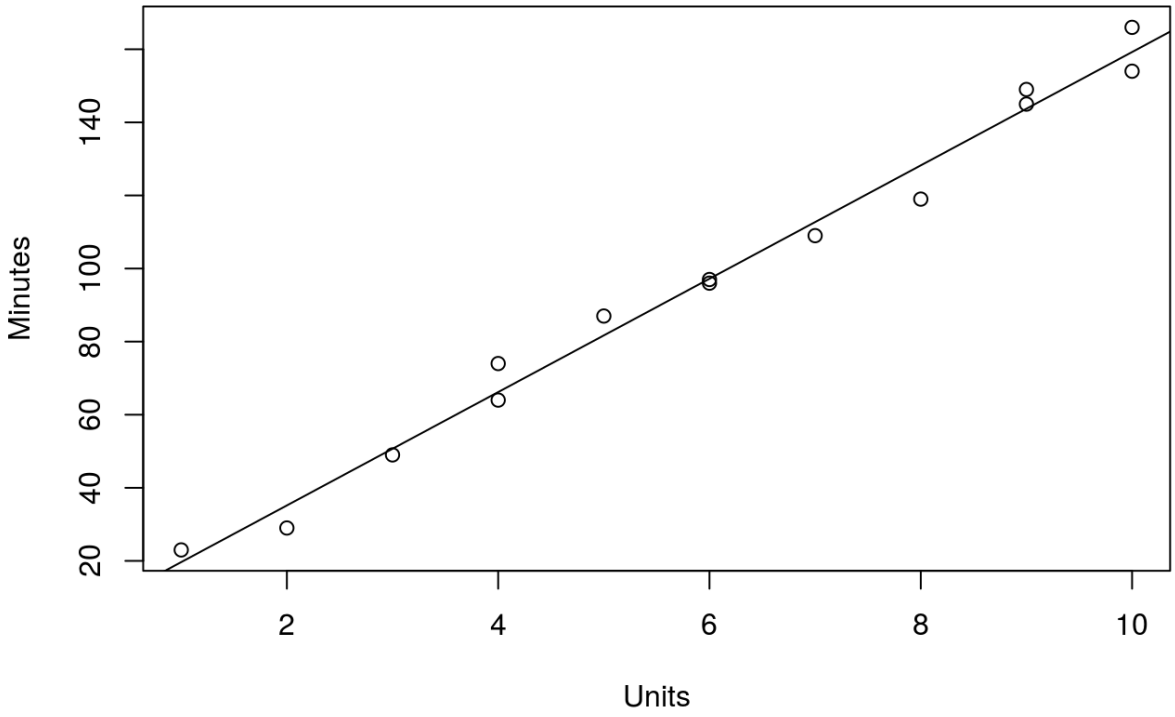
Can we get a smaller overall error with better choices of the model coefficients?

```
> beta.hat <- c(2, 15)
> e <- Y - X %*% beta.hat
> crossprod(e) # sum of squared residuals
     [,1]
[1,]  759
> plot(Minutes~Units, repairs)
> abline(a=2, b=15)
```



The choices $\beta_0=2$ and $\beta_1=15$ are even better. The points are closer to the line, giving an overall smaller error in the model (as measured by the sum of the squared errors). If you're following along in the textbook, you know that the choices of β_0 and β_1 which gives us the smallest possible value of SSE are $\beta_0 \approx 4.162$ and $\beta_1=15.509$:

```
> beta.hat <- c(4.162, 15.509)
> e <- Y - X %*% beta.hat
> crossprod(e) # sum of squared residuals
         [,1]
[1,] 348.8484
> plot(Minutes~Units, repairs)
> abline(a=4.162, b=15.509)
```



Try as you might, you will not find any other choices of $\hat{\beta}_0$ and $\hat{\beta}_1$ (aside from rounding errors) which will yield a smaller value of SSE.

Derivation of the Normal Equations

What follows is a brief justification, in terms of matrix algebra and calculus, for how we can find the vector $\boldsymbol{\beta}$ which minimizes SSE. If this level of mathematical wizardry is too much for you, don't get too frustrated with it! This derivation is provided solely for the sake of completeness, but you won't be assessed on the ability to understand or recreate any of the arguments.

!! perhaps can have each of the above lines revealed piecemeal, by clicking on a button to forward through !! First note that SSE can be viewed as both a scalar (real number), as well as

a 1×1 matrix $\sum_{j=1n}e_{2j}=[\sum_j e_{2j}]=[e_1e_2\cdots e_n]\begin{bmatrix}e_1\\e_2\\\vdots\\e_n\end{bmatrix}=e^Te=(Y-X\boldsymbol{\beta})^T(Y-X\boldsymbol{\beta})$

$=(Y^T-\boldsymbol{\beta}^TX^T)(Y-X\boldsymbol{\beta})=Y^TY-Y^TX\boldsymbol{\beta}-\boldsymbol{\beta}^TX^TY+\boldsymbol{\beta}^TX^TX\boldsymbol{\beta}=Y^TY-2\boldsymbol{\beta}^TX^TY+\boldsymbol{\beta}^TX^TX\boldsymbol{\beta}$

In the above we have used the fact that $(AB)^T=B^TA^T$ whenever A and B are matrices (of the right dimensions so that AB is defined). Also, in the last line we use the fact $Y^TX\boldsymbol{\beta}=(\boldsymbol{\beta}^TX^TY)^T=\boldsymbol{\beta}^TX^TY$, as all of these are 1×1 matrices, and hence equal to their own transpose.

Recall from your days in a calculus course that in order to find the maximum (or minimum) of some function, we can calculate the derivative of that function, set the

derivative equal to 0, and solve the resulting equation (with some caveats sprinkled in there). The same is true with functions of matrices as well! View SSE as a function of the variable $\boldsymbol{\beta}$, then differentiate with respect to this variable: $\frac{d}{d\boldsymbol{\beta}} SSE = 0 - 2\boldsymbol{X}^\intercal \boldsymbol{Y} + 2\boldsymbol{X}^\intercal \boldsymbol{X}\boldsymbol{\beta}$ Derivatives with matrices can be a bit tricky, but note that $\frac{d}{db}(y - bx)^2 = \frac{d}{db}(y^2 - 2xyb + b^2x^2) = -2xy + 2bx^2$ when x, y, and b are all just real numbers. The above is the matrix analogue of this fact.

We now set the derivative equal to 0 and solve this equation for $\boldsymbol{\beta}$: $2\boldsymbol{X}^\intercal \boldsymbol{X}\boldsymbol{\beta} = 2\boldsymbol{X}^\intercal \boldsymbol{Y}$ The above is called the **normal equation** associated with the linear regression model, and it yields the solution

$$\boldsymbol{\beta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1}\boldsymbol{X}^\intercal \boldsymbol{Y}$$

Note that the above solution makes the tacit assumption that $\boldsymbol{X}^\intercal \boldsymbol{X}$ has an inverse (which will typically be the case for us in this course).

Estimating the Regression Coefficients

There are a couple of routes by which you will want to obtain the least-squares estimates of the model coefficients in this course:

- using matrix operations in R to implement the formula $\boldsymbol{\beta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1}\boldsymbol{X}^\intercal \boldsymbol{Y}$
- using built-in R functions designed precisely for this purpose

While the mathematical theory that underlies many of the formulas you will encounter will not be properly investigated in this course, it is desirable that you are at least familiar with the notation and operations needed to implement them, if at least just to make the more convenient R functions less of a black box.

Open each tab to read more.

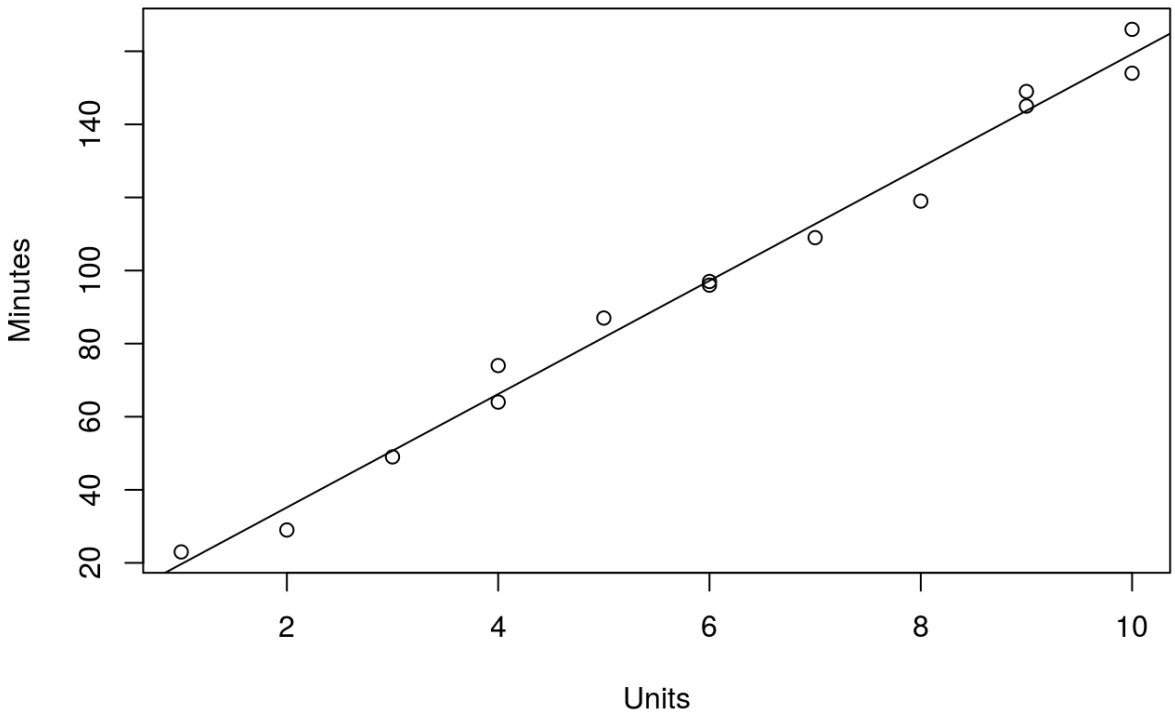
Route 1: Matrix Operation

Route 2: `lm` Function

Plotting the Regression Line

With simple linear regression, we have the ability to plot the regression line on top of the scatter plot of the data. The `abline` function does this for us quite nicely:

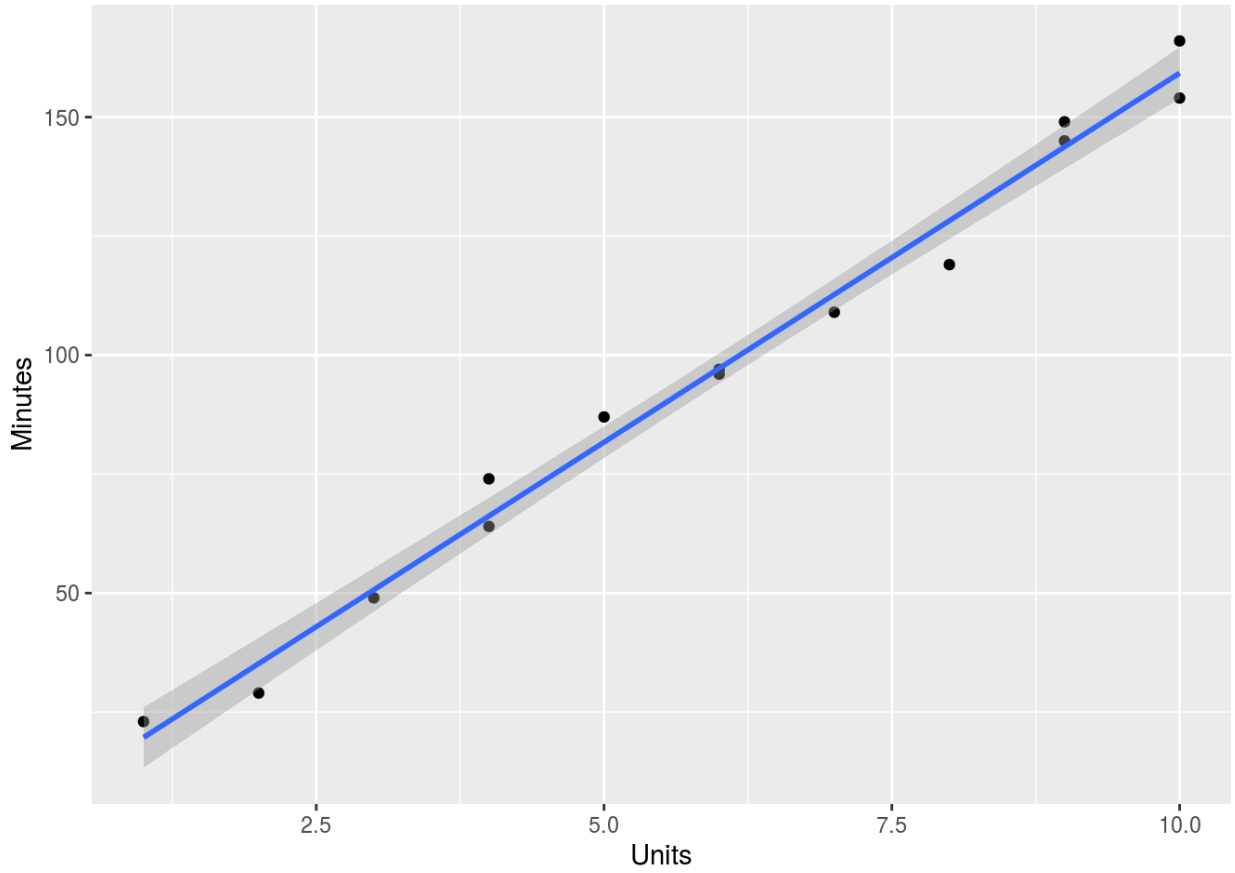
```
> plot(Minutes~Units, repairs)
> abline(repairs_lm)
```



- you could have also used the function call `abline(a=beta[1], b=beta[2])`, where the arguments `a` and `b` represent the intercept and slope of the straight line being drawn

Uou are again reminded of the `ggplot2` R package, which is part of the `tidyverse`, which is another very popular way of creating very customizable graphics in R. Here is a teaser on how such a plot could be created in with `ggplot`:

```
> library(ggplot2) # loads the ggplot2 package
> myPlot <- ggplot(repairs, aes(x=Units, y=Minutes)) +
+   geom_point() +
+   geom_smooth(method='lm')
> myPlot
```



You will not be required to use the `ggplot2` package in *this* course (I cannot speak for other instructors in the program), but if you will be performing data visualizations on a fairly regular basis then learning how to use this package effectively will be something you should put on your to-do list. Let's dissect the above code a bit.

Click 'Next' to review.

Next

Exercise

Put your R skills to the test with one more example. Consider the data frame `faithful`, which contains data on the duration of an eruption (measured in minutes) of the Old Faithful geyser at Yellowstone National Park and the amount of time that passes until the next eruption (also measured in minutes).

Use the given R application in Canvas (or of course you can use your own desktop version of R) to perform the following tasks:

- create a design matrix which models `waiting` as the response variable and `eruptions` as the explanatory variable in a simple linear regression model
- use matrix operators to calculate the model coefficient estimates $\boldsymbol{\beta}$, the set of fitted values $\hat{Y}$, and the set of residuals $e = \hat{Y} - Y$
- verify that $\beta_1 = \text{Cor}(x,y) \cdot s_y s_x$ and $\beta_0 = \bar{y} - \beta_1 \bar{x}$
- calculate the sum of the squared residuals, SSE
- plot the data in a scatter plot, with the estimated regression line overlaid on top of it

Then...

- investigate the plot and make note of any interesting features
- provide a physical interpretation to the estimated slope β_1
- decide whether any meaningful physical interpretation can be provided to the estimated intercept β_0
- predict what would happen to β_0 and β_1 if we changed the explanatory variable `eruptions` to be measured in seconds rather than minutes (then verify with the use of `lm`)

(The following code is provided as a sample. You may clear it to input new code for your exercise.)

Lesson 1.4: Simple Linear Regression Model

Required Reading

- Sections 2.4-2.5

Simple Linear Regression Model

After having inspected a scatter plot of two variables and decided that a linear relationship might be appropriate, it is desirable to estimate the line that is a best fit for the given data. We do this by supposing that the two variables x and y satisfy the following **simple linear regression model**: $y=\beta_0+\beta_1 x+\varepsilon$

Recall...

- y is called the **response variable**
- x is called an **explanatory variable**
- β_0 is called the **intercept**
- β_1 is called the **slope**
- ε is called an **error term**

We view y as being a random variable that can be picked from a population and observed.

- the explanatory variable could be viewed as being a *fixed constant* which could be chosen by the experimenter, in which case we have what is called a **fixed-effects model**
- alternatively, x might also be thought of as *random*, which would lead to a **random-effects model**

The mathematics that goes on behind the scenes for random-effects models becomes more complicated, but the computations that are used in estimating β_0 or β_1 will be the same. The error term ε is used to model the **statistical error**, which is the deviation of what is actually observed (y) from what is predicted by a model ($\beta_0+\beta_1 x$).
In order to *fit* this simple linear regression model, the experimenter goes and and collects n pairs of observations $(x_1,y_1),\ldots,(x_n,y_n)$ attached to sampling units sampled from some population of interest. With n instances of pairs (x,y), we then have n error terms ε_j in the linear model, which can now be expressed as $y_j=\beta_0+\beta_1 x_j+\varepsilon_j, j=1,2,\ldots,n$

Design Matrix

While there are many relatively painless ways to find point estimates for the coefficients β_0 and β_1 via R or other software packages, it is desirable that these functions not be a black boxLinks to an external site..

- We will investigate in some depth the *formulas* and *algorithms* that are needed to move from n pairs of numbers (x_j,y_j) to a set of other numbers which can be interpreted via statistical inference

- Unfortunately, these formulas and algorithms will have to remain something of a black box, as the mathematical theory used to develop these formulas is rather deep and time-consuming to learn
    - when feasible, some rough justifications for the various formulas will be provided

The simple linear regression model can be expressed as follows:

$$\boldsymbol{Y}=\boldsymbol{X}\boldsymbol{\beta}+\boldsymbol{\varepsilon}$$

where

$$\boldsymbol{Y}=\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix},\boldsymbol{\beta}=[\beta_0 \beta_1],\boldsymbol{X}=\begin{bmatrix} 1 & 1 & x_1 \\ 1 & x_2 & \vdots \\ 1 & x_n & \end{bmatrix},\text{and}\boldsymbol{\varepsilon}=\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

Recall how matrix multiplication and addition works: $\boldsymbol{X}\boldsymbol{\beta}+\boldsymbol{\varepsilon}=\begin{bmatrix} 1 & 1 & x_1 \\ 1 & x_2 & \vdots \\ 1 & x_n & \end{bmatrix}[\beta_0 \beta_1]+\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}=\begin{bmatrix} 1\cdot\beta_0+x_1\cdot\beta_1 \\ 1\cdot\beta_0+x_2\cdot\beta_1 \\ \vdots \\ 1\cdot\beta_0+x_n\cdot\beta_n \end{bmatrix}+\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}=\begin{bmatrix} \beta_0+\beta_1 x_1+\varepsilon_1 \\ \beta_0+\beta_1 x_2+\varepsilon_2 \\ \vdots \\ \beta_0+\beta_1 x_n+\varepsilon_n \end{bmatrix}=\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

We call the matrix $\boldsymbol{X}$ the **design matrix** for this model. Note that the design matrix can be created in R relatively simply by using the `cbind` function:

```
> n <- nrow(repairs)
> X <- cbind(rep(1, n), repairs$Units)
```

Try the `View` function on this design matrix.

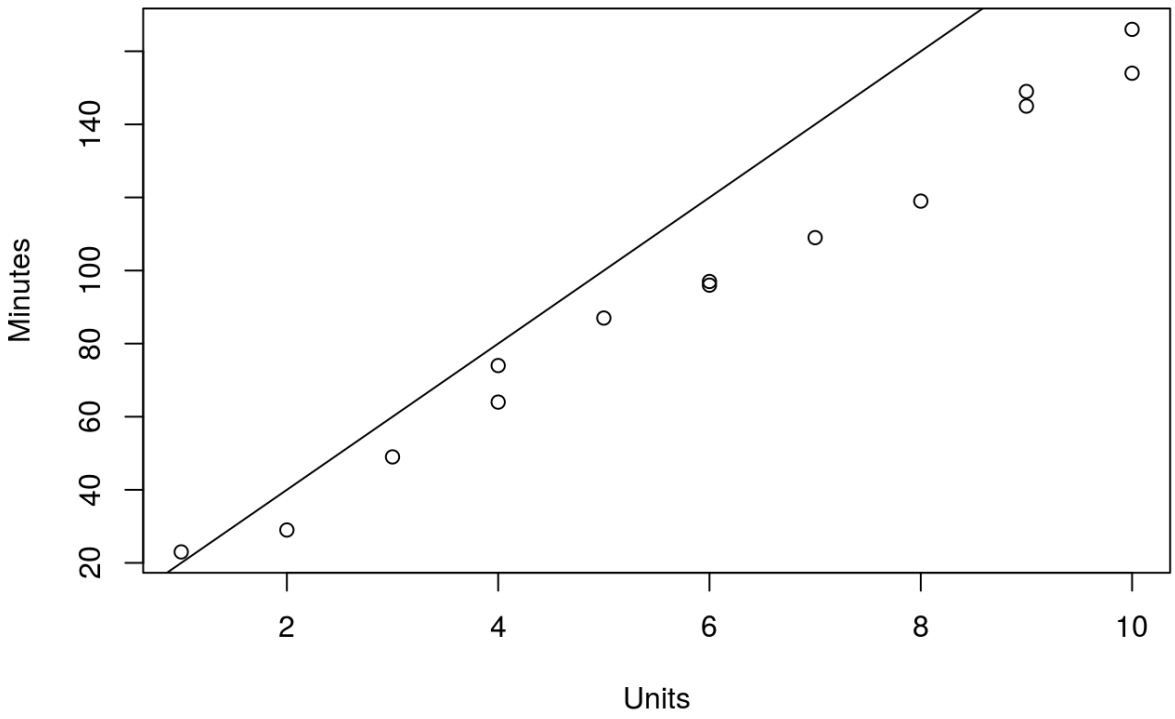
Suppose that we have two numbers $\hat{\beta}_0$ and $\hat{\beta}_1$ which we wish to use as estimates of the model coefficients β_0 and β_1. Then the **residuals**, or **observed errors**, are defined as $e_j=y_j-\hat{y}_j=y_j-(\hat{\beta}_0+\hat{\beta}_1 x_j)$, where $\hat{y}_j=\hat{\beta}_0+\hat{\beta}_1 x_j$ is called the **fitted value** of the response variable.
The goal in choosing the numbers $\hat{\beta}_0$ and $\hat{\beta}_1$ is to minimize the errors in estimating y_j with $\hat{y}_j$.

- one idea might be to choose $\hat{\beta}_0$ and $\hat{\beta}_1$ so as to minimize $\sum_{j=1}^{n}|e_j|$
    - while this seems like a natural idea, the mathematics behind this method are even messier than what will be considered
- instead, let us decide to minimize the **sum of the squared residuals**, $SSE=\sum_j e_j^2$ (the "E" in SSE stands for error)
- yet another method that could be used is the *orthogonal regression method*, as discussed in Exercise 2.14 of your textbook
    - if you're up to the challenge of a formal mathematical derivation, please feel free to try out that exercise and post your solution to

Piazza (a digital photo of a handwritten solution is likely the easiest route)

Define a couple of more matrices to represent the estimated model coefficients and the residuals: $\boldsymbol{\beta} = [\beta_0 \beta_1]$, and $\boldsymbol{e} = \begin{bmatrix} e_1 e_2 : e_n \end{bmatrix} = \begin{bmatrix} y_1 - (\beta_0 + \beta_1 x_1) y_2 - (\beta_0 + \beta_1 x_2) : y_n - (\beta_0 + \beta_1 x_n) \end{bmatrix} = \boldsymbol{Y} - \boldsymbol{X\beta}$ Let's try out a few different values of $\boldsymbol{\beta}$ to see what this *sum of squared errors* (SSE) looks like, and also overlay the associated line on top of the scatter plot:

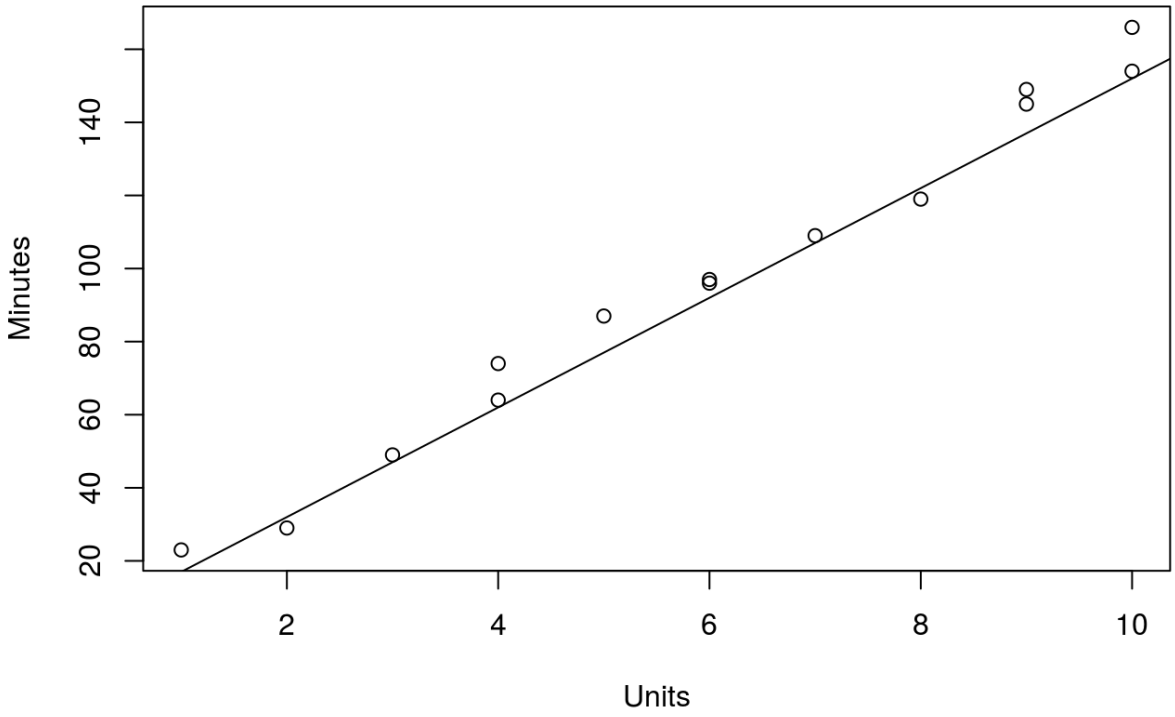
```
> beta.hat <- c(0, 20)
> Y <- repairs$Minutes
> e <- Y - X %*% beta.hat
> crossprod(e) # sum of squared residuals
     [,1]
[1,] 9917
> plot(Minutes~Units, repairs)
> abline(a=0, b=20) # plots a straight line with intercept equal to a and s
lope equal to b
```



Recall the `crossprod` function, when applied to a single vector, is just the sum of the squares of the elements in that vector; this could also be implemented with the command `sum(e^2)`. That is, when we choose $\beta_0=0$ and $\beta_1=20$, the sum of the squared residuals is 9917.

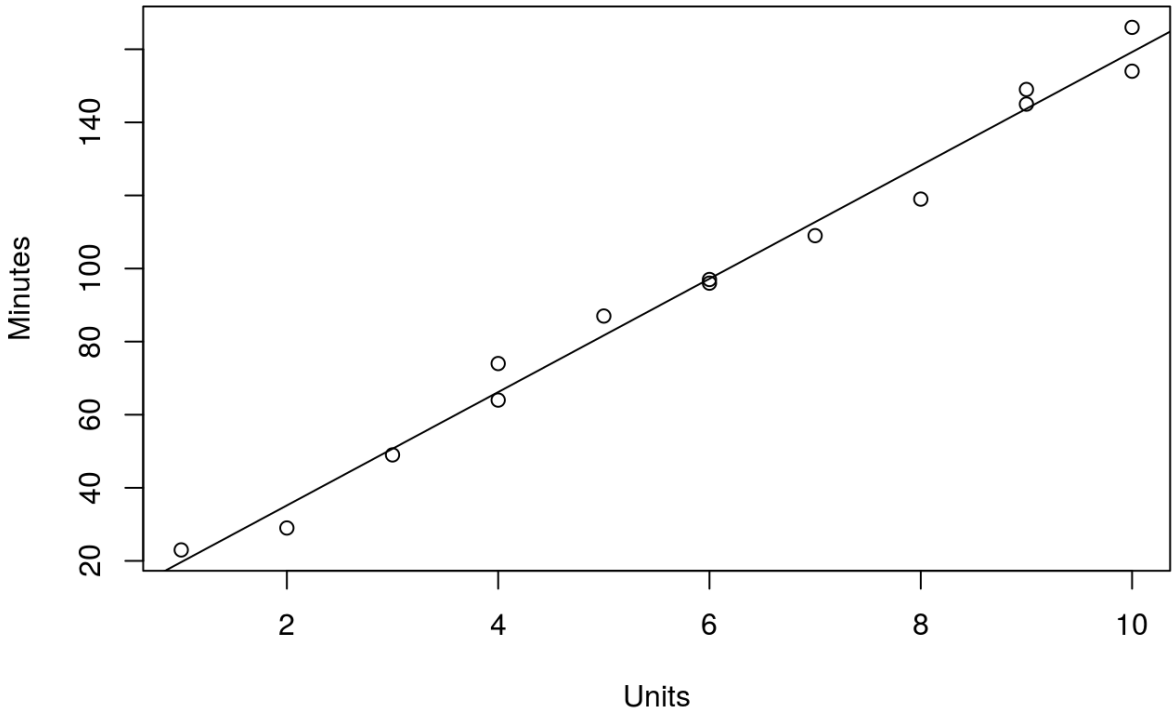
Can we get a smaller overall error with better choices of the model coefficients?

```
> beta.hat <- c(2, 15)
> e <- Y - X %*% beta.hat
> crossprod(e) # sum of squared residuals
     [,1]
[1,]  759
> plot(Minutes~Units, repairs)
> abline(a=2, b=15)
```



The choices $\beta_0=2$ and $\beta_1=15$ are even better. The points are closer to the line, giving an overall smaller error in the model (as measured by the sum of the squared errors). If you're following along in the textbook, you know that the choices of β_0 and β_1 which gives us the smallest possible value of SSE are $\beta_0 \approx 4.162$ and $\beta_1=15.509$:

```
> beta.hat <- c(4.162, 15.509)
> e <- Y - X %*% beta.hat
> crossprod(e) # sum of squared residuals
         [,1]
[1,] 348.8484
> plot(Minutes~Units, repairs)
> abline(a=4.162, b=15.509)
```



Try as you might, you will not find any other choices of $\hat{\beta}_0$ and $\hat{\beta}_1$ (aside from rounding errors) which will yield a smaller value of SSE.

Derivation of the Normal Equations

What follows is a brief justification, in terms of matrix algebra and calculus, for how we can find the vector $\boldsymbol{\beta}$ which minimizes SSE. If this level of mathematical wizardry is too much for you, don't get too frustrated with it! This derivation is provided solely for the sake of completeness, but you won't be assessed on the ability to understand or recreate any of the arguments.

!! perhaps can have each of the above lines revealed piecemeal, by clicking on a button to forward through !! First note that SSE can be viewed as both a scalar (real number), as well as

a 1×1 matrix $\sum_{j=1n}e_{2j}=[\sum_j e_{2j}]=[e_1e_2\cdots e_n]\begin{bmatrix}e_1\\e_2\\\vdots\\e_n\end{bmatrix}=e^Te=(Y-X\boldsymbol{\beta})^T(Y-X\boldsymbol{\beta})$

$=(Y^T-\boldsymbol{\beta}^TX^T)(Y-X\boldsymbol{\beta})=Y^TY-Y^TX\boldsymbol{\beta}-\boldsymbol{\beta}^TX^TY+\boldsymbol{\beta}^TX^TX\boldsymbol{\beta}=Y^TY-2\boldsymbol{\beta}^TX^TY+\boldsymbol{\beta}^TX^TX\boldsymbol{\beta}$

In the above we have used the fact that $(AB)^T=B^TA^T$ whenever A and B are matrices (of the right dimensions so that AB is defined). Also, in the last line we use the fact $Y^TX\boldsymbol{\beta}=(\boldsymbol{\beta}^TX^TY)^T=\boldsymbol{\beta}^TX^TY$, as all of these are 1×1 matrices, and hence equal to their own transpose.

Recall from your days in a calculus course that in order to find the maximum (or minimum) of some function, we can calculate the derivative of that function, set the

derivative equal to 0, and solve the resulting equation (with some caveats sprinkled in there). The same is true with functions of matrices as well! View SSE as a function of the variable $\boldsymbol{\beta}$, then differentiate with respect to this variable: $\frac{d}{d\boldsymbol{\beta}} SSE = 0 - 2\boldsymbol{X}^\intercal \boldsymbol{Y} + 2\boldsymbol{X}^\intercal \boldsymbol{X}\boldsymbol{\beta}$ Derivatives with matrices can be a bit tricky, but note that $\frac{d}{db}(y - bx)^2 = \frac{d}{db}(y^2 - 2xyb + b^2x^2) = -2xy + 2bx^2$ when x, y, and b are all just real numbers. The above is the matrix analogue of this fact.

We now set the derivative equal to 0 and solve this equation for $\boldsymbol{\beta}$: $2\boldsymbol{X}^\intercal \boldsymbol{X}\boldsymbol{\beta} = 2\boldsymbol{X}^\intercal \boldsymbol{Y}$ The above is called the **normal equation** associated with the linear regression model, and it yields the solution

$$\boldsymbol{\beta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1}\boldsymbol{X}^\intercal \boldsymbol{Y}$$

Note that the above solution makes the tacit assumption that $\boldsymbol{X}^\intercal \boldsymbol{X}$ has an inverse (which will typically be the case for us in this course).

Estimating the Regression Coefficients

There are a couple of routes by which you will want to obtain the least-squares estimates of the model coefficients in this course:

- using matrix operations in R to implement the formula $\boldsymbol{\beta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1}\boldsymbol{X}^\intercal \boldsymbol{Y}$
- using built-in R functions designed precisely for this purpose

While the mathematical theory that underlies many of the formulas you will encounter will not be properly investigated in this course, it is desirable that you are at least familiar with the notation and operations needed to implement them, if at least just to make the more convenient R functions less of a black box.

Open each tab to read more.

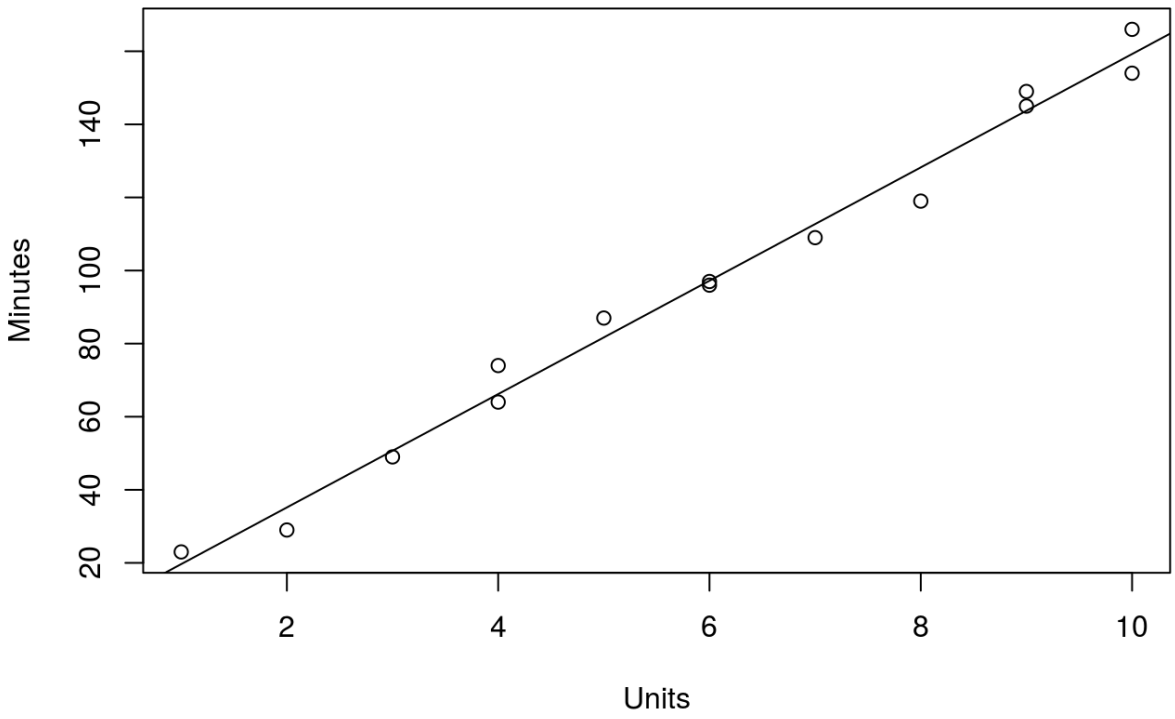
Route 1: Matrix Operation

Route 2: `lm` Function

Plotting the Regression Line

With simple linear regression, we have the ability to plot the regression line on top of the scatter plot of the data. The `abline` function does this for us quite nicely:

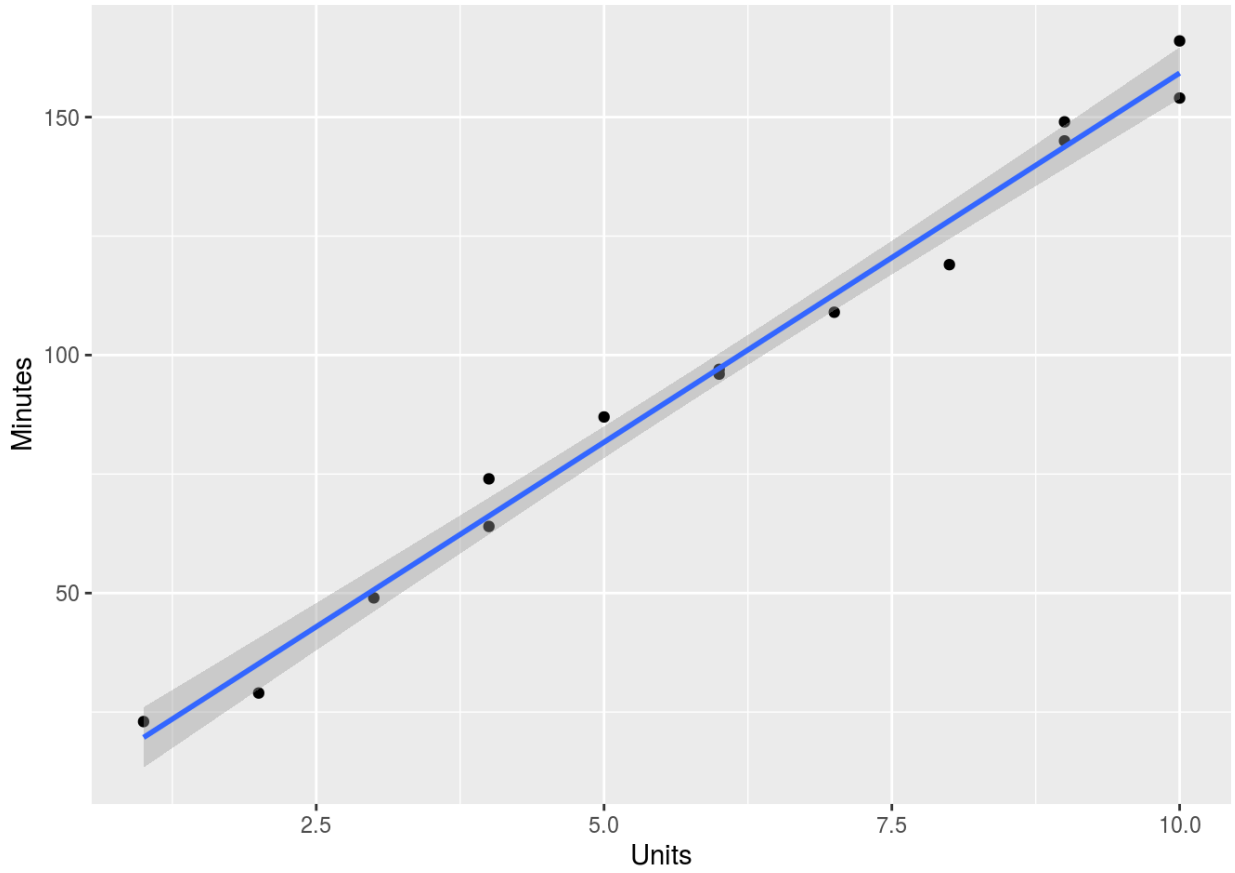
```
> plot(Minutes~Units, repairs)
> abline(repairs_lm)
```



- you could have also used the function call `abline(a=beta[1], b=beta[2])`, where the arguments `a` and `b` represent the intercept and slope of the straight line being drawn

Uou are again reminded of the `ggplot2` R package, which is part of the `tidyverse`, which is another very popular way of creating very customizable graphics in R. Here is a teaser on how such a plot could be created in with `ggplot`:

```
> library(ggplot2) # loads the ggplot2 package
> myPlot <- ggplot(repairs, aes(x=Units, y=Minutes)) +
+   geom_point() +
+   geom_smooth(method='lm')
> myPlot
```



You will not be required to use the `ggplot2` package in *this* course (I cannot speak for other instructors in the program), but if you will be performing data visualizations on a fairly regular basis then learning how to use this package effectively will be something you should put on your to-do list. Let's dissect the above code a bit.

Click 'Next' to review.

Next

Exercise

Put your R skills to the test with one more example. Consider the data frame `faithful`, which contains data on the duration of an eruption (measured in minutes) of the Old Faithful geyser at Yellowstone National Park and the amount of time that passes until the next eruption (also measured in minutes).

Use the given R application in Canvas (or of course you can use your own desktop version of R) to perform the following tasks:

- create a design matrix which models `waiting` as the response variable and `eruptions` as the explanatory variable in a simple linear regression model
- use matrix operators to calculate the model coefficient estimates $\boldsymbol{\beta}$, the set of fitted values $\hat{Y}$, and the set of residuals $e=\hat{Y}-Y$
- verify that $\beta_1=\text{Cor}(x,y)\cdot s_y s_x$ and $\beta_0=\bar{y}-\beta_1\bar{x}$
- calculate the sum of the squared residuals, SSE
- plot the data in a scatter plot, with the estimated regression line overlaid on top of it

Then...

- investigate the plot and make note of any interesting features
- provide a physical interpretation to the estimated slope β_1
- decide whether any meaningful physical interpretation can be provided to the estimated intercept β_0
- predict what would happen to β_0 and β_1 if we changed the explanatory variable `eruptions` to be measured in seconds rather than minutes (then verify with the use of `lm`)

(The following code is provided as a sample. You may clear it to input new code for your exercise.)

Lesson 1.4: Simple Linear Regression Model

Required Reading

- Sections 2.4-2.5

Simple Linear Regression Model

After having inspected a scatter plot of two variables and decided that a linear relationship might be appropriate, it is desirable to estimate the line that is a best fit for the given data. We do this by supposing that the two variables x and y satisfy the following **simple linear regression model**: $y=\beta_0+\beta_1 x+\varepsilon$

Recall...

- y is called the **response variable**
- x is called an **explanatory variable**
- β_0 is called the **intercept**
- β_1 is called the **slope**
- ε is called an **error term**

We view y as being a random variable that can be picked from a population and observed.

- the explanatory variable could be viewed as being a *fixed constant* which could be chosen by the experimenter, in which case we have what is called a **fixed-effects model**
- alternatively, x might also be thought of as *random*, which would lead to a **random-effects model**

The mathematics that goes on behind the scenes for random-effects models becomes more complicated, but the computations that are used in estimating β_0 or β_1 will be the same. The error term ε is used to model the **statistical error**, which is the deviation of what is actually observed (y) from what is predicted by a model ($\beta_0+\beta_1 x$).
In order to *fit* this simple linear regression model, the experimenter goes and and collects n pairs of observations $(x_1,y_1),\ldots,(x_n,y_n)$ attached to sampling units sampled from some population of interest. With n instances of pairs (x,y), we then have n error terms ε_j in the linear model, which can now be expressed as $y_j=\beta_0+\beta_1 x_j+\varepsilon_j, j=1,2,\ldots,n$

Design Matrix

While there are many relatively painless ways to find point estimates for the coefficients β_0 and β_1 via R or other software packages, it is desirable that these functions not be a [black boxLinks to an external site.](#).

- We will investigate in some depth the *formulas* and *algorithms* that are needed to move from n pairs of numbers (x_j, y_j) to a set of other numbers which can be interpreted via statistical inference
- Unfortunately, these formulas and algorithms will have to remain something of a black box, as the mathematical theory used to develop these formulas is rather deep and time-consuming to learn
    - when feasible, some rough justifications for the various formulas will be provided

The simple linear regression model can be expressed as follows:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

where

$$\boldsymbol{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \boldsymbol{\beta} = [\beta_0 \beta_1], \boldsymbol{X} = \begin{bmatrix} 1 & 1 \\ 1 & x_1 \\ x_2 & \vdots \\ x_n \end{bmatrix}, \text{and } \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

Recall how matrix multiplication and addition works: $\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} = \begin{bmatrix} 1 & 1 \\ 1 & x_1 \\ x_2 & \vdots \\ x_n \end{bmatrix} [\beta_0 \beta_1] + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix} = \begin{bmatrix} 1 \cdot \beta_0 + x_1 \cdot \beta_1 \\ 1 \cdot \beta_0 + x_2 \cdot \beta_1 \\ \vdots \\ 1 \cdot \beta_0 + x_n \cdot \beta_n \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix} = \begin{bmatrix} \beta_0 + \beta_1 x_1 + \varepsilon_1 \\ \beta_0 + \beta_1 x_2 + \varepsilon_2 \\ \vdots \\ \beta_0 + \beta_1 x_n + \varepsilon_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

We call the matrix $\boldsymbol{X}$ the **design matrix** for this model. Note that the design matrix can be created in R relatively simply by using the `cbind` function:

```
> n <- nrow(repairs)
> X <- cbind(rep(1, n), repairs$Units)
```

Try the `View` function on this design matrix.

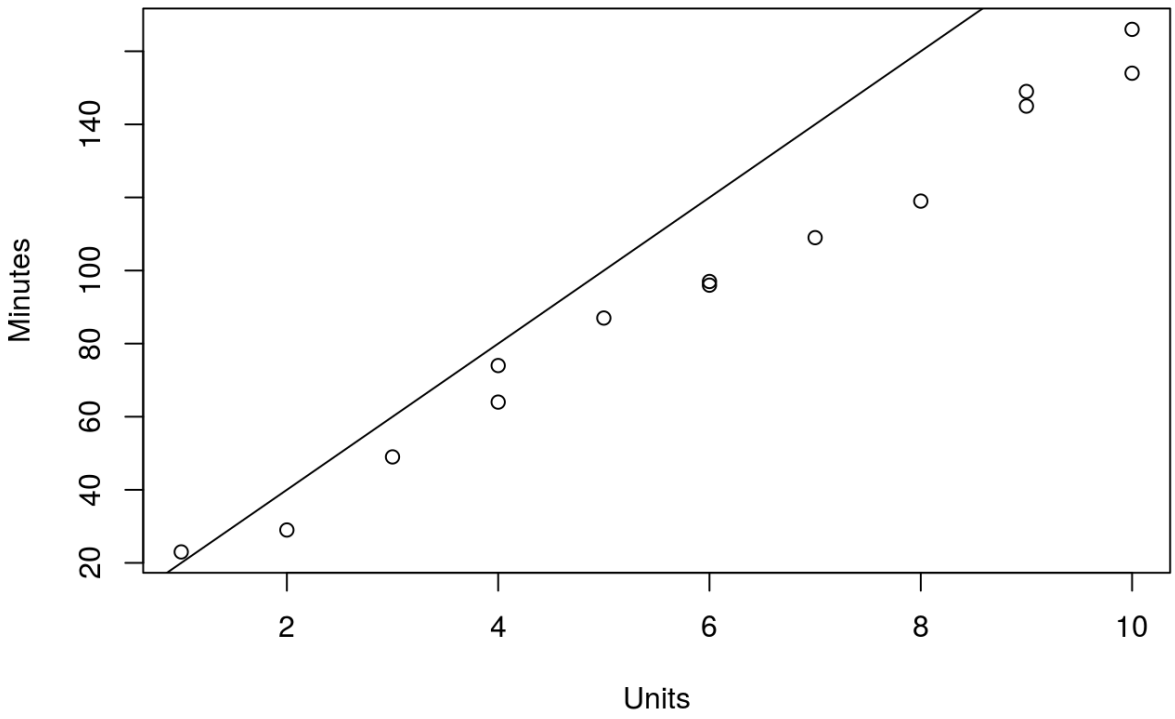
Suppose that we have two numbers $\hat{\beta}_0$ and $\hat{\beta}_1$ which we wish to use as estimates of the model coefficients β_0 and β_1. Then the **residuals**, or **observed errors**, are defined as $e_j = y_j - \hat{y}_j = y_j - (\hat{\beta}_0 + \hat{\beta}_1 x_j)$, where $\hat{y}_j = \hat{\beta}_0 + \hat{\beta}_1 x_j$ is called the **fitted value** of the response variable.
The goal in choosing the numbers $\hat{\beta}_0$ and $\hat{\beta}_1$ is to minimize the errors in estimating y_j with $\hat{y}_j$.

- one idea might be to choose $\hat{\beta}_0$ and $\hat{\beta}_1$ so as to minimize $\sum_{j=1}^{n} |e_j|$
    - while this seems like a natural idea, the mathematics behind this method are even messier than what will be considered
- instead, let us decide to minimize the **sum of the squared residuals**, $SSE = \sum_j e_j^2$ (the "E" in SSE stands for error)
- yet another method that could be used is the *orthogonal regression method*, as discussed in Exercise 2.14 of your textbook

- if you're up to the challenge of a formal mathematical derivation, please feel free to try out that exercise and post your solution to Piazza (a digital photo of a handwritten solution is likely the easiest route)

Define a couple of more matrices to represent the estimated model coefficients and the residuals: $\boldsymbol{\beta} = [\beta_0 \beta_1]$, and $\boldsymbol{e} = \begin{bmatrix} e_1 e_2 : e_n \end{bmatrix} = \begin{bmatrix} y_1 - (\beta_0 + \beta_1 x_1) y_2 - (\beta_0 + \beta_1 x_2) : y_n - (\beta_0 + \beta_1 x_n) \end{bmatrix} = \boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}$ Let's try out a few different values of $\boldsymbol{\beta}$ to see what this *sum of squared errors* (SSE) looks like, and also overlay the associated line on top of the scatter plot:

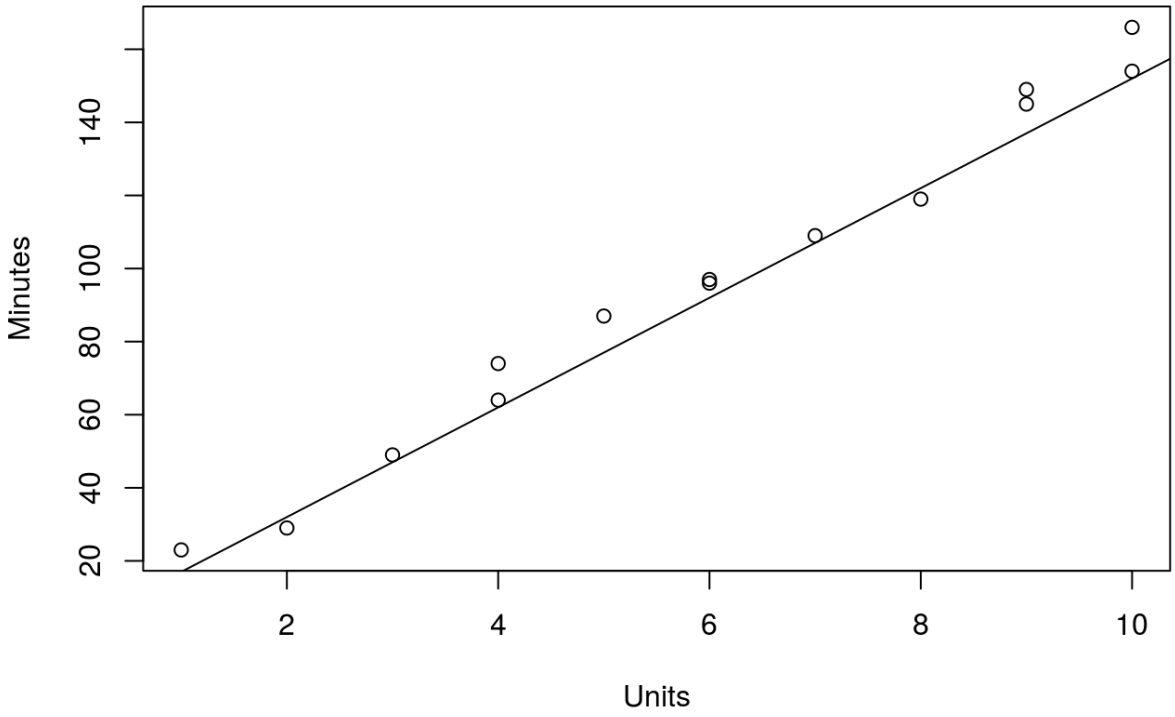
```
> beta.hat <- c(0, 20)
> Y <- repairs$Minutes
> e <- Y - X %*% beta.hat
> crossprod(e) # sum of squared residuals
     [,1]
[1,] 9917
> plot(Minutes~Units, repairs)
> abline(a=0, b=20) # plots a straight line with intercept equal to a and s
lope equal to b
```



Recall the `crossprod` function, when applied to a single vector, is just the sum of the squares of the elements in that vector; this could also be implemented with the command `sum(e^2)`. That is, when we choose $\beta_0=0$ and $\beta_1=20$, the sum of the squared residuals is 9917.

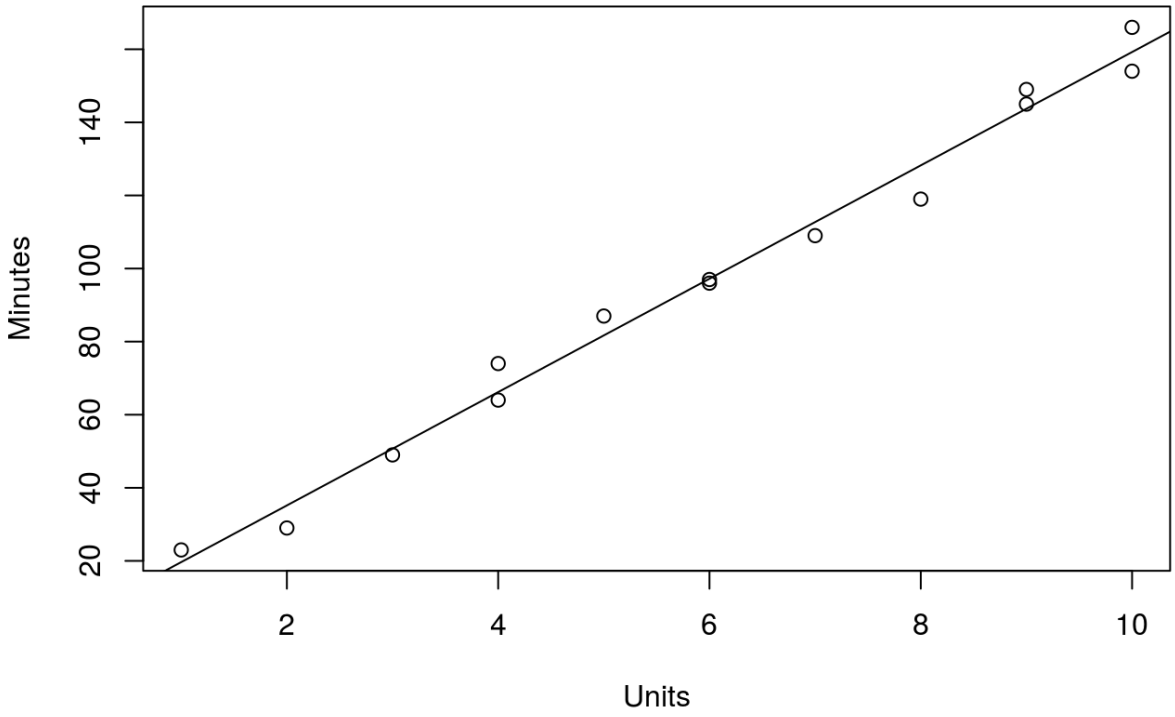
Can we get a smaller overall error with better choices of the model coefficients?

```
> beta.hat <- c(2, 15)
> e <- Y - X %*% beta.hat
> crossprod(e) # sum of squared residuals
     [,1]
[1,]  759
> plot(Minutes~Units, repairs)
> abline(a=2, b=15)
```



The choices $\beta_0=2$ and $\beta_1=15$ are even better. The points are closer to the line, giving an overall smaller error in the model (as measured by the sum of the squared errors). If you're following along in the textbook, you know that the choices of β_0 and β_1 which gives us the smallest possible value of SSE are $\beta_0 \approx 4.162$ and $\beta_1=15.509$:

```
> beta.hat <- c(4.162, 15.509)
> e <- Y - X %*% beta.hat
> crossprod(e) # sum of squared residuals
         [,1]
[1,] 348.8484
> plot(Minutes~Units, repairs)
> abline(a=4.162, b=15.509)
```



Try as you might, you will not find any other choices of $\hat{\beta}_0$ and $\hat{\beta}_1$ (aside from rounding errors) which will yield a smaller value of SSE.

Derivation of the Normal Equations

What follows is a brief justification, in terms of matrix algebra and calculus, for how we can find the vector $\boldsymbol{\beta}$ which minimizes SSE. If this level of mathematical wizardry is too much for you, don't get too frustrated with it! This derivation is provided solely for the sake of completeness, but you won't be assessed on the ability to understand or recreate any of the arguments.

!! perhaps can have each of the above lines revealed piecemeal, by clicking on a button to forward through !! First note that SSE can be viewed as both a scalar (real number), as well as

a 1×1 matrix $\sum_{j=1n}e_{2j}=[\sum_j e_{2j}]=[e_1e_2\cdots e_n]\begin{bmatrix}e_1\\e_2\\\vdots\\e_n\end{bmatrix}=e^Te=(Y-X\boldsymbol{\beta})^T(Y-X\boldsymbol{\beta})$

$=(Y^T-\boldsymbol{\beta}^TX^T)(Y-X\boldsymbol{\beta})=Y^TY-Y^TX\boldsymbol{\beta}-\boldsymbol{\beta}^TX^TY+\boldsymbol{\beta}^TX^TX\boldsymbol{\beta}=Y^TY-2\boldsymbol{\beta}^TX^TY+\boldsymbol{\beta}^TX^TX\boldsymbol{\beta}$

In the above we have used the fact that $(AB)^T=B^TA^T$ whenever A and B are matrices (of the right dimensions so that AB is defined). Also, in the last line we use the fact $Y^TX\boldsymbol{\beta}=(\boldsymbol{\beta}^TX^TY)^T=\boldsymbol{\beta}^TX^TY$, as all of these are 1×1 matrices, and hence equal to their own transpose.

Recall from your days in a calculus course that in order to find the maximum (or minimum) of some function, we can calculate the derivative of that function, set the

derivative equal to 0, and solve the resulting equation (with some caveats sprinkled in there). The same is true with functions of matrices as well! View SSE as a function of the variable $\boldsymbol{\beta}$, then differentiate with respect to this variable: $\frac{d}{d\boldsymbol{\beta}} SSE = 0 - 2\boldsymbol{X}^\intercal \boldsymbol{Y} + 2\boldsymbol{X}^\intercal \boldsymbol{X}\boldsymbol{\beta}$ Derivatives with matrices can be a bit tricky, but note that $\frac{d}{db}(y - bx)^2 = \frac{d}{db}(y^2 - 2xyb + b^2x^2) = -2xy + 2bx^2$ when x, y, and b are all just real numbers. The above is the matrix analogue of this fact.

We now set the derivative equal to 0 and solve this equation for $\boldsymbol{\beta}$: $2\boldsymbol{X}^\intercal \boldsymbol{X}\boldsymbol{\beta} = 2\boldsymbol{X}^\intercal \boldsymbol{Y}$ The above is called the **normal equation** associated with the linear regression model, and it yields the solution

$$\boldsymbol{\beta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1}\boldsymbol{X}^\intercal \boldsymbol{Y}$$

Note that the above solution makes the tacit assumption that $\boldsymbol{X}^\intercal \boldsymbol{X}$ has an inverse (which will typically be the case for us in this course).

Estimating the Regression Coefficients

There are a couple of routes by which you will want to obtain the least-squares estimates of the model coefficients in this course:

- using matrix operations in R to implement the formula $\boldsymbol{\beta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1}\boldsymbol{X}^\intercal \boldsymbol{Y}$
- using built-in R functions designed precisely for this purpose

While the mathematical theory that underlies many of the formulas you will encounter will not be properly investigated in this course, it is desirable that you are at least familiar with the notation and operations needed to implement them, if at least just to make the more convenient R functions less of a black box.

Open each tab to read more.

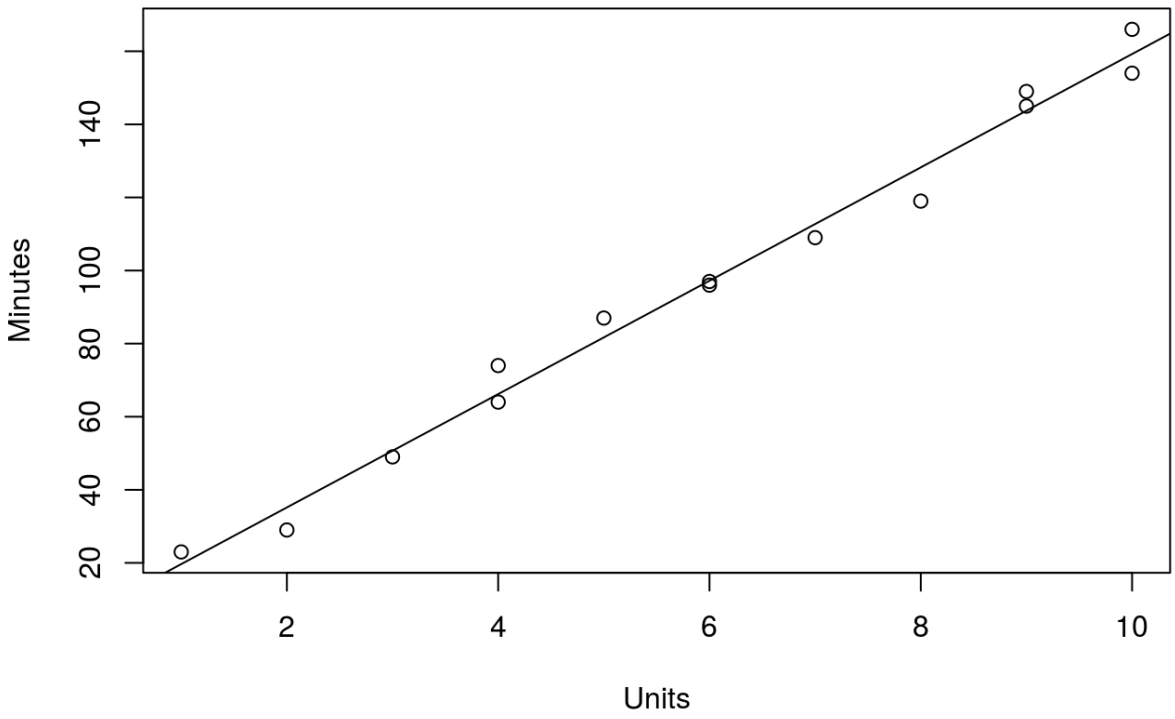
Route 1: Matrix Operation

Route 2: `lm` Function

Plotting the Regression Line

With simple linear regression, we have the ability to plot the regression line on top of the scatter plot of the data. The `abline` function does this for us quite nicely:

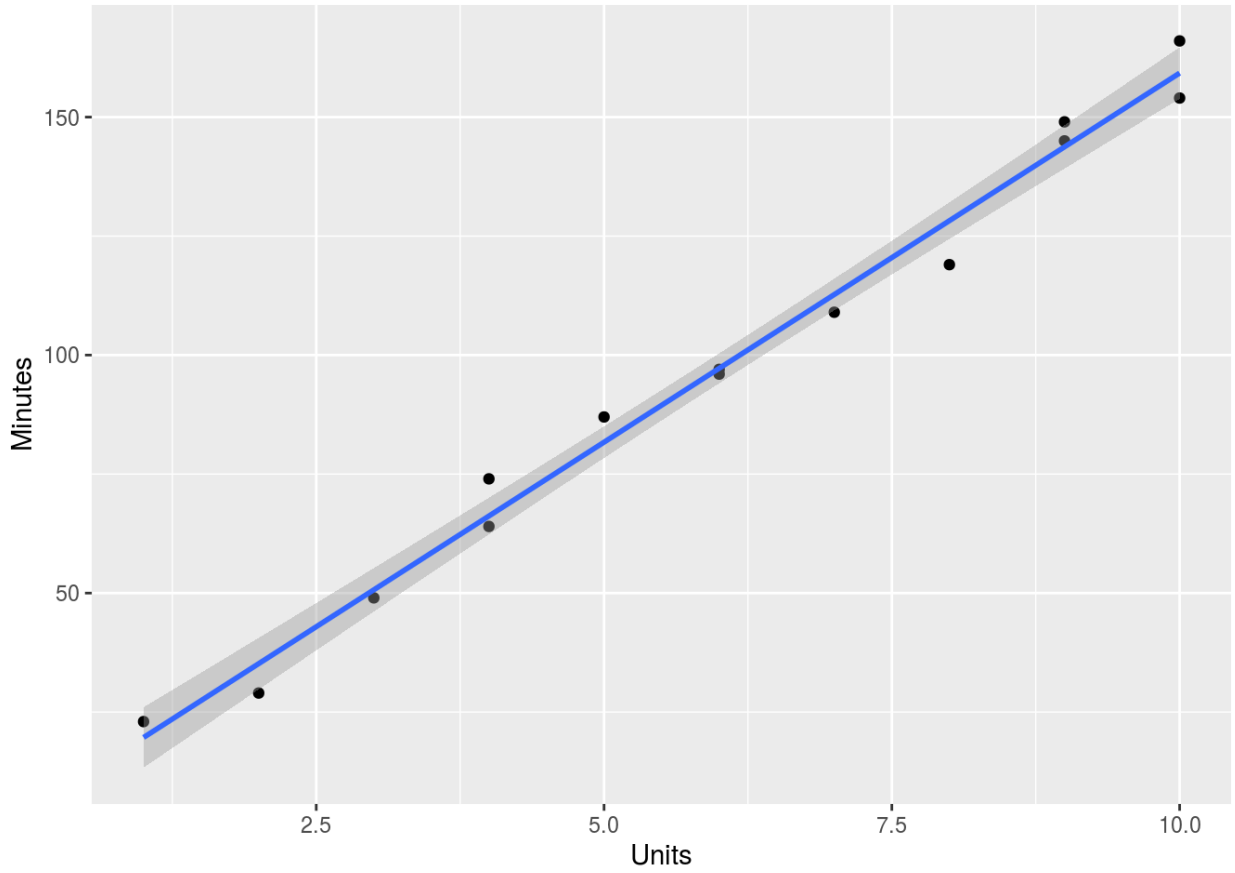
```
> plot(Minutes~Units, repairs)
> abline(repairs_lm)
```



- you could have also used the function call `abline(a=beta[1], b=beta[2])`, where the arguments `a` and `b` represent the intercept and slope of the straight line being drawn

Uou are again reminded of the `ggplot2` R package, which is part of the `tidyverse`, which is another very popular way of creating very customizable graphics in R. Here is a teaser on how such a plot could be created in with `ggplot`:

```
> library(ggplot2) # loads the ggplot2 package
> myPlot <- ggplot(repairs, aes(x=Units, y=Minutes)) +
+   geom_point() +
+   geom_smooth(method='lm')
> myPlot
```



You will not be required to use the `ggplot2` package in *this* course (I cannot speak for other instructors in the program), but if you will be performing data visualizations on a fairly regular basis then learning how to use this package effectively will be something you should put on your to-do list. Let's dissect the above code a bit.

Click 'Next' to review.

Next

Exercise

Put your R skills to the test with one more example. Consider the data frame `faithful`, which contains data on the duration of an eruption (measured in minutes) of the Old Faithful geyser at Yellowstone National Park and the amount of time that passes until the next eruption (also measured in minutes).

Use the given R application in Canvas (or of course you can use your own desktop version of R) to perform the following tasks:

- create a design matrix which models `waiting` as the response variable and `eruptions` as the explanatory variable in a simple linear regression model
- use matrix operators to calculate the model coefficient estimates $\boldsymbol{\beta}$, the set of fitted values $\hat{Y}$, and the set of residuals $e=\hat{Y}-Y$
- verify that $\beta_1 = \text{Cor}(x,y) \cdot s_y s_x$ and $\beta_0 = \bar{y} - \beta_1 \bar{x}$
- calculate the sum of the squared residuals, SSE
- plot the data in a scatter plot, with the estimated regression line overlaid on top of it

Then...

- investigate the plot and make note of any interesting features
- provide a physical interpretation to the estimated slope β_1
- decide whether any meaningful physical interpretation can be provided to the estimated intercept β_0
- predict what would happen to β_0 and β_1 if we changed the explanatory variable `eruptions` to be measured in seconds rather than minutes (then verify with the use of `lm`)

(The following code is provided as a sample. You may clear it to input new code for your exercise.)

Lesson 1.5: Statistical Inference in Simple Linear Regression

Introduction

Required Reading

- Sections 2.6-2.12

Note that probability actually plays no role in estimating the model coefficients β_0 and β_1. Obtaining the least-squares estimates is essentially an exercise in calculus (with some linear algebra thrown into the mix).

However, any value that can come from a fitted regression model will generally come from its ability to

- inform the experimenter about the relationship between an explanatory and response variable
- make useful predictions about a potential observation that could be made in the future

For these tasks, we do need the ability to model the uncertainty associated with observations, and probability is one of the best tools we have for such modeling.

- the *hypothesis test* will play a primary role in forming decisions about whether or not an explanatory variable has an important role in explaining away observed variation in a response variable
    - inference will be based solely upon p-values in this course, so there will be no more need to calculate critical values of rejection regions depending on some choice of α
- the *prediction interval* will play the role of being able to make a prediction about a future observation

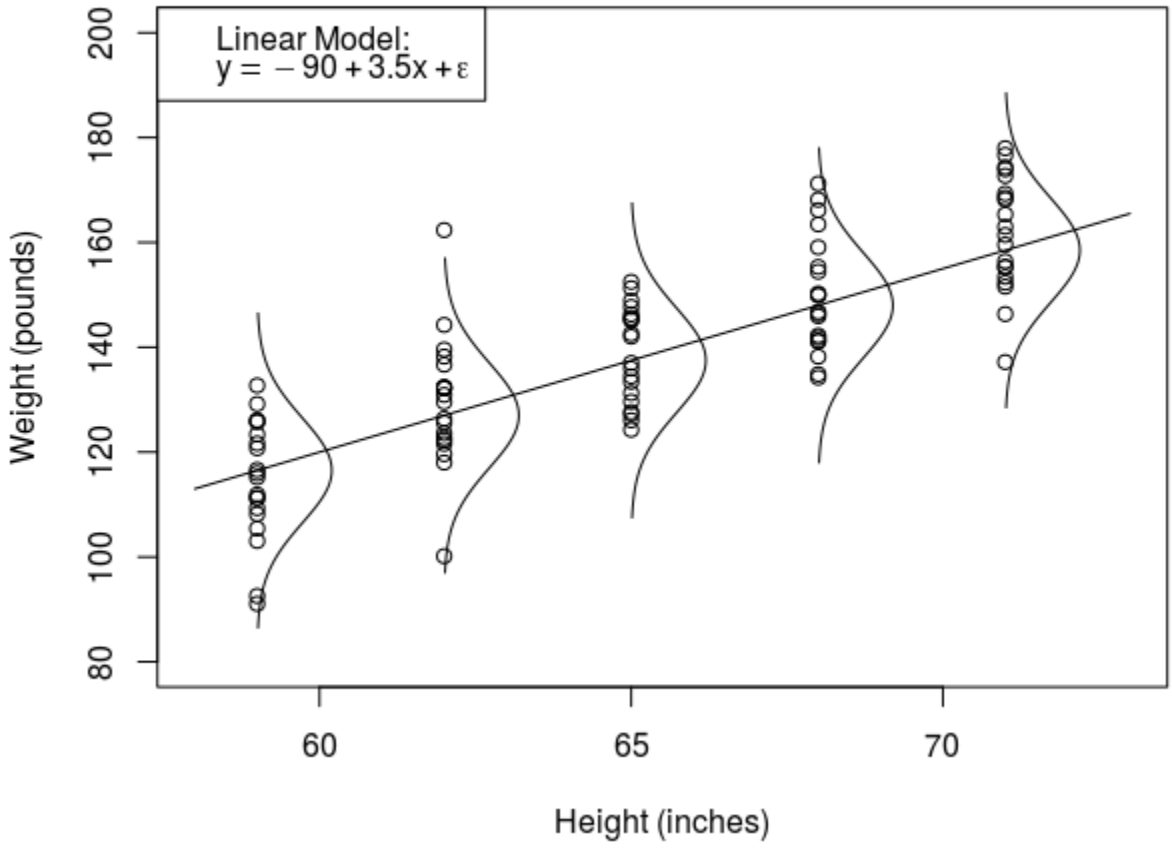
Estimating the Variability in the Error

We will return to the assumptions that lie behind a linear regression model at a later point, but you should recall from previous courses that one of the more typical assumptions in classical inference is that the data in question is normally distributed. It is this assumption which makes the use of distributions like Student's t, the χ^2, and the F distribution relevant.

For now, given the linear model $y=\beta_0+\beta_1 x+\varepsilon$

we will assume of the error term that ε is normally distributed with a mean of 0 and a variance of σ^2. Recall that the error term represents *deviations* of the observed response y from what is predicted by the model $(\beta_0+\beta_1 x)$.

To illustrate this concept further, consider the hypothetical sceneario where we model height and weights of adult women via the model $y=-90+3.5x+\varepsilon$ where the response y is weight (in pounds), the explanatory variable x is height (in inches), and the error is normally distributed with a standard deviation of 10 pounds. If this model were actually reflective of reality, we could envision sampling several sets of women from the population with varying heights (in practice, this would be an unreasonable sampling scheme) and potentially observe data as illustrated in the figure below:



According to this model, when height is fixed at some particular value, say $x=65$, weights will be normally distributed with a mean of $-90+3.5\cdot 65=137.5$ pounds, varying around this mean with a standard deviation of $\sigma=10$ (suggesting around 95% of weights will lie in the range $\mu\pm 2\sigma\approx 137.5\pm 2\cdot 10=[107.5,157.5]$).
The purpose of this illustration is to note that, not only must we find estimates of the model coefficients β_0 and β_1 when given a set of data, there is one more parameter that needs to be estimated: σ^2, the variance in the error.

The point estimates for the model coefficients were found to be $\boldsymbol{\beta} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{Y}$ If σ^2 is the variance of the error term, it stands to reason that a point estimate for σ^2 should be the variance of the *observed errors* (residuals). This is essentially correct: $\hat{\sigma}^2 = \frac{1}{n-2}\sum_{j=1}^{n}(e_j - \bar{e})^2 = \frac{1}{n-2}\sum_{j=1}^{n} e_j^2 = \frac{SSE}{n-2}$

- the method of least-squares estimation ensures us that $\bar{e} = 0$ is always true (so the residuals are centered at 0)
- we divide by $n-2$ (instead of $n-1$) in the formula for the sample variance

Recall that σ^2 is the average of the *square deviation* of a random variable from its mean: $\sigma^2 = \mathrm{E}[(Y - \mu_Y)^2]$

In estimating this quantity from a sample, we could use...

- $\hat{\sigma}^2 = \frac{1}{n}\sum_{j=1}^{n}(y_j - \mu_Y)^2$, with n degrees of freedom in the denominator
  - this presuppose that we "*know*" the true mean μ_Y, which is in practice never a reasonable assumption
- $\hat{\sigma}^2 = \frac{1}{n-1}\sum_{j=1}^{n}(y_j - \bar{y})^2$, with $n-1$ degrees of freedom in the denominator
  - essentially, one degree of freedom is used up in estimating μ_Y with $\bar{y}$
- $\hat{\sigma}^2 = \frac{1}{n-2}\sum_{j=1}^{n}(y_j - \beta_0 - \beta_1 x_j)^2$ in the context of simple linear regression
  - here, we now estimate μ_Y as being dependent upon some extra information provided by the regressor x_j
  - two degrees of freedom are used up in estimating μ_Y (via β_0 and β_1), leaving us with $n-2$ degrees of freedom in the denominator

In this course, we shall use the notation $\hat{\sigma}^2 = \frac{SSE}{n-2}$ and MSE interchangeably, though MSE is more typically reserved in the context of creating an ANOVA table. Then our estimate of the standard deviation, which is in units of y and has more interpretability, is $\hat{\sigma} = \sqrt{\hat{\sigma}^2} = \sqrt{\mathrm{T}frac{SSE}{n-2}}$

Let's obtain this estimate from the computer repair data. Make sure you have evaluated all of the commands in the R program from the previous lesson:

```
> sigma.sq <- SSE / (n-2)
> sigma.sq
        [,1]
[1,] 29.0707
> sigma <- sqrt(sigma.sq)
```

```
> sigma
         [,1]
[1,] 5.391725
```

These could have been obtained from the use of the matrix operations. Using the more convenient `lm` function, note that $\hat{\sigma}$ is output as the *Residual standard error* in the `summary` of a linear model:

```
> summary(repairs_lm)

Call:
lm(formula = Minutes ~ Units, data = repairs)

Residuals:
    Min      1Q  Median      3Q     Max
-9.2318 -3.3415 -0.7143  4.7769  7.8033

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    4.162      3.355    1.24    0.239
Units         15.509      0.505   30.71 8.92e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.392 on 12 degrees of freedom
Multiple R-squared:  0.9874,    Adjusted R-squared:  0.9864
F-statistic: 943.2 on 1 and 12 DF,  p-value: 8.916e-13
```

It would be nice to be able to extract that number from this output and store it as a variable, in case it should be needed for future computations. You could of course just copy down the numbers, such as

```
> sigma <- 5.39
```

but this could be prone to rounding or transcription errors. Let's first store the `summary` of the model with a variable name, and then apply the `names` function to this summary object:

```
> repairs_summ <- summary(repairs_lm)
> names(repairs_summ)
 [1] "call"          "terms"         "residuals"     "coefficients"
 [5] "aliased"       "sigma"         "df"            "r.squared"
 [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

Some of these items need not be worried about in this course, but you will note that there is a `sigma` in this list. The output of the `summary` function is at its core a list, and so this value of `sigma` can be extracted with the `$` operator:

```
> sigma <- repairs_summ$sigma
> sigma
[1] 5.391725
```

This could also be obtained with the `deviance` and `df.residual` functions applied to the output of `lm`:

```
> deviance(repairs_lm) # this is SSE
[1] 348.8484
> df.residual(repairs_lm) # this is n-2, the degrees of freedom
```

```
[1] 12
> deviance(repairs_lm) / df.residual(repairs_lm) # this is MSE
[1] 29.0707
> sqrt(deviance(repairs_lm) / df.residual(repairs_lm)) # this is sigma-hat
[1] 5.391725
```

While it is not overly common to perform statistical inference directly on σ, the mathematical theory implies that $\frac{SSE}{\sigma^2} \sim \chi^2_{2n-2}$ You could, if so desired, construct a confidence interval on σ^2: $\left(\frac{SSE}{\chi^2_{2n-2,1-\alpha/2}}, \frac{SSE}{\chi^2_{2n-2,\alpha/2}}\right)$ would be a $100(1-\alpha)$ confidence interval on σ^2. Take the square root of this to get a corresponding confidence interval on σ.

Return to the linear model of `waiting~eruptions` in the the `faithful` data set.

You may use the R-code simulator to answer the following questions. (The following code is provided as a sample. You may clear it to input new code for your exercise.) Click 'Run' to check your code.

Comparing Reduced and Full Models

The "goodness" of our linear model can be measured in part by determining whether or not x provides a sufficient amount of power in explaining away the variability in the response variable. This can be done by performing a hypothesis test to choose between two potential models:
$H_0: y = \beta_0 + \varepsilon$ against $H_a: y = \beta_0 + \beta_1 x + \varepsilon$

Consider a few illustrations of this notion before jumping into the compuations.

Click each tab to read more.

Duration of Employment
Computer Repairs

Inference on the Model Coefficients

We can view a hypothesis test on the goodness of fit for a simple linear regression model as comparing the reduced and full models: $H_0: y = \beta_0 + \varepsilon$ against $H_a: y = \beta_0 + \beta_1 x + \varepsilon$ Note that the "difference" between these two models is just the inclusion of the term $\beta_1 x$, so that the above hypotheses are equivalent to testing $H_0: \beta_1 = 0$ against $H_1: \beta_1 \neq 0$

Under the assumption that ε is normally distributed with a mean of 0 and a variance of σ^2 (and some other assumptions that will be investigated later in the course), mathematical theory implies that $\boldsymbol{\hat{\beta}}$ is normally distributed with mean $\boldsymbol{\beta}$ and covariance matrix $\sigma^2(\boldsymbol{X}^\top\boldsymbol{X})^{-1}$ What does it mean for $\boldsymbol{\hat{\beta}}$ to have a normal distribution?

- the individual components $\hat{\beta}_1$ and $\hat{\beta}_0$ both have normal distributions
- any linear combination $c_1\hat{\beta}_0+c_2\hat{\beta}_1$ also has a normal distribution, with mean and variance determined by the constants c_1, c_2, as well as the mean $\boldsymbol{\beta}$ and covariance matrix $\sigma^2(\boldsymbol{X}^\top\boldsymbol{X})^{-1}$)

In the case of simple linear regression, this covariance matrix has a relatively simple form:

$$\text{Cov}[\hat{\beta}_0\hat{\beta}_1]=[\text{Cov}(\hat{\beta}_0,\hat{\beta}_0)\text{Cov}(\hat{\beta}_0,\hat{\beta}_1)\text{Cov}(\hat{\beta}_0,\hat{\beta}_1)\text{Cov}(\hat{\beta}_1,\hat{\beta}_1)]=[\text{Var}(\hat{\beta}_0)\text{Cov}(\hat{\beta}_0,\hat{\beta}_1)\text{Cov}(\hat{\beta}_0,\hat{\beta}_1)\text{Var}(\hat{\beta}_1)]$$
$$=\begin{bmatrix}\sigma^2\left(\frac{1}{n}+\bar{x}^2\sum_j(x_j-\bar{x})^2\right) & -\sigma^2\bar{x}\sum_j(x_j-\bar{x})^2 \\ -\sigma^2\bar{x}\sum_j(x_j-\bar{x})^2 & \sigma^2\sum_j(x_j-\bar{x})^2\end{bmatrix}$$

Don't worry about these formulas! They are messy and rather unilluminating, and moreover do not generalize well to the situation in a multiple regression model.
The important part to remember is that $(\boldsymbol{X}^\top\boldsymbol{X})^{-1}$ could be called the **unscaled covariance matrix** of the coefficient estimates. Recall that this was one of the objects found in the `summary` of the `lm`:

```
> repairs_summ$cov.unscaled
             (Intercept)       Units
(Intercept)  0.38721805 -0.05263158
Units       -0.05263158  0.00877193
```

This is the same as the output of `solve(crossprod(X))` (only a bit nicer with row and column names). It is called *unscaled* as we need multiply all of the elements in this matrix by σ^2 to get the variance of $\hat{\beta}_1$ or $\hat{\beta}_0$ (or to get the covariance between $\hat{\beta}_0$ and $\hat{\beta}_1$).
We will have little need to consider the covariance between the model coefficients (at the moment), but their variance plays a direct role in inference about the model coefficients β_0 and β_1. Particularly, $\hat{\beta}_1-\beta_1\text{se}(\hat{\beta}_1)=\hat{\beta}_1-\beta_1\sigma/\sqrt{\sum_j(x_j-\bar{x})^2}\sim N(0,1)$ If σ were known to us (which should essentially never be assumed), a z-test for testing hypotheses about β_1 could be constructed.
As σ is not known, we may replace σ with the estimate $\hat{\sigma}$: $\hat{\beta}_1-\beta_1\hat{\text{se}}(\hat{\beta}_1)=\hat{\beta}_1-\beta_1\hat{\sigma}/\sqrt{\sum_j(x_j-\bar{x})^2}\sim t_{n-2}$

- recall that replacing the *constant* parameter σ with the *random* statistic $\hat{\sigma}$ introduces further variability into the standardization of $\hat{\beta}_1$, and we account for this extra variability by using Student's t distribution in lieu of the standard normal distribution

Similar results hold for the intercept estimate $\hat{\beta}_0$, though it is uncommon in practice to ever need to perform inference on the parameter β_0.

Matrix Operations

The variance of $\hat{\beta}_0$ and $\hat{\beta}_1$ are the elements on the diagonal of the covariance matrix $\sigma^2(X^\top X)^{-1}$ Point estimates of these variances are then obtained by replacing σ^2 with $\hat{\sigma}^2$, and then extracting the diagonal elements of this matrix:

```
> print(XtXinv <- solve(crossprod(X)))
            [,1]        [,2]
[1,]  0.38721805 -0.05263158
[2,] -0.05263158  0.00877193
> diag(XtXinv)
[1] 0.38721805 0.00877193
> sigma.sq * diag(XtXinv)
Warning in sigma.sq * diag(XtXinv): Recycling array of length 1 in array-ve
ctor arithmetic is deprecated.
  Use c() or as.vector() instead.
[1] 11.2566987  0.2550061
```

Oops! When using matrix operations, we must be mindful that a 1x1 matrix is still a matrix (even though it may also be thought of as a single number). We must explicity coerce `sigma.sq` into being a single number and then a warning message won't be printed:

```
> sigma.sq <- as.vector(sigma.sq)
> print(beta.var <- sigma.sq * diag(XtXinv))
```

```
[1] 11.2566987  0.2550061
```

The standard errors of either $\hat{\beta}_0$ or $\hat{\beta}_1$ are just the square roots of these variances:

```
> print(beta.se <- sqrt(beta.var))
[1] 3.3551004 0.5049813
```

Confidence intervals and t-tests can now be performed given the point estimates $\hat{\beta}_0$ and $\hat{\beta}_1$, along with their respective standard errors. For instance, a 90% confidence interval on β_1 is $\hat{\beta}_1 \pm t_{0.05,df} \hat{\text{se}}(\hat{\beta}_1)$ and calculated as follows:

```
> print(t.crit <- qt(.95, n-2)) # could be obtained from a t-table
[1] 1.782288
> beta.hat[2] + c(-1,1) * t.crit * beta.se[2]
[1] 14.60898 16.40902
```

- With 90% confidence, we can say that each extra unit that needs to be repaired adds anywhere from 14.6 to 16.4 minutes to the average length of a service call.
- Recall that we **do not** interpret confidence as meaning that there is a 90% chance that β_1 lies inside this interval.
- Rather, 90% of all such confidence intervals that could be calculated by replicating this experiment will contain the parameter β_1.

To test $H_0: \beta_1 = 0$ against $H_a: \beta_1 \neq 0$, use the test statistic $t = \frac{\hat{\beta}_1 - 0}{\hat{\text{se}}(\hat{\beta}_1)}$ and calculate the p-value:

```
> print(t <- beta.hat[2]/beta.se[2])
[1] 30.71203
> 2*pt(t, n-2, lower.tail=FALSE)
[1] 8.914699e-13
```

Recall that, in a *two-tailed test*, we must multiply the tail probability associated with the test statistic by 2.

Calculate the p-value in the test of H_0:β_1=12 against H_a:β_1>12, then check your answer by clicking 'Reveal.'

Use of `lm`

The `summary` function applied to the output of `lm` provides several pieces of information:

```
> repairs_summ

Call:
lm(formula = Minutes ~ Units, data = repairs)

Residuals:
    Min      1Q  Median      3Q     Max
-9.2318 -3.3415 -0.7143  4.7769  7.8033

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    4.162      3.355    1.24    0.239
Units         15.509      0.505   30.71 8.92e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.392 on 12 degrees of freedom
Multiple R-squared:  0.9874,    Adjusted R-squared:  0.9864
F-statistic: 943.2 on 1 and 12 DF,  p-value: 8.916e-13
```

- the function *call* (this will be useful when you are fitting many different models at once, as it helps to remind you of all the variables in the model)
- the five-number summary on the set of *residuals*

    - a more proper investigation of the residuals will be delayed until the third module
- a table of the estimated model coefficients ($\hat{\beta}_0$ and $\hat{\beta}_1$), their standard errors, and t-values and p-values in the test of $H_0: \beta_0 = 0$ or $H_0: \beta_1 = 0$ against their two-sided alternatives
    - note that $\hat{\beta}_j$, $\widehat{se}(\hat{\beta}_j)$, and the t-value and p-value agree with the computations obtained via matrix operations
- the *residual standard error*, which is $\hat{\sigma}$
- the *multiple* R^2 and *adjusted* R^2 (more will be said about these soon)
- the test statistic and p-value in a related F-test
    - we'll return to the F-test in the next module, but note that the p-value is precisely the same as the p-value in the t-test on β_1

What are some of the other functions that could be applied to the output of `lm`?

- `coefficients(repairs_lm)` prints the estimated model coefficients $\hat{\beta}_0$ and $\hat{\beta}_1$
- `deviance(repairs_lm)` prints SSE
- `df.residual(repairs_lm)` prints the error degrees of freedom
- `residuals(repairs_lm)` is the vector of residuals e_j
- `fitted(repairs_lm)` is the vector of fitted values $\hat{y}_j$
- `confint(repairs_lm)` is a set of 95% confidence intervals on $\hat{\beta}_0$ and $\hat{\beta}_1$
    - `confint(repairs_lm, level=0.9)` will report a set of 90% confidence intervals
- `vcov(repairs_lm)` is the estimated covariance matrix of $\hat{\boldsymbol{\beta}}$; namely, $\hat{\sigma}^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}$

You'll be getting plenty of practice with these functions throughout the course, but at least they're named in a relatively sane fashion!

Exercise

Return to the `faithful` data set, and write some R code which can be used to answer the following questions.

You may use the R-code simulator to answer the following questions. (The following code is provided as a sample. You may clear it to input new code for your exercise.) Click 'Run' to check your code.

Enter your answers for the following questions in the spaces provided. Click 'Check' to check your answers and click 'Next' to advance.

Enter your answers for the following questions in the spaces provided. Click 'Reveal' to check your answers.

1. What is the value of the t-statistic in the test of $H_0:\beta_1=0$ against $H_a:\beta_1\neq 0$?
2. Determine the p-value in the test of $H_0:\beta_1=10$ against $H_a:\beta_1>10$, rounded to the nearest 0.0001. You'll have to use the output from `summary` to do some calculations "by hand" (or rather, with `pt`).

Comparing Reduced and Full Models

The "goodness" of our linear model can be measured in part by determining whether or not x provides a sufficient amount of power in explaining away the variability in the response variable. This can be done by performing a hypothesis test to choose between two potential models:
$H_0:y=\beta_0+\varepsilon$ against $H_a:y=\beta_0+\beta_1 x+\varepsilon$

Consider a few illustrations of this notion before jumping into the compuations.

Click each tab to read more.

Duration of Employment
Computer Repairs

Inference on the Model Coefficients

We can view a hypothesis test on the goodness of fit for a simple linear regression model as comparing the reduced and full models: $H_0:y=\beta_0+\varepsilon$ against $H_a:y=\beta_0+\beta_1 x+\varepsilon$ Note that the "difference" between these two models is just the inclusion of the term $\beta_1 x$, so that the above hypotheses are equivalent to testing $H_0:\beta_1=0$ against $H_1:\beta_1\neq 0$
Under the assumption that ε is normally distributed with a mean of 0 and a variance of σ^2 (and some other assumptions that will be investigated later in the course), mathematical theory implies that $\boldsymbol{\beta}$ is normally distributed with mean $\boldsymbol{\beta}$ and covariance matrix $\sigma^2(X^TX)^{-1}$ What does it mean for $\boldsymbol{\beta}$ to have a normal distribution?

- the individual components β_1 and β_0 both have normal distributions

- any linear combination $c_1\hat{\beta}_0 + c_2\hat{\beta}_1$ also has a normal distribution, with mean and variance determined by the constants c_1, c_2, as well as the mean $\boldsymbol{\beta}$ and covariance matrix $\sigma^2(\boldsymbol{X}^\top\boldsymbol{X})^{-1}$

In the case of simple linear regression, this covariance matrix has a relatively simple form:

$$\text{Cov}[\hat{\beta}_0\hat{\beta}_1] = [\text{Cov}(\hat{\beta}_0,\hat{\beta}_0)\text{Cov}(\hat{\beta}_0,\hat{\beta}_1)\text{Cov}(\hat{\beta}_0,\hat{\beta}_1)\text{Cov}(\hat{\beta}_1,\hat{\beta}_1)] = [\text{Var}(\hat{\beta}_0)\text{Cov}(\hat{\beta}_0,\hat{\beta}_1)\text{Cov}(\hat{\beta}_0,\hat{\beta}_1)\text{Var}(\hat{\beta}_1)]$$
$$= \begin{bmatrix} \sigma^2\left(\frac{1}{n} + \bar{x}^2\sum_j (x_j-\bar{x})^2\right) & -\sigma^2\bar{x}\sum_j (x_j-\bar{x})^2 \\ -\sigma^2\bar{x}\sum_j (x_j-\bar{x})^2 & \sigma^2\sum_j (x_j-\bar{x})^2 \end{bmatrix}$$

Don't worry about these formulas! They are messy and rather unilluminating, and moreover do not generalize well to the situation in a multiple regression model.
The important part to remember is that $(\boldsymbol{X}^\top\boldsymbol{X})^{-1}$ could be called the **unscaled covariance matrix** of the coefficient estimates. Recall that this was one of the objects found in the `summary` of the `lm`:

```
> repairs_summ$cov.unscaled
            (Intercept)       Units
(Intercept)  0.38721805 -0.05263158
Units       -0.05263158  0.00877193
```

This is the same as the output of `solve(crossprod(X))` (only a bit nicer with row and column names). It is called *unscaled* as we need multiply all of the elements in this matrix by σ^2 to get the variance of $\hat{\beta}_1$ or $\hat{\beta}_0$ (or to get the covariance between $\hat{\beta}_0$ and $\hat{\beta}_1$).
We will have little need to consider the covariance between the model coefficients (at the moment), but their variance plays a direct role in inference about the model coefficients β_0 and β_1. Particularly, $\frac{\hat{\beta}_1 - \beta_1}{\text{se}(\hat{\beta}_1)} = \frac{\hat{\beta}_1 - \beta_1}{\sigma/\sqrt{\sum_j (x_j-\bar{x})^2}} \sim N(0,1)$ If σ were known to us (which should essentially never be assumed), a z-test for testing hypotheses about β_1 could be constructed.
As σ is not known, we may replace σ with the estimate $\hat{\sigma}$: $\frac{\hat{\beta}_1 - \beta_1}{\hat{\text{se}}(\hat{\beta}_1)} = \frac{\hat{\beta}_1 - \beta_1}{\hat{\sigma}/\sqrt{\sum_j (x_j-\bar{x})^2}} \sim t_{n-2}$

- recall that replacing the *constant* parameter σ with the *random* statistic $\hat{\sigma}$ introduces further variability into the standardization of $\hat{\beta}_1$, and we account for this extra variability by using Student's t distribution in lieu of the standard normal distribution

Similar results hold for the intercept estimate $\hat{\beta}_0$, though it is uncommon in practice to ever need to perform inference on the parameter β_0.

Matrix Operations

The variance of $\hat{\beta}_0$ and $\hat{\beta}_1$ are the elements on the diagonal of the covariance matrix $\sigma^2(X^TX)^{-1}$ Point estimates of these variances are then obtained by replacing σ^2 with $\hat{\sigma}^2$, and then extracting the diagonal elements of this matrix:

```
> print(XtXinv <- solve(crossprod(X)))
            [,1]        [,2]
[1,]  0.38721805 -0.05263158
[2,] -0.05263158  0.00877193
> diag(XtXinv)
[1] 0.38721805 0.00877193
> sigma.sq * diag(XtXinv)
Warning in sigma.sq * diag(XtXinv): Recycling array of length 1 in array-ve
ctor arithmetic is deprecated.
  Use c() or as.vector() instead.
[1] 11.2566987  0.2550061
```

Oops! When using matrix operations, we must be mindful that a 1x1 matrix is still a matrix (even though it may also be thought of as a single number). We must explicity coerce `sigma.sq` into being a single number and then a warning message won't be printed:

```
> sigma.sq <- as.vector(sigma.sq)
> print(beta.var <- sigma.sq * diag(XtXinv))
[1] 11.2566987  0.2550061
```

The standard errors of either $\hat{\beta}_0$ or $\hat{\beta}_1$ are just the square roots of these variances:

```
> print(beta.se <- sqrt(beta.var))
```

```
[1] 3.3551004 0.5049813
```

Confidence intervals and t-tests can now be performed given the point estimates $\beta\ 0$ and $\beta\ 1$, along with their respective standard errors. For instance, a 90% confidence interval on $\beta 1$ is $\beta\ 1 \pm t_{0.05,df} \hat{se}(\beta\ 1)$ and calculated as follows:

```
> print(t.crit <- qt(.95, n-2)) # could be obtained from a t-table
[1] 1.782288
> beta.hat[2] + c(-1,1) * t.crit * beta.se[2]
[1] 14.60898 16.40902
```

- With 90% confidence, we can say that each extra unit that needs to be repaired adds anywhere from 14.6 to 16.4 minutes to the average length of a service call.
- Recall that we **do not** interpret confidence as meaning that there is a 90% chance that $\beta 1$ lies inside this interval.
- Rather, 90% of all such confidence intervals that could be calculated by replicating this experiment will contain the parameter $\beta 1$.

To test $H_0: \beta 1 = 0$ against $H_a: \beta 1 \neq 0$, use the test statistic $t = \beta\ 1 - 0 \hat{se}(\beta\ 1)$ and calculate the p-value:

```
> print(t <- beta.hat[2]/beta.se[2])
[1] 30.71203
> 2*pt(t, n-2, lower.tail=FALSE)
[1] 8.914699e-13
```

Recall that, in a *two-tailed test*, we must multiply the tail probability associated with the test statistic by 2.

Calculate the p-value in the test of $H_0: \beta 1 = 12$ against $H_a: \beta 1 > 12$, then check your answer by clicking 'Reveal.'

Use of `lm`

The `summary` function applied to the output of `lm` provides several pieces of information:

```
> repairs_summ

Call:
lm(formula = Minutes ~ Units, data = repairs)

Residuals:
    Min      1Q  Median      3Q     Max
-9.2318 -3.3415 -0.7143  4.7769  7.8033

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    4.162      3.355    1.24    0.239
Units         15.509      0.505   30.71 8.92e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.392 on 12 degrees of freedom
Multiple R-squared:  0.9874,    Adjusted R-squared:  0.9864
F-statistic: 943.2 on 1 and 12 DF,  p-value: 8.916e-13
```

- the function *call* (this will be useful when you are fitting many different models at once, as it helps to remind you of all the variables in the model)
- the five-number summary on the set of *residuals*
    - a more proper investigation of the residuals will be delayed until the third module
- a table of the estimated model coefficients ($\hat{\beta}_0$ and $\hat{\beta}_1$), their standard errors, and t-values and p-values in the test of H_0:β_0=0 or H_0:β_1=0 against their two-sided alternatives

    - note that $\hat{\beta}_j$, $\widehat{se}(\hat{\beta}_j)$, and the t-value and p-value agree with the computations obtained via matrix operations
- the *residual standard error*, which is $\hat{\sigma}$
- the *multiple* R^2 and *adjusted* R^2 (more will be said about these soon)
- the test statistic and p-value in a related F-test
    - we'll return to the F-test in the next module, but note that the p-value is precisely the same as the p-value in the t-test on β_1

What are some of the other functions that could be applied to the output of `lm`?

- `coefficients(repairs_lm)` prints the estimated model coefficients $\hat{\beta}_0$ and $\hat{\beta}_1$
- `deviance(repairs_lm)` prints SSE
- `df.residual(repairs_lm)` prints the error degrees of freedom
- `residuals(repairs_lm)` is the vector of residuals e_j
- `fitted(repairs_lm)` is the vector of fitted values $\hat{y}_j$
- `confint(repairs_lm)` is a set of 95% confidence intervals on $\hat{\beta}_0$ and $\hat{\beta}_1$
    - `confint(repairs_lm, level=0.9)` will report a set of 90% confidence intervals
- `vcov(repairs_lm)` is the estimated covariance matrix of $\hat{\boldsymbol{\beta}}$; namely, $\hat{\sigma}^2(\boldsymbol{X}^\top\boldsymbol{X})^{-1}$

You'll be getting plenty of practice with these functions throughout the course, but at least they're named in a relatively sane fashion!

Exercise

Return to the `faithful` data set, and write some R code which can be used to answer the following questions.

You may use the R-code simulator to answer the following questions. (The following code is provided as a sample. You may clear it to input new code for your exercise.) Click 'Run' to check your code.

Enter your answers for the following questions in the spaces provided. Click 'Check' to check your answers and click 'Next' to advance.

Enter your answers for the following questions in the spaces provided. Click 'Reveal' to check your answers.

1. What is the value of the t-statistic in the test of H_0:β_1=0 against H_a:β_1≠0?

2. Determine the p-value in the test of $H_0: \beta_1 = 10$ against $H_a: \beta_1 > 10$, rounded to the nearest 0.0001. You'll have to use the output from `summary` to do some calculations "by hand" (or rather, with `pt`).