# Development for Learning Modules on Data Science for Energy and Buildings

*Catherine R Dressler*

## Contents

# 1 Introduction to Data Science for Energy and Buidings

The content of this section relies heavily on Agami Reddy's book, "Applied Data Analysis and Modeling for Energy Engineers and Scientists." We have also chosen to maintain the books figure and example numbers in order for students to easily follow along.

- Reddy, T.Agami. Applied Data Analysis and Modeling for Energy Engineers and Scientists. Springer, 2011.
- The R applications for the distributions are adapted from http://www.r-tutor.com/elementary-statistics.

Additional sources can be found in the references.

*By the completion of this section, the reader should be able to*

- *understand the basics of probability*
- *identify and interpret probability distributions*
- *understand the difference between classical and frequentist views of probability*
- *become comfortable with the r interface*

## 1.1 Fundamentals of Probability

To succeed in this class, it is important to master the basic concepts on which statistical analysis is built. In this section we will review some of the basics of probability.

There are two main schools of thought when it comes to probability, classical (or frequentist) and Bayesian. In this section we will focus on the **classical viewpoints** of probability. Predicting outcomes from random data can prove to be difficult. However, if a large set of data from the same experiment is evaluated, more stable predictions can be observed. This idea comes from the strong law of large numbers which states that the average of a sequence of independent random variables having the same distribution will converge to the mean of that distribution.

Random variables are values that represent all of the possible outcomes of chance experiments. These experiments include processes that are not entirely predictable, such as coin tosses or dice rolls. It is common practice to calculate the probability or expected value of these random variables. There are two types of random variables.

- A **discrete random variable** can take on a finite number of values.
- A **continuous random variable** can take on any value in an interval.

We will assume that the reader has a basic knowledge of discrete random variables and will begin this section with a focus on continuous random variables.

### 1.1.1 Density Functions

- The **probability density function** (PDF) expresses the probability of a random variable taking on various different values

For a discrete random variable, the function will take on a single value at discrete points along the $x$-axis. Figure 1 represents the probability distribution function for data that describes a the dry bulb temperature in New York for one year. It represents the probability of a certain temperature being observed. As you can see, the function for continuous random variables is itself continuous. In this case, one might be more interested in finding the probability of the occurrence of a range of values or temperatures. This probability can be found by finding the area under the curve of the probability density function. The area underneath the entire curve is known as the cumulative distribution function (CDF).

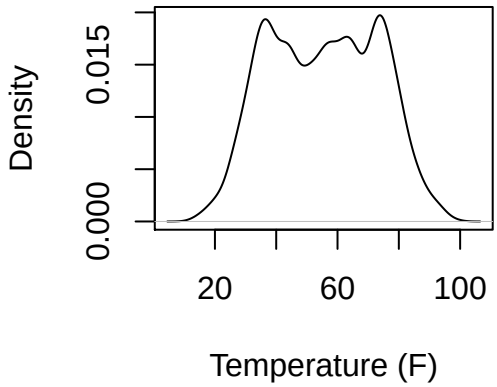



Figure 1: PDF of annual temperature data from New York.

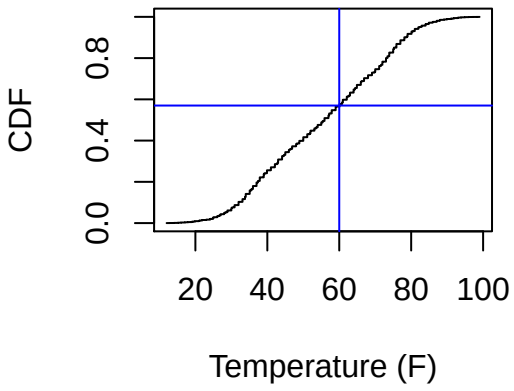



Figure 2: CDF of annual temperature data from New York

- The **cumulative distribution function** (CDF) represents the area under the curve of the, or integral, of the PDF. It expresses the probability that the random variable assumes a value **less than or equal** to various values. For example, the probability that the temperature is less than or equal to 60F in New York for during this year is around 56%.

### 1.1.2 Density Function Axioms for Continuous Random Variables

Let $X$ be a random variable with a probability distribution function, $f(x)$, and a cumulative distribution function, $F(x)$:

- The PDF cannot be negative: $f(x) \geq 0$, $-\infty < x < \infty$
- The probability of all outcomes must sum to unity:

$$\int_{-\infty}^{\infty} f(x)dx=1$$

- The CDF represents the area under the PDF enclosed in the range $-\infty < x < a$:

$$F(a) = p(X \leq a) = \int_{-\infty}^{a} f(x)dx$$

- The inverse relationship between $f(x)$ and $F(a)$, provided a derivative exits, is:

$$f(x) = \frac{dF(x)}{dx}$$

This leads to the probability of an outcome $a \leq X \leq b$ given by:

$$p(a \leq X \leq b) = \int_{a}^{b} f(x)dx = \int_{-\infty}^{b} f(x)dx - \int_{\infty}^{a} f(x)dx = F(b) - F(a)$$

**Example 2.3.2**: The operating life of a high efficiency air filter in an industrial plant is a random variable $X$ having the PDF:

$$f(x) = \frac{20}{(x+100)^3} \text{ for } x > 0$$

Find the probability that the filter will have an operating life of

i) at least 20 weeks
ii) anywhere between 80 and 120 weeks:

$$\text{CDF}=\int_{x}^{0} \frac{20}{(x+100)^3}dx = [-\frac{10}{(x+100)^2}]_{x}^{0}$$

- at least 20 weeks.

$$p(20 < X < \infty) = [-\frac{10}{(x+100)^2}]_{20}^{\infty} = 0.000694$$

- anywhere between 80 and 120 week

$$p(80 < X < 120) = [-\frac{10}{(x+100)^2}]_{80}^{120} = 0.000102$$

### 1.1.3 Simultaneous Outcomes of Multiple Variables

Probability distribution functions can be a very powerful tool because they allow us to look at simultaneous outcomes of multiple random variables.

Let $X$ and $Y$ be two random variables. The probability that they occur together can be represented by a function for any pair of values within the range of variability of the random variables $X$ and $Y$.

This function is referred to as the **joint probability density function** of $X$ and $Y$ which has to satisfy the following properties for continuous variables:

- $f(x,y) \leq 0$ for all (x,y)
- $\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)dx\mathrm{d}y = 1$

### 1.1.4 Joint, Marginal, and Conditional Distributions for Continuous Variables

Joint probability distribution of two **independent** events represents the case when the events both occur together. If $X$ and $Y$ are two independent random variables, their joint probability distribution will be the product of their marginal one:

$$f(x,y) = f(x)\,f(y)$$

The marginal distribution of $X$ given two jointly distributed random variables $X$ and $Y$, is simply the probability distribution of $X$, ignoring that of $Y$. This is determined for X as:

$$g(x) = \int_{-\infty}^{\infty} f(x,y)dy$$

The **conditional probability distribution** of $X$ given that $X = x$ for two jointly distributed random variables $X$ and $Y$ is:

$$f(X|Y) = \tfrac{f(x,y)}{g(x)},\ g(x) > 0$$

### 1.1.5 Expectations and Moments

Expectations and Moments are two important measures which allow us to summarize the characteristics of a probability function. To apply these measures to discrete variables, the integrals are replaced with summations

- **Mean**:

$$E[X] \equiv \mu = \int_{-\infty}^{\infty} xf(x)dx$$

- **Variance**: Measures dispersion

$$var[X] \equiv \sigma^2 = E[(X-\mu)^2] = \int_{-\infty}^{\infty} (x-\mu)^2 f(x)dx$$

- **Standard deviation**=$(\sigma)$
- **Coefficient of Variation (CV)**: relative error is often more important than actual error. Therefore, we use CV, which is defined as the percentage ratio of the standard deviation to the mean:

$$CV = 100\,(\tfrac{\sigma}{\mu})$$

### 1.1.6 Functions of Random Variables

The measures above can also be used for the case when the random variable $X$ is a function of several random variables:

$$X = a_0 + a_1X_1 + a_2X_2$$

- Important relationships regarding the mean:

$$E[a_0] = a_0$$
$$E[a_1X_1] = a_1E[X_1]$$
$$E[a_0 + a_1X_1 + a_2X_2] = a_0 + a_1E[X_1] + a_2E[X_2]$$

- Similarly, there are important relations regarding the variance:

$$var[a_0] = 0$$
$$var[a_1 X_1] = a_1^2 var[X_1]$$

- If the variables are independent:

$$var[a_0 + a_1 X_1 + a_2 X_2] = a_1^2 var[X_1] + a_2^2 var[X_2]$$

- If the variables are not independent we use covariance. **Covariance** is the measure of the tendency of two random variables to vary together. The covariance is defined as:

$$cov[X_1, X_2] = E[(X_1 - \mu_1)(X_2 - \mu_2)]$$

So for variables that are not independent, we modify the variance to be:

$$var[a_0 + a_1 X_1 + a_2 X_2] = a_1^2 var[X_1] + a_2^2 var[X_2] + 2a_1 a_2 \cdot cov[X_1, X_2]$$

## 1.2 Distributions for Discrete Variables

The content of this section relies heavily on Agami Reddy's book, "Applied Data Analysis and Modeling for Energy Engineers and Scientists." We have also chosen to maintain the books figure and example numbers in order for students to easily follow along.

*Reddy, T.Agami. Applied Data Analysis and Modeling for Energy Engineers and Scientists. Springer, 2011.*

### 1.2.1 Bernoulli Process:

- The Bernoulli process describes experiments involving repeated trials where only two complementary outcomes are possible:"success" or "failure"
- An experiment is considered a Bernoulli process if:
  - the successive trials are independent
  - the probability of success $p$ remains constant from one trial to the next

$$C(n,x) = \binom{n}{x}$$

### 1.2.2 Binomial Distribution:

- The binomial distribution is the PDF of the Bernoulli process. It describes the distribution which gives the probability of $x$ successes in $n$ independent trials, if the probability of success in any one trial is $p$. (Note: Outcomes must be Bernoulli trials)

$$B(x:n,p) = \binom{n}{x}p^x(1-p)^{n-x}$$
mean: $\mu = (np)$
variance: $\sigma^2 = np(1-p)$

**Example 2.4.1**

Let $k$ be the number of heads in $n = 4$ independent tosses of a coin. Then the mean of the distribution=$4 \cdot (1/2)$=2, and the variance $\sigma^2 = (4) \cdot (1/2) \cdot (1–1/2) = 1$. What is the probability of two successes (x=2) in four tosses?

$$B(2;4,0.5) = \binom{4}{2}(\tfrac{1}{2})^2(1-\tfrac{1}{2})^{(4-2)} = \tfrac{4\times3}{2} \times \tfrac{1}{4} \times \tfrac{1}{4} = \tfrac{3}{8}$$

**R Applications**

In R, the binomial distribution is written using the binom function. To solve for the probability of two successes in four coin tosses as in 2.4.1, we use the d prefix which gives the density of the distribution. We write dbinom(x,size,probability).

```
dbinom(2, size=4, prob=0.5)
```

```
## [1] 0.375
```

To find the probability of 2 or less successful tosses, we can use the cumulative distribution function for the binomial distribution. For this, the p prefix is used.

```
pbinom(2, size=4, prob=0.5)
```

```
## [1] 0.6875
```

To plot the PDF of a binomial distribution in R, write:

```
n=4
p=0.5
x=(1:20)
B=dbinom(x, n, p)
```

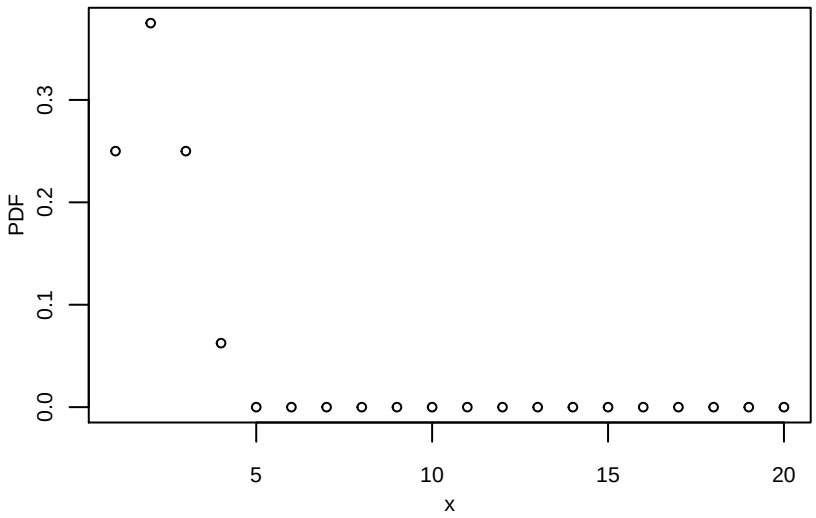



Figure 3: PDF of the binomial distribution

```
par(mgp=c(1.5,0.75,0))
plot(x,B,xlab="x",ylab="PDF", cex=0.6,
     cex.lab=0.7, cex.axis=0.7)
```

For large values of $n$, tables, such as Table A1, may be applied to the corresponding CDF, or *binomial probability sum*. This particular table illustrates the concept for $n = 15$ and $n = 20$ with varying values of $p$ and $r$.

$$B(r; n, p) = \sum_{x=0}^{r} B(x; n, p)$$

**Table A.1** Binomial probability sums $\sum_{x=0}^{r} b(x; n, p)$

| | | p | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| n | r | 0.10 | 0.20 | 0.25 | 0.30 | 0.40 | 0.50 | 0.60 | 0.70 | 0.80 | 0.90 |
| 15 | 0 | 0.2059 | 0.0352 | 0.0134 | 0.0047 | 0.0005 | 0.0000 | | | | |
| | 1 | 0.5490 | 0.1671 | 0.0802 | 0.0353 | 0.0052 | 0.0005 | 0.0000 | | | |
| | 2 | 0.8159 | 0.3980 | 0.2361 | 0.1268 | 0.0271 | 0.0037 | 0.0003 | 0.0000 | | |
| | 3 | 0.9444 | 0.6482 | 0.4613 | 0.2969 | 0.0905 | 0.0176 | 0.0019 | 0.0001 | | |
| | 4 | 0.9873 | 0.8358 | 0.6865 | 0.5155 | 0.2173 | 0.0592 | 0.0094 | 0.0007 | 0.0000 | |
| | 5 | 0.9978 | 0.9389 | 0.8516 | 0.7216 | 0.4032 | 0.1509 | 0.0338 | 0.0037 | 0.0001 | |
| | 6 | 0.9997 | 0.9819 | 0.9434 | 0.8689 | 0.6098 | 0.3036 | 0.0951 | 0.0152 | 0.0008 | |
| | 7 | 1.0000 | 0.9958 | 0.9827 | 0.9500 | 0.7869 | 0.5000 | 0.2131 | 0.0500 | 0.0042 | 0.0000 |
| | 8 | | 0.9992 | 0.9958 | 0.9848 | 0.9050 | 0.6964 | 0.3902 | 0.1311 | 0.0181 | 0.0003 |
| | 9 | | 0.9999 | 0.9992 | 0.9963 | 0.9662 | 0.8491 | 0.5968 | 0.2784 | 0.0611 | 0.0023 |
| | 10 | | 1.0000 | 0.9999 | 0.9993 | 0.9907 | 0.9408 | 0.7827 | 0.4845 | 0.1642 | 0.0127 |
| | 11 | | | 1.0000 | 0.9999 | 0.9981 | 0.9824 | 0.9095 | 0.7031 | 0.3518 | 0.0556 |
| | 12 | | | | 1.0000 | 0.9997 | 0.9963 | 0.9729 | 0.8732 | 0.6020 | 0.1841 |
| | 13 | | | | | 1.0000 | 0.9995 | 0.9948 | 0.9647 | 0.8329 | 0.4510 |
| | 14 | | | | | | 1.0000 | 0.9995 | 0.9953 | 0.9648 | 0.7941 |
| | 15 | | | | | | | 1.0000 | 1.0000 | 1.0000 | 1.0000 |

Figure 4: Binomial probabiltiy sums table. (Table A.1, Reddy 2011)

**Example 2.4.2**

The probability that a patient recovers from a type of cancer is 0.6. If 15 people are known to have contracted

this disease, then one can determine the probabilities of various types of cases using Table A1.

Let X be the number of people who survive.

- Find:
  - The probability that at least 5 survive:
  - The probability that there will be 5 to 8 survivors:
  - The probability that exactly 5 survive:
- The probability that at least 5 survive:

$$p(X \geq 5) = 1 - p(x < 5) = 1 - \sum_{x=0}^{4} B(x; 15, 0.6) = 1 - 0.0094 = .6606$$

- The probability that there will be 5 to 8 survivors:

$$p(5 \leq X \leq 8) = \sum_{x=0}^{8} B(x; 15, 0.6) - \sum_{x=0}^{4} B(x; 15, 0.6) = 0.3902 - 0.0094 = 0.3808$$

- The probability that exactly 5 survive:

$$p(X = 5) = \sum_{x=0}^{5} B(x; 15, 0.6) - \sum_{x=0}^{4} B(x; 15, 0.6) = 0.0338 - 0.0094 = 0.0244$$

**Table A.1** **Binomial Probability Sums $\sum_{x=0}^{r} b(x; n, p)$**

| | | p | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| n | r | 0.10 | 0.20 | 0.25 | 0.30 | 0.40 | 0.50 | 0.60 | 0.70 | 0.80 | 0.90 |
| 15 | 0 | 0.2059 | 0.0352 | 0.0134 | 0.0047 | 0.0005 | 0.0000 | | | | |
| | 1 | 0.5490 | 0.1671 | 0.0802 | 0.0353 | 0.0052 | 0.0005 | 0.0000 | | | |
| | 2 | 0.8159 | 0.3980 | 0.2361 | 0.1268 | 0.0271 | 0.0037 | 0.0003 | 0.0000 | | |
| | 3 | 0.9444 | 0.6482 | 0.4613 | 0.2969 | 0.0905 | 0.0176 | 0.0019 | 0.0001 | | |
| | 4 | 0.9873 | 0.8358 | 0.6865 | 0.5155 | 0.2173 | 0.0592 | 0.0094 | 0.0007 | 0.0000 | |
| | 5 | 0.9978 | 0.9389 | 0.8516 | 0.7216 | 0.4032 | 0.1509 | 0.0338 | 0.0037 | 0.0001 | |
| | 6 | 0.9997 | 0.9819 | 0.9434 | 0.8689 | 0.6098 | 0.3036 | 0.0951 | 0.0152 | 0.0008 | |
| | 7 | 1.0000 | 0.9958 | 0.9827 | 0.9500 | 0.7869 | 0.5000 | 0.2131 | 0.0500 | 0.0042 | 0.0000 |
| | 8 | | 0.9992 | 0.9958 | 0.9848 | 0.9050 | 0.6964 | 0.3902 | 0.1311 | 0.0181 | 0.0003 |
| | 9 | | 0.9999 | 0.9992 | 0.9963 | 0.9662 | 0.8491 | 0.5968 | 0.2784 | 0.0611 | 0.0023 |
| | 10 | | 1.0000 | 0.9999 | 0.9993 | 0.9907 | 0.9408 | 0.7827 | 0.4845 | 0.1642 | 0.0127 |
| | 11 | | | 1.0000 | 0.9999 | 0.9981 | 0.9824 | 0.9095 | 0.7031 | 0.3518 | 0.0556 |
| | 12 | | | | 1.0000 | 0.9997 | 0.9963 | 0.9729 | 0.8732 | 0.6020 | 0.1841 |
| | 13 | | | | | 1.0000 | 0.9995 | 0.9948 | 0.9647 | 0.8329 | 0.4510 |
| | 14 | | | | | | 1.0000 | 0.9995 | 0.9953 | 0.9648 | 0.7941 |
| | 15 | | | | | | | 1.0000 | 1.0000 | 1.0000 | 1.0000 |

Figure 5: Binomial probabiltiy sums table for example 2.4.2 (Table A.1, Reddy 2011)

Depending on the values of $p$ and $n$, the binomial distribution will be skewed. You can see from Figure 6,7,and 8, that as changed, the peak of the distribution shifts. As the number of trials, $n$, changes, the shape of the distribution changes.

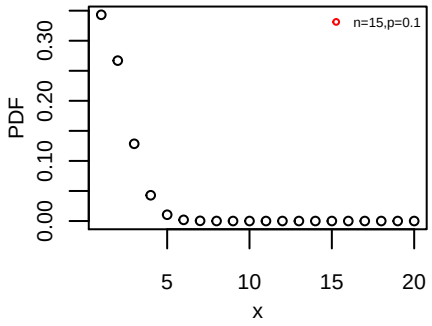



Figure 6: The effects of different values of p and and on the binomial probability distribution: n=12, p=0.1

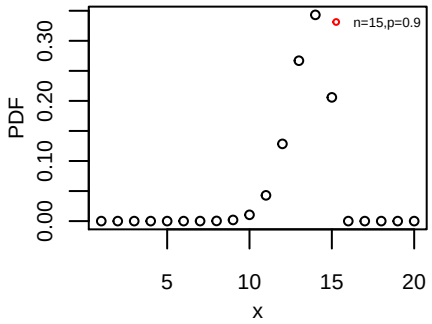



Figure 7: The effects of different values of p and and on the binomial probability distribution: n=15,p=0.9

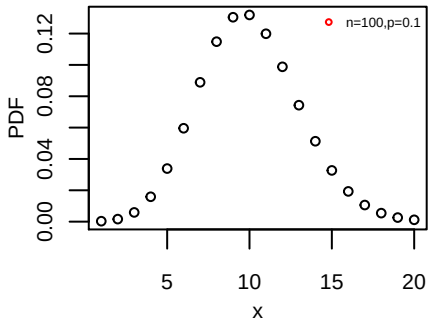



Figure 8: The effects of different values of p and and on the binomial probability distribution: n=100, p=0.1

### 1.2.3 Geometric Distribution

When one wants to know the probability of an event to occur for the first time, the geometric distribution is applied.

Let n be the random variable representing the number of trials until the event does occur. Then the probability density function is given by:

- $G(n;p) = p(1-p)^{n-1}$, n=1,2,3,....

Sometimes the time between two instances of the same event, i.e. the **recurrence time**, is of interest. Since the events are assumed independent, the mean recurrence time ($T$) between two consecutive events is simply the expected value of the Bernoulli distribution:

- $\bar{T} = E(T) = \sum_{t=1}^{\infty} t \cdot p(1-p)^{t-1} = p[1 + 2(1-p) + 3(1-p)^2] \approx \frac{1}{p}$

**Example 2.4.3**

The design code for buildings in a certain coastal region specifies the 50-yr wind as the "design wind", i.e., a wind velocity with a return period of 50 years, or one which may be expected to occur once every 50 years. What are the probabilities that:

The design wind is encountered in any given year?

- $p = \frac{1}{T} = \frac{1}{50} = 0.02$

The design wind is encountered during the fifth year of a newly constructed building?

- $G(5; 0.02) = (0.02) \cdot (1 - 0.02)^4 = 0.018$

The design wind is encountered within the first 5 years?

- $G(n \leq 5 : p) = \sum_{i=1}^{5} (0.02) \cdot (1 - 0.02)^{t-1} = 0.02 + 0.0196 + 0.0192 + 0.0188 + 0.0184 = 0.096$

**R Applications**

For the geometric distribution in R, we use the geom() function.

- From Example 2.4.3, to find the probability that the design wind is encountered during the fifth year of a newly constructed building we write dgeom(x,p):

```
p=1/50
x=5
dgeom(x,p)
```

```
## [1] 0.01807842
```

To plot this PDF write:

```
p=1/50
x=(1:400)
G=dgeom(x,prob=p)
par(mgp=c(1,0.5,0))
plot(x,G,xlab="x", ylab="PDF", cex=0.6,
     cex.lab=0.7, cex.axis=0.7, cex.main=0.7)
```

To plot the CDF we write:

```
p=1/50
x=(1:400)
G=pgeom(x,prob=p)
```

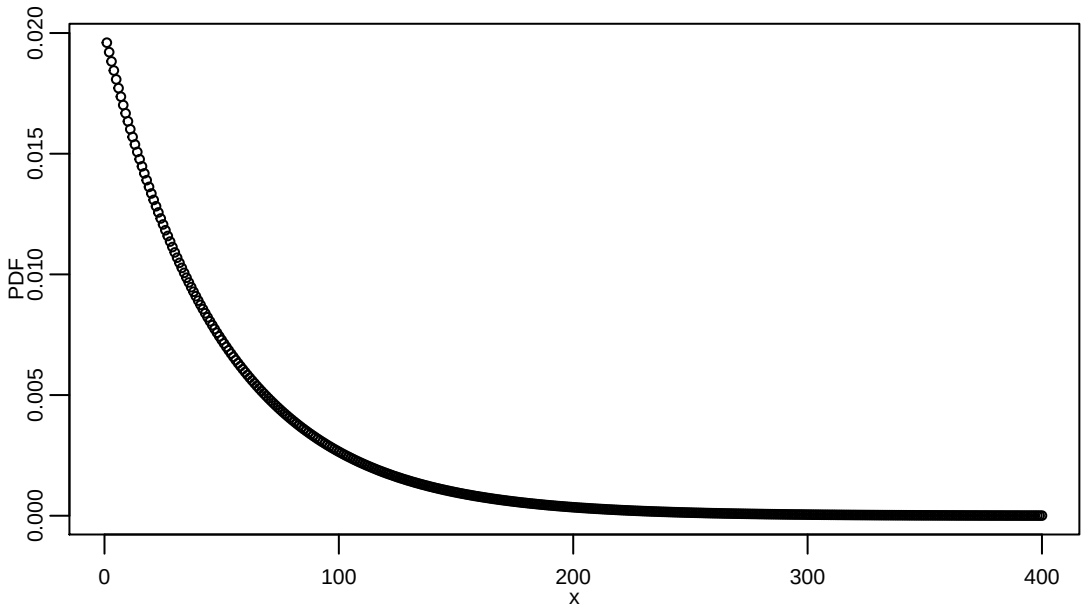



Figure 9: The PDF of design wind experienced in year x.

```
par(mgp=c(1,0.5,0))
plot(x,G,xlab="x", ylab="CDF", cex=0.6, cex.lab=0.7,
     cex.axis=0.7, cex.main=0.7)
```

### 1.2.4 Poisson Distribution

Poisson experiments involve the **number of outcomes** of a random variable $X$ which occur per unit time. The three main characteristics are:

- independent outcomes
- the probability that a single outcome will occur during a very short time is proportional to the length of the time interval
- the probability that more than one outcome occurs during a very short time is negligible

These conditions lead to the **Poisson distribution** where $\lambda$t remains constant. It is defined as:

$$P(x;\lambda t) = \frac{(\lambda t)^x exp(-\lambda t)}{x!}, \text{ x=0,1,2,3...}$$

where $\lambda$ is the **mean occurence rate**, or the rate of average occurrences of the event.

- A unique feature of the Poisson Distribution regarding its mean and variance:

$$\mu(X) = \sigma^2(X) = \lambda t = np$$

Some common applications for the Poisson distribution includes:

- Number of faults in a length of cable
- Number of cars in a fixed length of roadway
- Number of cars passing a point in a fixed time interval
- Counts of $\alpha$-particles in a radioactive decay
- Number of arrivals in an interval of time
- Number of noticeable surface defects found on a new automobile

**Example 2.4.8**

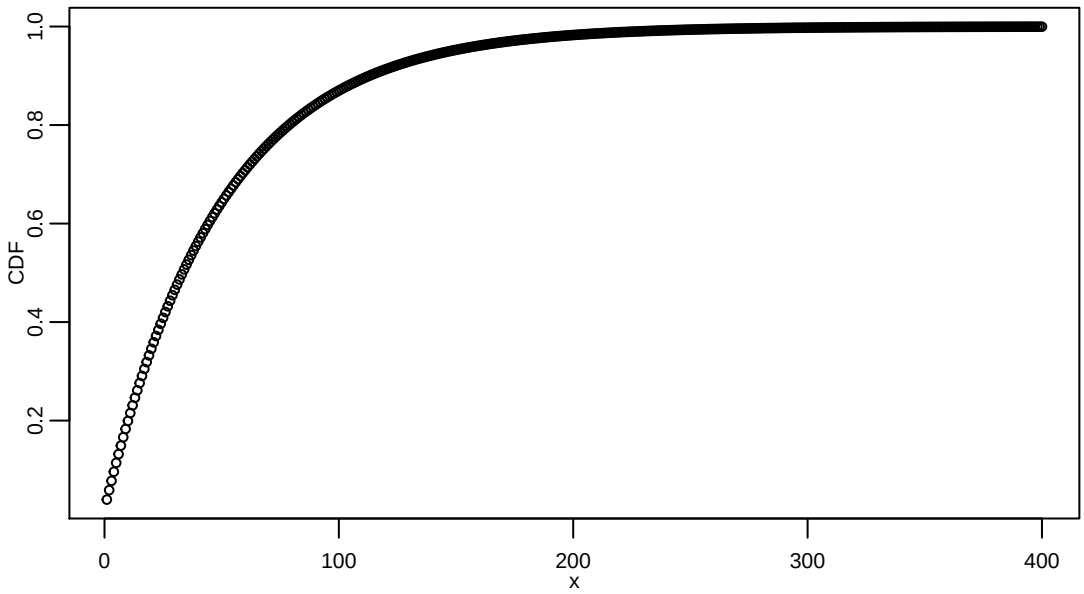



Figure 10: The CDF of design wind experienced in year x.

Historical records at Phoenix, AZ indicate that on an average there will be 4 dust storms per year. Assuming a Poisson distribution, compute the probabilities of the following events:

- That there would not be any storms at all during a year: $p(X = 0) = \frac{(4^0)\cdot(e^{-4})}{0!} = 0.018$
- The probability that there will be four storms during a year: $p(X = 4) = \frac{(4^4)\cdot(e^{-4})}{4!} = 0.195$

Note that though the average is four, the probability of four storms actually encountering four storms in a year is less than 20%.

**R Applications**

To calculate Poisson distributions in R, we use the pois() function. To find the number of storms per year, we use the dpois($x, \lambda$) function for the PDF. From the first part of the example above we find the probability of there not being any storms in a year.

```
lambda=4
x=(0)
dpois(x, lambda)
```

```
## [1] 0.01831564
```

To plot this PDF in R, write the following.

```
lambda=4
x=(1:20)
P=dpois(x, lambda)
par(mgp=c(1,0.5,0))
plot(x,P, xlab="x", ylab="PDF",main="PDF of Number of Dust Storms in a year", cex=0.6,
     cex.lab=0.7, cex.axis=0.7, cex.main=0.7)
```

To find the probability of there being less than or equal to a certain amount of storms per year in R, we use ppois() for the CDF.

The probability that there will be four storms or less in a year is about 63%

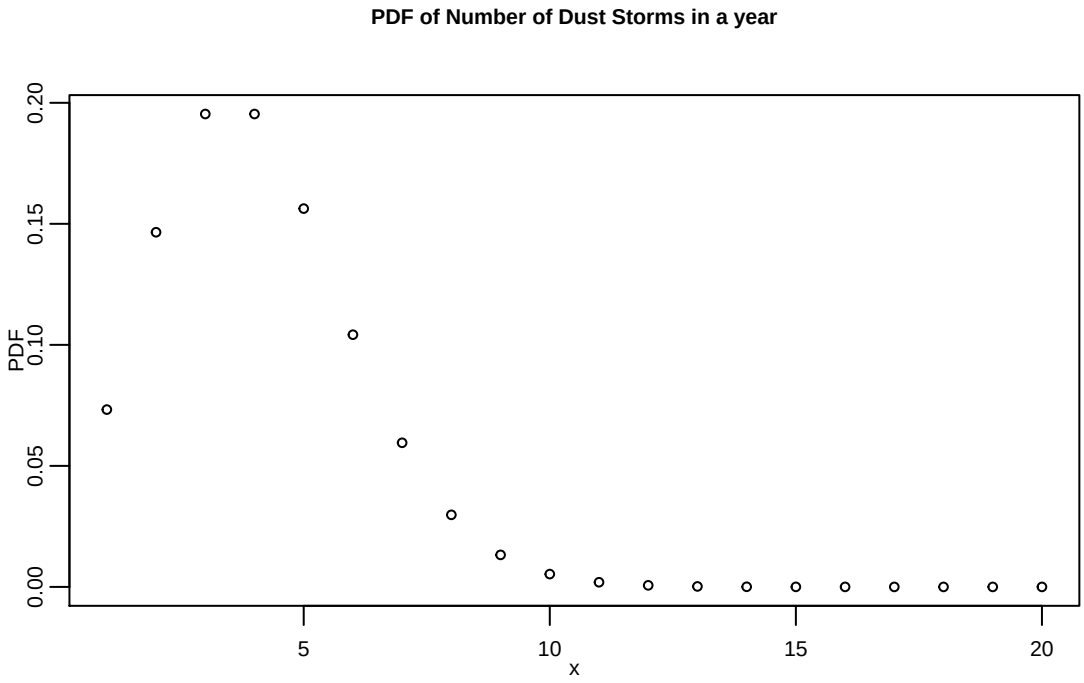



Figure 11: The PDF for dust storms in a year x

```
lambda=4
x=(4)
ppois(x, lambda)
```

```
## [1] 0.6288369
```

Similarly, to graph this CDF, we write:

```
lambda=4
x=(1:20)
P=ppois(x, lambda)
par(mgp=c(1,0.5,0))
plot(x,P, xlab="x", ylab="PDF",main="PDF of Number of Dust Storms in a year", cex=0.6,
     cex.lab=0.7, cex.axis=0.7, cex.main=0.7)
```

Similar to the binomial distribution, values for the cumulative Poisson distribution can be read off of a table (Table A.2) for certain combinations of the two parameters.

## 1.3 Distributions for Continuous Variables

The content of this section relies heavily on Agami Reddy's book, "Applied Data Analysis and Modeling for Energy Engineers and Scientists." We have also chosen to maintain the books figure and example numbers in order for students to easily follow along.

*Reddy, T.Agami. Applied Data Analysis and Modeling for Energy Engineers and Scientists. Springer, 2011.*

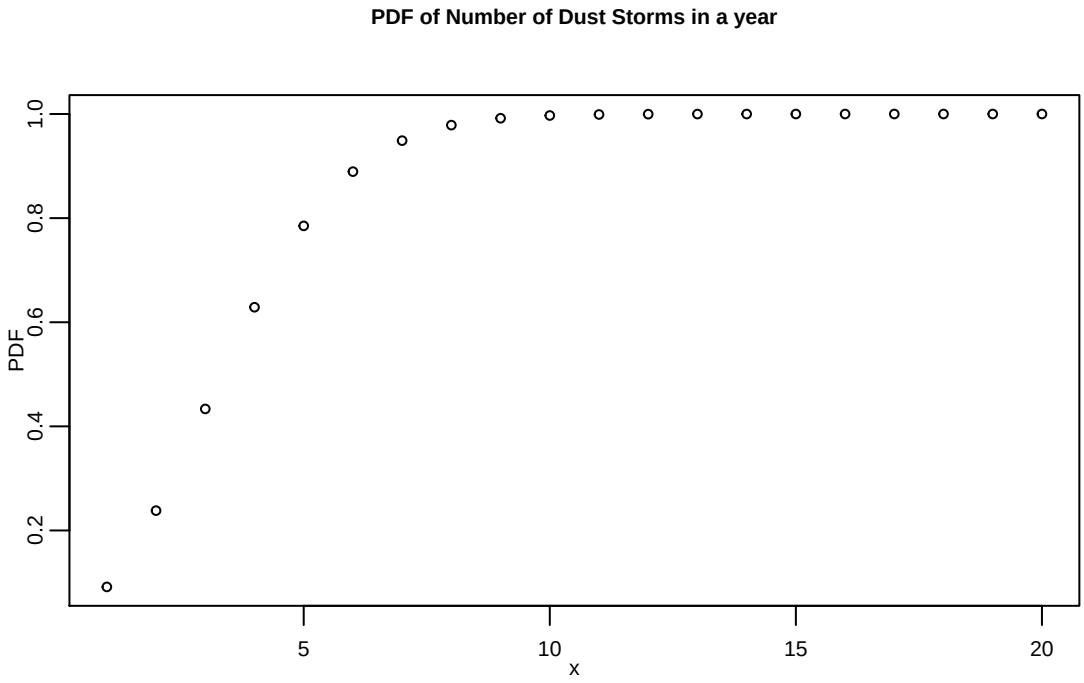



Figure 12: The PDF for dust storms in a year x

### 1.3.1 Gaussian Distribution

The Gaussian distribution is a special case of the binomial distribution where $n$ is large ($n > 30$). The Gaussian distribution is considered on of the most important of all distributions in statistics and is used to model events which occur by chance. The PDF is given by:

- $N(x : \mu, \sigma) = \frac{1}{\sigma(2\pi)^{1/2}} exp[-\frac{(x-\mu)}{\sigma}]^2]$

where $\mu$ and $\sigma$ are the mean and the standard deviation respectively.

Examples include:

- variation of dimensions of mass-produced items during manufacturing
- experimental errors
- variability in measurable biological traits

**Standard Normal Distribution** For this distribution, it is often convenient to standardize the random variable into a new random variable with zero mean and variance of unity.

- $z \equiv \frac{x-\mu}{\sigma}$

This results in the **standard normal curce or the z-curve**:

- $N(z; 0, 1) = \frac{1}{\sqrt{2\pi}} exp(-z^2/2)$

The standard normal distribution is used to determine the probability of the variate having a value between a certain interval, say $z_1$ and $z_2$. Recall that the probability can be obtained from a continuous random variable by finding the area under the curve. In this case:

- $N(z_1 \leq x \leq z_2) = \int_{z_1}^{z_2} \frac{1}{\sqrt{2\pi}} exp(-z^2/2)\, \mathrm{d}z$

Tables, like Table A.3 can make it easy to find these probabilities for any $z_k$. Note that for $z = 0$, the probability given by the shaded area is 0.5 or 50%.

**Example 2.4.9**

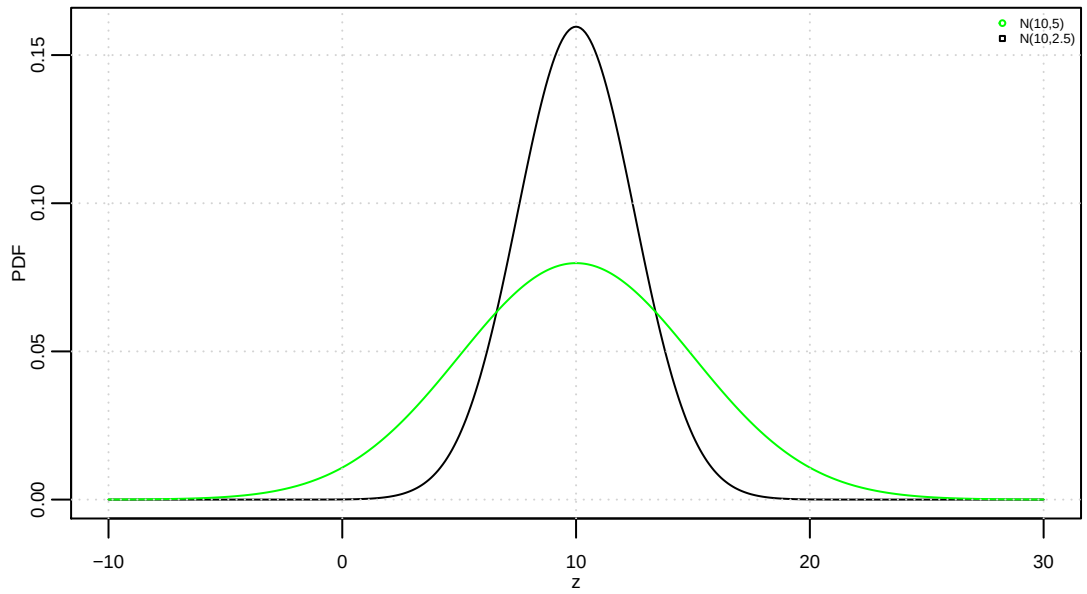



Figure 13: Two standard normal z-curves with different standard deviation 5 (green) and 2.5 (black)

Resistors have a nominal value of 100 ohms but their actual values are normally distributed with a mean = 100.6 ohms and standard deviation = 3 ohms. Find the percentage of resistors that will have values:

- higher than the normal rating.
  - The standard normal variable $z(x = 100) = \frac{100-100.6}{3} = -0.2$.
  - From Table A3, this corresponds to a probability of (1-0.4207)=0.5793 or 57.93%.
- within 3 ohms of the nominal rating (i.e. between 97 and 103 ohms). The **lower limit** $z_1 = \frac{97-100.6}{3} = -1.2$, and the tabulated probability from Table A3 is p(z=-1.2)=0.1151 The **upper limit** is: $z_2 = \frac{103-100.6}{3} = 0.8$
  - First, determine the probability about the negative value symmetric about 0, i.e. p(z=-0.8)=0.2119 from Table A3
  - Since the total area under the curve is 1.0, p(z=0.8)=1.0-0.2119=0.7881.
  - Finally, the required probability p(-1.2<z<0.8)=(0.7881-0.1151)

**R Applications**

To create a graph of the PDF in R for the Gaussian distribution we use the norm() function

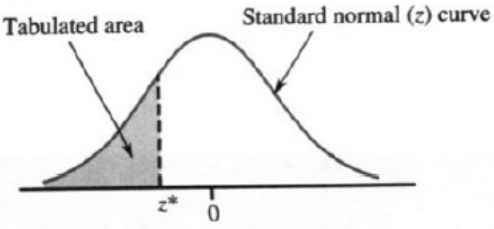
For a distribution with a mean=10 and $\sigma^2$=2.5 write:

```
x = seq(-10, 30, 0.1)
y = dnorm(x, mean = 10, sd = 2.5)
par(mgp = c(1, 0.5, 0))
plot(x, y, type = "l", xlab = "x", ylab = "PDF", cex = 0.6, cex.lab = 0.6, cex.axis = 0.6)
```

For the CDF (Figure 10) of the Graph above, in R Studio we write:

```
y=pnorm(x,mean=10,sd=2.5)
par(mgp=c(1,0.5,0))
plot(x,y,type="l", xlab="x", ylab="CDF", cex=0.6, cex.lab=0.6, cex.axis=0.6)
grid()
```

**Table A.3** The standard normal distribution (cumulative z curve areas)




| z* | .00 | .01 | .02 | .03 | .04 | .05 | .06 | .07 | .08 | .09 |
|---|---|---|---|---|---|---|---|---|---|---|
| **−3.8** | .0001 | .0001 | .0001 | .0001 | .0001 | .0001 | .0001 | .0001 | .0001 | .0000 |
| **−3.7** | .0001 | .0001 | .0001 | .0001 | .0001 | .0001 | .0001 | .0001 | .0001 | .0001 |
| **−3.6** | .0002 | .0002 | .0001 | .0001 | .0001 | .0001 | .0001 | .0001 | .0001 | .0001 |
| **−3.5** | .0002 | .0002 | .0002 | .0002 | .0002 | .0002 | .0002 | .0002 | .0002 | .0002 |
| **−3.4** | .0003 | .0003 | .0003 | .0003 | .0003 | .0003 | .0003 | .0003 | .0003 | .0002 |
| **−3.3** | .0005 | .0005 | .0005 | .0004 | .0004 | .0004 | .0004 | .0004 | .0004 | .0003 |
| **−3.2** | .0007 | .0007 | .0006 | .0006 | .0006 | .0006 | .0006 | .0005 | .0005 | .0005 |
| **−3.1** | .0010 | .0009 | .0009 | .0009 | .0008 | .0008 | .0008 | .0008 | .0007 | .0007 |
| **−3.0** | .0013 | .0013 | .0013 | .0012 | .0012 | .0011 | .0011 | .0011 | .0010 | .0010 |
| **−2.9** | .0019 | .0018 | .0018 | .0017 | .0016 | .0016 | .0015 | .0015 | .0014 | .0014 |
| **−2.8** | .0026 | .0025 | .0024 | .0023 | .0023 | .0022 | .0021 | .0021 | .0020 | .0019 |
| **−2.7** | .0035 | .0034 | .0033 | .0032 | .0031 | .0030 | .0029 | .0028 | .0027 | .0026 |
| **−2.6** | .0047 | .0045 | .0044 | .0043 | .0041 | .0040 | .0039 | .0038 | .0037 | .0036 |
| **−2.5** | .0062 | .0060 | .0059 | .0057 | .0055 | .0054 | .0052 | .0051 | .0049 | .0048 |
| **−2.4** | .0082 | .0080 | .0078 | .0075 | .0073 | .0071 | .0069 | .0068 | .0066 | .0064 |
| **−2.3** | .0107 | .0104 | .0102 | .0099 | .0096 | .0094 | .0091 | .0089 | .0087 | .0084 |
| **−2.2** | .0139 | .0136 | .0132 | .0129 | .0125 | .0122 | .0119 | .0116 | .0113 | .0110 |
| **−2.1** | .0179 | .0174 | .0170 | .0166 | .0162 | .0158 | .0154 | .0150 | .0146 | .0143 |
| **−2.0** | .0228 | .0222 | .0217 | .0212 | .0207 | .0202 | .0197 | .0192 | .0188 | .0183 |
| **−1.9** | .0287 | .0281 | .0274 | .0268 | .0262 | .0256 | .0250 | .0244 | .0239 | .0233 |
| **−1.8** | .0359 | .0351 | .0344 | .0336 | .0329 | .0322 | .0314 | .0307 | .0301 | .0294 |
| **−1.7** | .0446 | .0436 | .0427 | .0418 | .0409 | .0401 | .0392 | .0384 | .0375 | .0367 |
| **−1.6** | .0548 | .0537 | .0526 | .0516 | .0505 | .0495 | .0485 | .0475 | .0465 | .0455 |
| **−1.5** | .0668 | .0655 | .0643 | .0630 | .0618 | .0606 | .0594 | .0582 | .0571 | .0559 |
| **−1.4** | .0808 | .0793 | .0778 | .0764 | .0749 | .0735 | .0721 | .0708 | .0694 | .0681 |
| **−1.3** | .0968 | .0951 | .0934 | .0918 | .0901 | .0885 | .0869 | .0853 | .0838 | .0823 |
| **−1.2** | .1151 | .1131 | .1112 | .1093 | .1075 | .1056 | .1038 | .1020 | .1003 | .0985 |
| **−1.1** | .1357 | .1335 | .1314 | .1292 | .1271 | .1251 | .1230 | .1210 | .1190 | .1170 |
| **−1.0** | .1587 | .1562 | .1539 | .1515 | .1492 | .1469 | .1446 | .1423 | .1401 | .1379 |
| **−0.9** | .1841 | .1814 | .1788 | .1762 | .1736 | .1711 | .1685 | .1660 | .1635 | .1611 |
| **−0.8** | .2119 | .2090 | .2061 | .2033 | .2005 | .1977 | .1949 | .1922 | .1894 | .1867 |
| **−0.7** | .2420 | .2389 | .2358 | .2327 | .2296 | .2266 | .2236 | .2206 | .2177 | .2148 |
| **−0.6** | .2743 | .2709 | .2676 | .2643 | .2611 | .2578 | .2546 | .2514 | .2483 | .2451 |
| **−0.5** | .3085 | .3050 | .3015 | .2981 | .2946 | .2912 | .2877 | .2843 | .2810 | .2776 |
| **−0.4** | .3446 | .3409 | .3372 | .3336 | .3300 | .3264 | .3228 | .3192 | .3156 | .3121 |
| **−0.3** | .3821 | .3783 | .3745 | .3707 | .3669 | .3632 | .3594 | .3557 | .3520 | .3483 |
| **−0.2** | .4207 | .4168 | .4129 | .4090 | .4052 | .4013 | .3974 | .3936 | .3897 | .3859 |
| **−0.1** | .4602 | .4562 | .4522 | .4483 | .4443 | .4404 | .4364 | .4325 | .4286 | .4247 |
| **−0.0** | .5000 | .4960 | .4920 | .4880 | .4840 | .4801 | .4761 | .4721 | .4681 | .4641 |

Figure 14: The standard normal distribution table. (Table A.3, Reddy 2011)

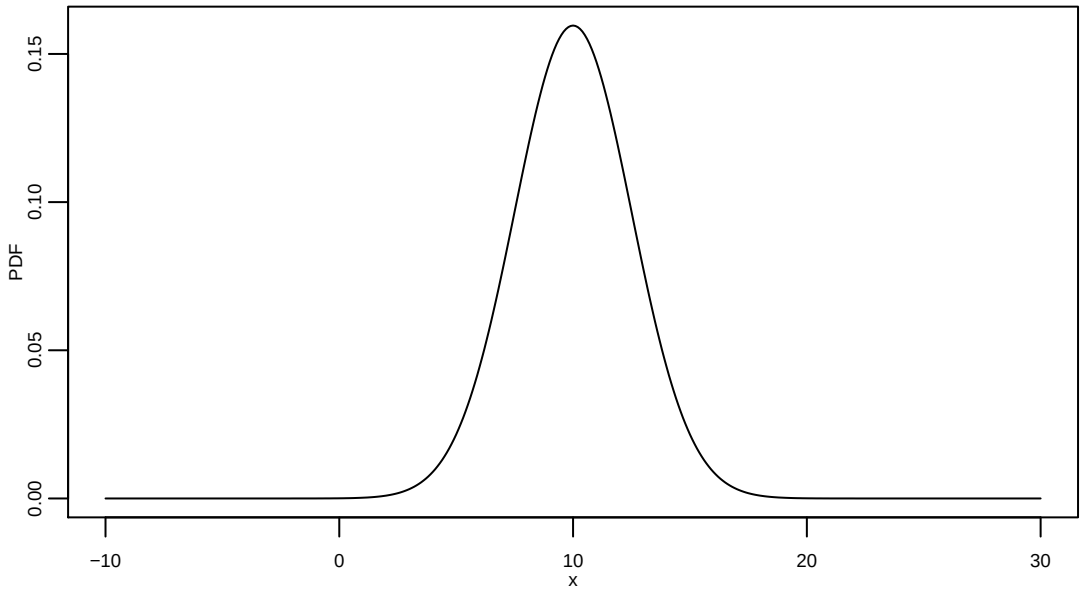



Figure 15: A PDF for the Gaussian distribution with mean=10 and standard deviation=2.5

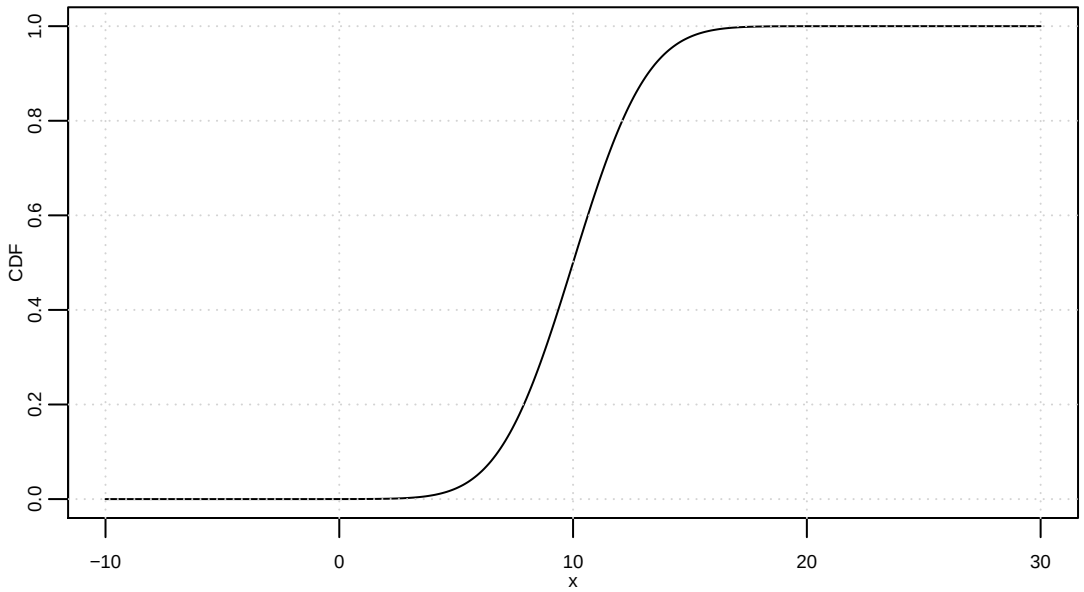



Figure 16: A CDF for the Gaussian distribution with mean=10 and standard deviation=2.5

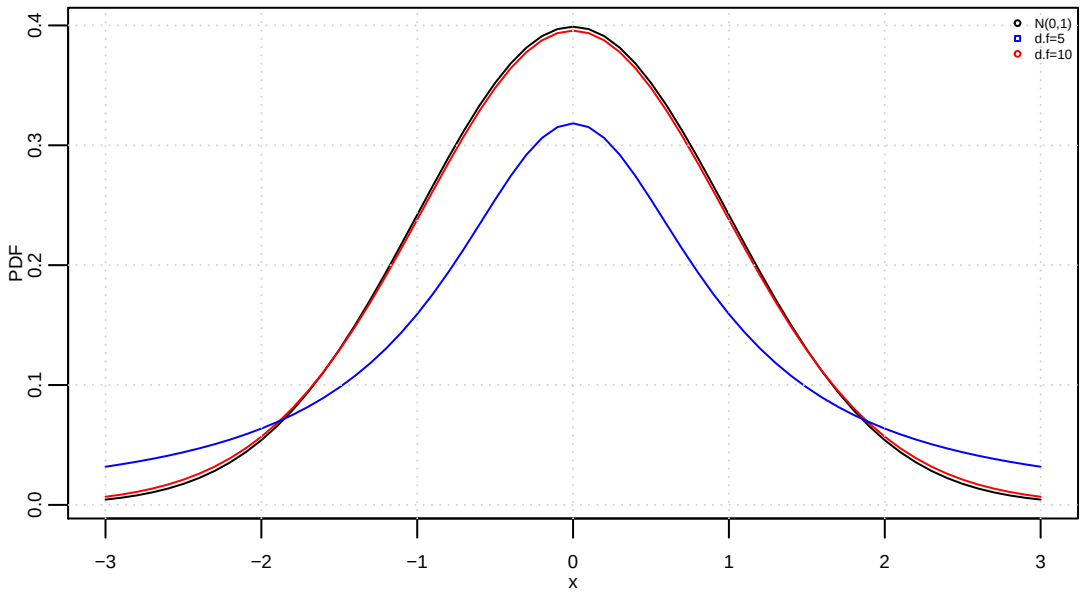



Figure 17: Shows the comparison of a Gaussian curves to two different Student-t curves. As degrees of freedom decrease, the PDF for the Student-t distribution flattens out and deviates increasingly from the normal distribution

### 1.3.2 Student t Distribution

When random samples are small (n<30), the Student-$t$ distribution instead of the normal distribution should be used. - Assume that a sampled population is approximately normally distributed, then the random variable $t = \frac{x-\mu}{s\sqrt{n}}$ has the Student-t distribution $t(\mu, s, v)$,

where s is the sample standard deviation and v is the degrees of freedom

Qualitatively the Student-$t$ distributions are similar to the standard normal distributions. In the context of finding probability from the area under the PDF curve, the differences are large enough to consider.

**R Application**

For the Student-$t$ distribution in R we use the t function. To create the PDF above we use the dt(data, mean, sd) function.

### 1.3.3 Lognormal Distribution

If the variate $X$ is such that $\log(X)$ is normally distributed, then the distribution of $X$ is said to be lognormal. Where $X$ ranges from $-\infty$ to $+\infty$, $\log(X)$ would range from 0 to $+\infty$. This creates a skewed distribution. This distribution is appropriate for non-negative outcomes which:

- are the product of a number of quantities
- captures the non-negative nature of many variables occurring in practice

It is characterized by two parameters, the mean and variance $(\mu, \sigma)$, and is defined by:

$$L(x; \mu, \sigma) = \frac{1}{\sigma x(\sqrt{2\pi})} exp[-\frac{(lnx-\mu)^2}{2\sigma^2}] = 0, x \geq 0$$

Applications include:

- flood frequency
- crack growth or wear

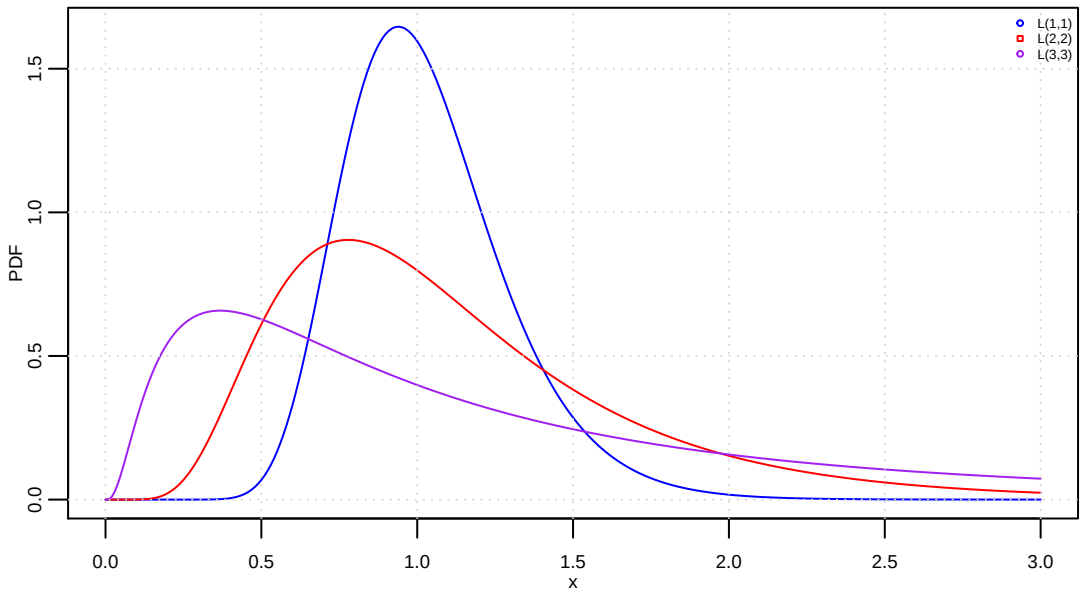



Figure 18: Lognormal distributions for different mean and standard deviation values

- pollutants produced by chemical plants
- threshold values for drug dosage
- infiltration rates in buildings

**Example 2.4.11**

Concentration of pollutants produced by chemical plants in known to resemble lognormal distributions and it is used to evaluate issues regarding compliance of government regulations. The concentration of a certain pollutant, in parts per million (ppm), is assumed lognormal with parameters $\mu$=4.6 and $\sigma$=1.5.

- What is the probability that the concentration exceeds 10 ppm?
  - To solve you can either use the equation or Table A3!

$L(X > 10) = N[ln(10), 4.6, 1.5] = N[\frac{ln(10)-4.6}{1.5}] = N(-1.531) = 0.0630$

**R Applications**

For lognormal Distributions we use the lnorm() function.

To plot the PDF write:

```
x=seq(0,3,0.01)
y=dlnorm(x,meanlog=0,sdlog=0.25)
par(mgp=c(1,0.5,0))
plot(x,y,type="l", xlab="x", ylab="PDF", cex=0.6, cex.lab=0.6, cex.axis=0.6)
grid()
```

### 1.3.4 Gamma Distribution

Gamma distributions, derived from the gamma function, are good for modeling random phenomena which can only be positive and are unimodal.

Recall the gamma function:

$\Gamma_x(\alpha) = \int_0^x \mathrm{x}^{\alpha-1} e^{-x}\, \mathrm{d}x$

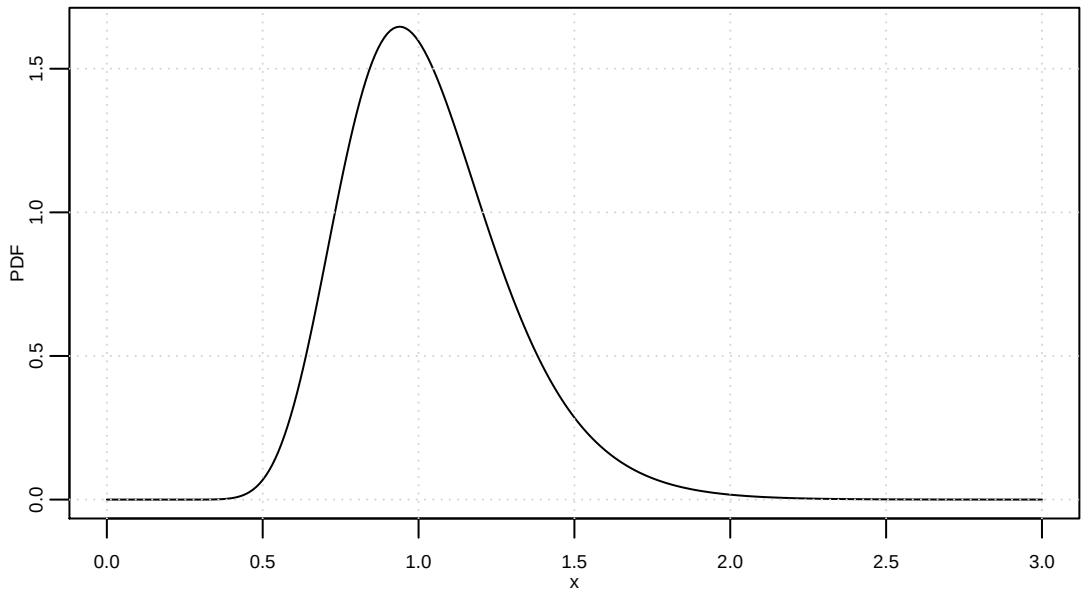



Figure 19: PDF for lognormal distribution.

for non-negative integers k:

$\Gamma(k+1) = k!$

Thus, the gamma distribution for a continuous random variable is given by:

$G(x;\alpha,\lambda) = \lambda^{\alpha} e^{-\lambda x} \frac{x^{\alpha-1}}{(\alpha-1)!}$ if $x > 0$

$= 0$ elsewhere

The mean and variance are given by:

$\mu = \alpha/\lambda$ and $\sigma^2 = \alpha/\lambda^2$

Where $\alpha$ is the shape factor and $\lambda$ is the scale parameter. Variation in these two parameters can provide a variety of different shapes. By changing these parameters in the gamma distributions, other important distributions can be derived. For example, if $\alpha \to \infty$ and $\lambda = 1$, then the gamma distribution approaches the normal. When $\alpha = 1$ the exponential distribution is obtained.

To model the gamma function in R we use the gamma() function with the appropriate prefix.

### 1.3.5 Weibull Distribution

The Weibull distribution is widely used in applications involving reliability and life testing. For example, this distribution can model time to failure. The Weibull distribution has a density function given by:

$W(x;\alpha,\beta) = \frac{\alpha}{\beta^{\alpha}} x^{\alpha-1} exp[-(x/\beta)^{\alpha}]$ For $x > 0$

$= 0$ elsewhere

Like the gamma distribution, the parameters $\alpha$ and $\beta$ are the shape and scale factors respectively. The Weibull distribution has mean:

$\mu = \beta\Gamma(1 + \frac{1}{\alpha})$

To model the gamma function in R we use the weibull() function with the appropriate prefix.

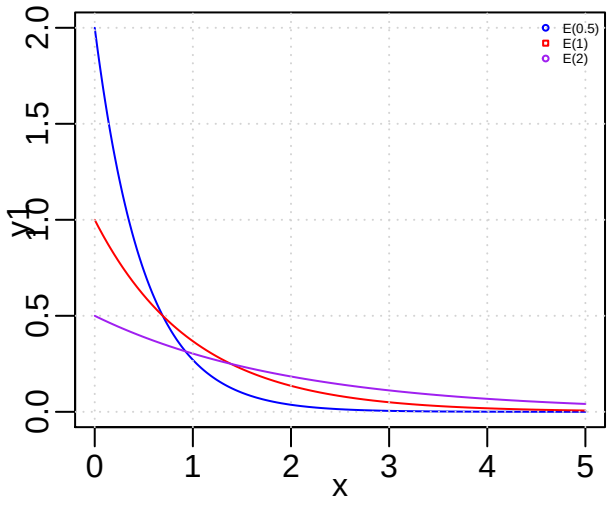



Figure 20: Exponential distributions for three different values of the parameter lambda

### 1.3.6 Exponental Distribution

The **exponential distribution** is the continuous equivalent to the geometric distribution for discrete cases. It is used to model the **interval between two occurrences**. Its PDF is defined by:

$$E(x;\lambda) = \lambda e^{-\lambda x}, \text{ if X>0}$$
$$= 0 \text{ , otherwise}$$

where $\lambda$ is the mean value per unit time or distance.

The mean and variance of the exponential distribution are:

$$\mu = 1/\lambda \text{ and } \sigma^2 = 1/\lambda^2$$

The CDF of the exponential distribution is

$$CDF[E(a,\lambda)] = \int_0^a \lambda e^{-\lambda x} dx = 1 - e^{-\lambda a}$$

Common applications include:

- distance between consecutive faults in a cables
- time between chance failures of a component or system
- the time between consecutive emissions of particles
- the times between successive arrivals at a service facility

**Example 2.4.12** Temporary disruptions to the power grid can occur due to random events such as lightning, transformer failures, forest fires, etc. The exponential distribution has been know to be a good function to model such failures. If these occur on average say once every 2.5 years, then

$$\lambda = 1/2.5\text{=}0.40 \text{ per year.}$$

- What is the probability that there will be at least one disruption next year?

  $CDF[E(X \leq 1;\lambda)] = 1 - e^{-0.4(1)} = 1 - 0.6703 = 0.3297$

- What is the probability that there will be no more than two disruptions next year? This is the complement of at least two disruptions.

  1-$CDF[E(X \leq 2;\lambda)] = 1 - [1 - e^{-0.4(1)}] = 0.4493$

**R Applications**

To use the exponential distribution in R we use the exp function. To create the CDF above in R, we use pexp(x,rate), where rate =$1/\lambda$

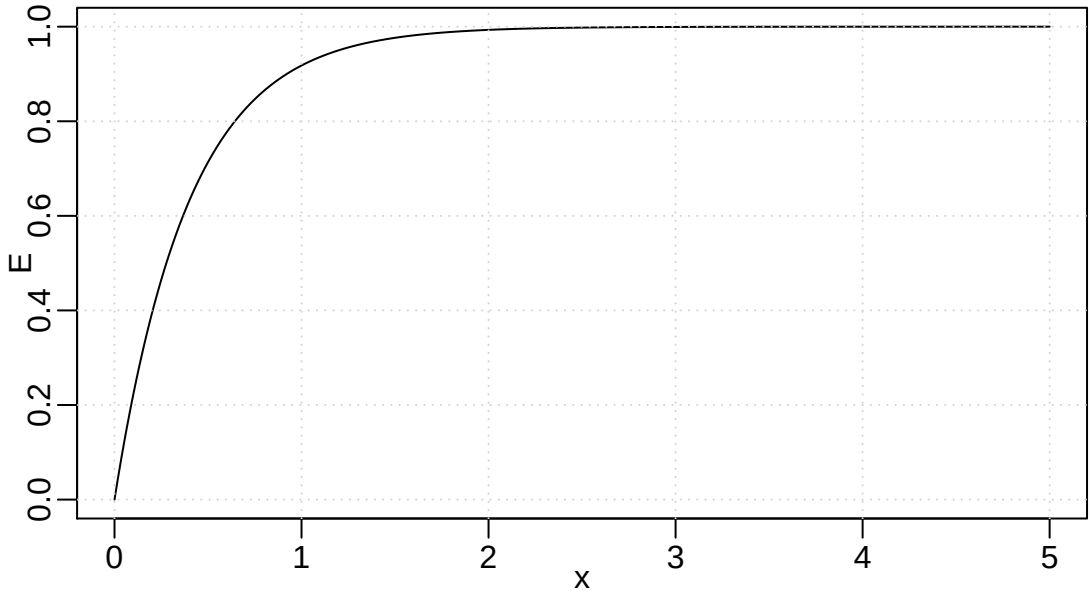



Figure 21: CDF of the exponential distriction

```
x=seq(0,5,0.01)
E=pexp(x,1/0.40)
par(mgp=c(1,0.5,0))
plot(x,E,type="l")
grid()
```

### 1.3.7 Chi-squared Distribution

**Chi-squared distributions** are often used in inferential statistics as a **test of significance for hypothesis testing** and analysis of variance type problems. Just like the t-statistics, there is a family of distributions for different values of v. Note that the distribution cannot assume negative values, and that it is positively skewed.

- The PDF of the Chi-squared Distribution is defined as:
  - $X^2(x;v) = \frac{1}{2^{v/2-1}\Gamma(v/2)} x^{v/2-1} e^{-x/2}$ x>0
  - =0 elsewhere
- The mean and variance are: $\mu = v$ and $\sigma^2 = 2v$

**R Applications** For the Chi-squared distribution in R we use the chisq function. To plot the PDF for the Chi-squared distribution with 7 degrees of freedom we write dchisq(x,degrees of freedom) :

```
x=seq(0,20,0.01)
df=7
C=dchisq(x,df)
plot(x,C,type="l", ylab="PDF")
```

### 1.3.8 F Distribution

The F distribution allows for the **comparison between two or more sample variances**. This distribution is defined as the ratio of two independent chi-square random variables, each divided by its degrees of freedom.

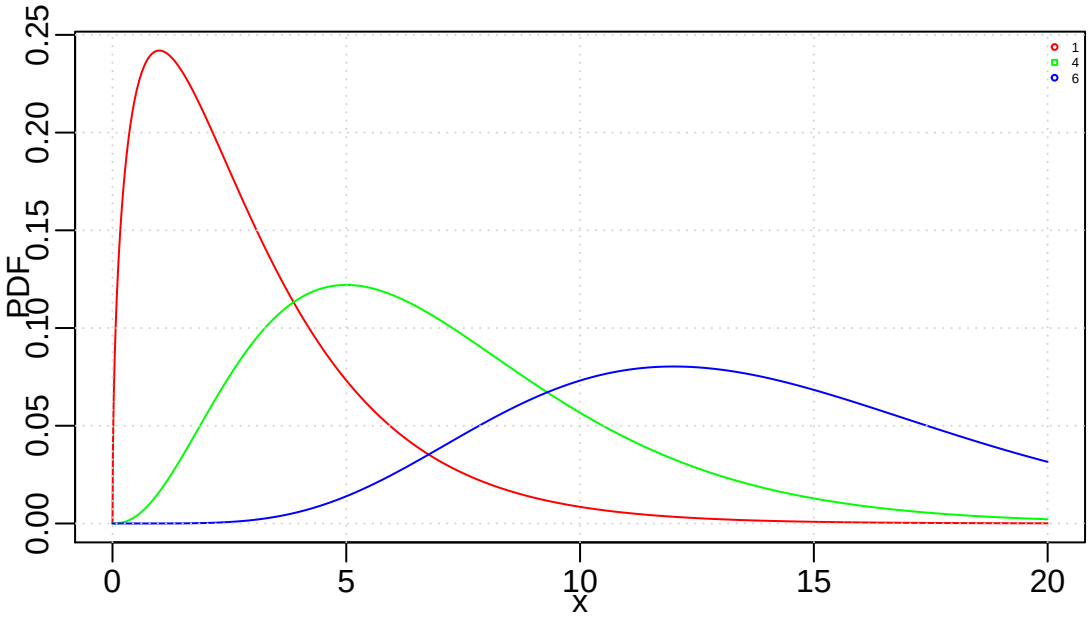



Figure 22: Chi-square distributions for different values of the variable v denoting the degrees of freedom

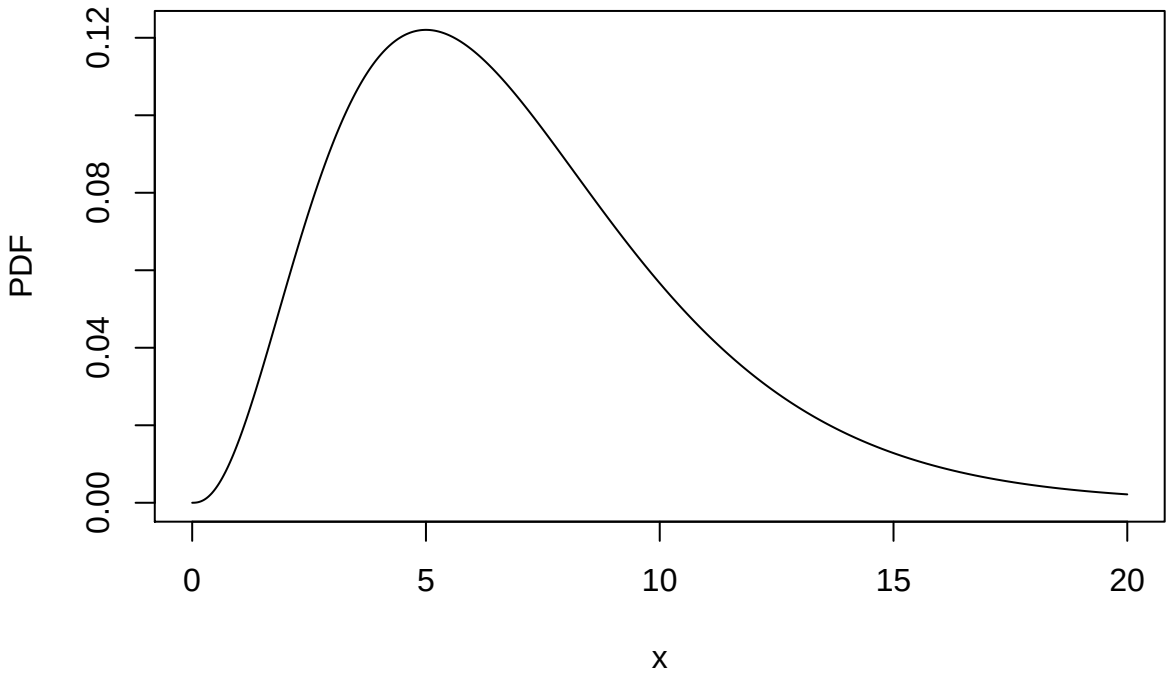



Figure 23: PDF for the Chi-square distribution with 7 degrees of freedom

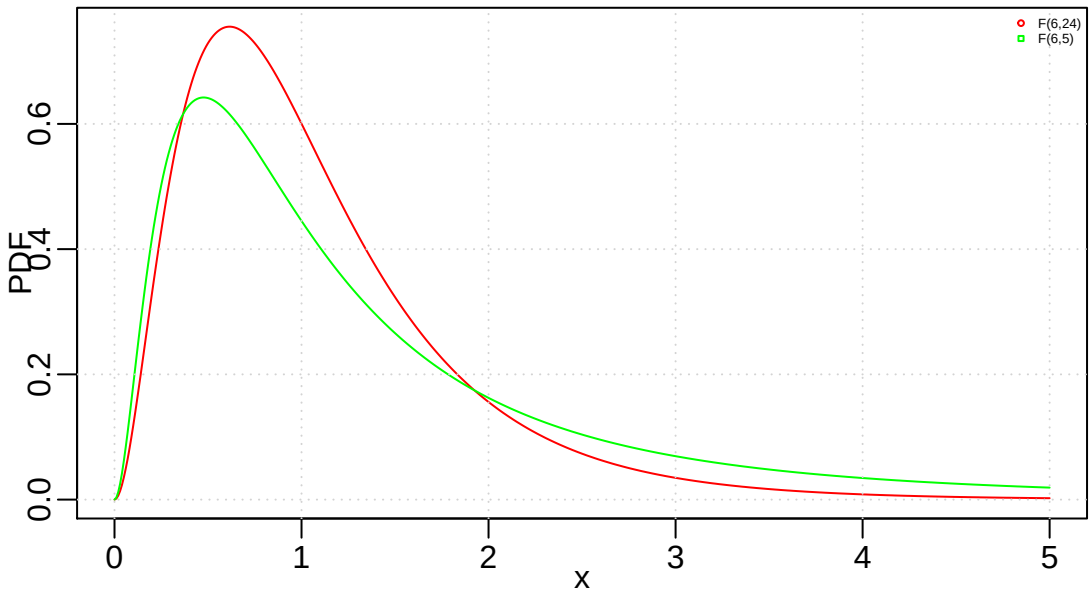



Figure 24: F-distributions for two different combinations of the random variables

It is represented by a family of plots where each plot is specific to a set numbers representing the degrees of freedom of the two random variables $(v_1, v_2)$.

**R Application**

To use the F distribution in R we use the f(x,degrees of freedom) function.

### 1.3.9 Uniform Distribution

The **uniform distribution** is the simplest of all PDFs and applies to both continuous and discrete data whose outcomes are all have equal probabilities.

- For the discrete case, where $X$ can assume values $x_1, x_2, ....x_k$, the PDF is defined by:
  - $U(x;k) = \frac{1}{k}$

with mean= $\frac{\sum_{i=1}^{k} x_i}{k}$ and variance $\frac{\sum_{i=1}^{k}(x_i - \mu)^2}{k}$

**R Application**

For uniform distributions in R we us the unif function. To find the values from the CDF we write punif(x,d,c), where $c$ and $d$ are the lower and upper bounds of the interval.

- **Example 2.4.14**: A random variable $X$ has a uniform distribution with $c = -5$ and $d = 10$. On average what proportion will have a negative value?

```
x=(-0)
d=-5
c=10
punif(x,d,c)
```

```
## [1] 0.3333333
```

### 1.3.10 Beta Distribution

The **beta distribution** is very versatile and is appropriate for discrete random variables between 0 and 1 and for those representing proportions. It is a two-parameter model which is defined by:

- $\beta(x;p,q) = \frac{(p+q+1)!}{(p-1)!(q-1)!}x^{p-1}(1-x)^{q-1}$

Note that this distribution is widely used in Bayesian probability problems.

Depending on the values of $p$ and $q$, a wide variety of curves are possible.

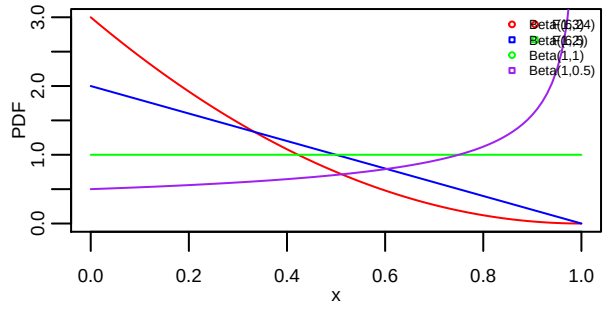



Figure 25: PDF for the beta distribution for p=1 different values of q.

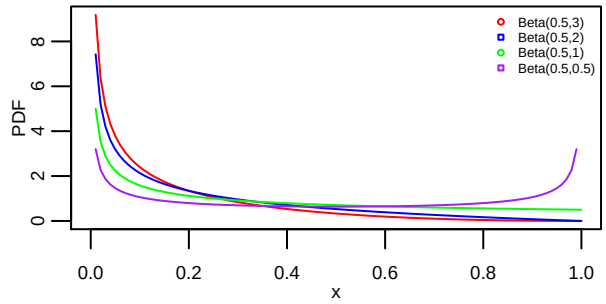



Figure 26: PDF for the beta distribution for p=0.5 and different values of q.

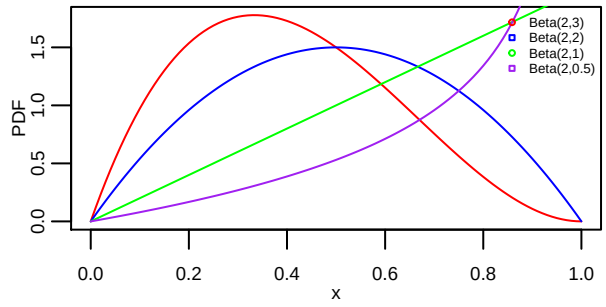



Figure 27: PDF for the beta distribution for p=2 and different values of q.

**R Application**

To utilize the $\beta$ distribution in R we use the beta function. For example, to create Figure 15, we use dbeta(x,p,q):

```
x=seq(0,1,0.01)
B1=dbeta(x,1,3)
```

```r
B2=dbeta(x,1,2)
B3=dbeta(x,1,1)
B4=dbeta(x,1,0.5)
par(mgp=c(1,0.5,0))
plot(x,B1,type="l",col="red", xlab="x", ylab="PDF")
lines(x,B2,col="blue")
lines(x,B3,col="green")
lines(x,B4,col="purple")
legend("topright",cex = 0.4, c("Beta(0.5,3)", "Beta(0.5,2)", "Beta(0.5,1)",
"Beta(0.5,0.5)"), col = c("red", "blue", "green", "purple"), bty="n",pch = 21:22)
```

### 1.3.11 R - Function Tables

Below is a table of the summary of a the R functions found in this section.

| Distribution | R function |
|---|---|
| Binomial | binom |
| Geometric | geom |
| Poisson | pois |
| Gaussian | norm |
| Student-t | t |
| Lognormal | lnorm |
| Gamma | gamma |
| Weibull | weibull |
| Exponential | exp |
| Chi-squared | chisq |
| F | f |
| Uniform | unif |
| Beta | beta |

The following table contains the different prefixes to the above functions. In this table, the normal distribution was used as an example.

| Function | Syntax | Purpose |
|---|---|---|
| r | rnorm(n,mean) | Generates random number from distribution |
| d | dnorm(x,mean) | Gives the density of the distribution |
| p | pnorm(q,mean) | Gives the area under the distributions curve |
| q | qnorm(p,mean) | Gives the value at which the CDF of the distribution is at a certain percentile |

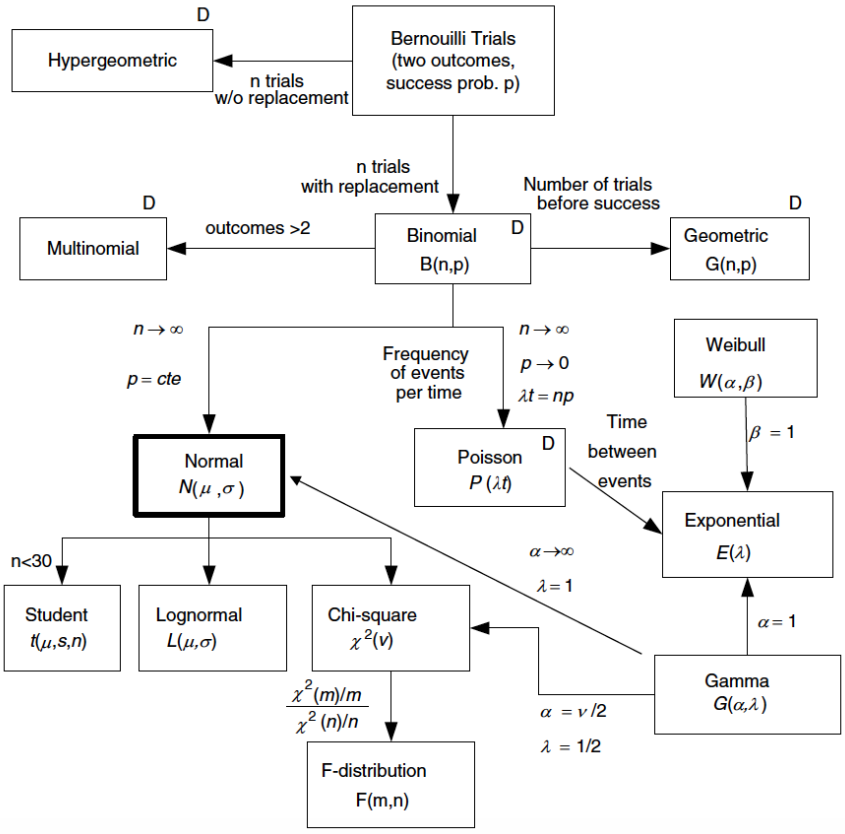



**Fig. 2.9** Genealogy between different important probability distribution functions. Those that are discrete functions are represented by "D" while the rest are continuous functions. (Adapted with modification from R. E. Lave, Jr. of Stanford University)

Figure 28: Figure showing the relationships between each distribution. The distributions that are dicrete are marked with a D. (Figure 2.9, Reddy 2011)

## 1.4 Parameter Estimation

The majority of the material and examples provided in this section were adapted from the following sources. Additional sources are listed in the references.

- Reddy, T.Agami. Applied Data Analysis and Modeling for Energy Engineers and Scientists. Springer, 2011.
- Stat 414/415: Probability Theory and Mathematical Statistics, PennState Eberly College of Science. (2017). The Pennsylvania State University. Retrieved from https://onlinecourses.science.psu.edu/stat414/node/213.

Parameter estimation problems can be thought of as optimization problems.

Suppose we have a random sample $x_1, x_2, ..., x_n$ with a probability distribution function with parameter $\theta$. We want to obtain a good "point" estimate of $\theta$, or $\hat{\theta}$, such that we can closely recreate the observed values of the random sample.

In this section we will explore two common parameter estimation methods:

- **Methods of Moments**
- **Maximum Likelihood Estimation**

Both of these methods are used to find estimates of distribution parameters when sample data is available.

A good estimator will

- be **unbiased**: If the statistic is neither an underestimate nor an overestimate of a population parameter, then that statistic is said to be unbiased. i.e.. If the estimator, or parameter mean, equals the parameter, then the estimator is unbiased:
  - Let $\theta$ be the parameter of the a probability distribution and $\hat{\theta}$ be the estimator of that parameter. Then, if $E[\hat{\theta}] = \theta$, then the statistic $\hat{\theta}$ is an unbiased estimator of the parameter $\theta$.
- have **small variance**: i.e. small standard deviation
- be **efficient**: the estimation has the smallest mean squared error (MSE).
  - $MSE(\hat{\theta}) = E(\hat{\theta} - \theta)^2$
- be **Consistent**: if sample size $n$ goes to $+\infty$, $\hat{\theta}$ will converge in probability to $\theta$

### 1.4.1 Method of Moments Estimation (MME)

In method of moments estimation parameters are estimated by equating **sample moments** with **theoretical moments**.

Moments of a distribution can be used to characterize the shape of the distribution. The formula for the different types of moments discussed in this section are found below:

**Sample Moments**:

- About the origin: $M_k = \frac{1}{n}\sum_{i=1}^{n}(x_i)^k = \bar{x}$ is the $k^{th}$ sample moment, for $k = 1, 2, \ldots$
- About the mean: $M_k = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^k$ is the $k^{th}$ sample moment for $k = 1, 2, \ldots$

**Theoretical Moments**:

- About the origin: $E(x^k) = \int_{-\infty}^{\infty} x_i^k f(x|\theta)d\theta = \mu$ is the $k^{th}$ theoretical moment of the distribution for k=1,2,...

- About the mean: $E[(x-\mu)^k] = \int_{-\infty}^{\infty}(x_i-\mu)^k f(x|\theta)d\theta$ is the $k^{th}$ theoretical moment of the distribution for k=1,2,...

**MME: The Basics**

- Equate the first sample moment about the origin $M_1 = \frac{1}{n}\sum_{i=1}^{n} x_i = \bar{x}$ to the first theoretical moment $E(x)$.
- Equate the second sample moment about the origin $M_2 = \frac{1}{n}\sum_{i=1}^{n}(x_i)^2$ to the second theoretical moment $E(x^2)$
- Continue this process until you have as many equations as you have distribution parameters
- Solve for parameters

Lets look at some examples.

**MME Example - Bernoulli**

Let $x_1, x_2, ..., x_n$ be Bernoulli random variables with parameter $p$. What is the method of moments estimator of p?

The first theoretical moment about the origin

- $E(x_i) = p$

Since we are trying to estimate one parameter we only need one equation. We find this by equating the first theoretical moment about the origin with the corresponding sample moment:

- $p = \frac{1}{n}\sum_{i=1}^{n}(x_i)$

Since the goal is to solve for $p$, our work is done:

- $\hat{p} = \frac{1}{n}\sum_{i=1}^{n}(x_i)$

**MME Example - Gaussian**

Let $x_1, x_2, ..., x_n$ be normal random variables with mean, $\mu$, and variance, $\sigma^2$. What are the method of moments estimators of $\mu$ and $\sigma^2$?

In this case we have two parameters and therefore need two equations:

- 1st moment: $E(x_i) = \mu = \frac{1}{n}\sum_{i=1}^{n} x_i$
- 2nd moment: $E(x_i^2) = \sigma^2 + \mu^2 = \sum_{i=1}^{n}(x_i^2)$

The first equation tells us that the method of moments estimator for the mean, $\mu$, is the sample mean:

- $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i = \bar{x}$

Then, by substituting the sample mean in for $\mu$ in the second equation and solving for $\sigma^2$, we get that the method of moments estimator for the variance $\sigma^2$ is:

- $\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$

**MME Trivial Example - Gamma**

A similar process can be used to find the parameter estimators using the theoretical and sample moments about the mean.

Let $x_1, x_2, ..., x_n$ be gamma random variables with parameters $\alpha$ and $\theta$. Again since we have two parameters, we need two equations:

- $E(x) = \alpha\theta = \frac{1}{n}\sum_{i=1}^{n} x_i = \bar{x}$
- $Var(x) = \alpha\theta^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$

Solving for $\alpha$ in the 1st equation we get:

- $\alpha = \frac{\bar{x}}{\theta}$

By substituting $\alpha$ into the second equation:

- $\alpha\theta^2 = (\frac{\bar{x}}{\theta})\theta^2 = \bar{x}\theta = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$

We then can solve for $\theta$:

- $\hat{\theta} = \frac{1}{n\bar{x}}\sum_{i=1}^{n}(x_i - \bar{x})^2$

Lastly, by substituting $\hat{\theta}$ into the equation for $\alpha$, we get:

- $\hat{\alpha} = \frac{\bar{x}}{\hat{\theta}} = \frac{\bar{x}}{(1/n\bar{x})\sum_{i=1}^{n}(x_i - \bar{x})^2} = \frac{n\bar{x}^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$

### 1.4.2 Maximum likelihood Estimation (MLE)

**MLE: The Basics**

It would make sense that to find the best estimate of the parameter as that which maximizes the probability or "likelihood" of getting the observed data. Thus, in maximum likelihood estimation, we use the likelihood function.

Suppose we have an independent and identically distributed random sample, $x_1, x_2, ..., x_n$ for which the probability density function of each $x_i$ is $f(x_i; \theta)$, then the **likelihood function** of $x_1, x_2, ..., x_n$ is:

$$L(\theta) = P(x_1 = x_1, x_2 = x_2, ..., x_n = x_n) = f(x_1; \theta)f(x_2; \theta)\cdots f(x_n; \theta) = \prod_{i=1}^{n} f(x_i; \theta)$$

We then need to find the value of the parameter, in this case $\theta$, which maximizes the likelihood function. The process to achieve this is shown in the examples below.

**MLE - Bernoulli Distribution**

If $x_i$ are independent Bernoulli random variables with unknown parameter $p$, then we know that the probability mass function of each $x_i$ is:

- $f(x_i; p) = p^{x_i}(1-p)^{1-x_i}$

for $x_i = 0$ or 1 and $0 < p < 1$. Therefore, the likelihood function $L(p)$ is:

$$L(p) = \prod_{i=1}^{n} f(x_i; p) = p^{x_1}(1-p)^{1-x_1}p^{x_2}(1-p)^{1-x_2}...p^{x_n}(1-p)^{1-x_n}$$

for $0 < p < 1$. After summing the exponents we get:

- $L(p) = p^{\sum x_i}(1-p)^{n-\sum x_i}$

We now need to find the $p$ that maximizes the likelihood function $L(p)$.To do this we must differentiate $L(p)$ with respect to $p$.

Since the natural logarithm is a strictly monotonically increasing function of $x$, the value of $p$ that maximizes the natural logarithm of the likelihood function, $\ln(L(p))$, is also the value of $p$ that maximizes the likelihood function $L(p)$ itself. Therefore, we us the **log likelihood function**.

- $\ln L(p) = (\sum x_i)\ln(p) + (n - \sum x_i)ln(1-p)$

Now, taking the derivative of the log likelihood, and setting to 0, we get:

- $\frac{\mathrm{d}\ln L(p)}{\mathrm{d}p} = \frac{\sum x_i}{p} - \frac{n-\sum x_i}{1-p} = 0$

Now, multiplying through by $p(1-p)$, we get:

- $(\sum x_i)(1-p) - (n - \sum x_i)p = 0$

We can now cancel terms and we are left with:

- $\sum x_i - np = 0$

All that is left to do now is to solve for $p$.

- $\hat{p} = \frac{1}{n}\sum_{i=1}^{n}(x_i)$

Note that the method of moments estimator is the same as the maximum likelihood estimator for the Bernoulli distribution.

**MLE - Normal Distribution** (Adapted from https://onlinecourses.science.psu.edu/stat414/node/191)

Let $x_1, x_2, ..., x_n$ be a random sample from a normal distribution with unknown mean $\mu$ and variance $\sigma^2$. Find the maximum likelihood estimators of mean $\hat{\mu}$ and variance $\hat{\sigma^2}$.

We first write the PDF of the normal distribution as a function of $\theta_1 = \mu$ and $\theta_2 = \sigma^2$:

- $f(x_i;\theta_1,\theta_2) = \frac{1}{\sqrt{\theta_2 2\pi}}\exp[-\frac{(x_i-\theta_1)^2}{2\theta_2}]$

Then the likelihood function is:

- $L(\theta_1,\theta_2) = \prod_{i=1}^{n} f(x_i;\theta_1,\theta_2) = \theta_2^{-n/2}(2\pi)^{-n/2}\exp[-\frac{1}{2\theta_2}\sum_{i=1}^{n}(x_i-\theta_1)^2]$

The log likelihood function is:

- $\ln L(\theta_1,\theta_2) = -\frac{n}{2}\ln\theta_2 - \frac{n}{2}\ln(2\pi) - \frac{\sum(x_i-\theta_1)^2}{2\theta_2}$

We then take the derivative with respect to $\theta_1$ ($\mu$), and set it to zero.

- $\frac{\mathrm{d}\ln L(\theta_1,\theta_2)}{\mathrm{d}\theta_1} = \frac{\sum(x_i-\theta_1)}{\theta_2} = 0$

After multiplying through by $\theta_2$, and distributing the summation, we get:

- $\sum x_i - n\theta_1 = 0$

Solving for the estimate of $\theta_1$ we get:

- $\hat{\theta}_1 = \hat{\mu} = \frac{\sum x_i}{n} = \bar{x}$

Completing the same processes by taking the derivative of the log likelihood function with respect to $\theta_2$, we get:

- $\hat{\theta}_2 = \hat{\sigma}^2 = \frac{\sum (x_i - \bar{x})^2}{n}$

In the end we get the parameter estimators for the normal distribution to be:

- $\hat{\mu} = \frac{\sum x_i}{n} = \bar{x}$ and $\hat{\sigma}^2 = \frac{\sum (x_i - \bar{x})^2}{n}$

**MLE for Exponential Distribution**

The exponential distribution give by the following one parameter model:

- $E(x; \lambda) = \lambda e^{-\lambda x}$ if x>0

  = 0 otherwise

The likelihood function is:

- $L(\lambda | x_i) = \prod_{i=1}^{n} p(x_i | \lambda) = (\lambda e^{-\lambda x_1})(\lambda e^{-\lambda x_2})(\lambda e^{-\lambda x_3})... = \lambda^n e^{-\lambda \sum x_i}$

Taking the logs, we get:

- $\ln L(\lambda) = n \ln \lambda - \lambda \sum x_i$

Differentiating with respect to $\lambda$ and setting it to zero:

- $\frac{\mathrm{d} \ln L(\lambda)}{\mathrm{d}\lambda} = \frac{n}{\lambda} - \sum x_i = 0$, from which $\lambda = \frac{n}{\sum x_i} = \frac{1}{\bar{x}}$

**Example 10.4.1 - MLE for Exponential Distribution**

The lifetime of several products and appliances can be described by the exponential distribution given by the following one parameter model.

- $E(x, \lambda) = \lambda e^{-\lambda x}$ if $x > 0$
- `= 0 otherwise`

Sixteen appliances have been tested and operation life data in hours are assembled in the following table.

| Table 10.14 |
|---|
| 2,100 2,107 2,128 2,138 2,167 2,374 2,412 2,435 |
| 2,438 2,246 2,596, 2,692 2,738 2,985 2,996 3,369 |

The parameter $\lambda$ is to be estimated using MLE.

From the trivial example for the exponential distribution we know that the MLE parameter estimator is:

- $\lambda = \frac{n}{\sum x_i} = \frac{1}{\bar{x}}$

From this we can solve for $\lambda$:

- $\lambda = (2508.188)^{-1} = 0.000399$

**Example 10.4.2 - MLE for Weibull distribution** The observations assembled in Table 10.15 are values of wind speed (m/s) at a certain location. The Weibull distribution with parameters $(\alpha, \beta)$ describes this sample.

- $p(x) = \frac{\alpha}{\beta^\alpha} x^{\alpha - 1} e^{-(x/\beta)^\alpha}$

Estimate the values of the parameters using MLE:

| Table 10.15 |
|---|
| 4.7 5.8 6.5 6.9 7.2 7.4 7.7 7.9 8.0 |

| Table 10.15 |
|---|
| 8.1 8.2 8.4 8.6 8.9 9.1 9.5 10.1 10.4 |

After taking the log of the likelihood function of the Weibull distribution, taking the derivative and maximizing it by setting it equal to zero, the parameter estimators for the Weibull distribution are found to be:

- $\alpha = [\frac{\sum x_i^\alpha ln(x_i)}{\sum x_i^\alpha} - \frac{\sum ln(x_i)}{n}]^{-1}$
- $\beta = \left[\frac{\sum x_i^\alpha}{n}\right]^{1/\alpha}$

From here we can solve for the parameters and we get $\alpha = 7.9686$ and $\beta = 0.833$

**Example - MLE for Normal Distribution**

Suppose the weights of randomly selected American female college students are normally distributed with unknown mean $\mu$ and standard deviation $\sigma$. A random sample of 10 American female college students yielded the following weights(lbs).

- Find the mean weight, $\mu$.

  115, 122, 130, 127, 149, 160, 152, 138, 149, 180

We know that the maximum likelihood estimate of a normal distribution for $\mu$ is:

- $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i = \bar{x}$

From here we can easily solve for the mean, $\mu$:

- $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{10}(115 + ... + 180) = 142.2$ pounds

**Advantages of MLE**

- Though biased for small samples, the bias reduces as the sample size increases and the estimator (parameter mean) converges to the true mean.
- MLE is generally superior to MME is terms of yielding a estimator with minimum variance.
- MLE is straightforward and easily solved by computers.
- MLE can handle any type of error distribution where MME is limited to normally distributed error.
- MLE allows generation of estimators of parameters that are generally more efficient (i.e have smaller mean square error) and consistent than those found using MME.

### 1.4.3 MME and MLE in R

ExtDist is a very useful R package when estimating MME and MLE parameters, which is basically an extension of the distribution functions that we have previously introduced. With this package you can also create PDFs an CDFs as we did before.

Please note: The default setting for the method that is used (i.e. MME and MLE) changes for each distribution. Typically, the default is set to some form of MLE. There are two types of MLE in this package to be aware of:

- **Analytical MLE**: MLE calculated by taking the derivative of the log likelihood function, setting the result to zero and solving for the parameter. The data is then plugged into this parameter estimation equation. This is the process that we have shown in this lecture.
- **Numerical MLE**: MLE found by numerically maximizing the parameter in the likelihood function.

You can specify which method you use by modifying the method argument:

- method="moments, MOM, analytical.MLE, numerical.MLE" depending on the PDF.

Please look at the ExtDist vignette for more details.

**Example 10.4.1 - R Applications** Using the ExtDist package, we are able to find the parameters of an exponential distribution with the eExp() function:

```
op_life <- c(2100,2107,2128,2138,2167,
2374,2412,2435,2438, 2456,2596,2692,2738
,2985,2996,3369)
est.par <- eExp(op_life); est.par
```

```
##
## Parameters for the Exp distribution.
## (found using the  analytical.MLE method.)
##
##  Parameter  Type     Estimate         S.E.
##       rate scale 0.0003986943 9.967357e-05
```

**Example - MLE for Normal Distribution**

Suppose the weights of randomly selected American female college students are normally distributed with unknown mean $\mu$ and standard deviation $\sigma$ . A random sample of 10 American female college students yielded the following weights(lbs).

- Find the mean weight, $\mu$.

  115, 122, 130, 127, 149, 160, 152, 138, 149, 180

Using the ExtDist Package, we are able to find the parameters of an normal distribution with the eNormal function where the shape is $\mu$ and the scale is $\sigma$:

```
weight <- c(115, 122, 130, 127, 149, 160, 152,
138, 149, 180)
est.par <- eNormal(weight); est.par
```

```
##
## Parameters for the Normal distribution.
## (found using the  unbiased.MLE method.)
##
##  Parameter     Type  Estimate     S.E.
##       mean location 142.20000 6.217895
##         sd    scale  19.66271 4.634546
```

**MLE using the MASS package**

It is also very common to use the fitdistr() function from the MASS package. Using the same example above, we will use the MASS package.

Again, using example 10.4.1, we find the parameters of an exponential distribution, but this time with the fitdistr() function:

```
library(MASS)
op_life <- c(2100,2107,2128,2138,2167,
2374,2412,2435,2438, 2456,2596,2692,2738
,2985,2996,3369)
est.par1 <- fitdistr(op_life, "exponential")
est.par
```

```
##
## Parameters for the Normal distribution.
```

```
## (found using the  unbiased.MLE method.)
##
##  Parameter     Type  Estimate     S.E.
##       mean location 142.20000 6.217895
##         sd    scale  19.66271 4.634546
```

**MME Example - Gamma Distribution**

Using the ExtDist Package, we are able to find the MME parameters of a Gamma distribution with the eGamma function. The default of the eGamma function is also set to MLE, so here we need to set the method="moments":

```
data <- c(16, 11.6, 19.9, 18.6, 18, 13.1, 29.1,
10.3, 12.2, 15.6, 12.7, 13.1,19.2, 19.5, 23, 6.7,
7.1, 14.3, 20.6, 25.6,8.2, 34.4, 16.1, 10.2, 12.3)
est.par <- eGamma(data,method="moments"); est.par
```

```
##
## Parameters for the Gamma distribution.
## (found using the  moments method.)
##
##  Parameter  Type  Estimate     S.E.
##      shape shape 5.8488498 1.935982
##      scale scale 0.3589132 1.609336
```

## 1.5 Bayesian Probability

The majority of the material and examples provided in this section were adapted from the following sources. Additional sources are listed in the references.

- Reddy, T.Agami. Applied Data Analysis and Modeling for Energy Engineers and Scientists. Springer, 2011. *
- Elfessi, E. Reineke, D. (2001) A Bayesian Look at Classical Estimations: The Exponential Distribution. Journal of Statistics Eduation Volume 9, Number 1. Retrieved from http://ww2.amstat.org/publications/jse/v9n1/elfessi.html.*
- The Pennsylvania State University. Bayesian Estimation. (2017). Retrieved from https://onlinecourses.science.psu.edu/stat414/node/213.*

As previously discussed, there are two main schools of thought concerning probability:

- The classical or **frequentist** view interprets probability as the long run frequency and excludes all subjective insights.
- The Bayesian or **subjectivist** view integrates prior knowledge with observed events.

The Bayesian approach argues that frequentists too easily leave out any subjective insights into the probability of an event occurring, even when one might have a good idea of the most likely outcome.

Note: Both converge to the same results as more data is gathered. The Bayesian approach is only beneficial when the available data sets are small. **All statisticians accept Bayes Theorem, it is the interpretation of probability which is controversial.**

### 1.5.1 Bayes' Theroem

Bayes' Theorem is a formula that allows for the inclusion of prior information in multi-stage experiments. Previously, we focused on continuous random variables and their properties. Here it is easy to derive for discrete random variables that:

Joint probability of event $A$ and $B$:

- $p(A \cap B) = p(A|B)p(B)$

Marginal probability of event $A$:

- $p(A) = p(A \cap B) + p(A \cap \bar{B})$

Conditional probability that event $B$ occurs given event $A$ has already occurred:

- $p(B|A) = \frac{p(A \cap B)}{p(A)}$

Substituting $p(A)$ from the marginal probability into the conditional probability we get:

- $p(B|A) = \frac{p(A \cap B)}{p(A \cap B) + p(A \cap \bar{B})}$

We can also re-arrange the conditional probability to be:

- $p(A \cap B) = p(A)p(B|A) = p(B)p(A|B)$

Putting this all together we get the **law of total probability** or **Bayes' Theorem**:

- $p(B|A) = \frac{p(A|B)p(B)}{p(A|B)p(B) + p(A|\bar{B})p(\bar{B})}$

Bayes' theorem is therefore simply a restatement of the conditional probability. So why is it so insightful?

- The probability is now in terms of it disjoint parts $\{B, \bar{B}\}$

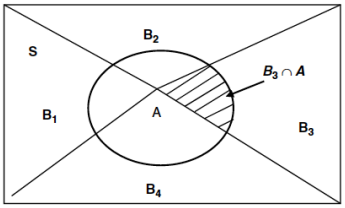



Figure 29: The sample space with 4 events. Given that event A has occured, the conditional probability of B3 is represented by the shaded area. (Figure 2.25, Reddy 2011)

- The probabilities have been "flipped" i.e. $p(B|A)$ is now expressed in terms of $p(A|B)$. For example if $A$ and $B$ are events and $A$ is observed while $B$ is not, the probability of $B$ can be inferred from the "flip" probability.

Bayes' theorem is made up of:

- **prior probability** $p(B)$: The probability of $B$, which represents the opinion before any data was collected
- **posterior probabilty** $p(B|A)$: Represents the opinion revised in light of new data
- **likelihood** $p(A|B)$: The probability of $A$ given $B$ also known as the conditional probability.

Bayes' theorem can now be written in words as:

Posterior probabilty of event B given event A $= \frac{\text{(Likelihood of A given B)(Prior probabilty of B)}}{\text{Prior probabilty of A}}$

### 1.5.2 Bayes' Theorem for Multiple Events

So far we have discussed Bayes' Theorem for the case when only one of two events is possible. Lets now consider the case where there are n events, $B_1, \ldots, B_n$ which are disjoint (no elements in common) and make up the entire sample space in Figure 29.

From Figure 29, if an observable event A has already occurred, the conditional probability of $B_3 : p(B_3|A) = \frac{p(B_3 \cap A)}{p(A)}$ which describes the **ratio of the hatched area to the total are inside the ellipse**.

The law of total probability states that the probability of an event $A$ is the sum of its disjoint parts:

$$p(A) = \sum_{j=1}^{n} p(A \cap B_j) = \sum_{j=1}^{n} p(A|B_j)p(B_j)$$

Therefore, Bayes' theorem for multiple events is:

$$p(B_i|A) = \frac{p(A \cap B_i)}{p(A)} = \frac{p(A|B_i)p(B_i)}{\sum_{j=1}^{n} p(A|B_j)p(B_j)}$$

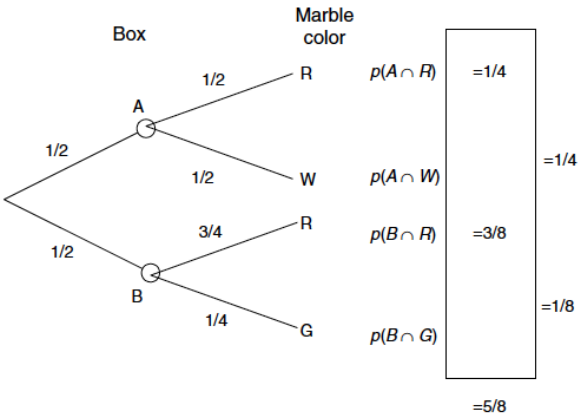



**Fig. 2.3** The first stage of the forward probability tree diagram involves selecting a box (either A or B) while the second stage involves drawing a marble which can be red (R), white (W) or green (G) in color. The total probability of drawing a red marble is 5/8

Figure 30: A tree diagram which depicts a two-stage process for determining the total probability of choosing each type of marble. (Figure 2.3, Reddy 2011)

### Example 2.5.1

Consider a two-stage experiment where there are two boxes with red, white and green marbles inside. In this example, we already know some information about the marbles in the box.

- Box 1: 1 red and 1 white
- Box 2: 4 red and 1 green

Giving the following tree diagram:

We will now try to do this example going the other direction. Let's say that a red marble has already been chosen. This prior knowledge, the prior probabilities R, W, and G, can be used to determine from which box the marble came from, the posterior probabilities.

- $p(B|R) = \frac{\frac{1}{2}\cdot\frac{3}{4}}{\frac{1}{2}\cdot\frac{1}{2}+\frac{1}{2}\cdot\frac{3}{4}} = \frac{3}{5}$
- $p(A|R) = \frac{\frac{1}{2}\cdot\frac{1}{2}}{\frac{1}{2}\cdot\frac{1}{2}+\frac{1}{2}\cdot\frac{3}{4}} = \frac{2}{5}$

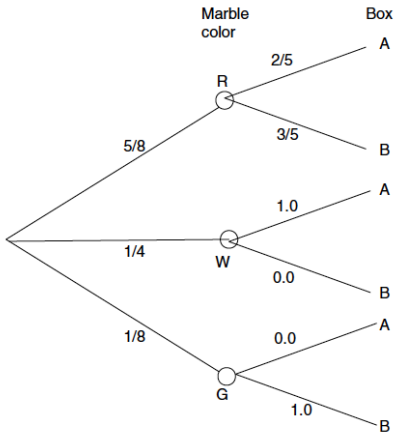



**Fig. 2.26** The probabilities of the reverse tree diagram at each stage are indicated. If a red marble (R) is picked, the probabilities that it came from either Box A or Box B are 2/5 and 3/5 respectively

Figure 31: A reverse tree diagram which, knowning the color of the marble already chosen, can help find the probabiltiy of it coming from a specific box. (Figure 2.26, Reddy 2011)

**Example 2.5.2 Fault Detection System**

- A fault detection system correctly identifies faulty operation when indeed it is faulty (this is referred to as **sensitivity**) 90% of the time. This implies that there is a probability $p = 0.10$ of a **false negative** occurring (i.e., a missed opportunity of signaling a fault).
- Also, the fault detection system correctly predicts status (or **specificity**) of the detection system (i.e., system identified as healthy when indeed it is so) is 0.95, implying that the **false positive or false alarm rate** is 0.05.
- Finally, historic data seem to indicate that the large piece of equipment tends to develop faults only 1% of the time.

**Priors**:

- Probability of getting fault-free equipment $P(A) = 0.99$
- Probability of getting faulty equipment $P(B) = 0.01$

**Likelihoods**:

- Probability of getting a correct fault-free reading $P(A1|A) = 0.95$
- Probability of getting a false fault-free reading $P(A2|A) = 0.05$
- Probability of getting a correct faulty reading $P(B1|B) = 0.90$
- Probability of getting a false faulty reading $P(B2|B) = 0.10$

Suppose a fault has been signaled. What is the probability that this a false alarm? Find the **posterior probability** of fault free equipment (event A) has occurred given a fault alarm (event A2), $p(A|A2)$.

- $p(A|A2) = \frac{p(A|B)p(B)}{p(A|B)p(B)+p(A|\bar{B})p(\bar{B})} = \frac{(0.99)(0.05)}{(0.99)(0.05)+(0.01)(0.90)} = 0.846$

This means that there is a 84.6% chance of getting a false positive! This is very high for practical situations and could well result in the operator disabling the fault detection system altogether.

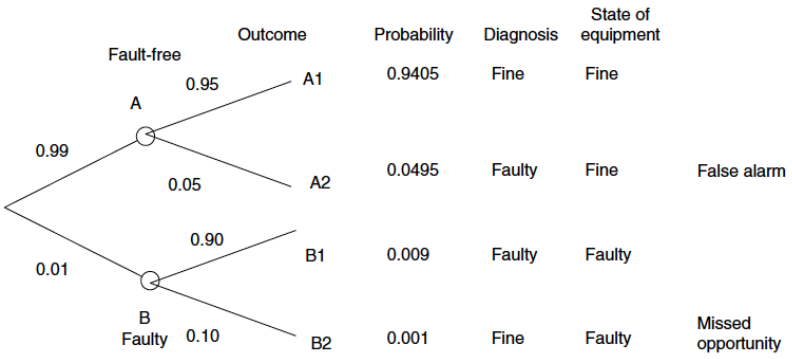



**Fig. 2.27** The forward tree diagram showing the four events which may result when monitoring the performance of a piece of equipment

Figure 32: The forward decision giving the probabiltiy of occurance of each event. (Figure 2.27, Reddy 2011)

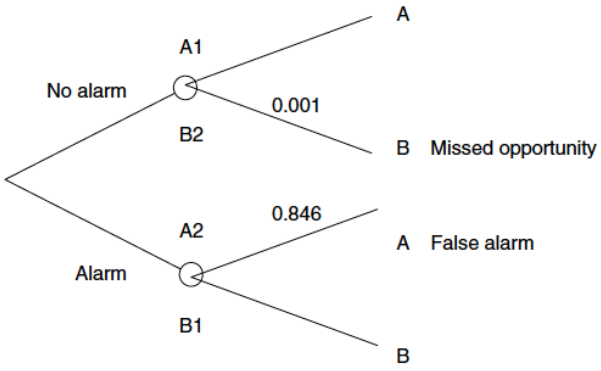



**Fig. 2.28** Reverse tree diagram depicting two possibilities. If an alarm sounds, it could be either an erroneous one (outcome A from A2) or a valid one (B from B1). Further, if no alarm sounds, there is still the possibility of missed opportunity (outcome B from B2). The probability that it is a false alarm is 0.846 which is too high to be acceptable in practice. How to decrease this is discussed in the text

Figure 33: The reverse tree diagram to be used after an event has already taken place. (Figure 2.28, Reddy 2011)

### 1.5.3 Applications to Continuous Probability Variables

Let $x$ be the random variable with a prior PDF denoted by $p(x)$. Though any distribution could be chosen, the Beta distribution is particularly convenient, and is widely used to characterize prior PDF. The Beta distribution can be rewritten to yield the **prior**:

- $p(x) \propto x^{\alpha}(1-x)^{b}$

Let $L(x)$ represent the conditional probability or likelihood function of observing $y$ "successes" out of $n$ observations. Then the posterior probability is given by:

- $f(x|y) \propto L(x)p(x)$

We now want to hold $p$ constant and study the behavior of the PDF of $x$. This gives the likelihood function:

- $L(x) = \binom{n}{y} x^{y}(1-x)^{n-y}, \quad 0 \leq x \leq 1$

We can multiply this likelihood function by the Beta prior to the get the posterior distribution:

- $f(x|y) = kx^{a+y}(1-x)^{b+n-y}$

where K is independent of x and is a normalization constant.

Note: The classical approach would use the likelihood function with exponents $y$ and $n-y$. You can see here that the posterior exponents have been inflated to $a+y$ and $b+n-y$. This is because the information contained in the prior has the net result of artificially augmenting the number of observations taken. Thus, Bayes' theorem is especially effective when few observations are available.

Recall from the higher the values of the exponents $a$ and $b$ in the Beta distribution, "peakier" distribution is indicative of the prior distribution being relatively well defined.

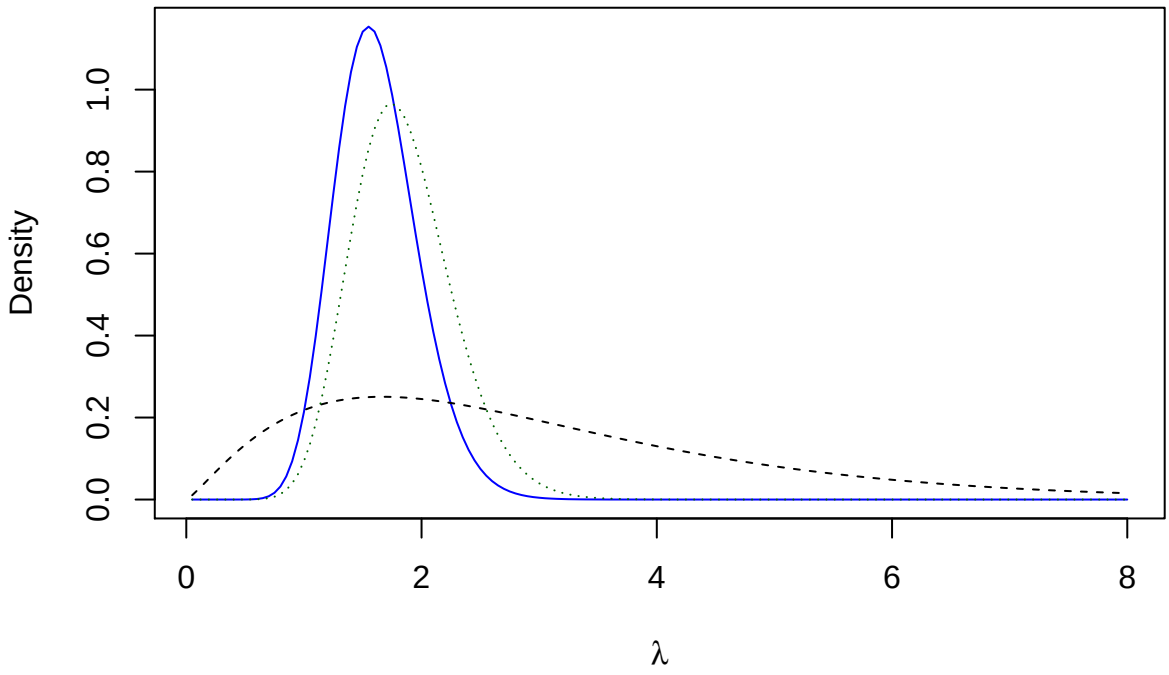



Figure 34: Plots of the prior, likelihood and the posterior functions

Plots of the prior, likelihood and the posterior functions are shown in Figure 34 . Notice how the posterior distribution has become more peaked reflective of the fact that the single test data has provided the analyst with more information than that contained in either the prior or the likelihood function.

**Example - Enhancing historical record of wind velocity using the Baysian approach**

Buildings are designed to withstand a maximum wind speed which depends on the location. The probability $x$ that the wind speed will not exceed 120 km/hr more than once in 5 years is to be determined. Past records of wind speeds of a nearby location indicated that the following beta distribution would be an acceptable prior for the probability distribution:

- $p(x) = 20x^3(1-x) \quad \text{for } 0 \leq x \leq 1$

In this case, the likelihood that the annual maximum wind speed will exceed 120 km/hr in 1 out of 5 years is given by:

- $L(x) = \binom{5}{4}x^4(1-x) = 5x^4(1-x)$

Hence the posterior probability is:

- $f(x|y) = k(L(x) * p(x) = k[5x^4(1-x)][20x^3(1-x)] = 100Kx^7(1-x)^2$

Where the constant $k$ can be found from the normalization criterion:

- $k = \left[\int_0^1 100x^7(1-x)^2 dx\right]^{-1} = 3.6$

Finally, the posterior PDF is given by

- $f(x|y) = 360x^7(1-x)^2 \quad \text{for } 0 \leq x \leq 1$

## 1.6 Bayesian Parameter Estimation

We have already explored some common classical methods of parameter estimation such as maximum likelihood estimation and method of moments. In this section we will discuss Bayesian parameter estimation. We will also discuss some interesting relationships between the two different approaches.

As we have previously seen, Bayes' Theorem is expressed as:

- $P(B_i|A) = \frac{P(B_i)P(A|B_i)}{\sum_{j=1}^{m} P(B_j)P(A|B_j)}$

where A is an event and $B_i$ are mutually exclusive and collectively exhausted events.

This same result can be translated into random variables. Let $X$ and $Y$ be continuous random variables and let $f_X(x)$ be the prior density of $X$ and $f(y|x)$ be the conditional density of $Y$ given $X$. Then in general, Bayes theorem is written as:

- $f(x|y) = \frac{f_X(x)f(y|x)}{\int_{-\infty}^{\infty} f_X(x)f(y|x)du}$

### 1.6.1 The Basics

We can know use our knowledge of Bayes' Theorem for continuous random variables and apply it to parameter estimation. Let $Y$ be a statistic described by parameter $u$. In order to estimate the parameter $u$ using the Bayesian method we need to know two things:

1. The **prior probability density function**, $f_U(u)$
2. The conditional probability density function, or the **likelihood function**, $g(y|u)$

We can then treat $g(y, u) = g(y|u)f_U(u)$ as the joint PDF of the statistic $Y$ and the parameter $u$ (the numerator in Bayes' theorem.)

Then, we can find the marginal distribution (the denominator in Bayes' theorem)

- $g_1(y) = \int_{-\infty}^{\infty} g(y, u)du = \int_{-\infty}^{\infty} f_U(u)g(y|u)du$

Using Bayes theorem we can then find the posterior PDF of parameter $u$:

- $g(u|y) = \frac{g(y,u)}{g_1(y)} = \frac{f_U(u)g(y|u)}{g_1(y)}$

To this point, we have done nothing new. We now want to find the parameter estimate $\hat{u}$. To do this, we consider the mean squared error:

Suppose we are trying to estimate the value of an unobserved parameter $u$, given that we have observed $Y = y$. In general, our estimate $\hat{u}$ is a function of y:

- $\hat{u} = g(y)$

The error in our estimate given by:

- $(u - \hat{u})$

We are often interested in minimizing the mean square error given by:

- $MSE(\hat{u}) = E[(u - \hat{u})^2] = E[u - g(y))^2]$

The mean square error (MSE) measures the difference, or error, between the true value and what is estimated. Therefore, a good estimator is one that minimizes this error.

Let's now try to find the minimum MSE among all possible estimators. Consider first the case that we would like to estimate $u$ without any observations. Then the MSE would be:

- $MSE = E[(u - g(y))^2]$
- $= E[u^2] - 2g(y)E[u] + g(y)^2$

We can then take the derivative with respect to $g(y)$.

- $-2E[u] + 2g(y) = 0$

By setting this equal to zero and solving for $g(y)$, we conclude that the minimum value of $g(y)$ is given by:

- $g(y) = E[u]$

Now consider if we have already observed $Y = y$. We can then repeat the above argument, with the only difference being that everything is conditioned on $Y = y$:

- $MSE = E[(u - g(u|y))^2]$
- $= E[u^2|y] - 2g(y)E[u|y] + g(y)^2$

Again, we minimize the value of $g(y)$ by differentiating the above quadratic function, setting it equal to zero, and solving for $g(y)$ to obtain:

- $\hat{u} = g(y) = E[u|y]$

Therefore, since $g(y) = \hat{u}$ we can conclude that $\hat{u} = E[u|y]$, i.e. the conditional expectation of u given y, has the lowest MSE among all other estimators $g(y)$, and is therefore the best estimate of $u$.

Lets try an example!

**Example 1**

Consider a random sample of independent observations $X_1, ..., X_n$ from an exponential distribution with probability density function:

- $f(x, \theta) = \theta \exp(-\theta x), \quad x > 0, \theta > 0$

Suppose the prior distribution of the parameter $\theta$ is of the form of the Gamma distribution:

- $\pi(\theta) = \theta^{\alpha-1} \exp(-\beta\theta), \quad \theta > 0, -\infty < \alpha < \infty, \beta \geq 0$

We then apply Bayes Theorem:

- $\pi(\theta|X_1, ..., X_n) \propto \frac{\pi(\theta) f(X_1,...,X_n|\theta)}{f(X_1,...,X_n)}$

where $f(X_1, ..., X_n)$ is the marginal distribution of $X$. The posterior distribution is then:

- $\pi(\theta|X_1, ..., X_n) \propto \theta^{n+\alpha-1} exp(-\theta(\beta + \sum_{i=1}^{n} X_i))$

The estimator $\hat{\theta}$ is derived by choosing that value of $\theta$ which minimizes the mean squared error $E[(\hat{\theta} - \theta)^2]$. The Bayes estimator of $\theta$ is given by:

$$\hat{\theta}_B = E[\theta|X_1, ..., X_n] = \frac{\int_0^\infty \theta(\theta^{n+\alpha-1} exp(-\theta(\beta + \sum X_n))d\theta)}{\int_0^\infty \theta^{n+\alpha-1} exp(-\theta(\beta + \sum X_n))d\theta)} = \frac{\frac{\alpha+n}{n}}{\beta + \sum_{i=1} X_i}$$

### 1.6.2 Bayesian and Classical Parameter Estimation

We have already shown the classical approach for finding the maximum likelihood estimator of $\theta$ for the exponential distribution. Recall that:

- $\hat{\theta} = \frac{n}{\sum_{i=1}^{n} X_i}$

From the previous example, we found that the estimation of $\theta$ using the Bayesian approach is:

- $\hat{\theta}_B = \frac{\alpha+n}{\beta+\sum_{i=1}^{n} x_i}$

We can see from this example that there is a connection between the Bayesian and classical estimators. By changing the Bayesian values of the parameters we are able to get to the classical estimator. In this case, we let $\alpha = 0$ and $\beta = 0$.

By changing the values of the Bayesian parameters $\alpha$ and $\beta$, you are also able to find other classical estimators. This is one useful relationship between the Bayesian and Classical methods.

### 1.6.3 R Applications

- The following example was created by Dr. Rajagopalan Balaji. http://civil.colorado.edu/~balajir/CVEN5454/R-sessions/sess3/Rjags-Gamma-param-example.R

JAGS is a program used for Bayesian model analysis using Marov chain Monte Carlo. JAGS provides a way to generate samples from the posterior distribution of the model parameters. In order to work with JAGS in r, the rjags package must be downloaded. In order to use rjags in R JAGS must be installed first. To introduce this package we will show an example using a gamma distribtion. Monthly flow data from Lees Ferry will be used. This data can be found at http://civil.colorado.edu/~balajir/r-session-files/Leesferry-mon-data.txt.

First, we will fit the PDF for September flow at Lees Ferry.

```
library(MASS)
library(rjags)
```

```
## Loading required package: coda
```

```
## Warning: package 'coda' was built under R version 3.3.2
```

```
## Linked to JAGS 4.2.0
```

```
## Loaded modules: basemod,bugs
```

```
test=matrix(scan("http://civil.colorado.edu/~balajir/r-session-files/Leesferry-mon-data.txt"),ncol=13,by

flows=test[,2:13]/10^6     #get it into MAF

y = flows[,9]    ## Sep flow
```

Using the MASS package and maximum likelihood estimation, we will fit the data to the the gamma distribution and find the parameter estimates.

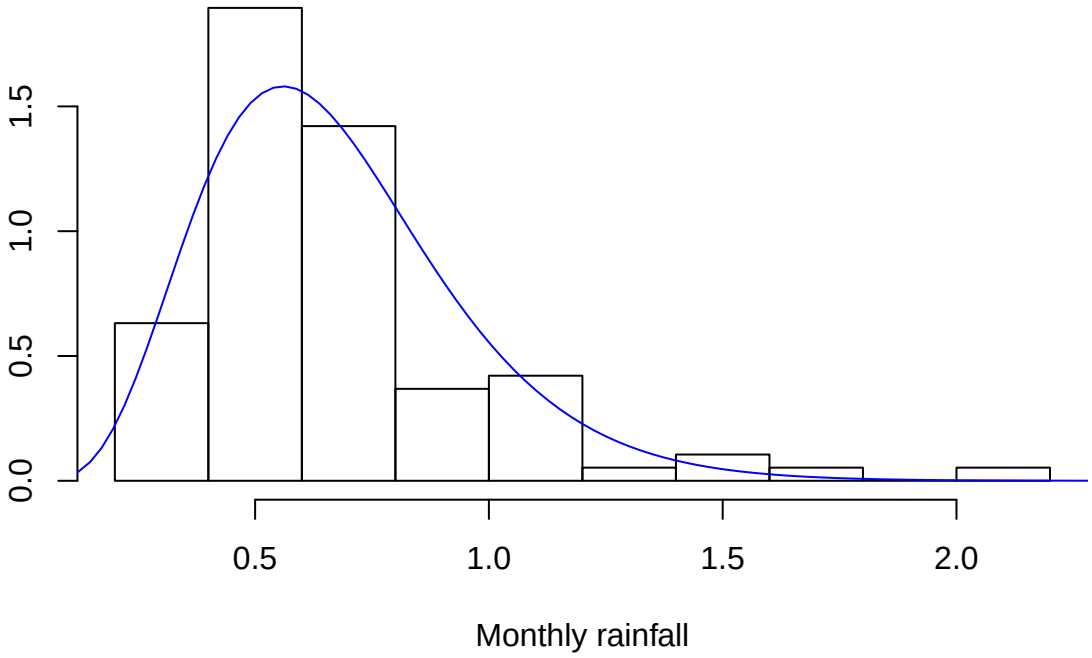
```
zz=fitdistr(y,"gamma")
shapefit = zz$estimate[1]
ratefit = zz$estimate[2]
```

We can visualized the data by overlaying the PDF on the histogram.

```
xeval=seq(max(0,min(y)-sd(y)),max(y)+sd(y),length=100)

xdensitygamma=dgamma(xeval,shape=zz$estimate[1],scale=1/zz$estimate[2])

zz=hist(y,freq=FALSE,plot=FALSE)
```

```
## Warning in hist.default(y, freq = FALSE, plot = FALSE): argument 'freq' is
## not made use of
```

```
hist(y,xlab="Monthly rainfall", ylab="", freq=FALSE,
main="",ylim=range(c(zz$density, xdensitygamma)))
lines(xeval,xdensitygamma,col="blue")
```




Now we can preform Bayesian parameter estimation using the rjags package.

Here we will designate

- lambda = rate
- scale = 1/rate
- rr = shape

```
N = length(y)
model_string <- "
model{
  for(i in 1:N) {
    y[i] ~ dgamma(rr,lambda)
  }
  lambda ~ dgamma(0.01,0.01)
  rr ~ dgamma(0.01,0.01)
}
"
 model1.spec<-textConnection(model_string)
```

The first line:

- y[i] ~ dgamma(rr,lambda)

specify that y follows a gamma distribution using the ~ to mean "distributed as", with parameters rr (shape) and lambda (rate).

The next two lines describes the priors

- lambda ~ dgamma(0.01,0.01) rr ~ dgamma(0.01,0.01)

We will now compile the model with the jags.model() function, get samples with the coda.samples() function and pull out the posterior simulations of lambda and rr using the the msms_samples function.

**Note**: After compiling, we specify a "burn-in" period of 1000 iterations with the update(model, 1000) function. Since rjags is a Markov chain based program, doing this will bring the Markov chain to a steady state.

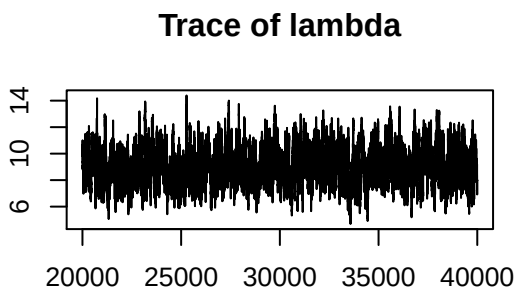
```
model <- jags.model(model1.spec, data = list(y = y,N=N), n.chains = 3, n.adapt= 10000)
```

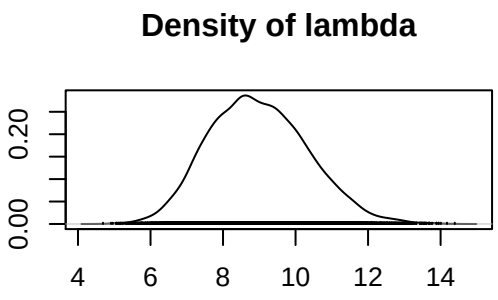
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 95
##    Unobserved stochastic nodes: 2
##    Total graph size: 102
##
## Initializing model
```

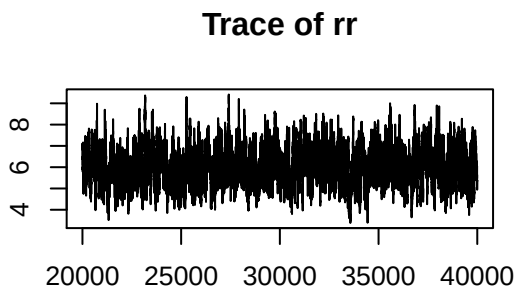
```
update(model, 10000);
mcmc_samples <- coda.samples(model, variable.names=c("rr","lambda"), n.iter=20000)

postsamp = mcmc_samples[[1]]
summary(postsamp)
```

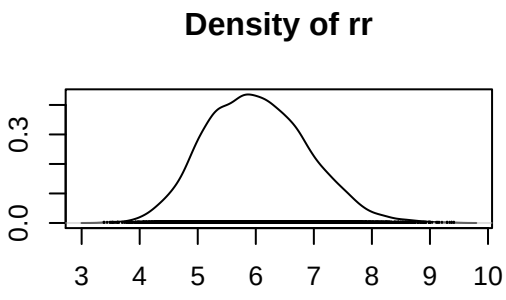
```
##
## Iterations = 20001:40000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 20000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##         Mean    SD Naive SE Time-series SE
## lambda 8.981 1.351 0.009556        0.05615
## rr     6.019 0.871 0.006159        0.03636
##
## 2. Quantiles for each variable:
##
##         2.5%   25%   50%   75%  97.5%
## lambda 6.563 8.001 8.907 9.891 11.744
## rr     4.458 5.381 5.975 6.603  7.799
```

```
plot(postsamp)
```



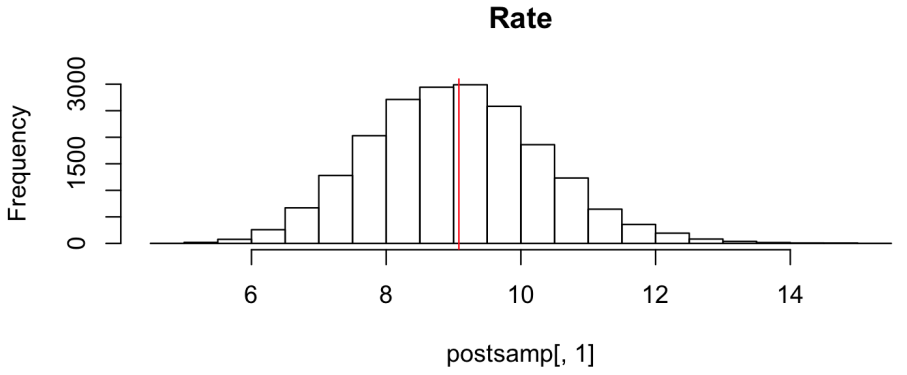



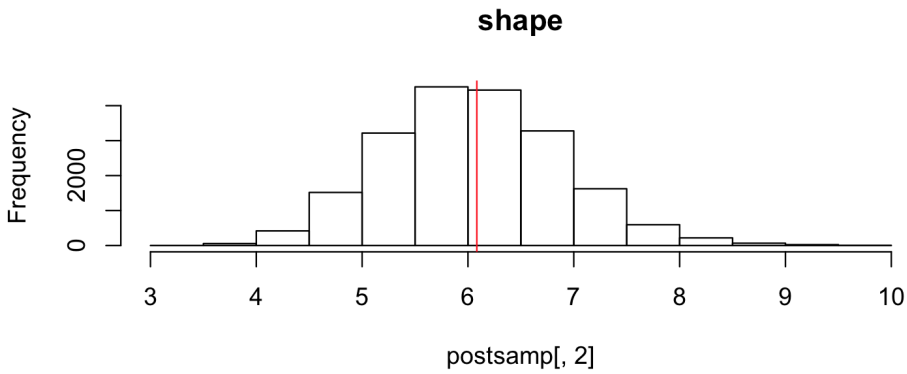




We can now view the posterior distribution of the parameters with the MLE estimates in red.

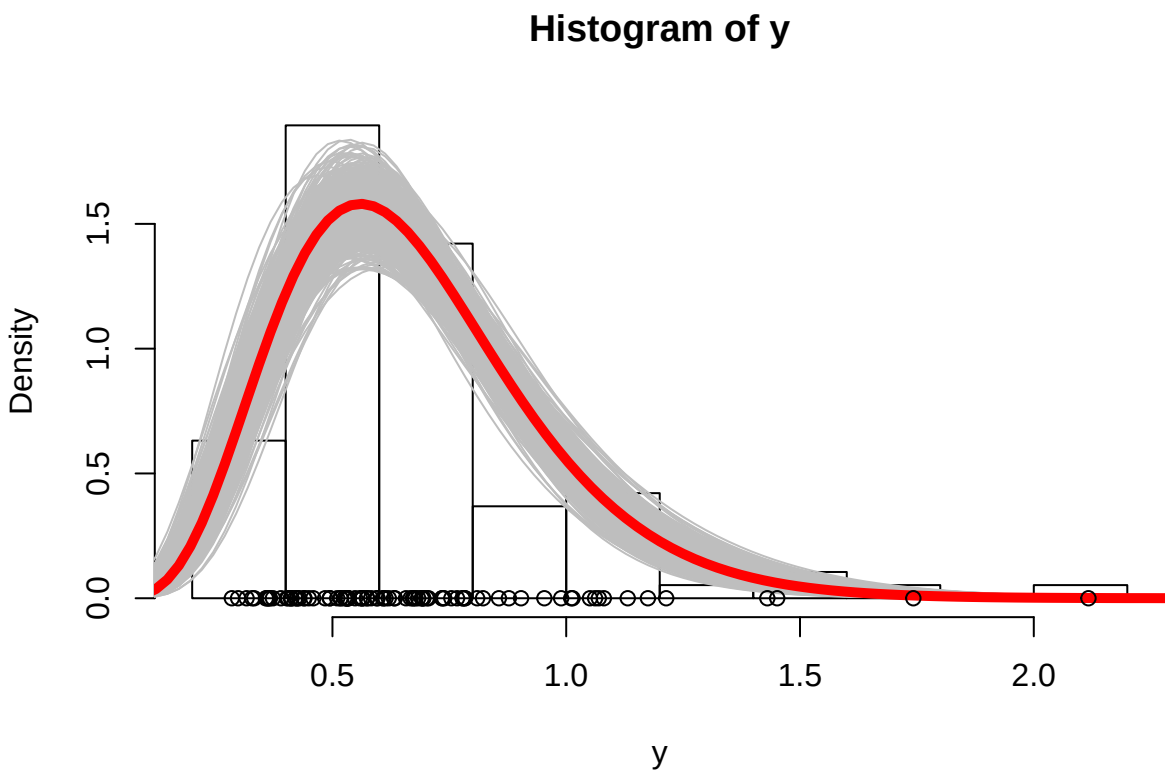
```
dev.new()
par(mfcol=c(2,1))
hist(postsamp[,1], main="Rate")   #lambda  = rate
abline(v=ratefit,col="red")

hist(postsamp[,2],main="shape")   #rr = shape
abline(v=shapefit,col="red")
```




Lastly, we will fit the gamma distribution, plot the PDFs from posterior parameters as grey scale, and overlay the PDF from MLE in red.

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

hist(y,freq=FALSE)

for(i in 1:500){
lines(xeval, dgamma(xeval, shape=postsamp[i,2], scale=1/postsamp[i,1]),col="grey")
}
lines(xeval,xdensitygamma,col="red",lwd=5)

points(y,rep(0,length(y)))
```


Histogram of y
Density
0.0
0.5
1.0
1.5
0.5
1.0
1.5
2.0
y

## 1.7 Descriptive Measures

The majority of the material provided in this section were adapted from the following sources. Additional sources are listed in the references.

- Reddy, T.Agami. Applied Data Analysis and Modeling for Energy Engineers and Scientists. Springer, 2011.

Descriptive summary measures of sample data are meant to characterize important statistical features. These measure make it much easier to interpret and understand the data. Below is several quick explanations on number of important measures.

a. **Mean** of a set or same of numbers is:

$x_{mean} = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ , where $n$ = sample size, and $x_i$ = individual reading

b. **Weighted mean** of a set of n numbers is:

$x = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i}$ , where $w_i$ is the weight for group $i$

c. **Geometric mean** is more appropriate when studying phenomena that exhibit exponential behavior (ex: population growth or biological processes)

$x_{geometric} = [x_1, x_2..., x_n]^{1/n}$

d. **Mode** is the value of the variate which occurs most frequently. When the variate is discrete, the mean may not take on a value that is not possible for the variate to take on. In this case, using the mode is a more appropriate value.

e. **Median** is the middle value of all variates. When mean is excessively influenced by the extreme observations, the medium may provide a more robust indicator of the central tendency of the data. In the case of an even number of observations, the median is the mean of the middle two numbers.

f. **Range** is the difference between the largest and the smallest observation values.

g. **Percentiles** are used to separate the data into bins. Let p by a number between 0 and 1. Then, the (100p)the (also called the quantile), represents the data value where 100p% of the data are lower. Thus 90% of the data will be below the 90th percentile, and the median represented by the 50th percentile.

h. **Inter-quartile range** (IQR) cuts the more extreme values in a distribution. It is the range which covers the middle 50% of the observations and is the difference between the lower quartile and the upper quartile.

i. **Deviation** of a number $x_i$ in a set of $n$ numbers is a measure of dispersion of the data from the mean, and is given by:

$d_i = (x_i - \bar{x})$

j. the **mean deviation** of a set of $n$ numbers is the mean of the absolute deviations:

$\bar{d} = \frac{1}{n}\sum_{i=1}^{n} |d_i|$

k. The **variance** or the mean square error (MSE) of a set of $n$ numbers is:

$s_x{}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{s_{xx}}{n-1}$ where $s_{xx} = \sum_{i=1}^{n}(x_i - \bar{x})^2$ sum of squares

l. the **standard deviation** of a set of $n$ numbers is

$s_x = \sqrt{\frac{s_{xx}}{n-1}}$

The more variation in a data set, the bigger the standard deviation, which is a measure of the actual absolute error.

m. **Coefficient of variation** is a measure of the relative error, and is often more appropriate than the standard deviation.

$CV = s_x/\bar{x}$

n. **Trimmed mean** is used for cases when the sample mean is sensitive to outliers and may bias the analysis results. The sample median is more robust since it is not effected by outliers. However, non-parametric tests which use the median are less efficient than parametric tests in general. Hence, the trimmed mean value, which is less sensitive to outliers, is used.

- A trimming percentage, 100r%, is selected. $0 < r < 0.25$ is recommended.
- Suppose one has a data set with $n = 20$. Selecting $r = 0.1$ implies that the trimming percentage is 10% (i.e. two observations). Then, two of the largest values and two of the smallest values of the data set are rejected prior to subsequent analysis.

o. **Covariance** provides a representation of the strength of the linear relationship between two variables $x$ and $y$:

- $cov(xy) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$ where $\bar{x}$ and $\bar{y}$ are the mean values of variables x and y.

### 1.7.1 Data Transformations

a. **Decimal scaling** moves the decimal point but still preserves most of the original data. The specific observations of a given variable may be divided by $10^x$ where $x$ is the minimum value so that all the observations are scaled between -1 and 1.

- For example, suppose the largest value in a sample is 289 and the smallest value is -150, then since $x = 3$, all observations are divided by 1000 so as to lie between [0.289 and -0.150].

b. **Min-max scaling** allows for better distribution of observations over the range of variation than does decimal scaling. It does this by redistributing the values to lie between [-1 and 1]. Hence, each observation is normalized as follows:

$z_i = \frac{x_i - x_{min}}{x_{max} - x_{min}}$ where $x_{max}$ and $x_{min}$ are the maximum and minimum numerical values respectively of the x variable.

c. **Standard deviation scaling** is widely used in multivariate statistical analysis but transforms the data into a form unrecognizable from the original data. Here, each observation is normalized as follows:

- $z_i = \frac{x_i - \bar{x}}{s_x}$ where $\bar{x}$ and $s_x$ are the mean and the standard deviation respectively of the $x$ variable.

# 2 Regression

The majority of the material and examples provided in this section were adapted from the following sources. Additional sources are listed in the references.

- Reddy, T.Agami. Applied Data Analysis and Modeling for Energy Engineers and Scientists. Springer, 2011.
- James, Witten, Hastie, and Tibshirani. (2013). An Introduction to Statistical Learning. New York: Springer

*By the completion of this section, the reader should be able to*

- *understand the benefits and limitations*
- *understand how to choose the best approach for different types of data*
- *be able to identify the best model through residual diagnosis and other methods such as stepwise regression*
- *use R to fit, diagnose, and select both simple and multiple linear models*
- *motivate the need for local regression models*

## 2.1 Ordinary Least Squares (OLS)

It is often helpful when analyzing data to create a statistical model which describes a relationship between the variation of the response (dependent) and the regressor (independent) variables. If observations are taken from both variables, one can build a mathematical model from this information, which can then be used as a predictive tool. Regression analysis allows us to analyze these relationships and build these types of predictive models.

The mains goals of regression:

- to identify the best model
- to determine the best values of the model parameters
- to explain variation in response for different values of the regressors

In general, for OLS we assume the true relationship takes the linear form, $Y = \beta_0 + \beta_1 X + \epsilon$, where $\beta_i$ are coefficients (in this case intercept and slope), $Y$ is the response, $X$ is the regressor, and $\epsilon$ is the error term. Ordinary least squares aims to achieve these goals by finding model coefficients in order to create a best fit line which minimizes the squared sum of the vertical deviations or residual:

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

where $y_i$ is the ith observed response variable, $\hat{y}_i$ is the ith value of $y$ estimated by the regression model, and n is the number of observations.

### 2.1.1 Measuring Model Accuracy

There are several important **model evalutation measures** which will help decipher one model fit over another. Some important measures include:

a) The coefficient of determination $R^2$, where $0 \leq R^2 \leq 1$:

- The better the fit of the linear model to the data the closer to 1 $R^2$ will be. A perfect fit would give $R^2 = 1$, while $R^2 = 0$ would mean that no linear relationships exist.

b) Root mean square error (RMSE): An estimate of the magnitude of the absolute error of the model. RMSE represents the average amount that the response will deviate from the true regression line

- $RMSE = \sqrt{\frac{SSE}{n-k}}$

- where $n$ is the number of observations and
- SSE is the error sum of squares of the residuals $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$. This represents the variation about the regression line.
- Also referred to as standard error of the estimate

Note: A normalized measure can also be used and is often beneficial. The **coefficient of variation of the RMSE** or the **CV** is defined as $CV = \frac{RMSE}{\bar{y}}$

c) The mean bias error (MBE): The mean difference between the actual data values and the model prediction values:

- $MBE = \frac{1}{n-k}\sum_{i=1}^{n}(y_i - \hat{y})$

d) The mean absolute deviation (MAD): The mean absolute difference between the actual data values and the model predicted values:

- $MAD = \frac{1}{n-k}\sum_{i=1}^{n}|y_i - \hat{y}_i|$

e) The **F-statistic** tests for significance of the overall regression model and is defined as:

- $F = \frac{\text{variance explained by the regression}}{\text{variance not explained by the regression}}$

Therefore, the smaller the F-statistic the poorer the regression model

The **quality of the regression coefficients** must also be analyzed. Are there significant relationships between the response and the regressor variables?

a) Standard error of the coefficients ($SE(\hat{\beta}_0)^2$): How close the estimates of the coefficients are to the true values of the coefficients

- $SE(\hat{\beta}_0)^2 = \sigma^2[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}]$
- $SE(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$

b) Confidence Intervals: a 95% confidence interval is defined as a range of values such that with 95% probability, the range will contain the true (yet unknown) value of the parameter. This range has an upper and lower limit. There is a 95% probability that the interval

$[\hat{\beta}_1 - 2SE(\hat{\beta}_1), \hat{\beta}_1 + 2SE(\hat{\beta}_1)]$

will contain the true value of $\hat{\beta}_1$

c) **Hyposthesis testing - the t-statistic**:

The null hypothesis:

- $H_0$: There is no relationship between $X$ and $Y$ ($\beta_1 = 0$)

The alternative hypothesis:

- $H_a$: There is some relationship between X and Y ($\beta_1 \neq 0$)

In order to reject the null hypothesis, we use the t-statistic which measures the number of standard deviations that $\hat{\beta}_1$ is away from 0.

- $t = \frac{\hat{\beta}_1 - 0}{SE(\hat{\beta}_1)}$

To interpret the $t$-statistic we need to find the probability of observing any values equal or larger to $t$. This probability is know as the p-value. A small $p$-value indicates it is unlikely to observe such a substantial association between the predictor and the response due to chance, without any association between the predictor and the response. Therefore, a small $p$-value indicates that there is an association between the predictor and response. The cut off it typically less than 5% of the number of observations

### 2.1.2 Diagnosing the Model

When performing OLS there are several potential problems due to the assumptions that it makes about the data. Therefore it is also very important to **diagnose the models by looking at the residual**. The residual can often tell us very important information about error in the model. Some common trends to watch out for include:

1. Nonlinearity of the Data

- Assumption: There is a linear relationship between the predictors and the response. If the true relationship is far from linear, then the conclusion that we draw from the model, including prediction accuracy, may be discredited.
- What to look for in the residuals: If we plot the residuals with the predictors it is easy to see non-linearity.

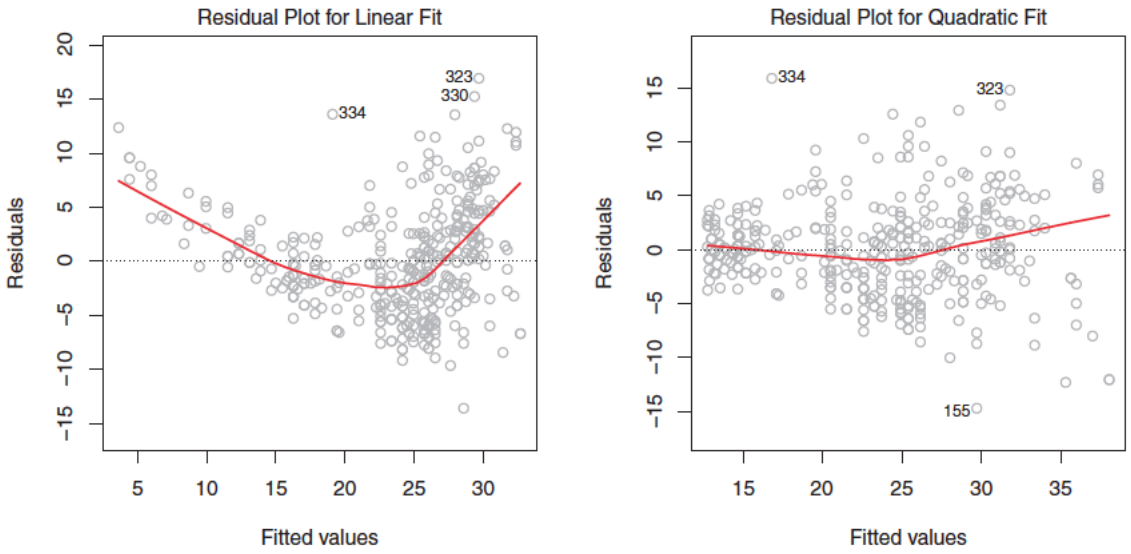



Figure 35: Plot of residuals for a linear and quadratic fit to non-linear data.(Figure 3.9, James, Witten, Hastie, & Harlow, 2013)

From the figure above, the left panel depicts a linear function being used to model a quadratic variation. The **bow shape** of the residual is a clear indication nonlinearity. The right panel depicts a quadratic term in the function being used to model the data. You can see here that the residual do not appear to follow a specific pattern which suggests an improved fit.

2. Correlation of Error Terms

- Assumption: The error terms are uncorrelated (i.e. autocorrelation exists). If the error terms are correlated, estimated standard errors will underestimate the true standard errors. Thus, confidence intervals will be narrower and $p$ values may be lower indicating significant variables when the statistic is truly insignificant.

- What to look for in the residuals: Autocorrelation is present if the residuals show a trend of pattern of clusters above or below the zero value that can be discerned visually. In figure 36, The top panel contain uncorrelated error terms. You can see that there are no obvious patterns from adjacent residuals. The bottom panel represents correlated errors. Now you can see that the adjacent residual seem to take on similar values.

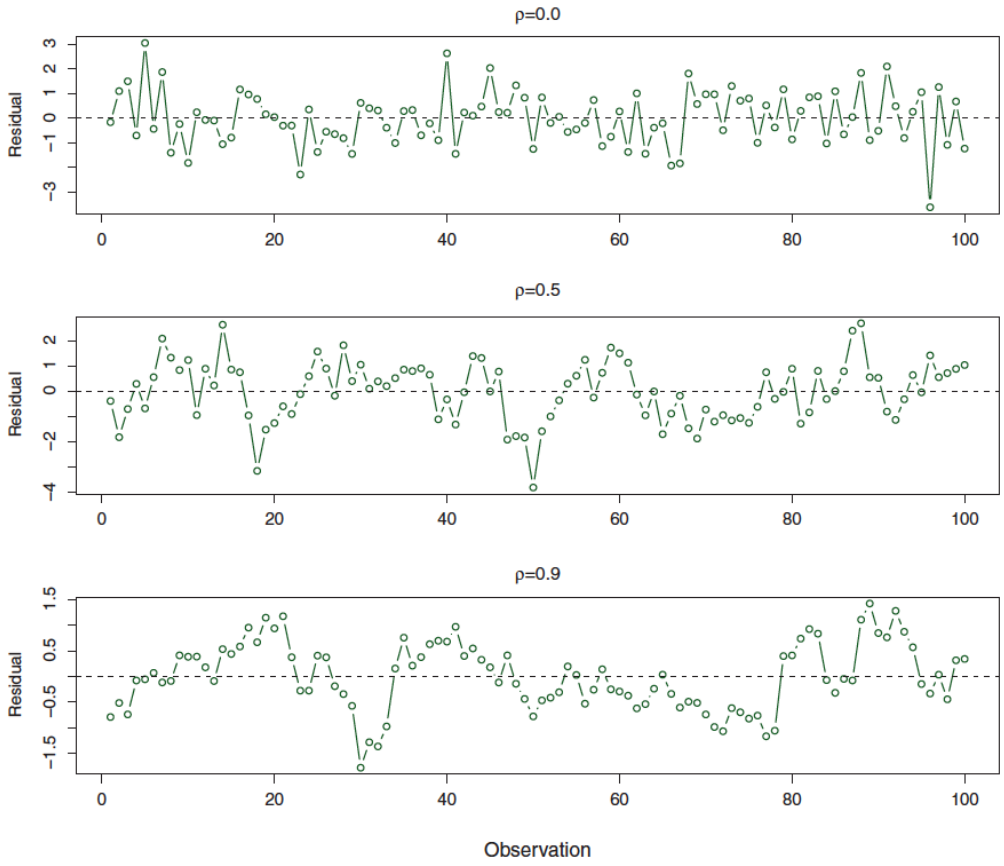



Figure 36: Visualization of correlation of error terms.(Figure 3.10, James, Witten, Hastie, & Harlow, 2013)

3. Non-constant Variance of Error Terms (Heteroscedasticity)

- Assumption: Error terms have a constant variance. Standard error, confidence intervals, and hypothesis testing all rely on this assumption.
- What to look for in the residuals: Heteroscedasticity can be indicated in the residual plot by a funnel shape. This issue can be resolved by transforming the response $Y$ using a concave function such a $\log Y$ or $\sqrt{Y}$.

4. Outliers

- An outlier is a point that is far from the value predicted by the model. You can see in Figure 38 that the outlier is uncharacteristically far from the rest of the residuals. If the outlier is not influential to the regression results, it can simply be omitted. A further discussion on how to identify outliers in the residual plots is provided later in this section.

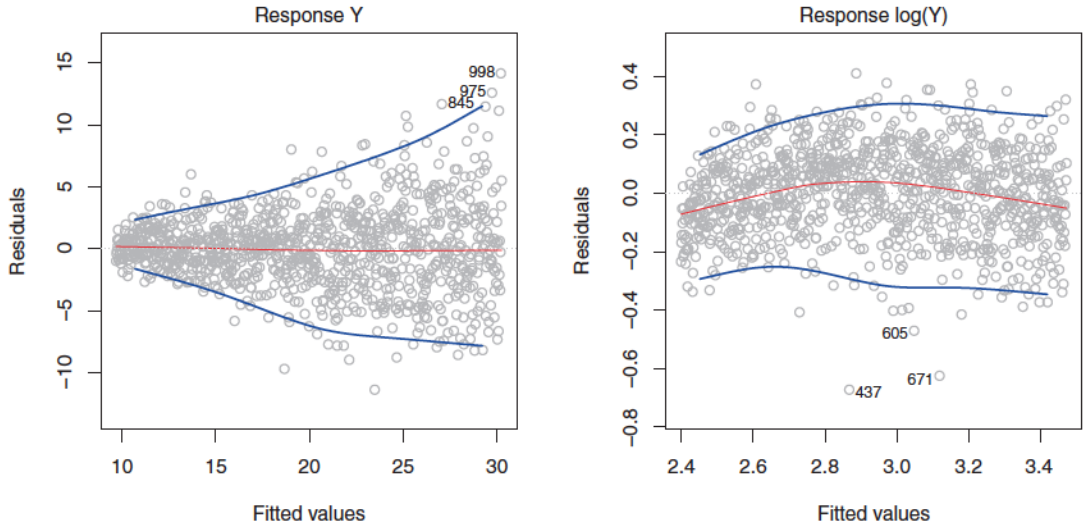



Figure 37: Checking for heteroscedasticity in the residuals. Heteroscedasticity is indicated by the funnel shape (left). By performing a log transformation you can see that heteroscedasticity has diminished (right).(Figure 3.11, James, Witten, Hastie, & Harlow, 2013)

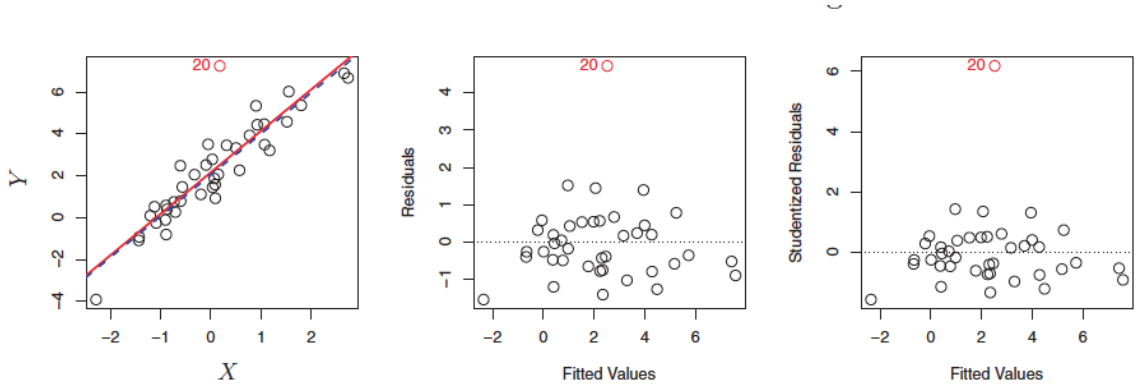



Figure 38: Visualization of outliers in the data.(Figure 3.12, James, Witten, Hastie, & Harlow, 2013)

### 2.1.3 R Applications

#### Example 1: Simple Linear Regression

Example 5.4.1 - Part load performance of fans (and pumps):

Part-load performance curves do not follow the idealized fan laws due to various irreversible losses. For example, decreasing the flow rate by half of the rated flow does not result in a 7/8th decrease in its rated power consumption. Hence, actual tests are performed for such equipment under different levels of loading. The performance tests of the flow rate and the power consumed are then normalized by the rated or 100% load conditions called part load ratio (PLR) and fractional full-load power (FFLP) respectively. Polynomial models can then be fit between these two quantities. The following data was obtained from laboratory tests on a variable speed drive (VSD) control which is a very energy efficient control option.

#### Visual Analysis

First, lets look at the data

```
PLR <- c(0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0)
FFLP <- c(0.05,0.11,0.19,0.28,0.39,0.51,0.68,0.84,1.0)
plot(FFLP ~ PLR)
```

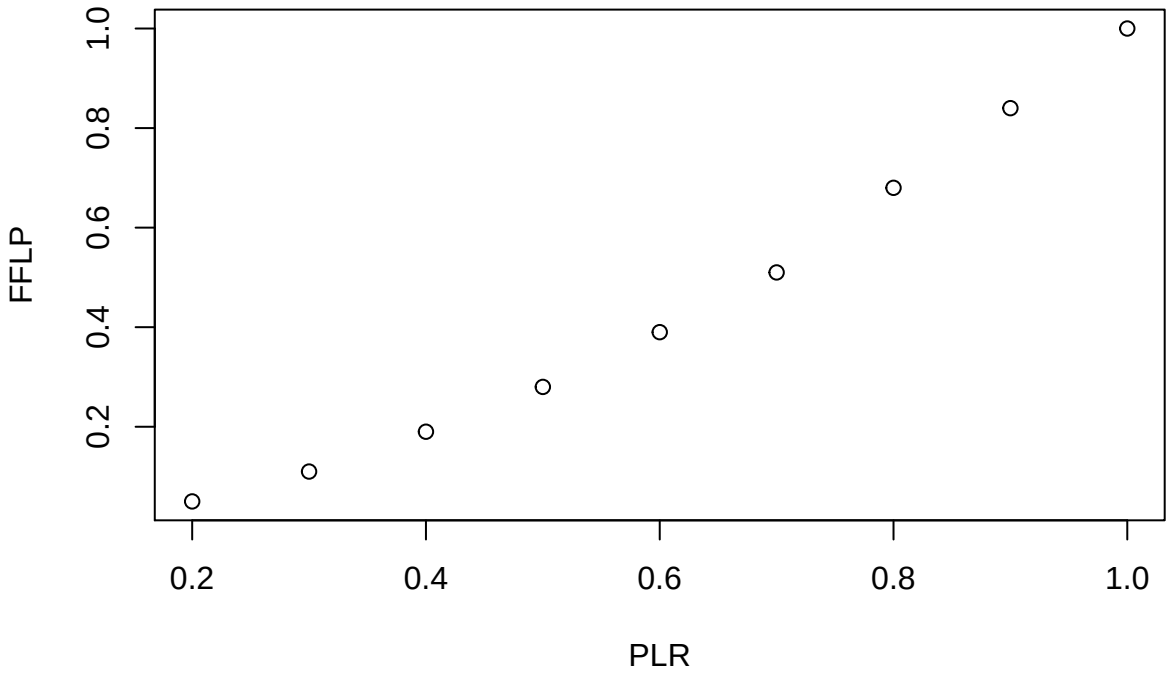



Figure 39: PLR and FFLP of VSD fan controls.

After looking at the data, we can already start to make some guesses as to which type of model might fit this data best. It looks like a quadratic function might be a good fit. Let's try some different models to compare.

#### Linear Model

```
PLR <- c(0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0)
FFLP <- c(0.05,0.11,0.19,0.28,0.39,0.51,0.68,0.84,1.0)
fit.lin=lm(FFLP~PLR)
plot(FFLP ~ PLR)
abline(fit.lin, col='purple')
```

#### Quadratic model

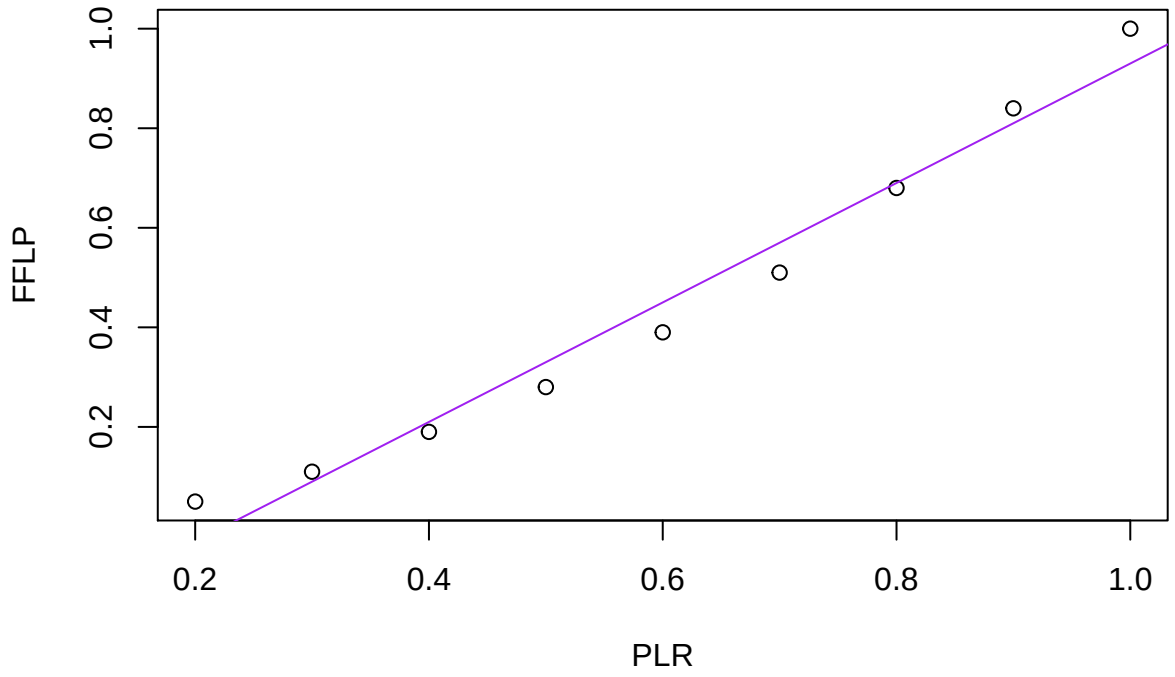



Figure 40: Linear fit to the fan data

```
PLR <- c(0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0)
FFLP <- c(0.05,0.11,0.19,0.28,0.39,0.51,0.68,0.84,1.0)
fit.quad=lm(FFLP~PLR+I(PLR^2))
plot(FFLP~PLR)
lines(PLR, fitted(fit.quad), col='red')
```

**Quadratic model w/o intercept**

```
PLR <- c(0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0)
FFLP <- c(0.05,0.11,0.19,0.28,0.39,0.51,0.68,0.84,1.0)
fit.quad2=lm(FFLP~PLR+I(PLR^2)-1)
plot(FFLP~PLR)
lines(PLR, fitted(fit.quad2), col='blue')
```

**Cubic Model**

```
PLR <- c(0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0)
FFLP <- c(0.05,0.11,0.19,0.28,0.39,0.51,0.68,0.84,1.0)
fit.cube=lm(FFLP~PLR+I(PLR^2)+I(PLR^3))
plot(FFLP~PLR)
lines(PLR, fitted(fit.cube), col='green')
```

From a comparison of all of the tested models, it is easy to see that the linear model is not the best fit. The other three polynomial models all follow the data more closely. Let's look at the quantitative results from each to determine which model is the best selection.

**Quantitative Analysis - Linear Model**

The summary() functions allows us to see the equations of each fitted model and gives us a first look at how accurate our models and estimates are.

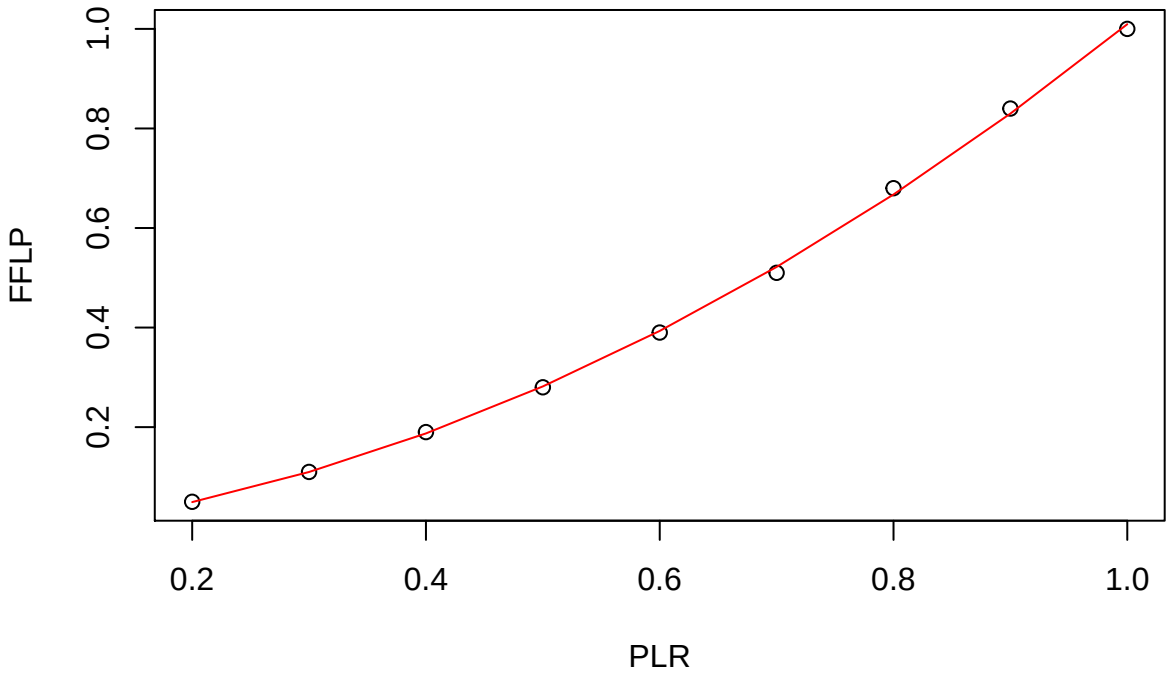



Figure 41: Quadratic fit to the fan data

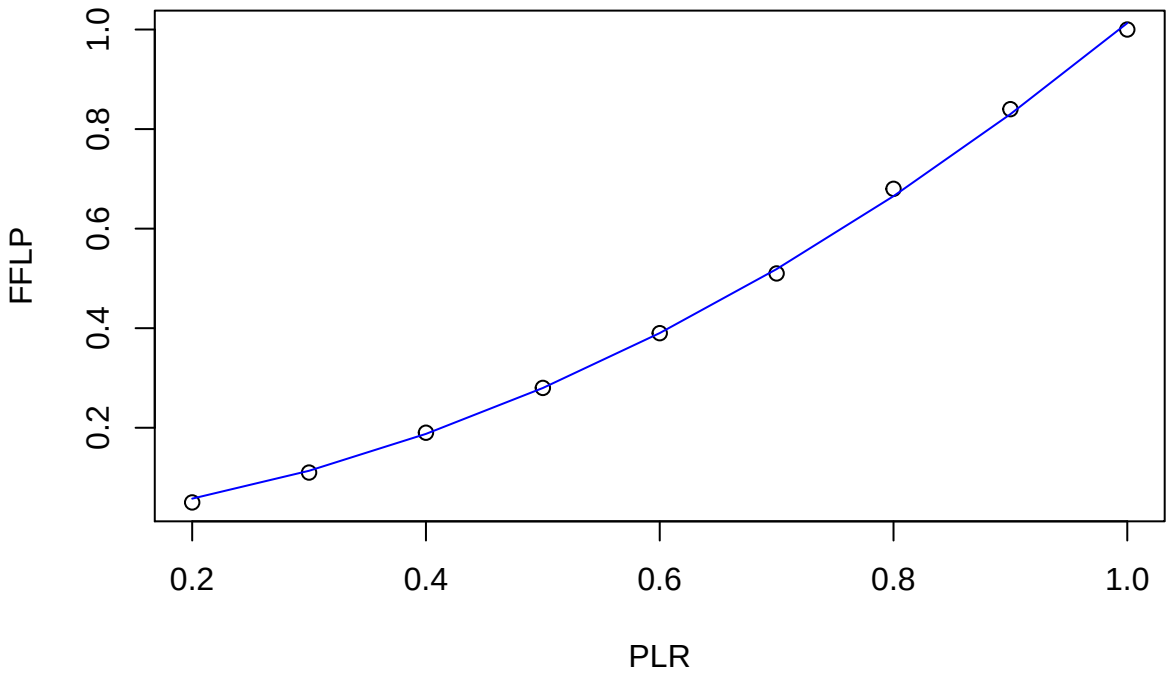



Figure 42: Quadratic without intercept model fit to the fan data

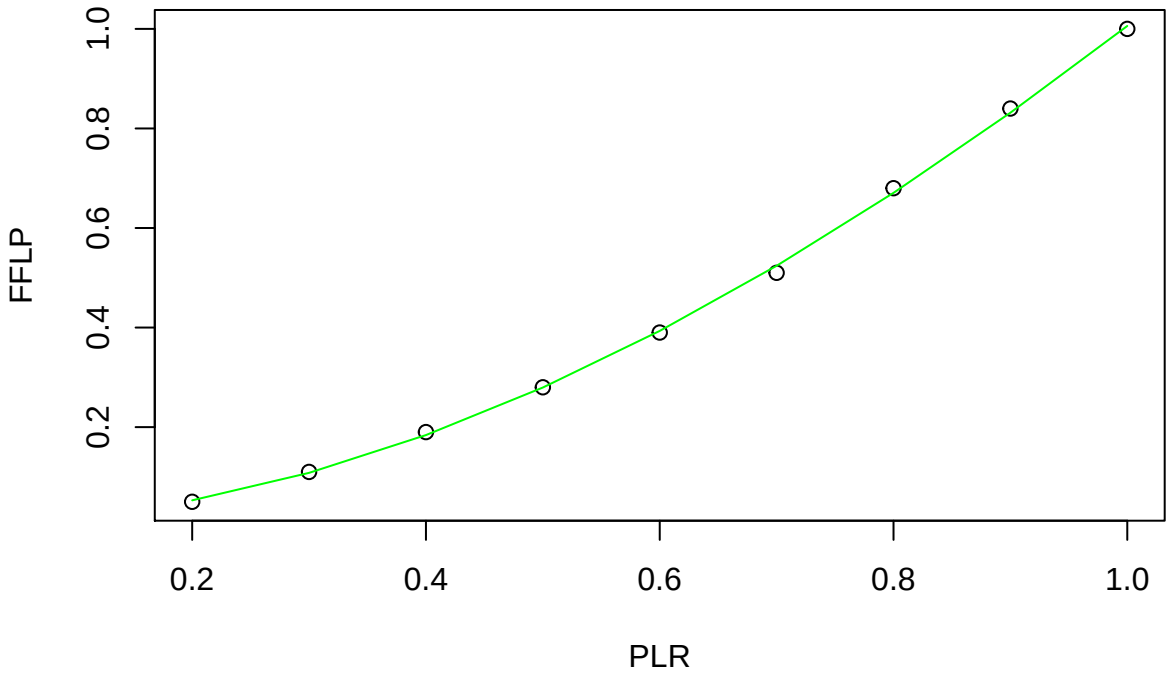



Figure 43: Cubic model fit to the fan data

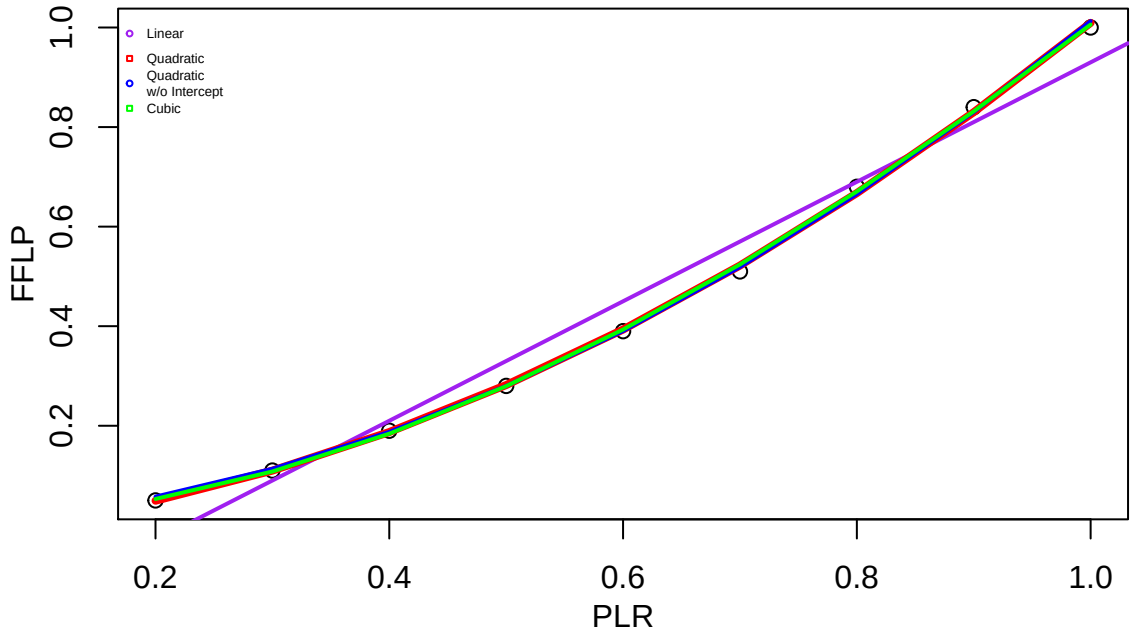



Figure 44: A comparison of all model regressions againts the actual fan data

```
summary(fit.lin)
```

```
##
## Call:
## lm(formula = FFLP ~ PLR)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -0.06  -0.05  -0.01   0.03   0.08
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.27000    0.04813   -5.61 0.000807 ***
## PLR          1.20000    0.07368   16.29 8.01e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05707 on 7 degrees of freedom
## Multiple R-squared:  0.9743, Adjusted R-squared:  0.9706
## F-statistic: 265.3 on 1 and 7 DF,  p-value: 8.011e-07
```

$y = -0.27 + 1.2PLR$

**Quantitative Analysis - Quadratic Model**

```
summary(fit.quad)
```

```
##
## Call:
## lm(formula = FFLP ~ PLR + I(PLR^2))
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0117965 -0.0032900  0.0001515  0.0026840  0.0126840
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.02048    0.01731  -1.183   0.2816
## PLR          0.17922    0.06434   2.785   0.0318 *
## I(PLR^2)     0.85065    0.05269  16.145 3.59e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.009247 on 6 degrees of freedom
## Multiple R-squared:  0.9994, Adjusted R-squared:  0.9992
## F-statistic:  5183 on 2 and 6 DF,  p-value: 1.936e-10
```

$y = -0.02048 + 0.17922PLR + 0.85065PLR^2$

**Quantitative Analysis - Quadratic Model Without a Intercept**

```
summary(fit.quad2)
```

```
##
## Call:
## lm(formula = FFLP ~ PLR + I(PLR^2) - 1)
##
```

```
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0128994 -0.0075733 -0.0002297  0.0023501  0.0146871
##
## Coefficients:
##          Estimate Std. Error t value Pr(>|t|)
## PLR       0.10661    0.01982   5.379  0.00103 **
## I(PLR^2)  0.90629    0.02440  37.143 2.67e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.009507 on 7 degrees of freedom
## Multiple R-squared:  0.9998, Adjusted R-squared:  0.9997
## F-statistic: 1.499e+04 on 2 and 7 DF,  p-value: 1.945e-13
```

$y = 0.10661 + 0.90629PLR^2 - 1$

**Quantitative Analysis - Cubic Model**

```
summary(fit.cube)
```

```
##
## Call:
## lm(formula = FFLP ~ PLR + I(PLR^2) + I(PLR^3))
##
## Residuals:
##          1          2          3          4          5          6
## -0.0029293  0.0019192  0.0059668  0.0004762 -0.0032900 -0.0140693
##          7          8          9
##  0.0094012  0.0083838 -0.0058586
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.01008    0.04057   0.248   0.8137
## PLR         -0.02322    0.25052  -0.093   0.9298
## I(PLR^2)     1.22944    0.45541   2.700   0.0428 *
## I(PLR^3)    -0.21044    0.25122  -0.838   0.4404
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.009485 on 5 degrees of freedom
## Multiple R-squared:  0.9995, Adjusted R-squared:  0.9992
## F-statistic:  3284 on 3 and 5 DF,  p-value: 1.181e-08
```

$y = 0.01008 - 0.02322PLR + 1.22944PLR^2 - 0.21044PLR^3$

**Quantitative Analysis - Conclusions**

- $R^2$: The Linear model has the lowest $R^2$ value of 0.9743 making it the worst fit, which is what we expected. The quadratic model without an intercept has the highest value at 0.9998 making it the best fit in terms of $R^2$.
- **Standard Error**: Again, the linear model has the highest value for standard error at 0.05707. The quadratic model has the lowest residual standard error at 0.009247. However, all of the polynomial models have very close residual standard errors.
- **F-Statistic**: While all of the models have fairly significant regression fits, The quadratic model without an intercept has a F-statistic that much larger than any of the other, indicating that the overall fit of this model is the best.

- **Parameter Significance**: It should be noted that while the quadratic and the quadratic without an intercept models have similar results, the intercept term for the quadratic model in not significant. Thus, the quadratic model without an intercept may be a better fit.

From this analysis it is clear that the linear model is not a good fit. The quadratic model without an intercept appears to be the best.

### Analysis of Variance - Linear Model

We can also see an analysis of the variance of each model using the anova() function:

```
anova(fit.lin)
```

```
## Analysis of Variance Table
##
## Response: FFLP
##           Df Sum Sq Mean Sq F value    Pr(>F)
## PLR        1 0.8640 0.86400  265.26 8.011e-07 ***
## Residuals  7 0.0228 0.00326
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

### Analysis of Variance - Quadratic Model

```
anova(fit.quad)
```

```
## Analysis of Variance Table
##
## Response: FFLP
##           Df  Sum Sq Mean Sq  F value    Pr(>F)
## PLR        1 0.86400 0.86400 10105.52 6.531e-11 ***
## I(PLR^2)   1 0.02229 0.02229   260.67 3.590e-06 ***
## Residuals  6 0.00051 0.00009
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

### Analysis of Variance - Quadratic Without Intercept Model

```
anova(fit.quad2)
```

```
## Analysis of Variance Table
##
## Response: FFLP
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## PLR        1 2.58398 2.58398 28592.2 6.677e-14 ***
## I(PLR^2)   1 0.12468 0.12468  1379.6 2.665e-09 ***
## Residuals  7 0.00063 0.00009
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

### Analysis of Variance - Cubic Model

```
anova(fit.cube)
```

```
## Analysis of Variance Table
##
## Response: FFLP
##           Df  Sum Sq Mean Sq   F value    Pr(>F)
## PLR        1 0.86400 0.86400 9603.0794 2.098e-09 ***
## I(PLR^2)   1 0.02229 0.02229  247.7129 1.883e-05 ***
```

```
## I(PLR^3)   1 0.00006 0.00006    0.7017    0.4404
## Residuals  5 0.00045 0.00009
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

**Analysis of Variance - Conclusion**

All of the models have statistically significant relationships between FFLP and PLR at the 95% confidence level. Again, the quadratic model without an intercept has the lowest $p$ value and therefore the most significant relationships.

**Diagnose the Model: Analysis of Residuals**

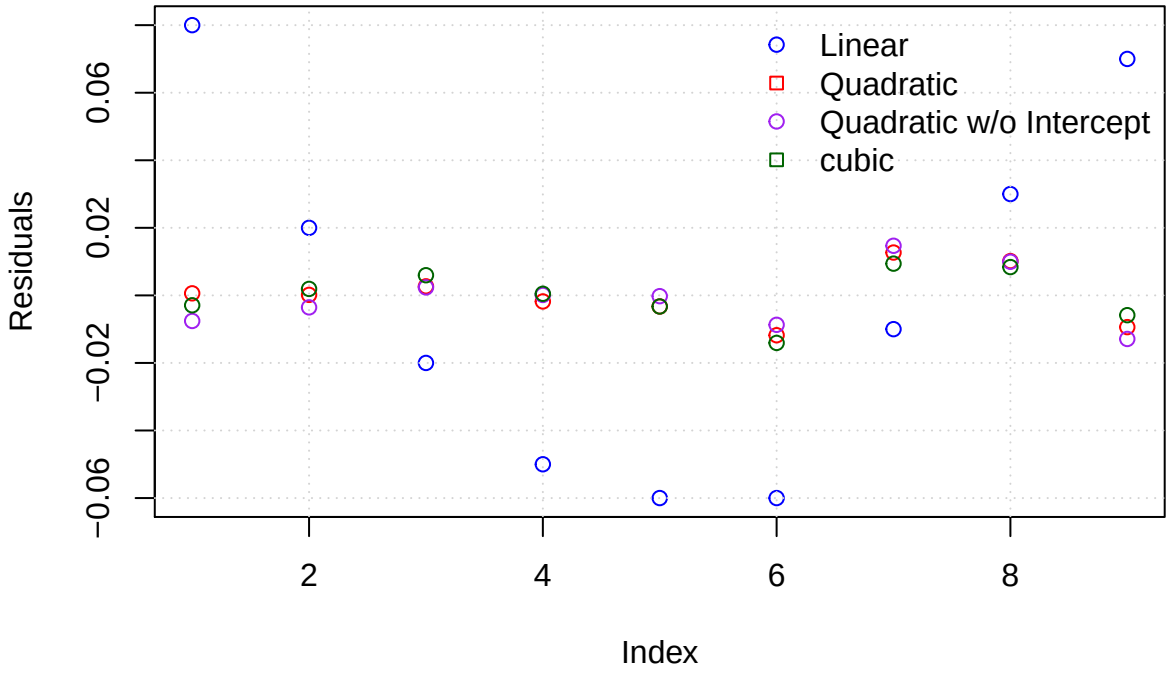



Figure 45: Residual plot of all four models

**Durbin-Watson Test**

Autocorrelation is present if residuals show a trend or a pattern. The most commonly used test for autocorrelation involving model residuals is the Durbin-Watson (DW) Test. If there is no autocorrelation present, the expected value of DW is 2. If the model underfits, DW would be less than 2 and if it over fits DW will be greater than 2 (the limiting range being 0-4). We can do this in R with the dwtest() function in the lmtest package. Lets look at the linear model compared to the quadratic model without an intercept:

```
dwtest(fit.lin)
```

```
##
##  Durbin-Watson test
##
## data:  fit.lin
## DW = 0.52193, p-value = 9.017e-06
## alternative hypothesis: true autocorrelation is greater than 0
```

```
dwtest(fit.quad2)
```

```
##
##  Durbin-Watson test
```

```
##
## data:  fit.quad2
## DW = 1.9286, p-value = 0.2912
## alternative hypothesis: true autocorrelation is greater than 0
```

The quadratic model without an intercept experiences minimal autocorrelation and is a good fit. The linear model is under fit and is not able to explain enough of the variation in the response variable.

**Diagnose the Model: Analysis of Residuals - Linear Model**

We can also create other useful plots in R. Using these plots we can analyze the residuals and check for heteroscedasticity and whether or not the errors are normally distributed with Q-Q plot:

```
par(mfrow=c(2,2))
par(mgp=c(2,1,0))
plot(fit.lin)
```

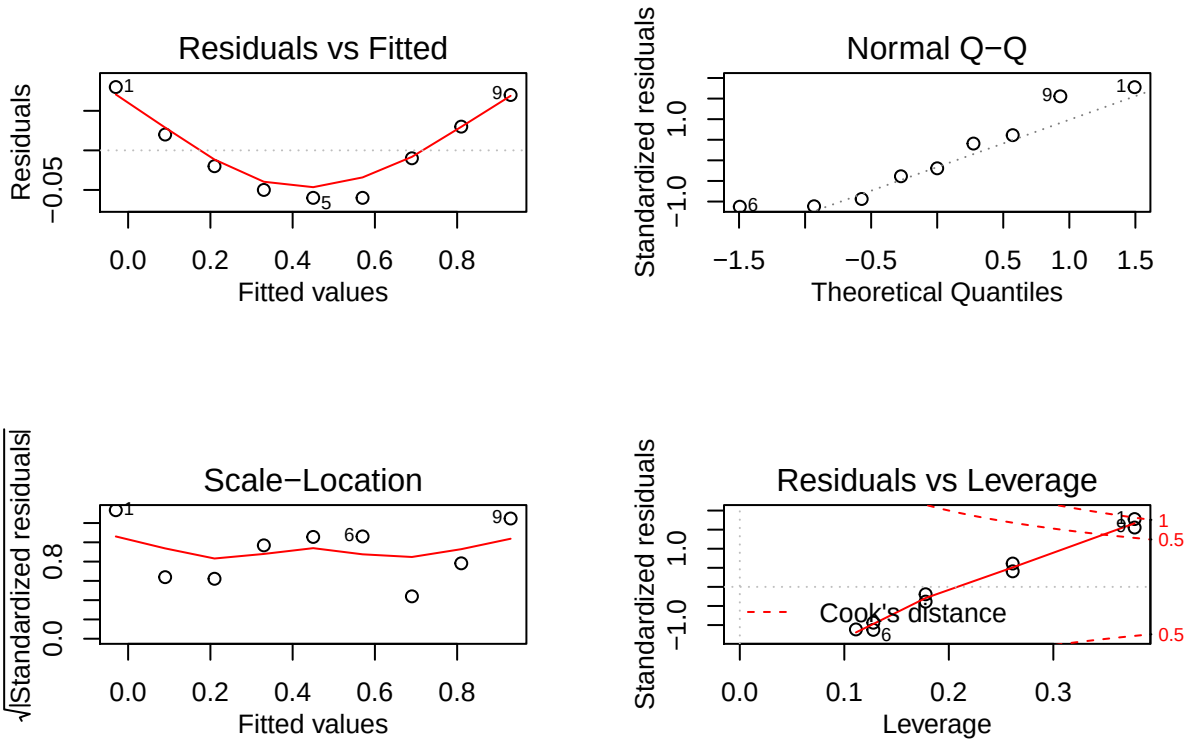



Figure 46: Residual diagnosis of the linear fit

**Diagnose the Model: Analysis of Residuals - Quadratic Without Intercept**

```
par(mfrow=c(2,2))
par(mgp=c(2,1,0))
plot(fit.quad2)
```

**Heteroscedasticity**

The presence of non-constant variance in residuals (heteroscedasticity) is a very common violation. You can see if heteroscedasticity exists by a bow or funnel shape in the residual plots. We are interested in the plot in the top, left hand corner. If there is no heteroscedasticity then we will expect to see residuals distributed randomly and a straight red line.

- Linear model: It is easy to see that there is heteroscedasticity in the residual plot due to the curved red line and the non constant variance.

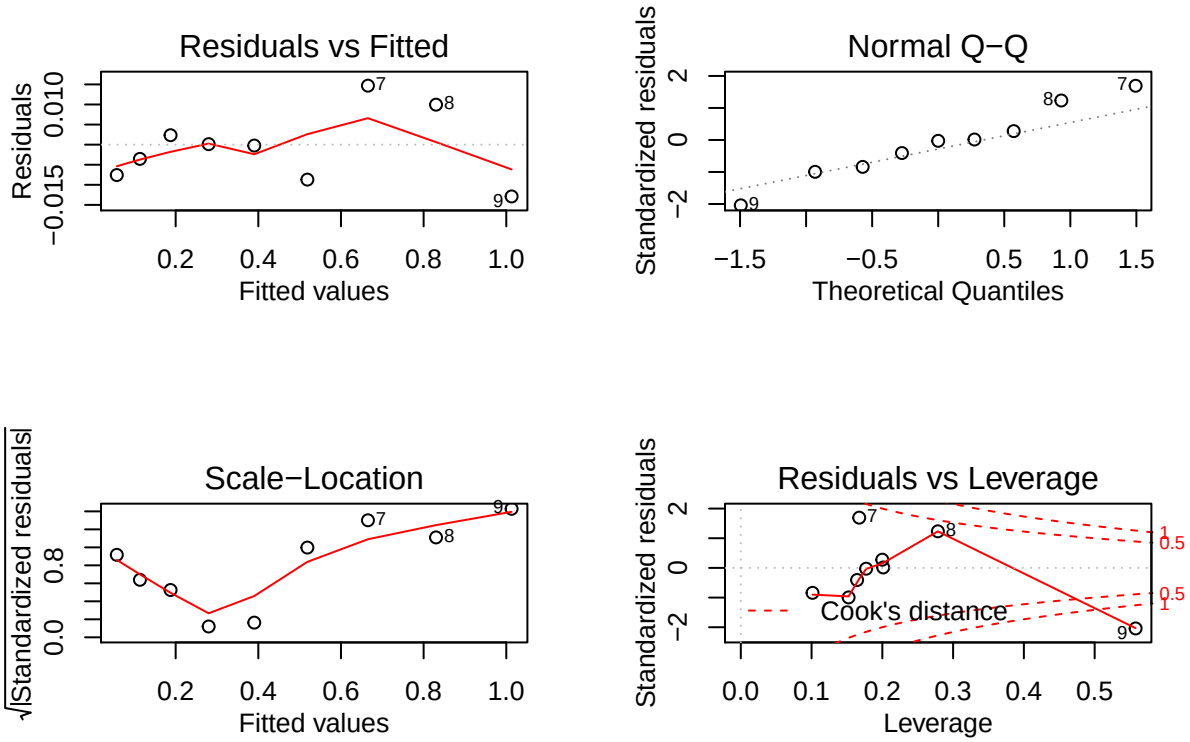



Figure 47: Residual diagnosis of the quadratic fit without intercept

- Quadratic model without intercept: Here you can see that there is no real pattern that the residuals follow but it does seem to have a possible funnel shape. Therefore, heteroscedasticity may exist, but the variance seems to be more constant than that of the linear model

In order to remedy heteroscedasticity, transformations of the response variable, like the log transformation, are often used.

**Q-Q Plot**

Q-Q plots are probability plots which gives a way to graphically compare two probability distributions using quantiles. In OLS, we want our variables to be normally distributed. This is indicated on a Q-Q plot by points that approximately follow a straight line across the diagonal. Below are some examples of different types of Q-Q plots and their meaning.

**Scale-Location Plot**

The scale-location plot can help one determine if homoscedasticity is present, i.e. the assumption that equal variance between the residuals exists. You want the red line to be horizontal with randomly distributed points. In the linear case, it not appear that the points are completely randomly distributed about the red line. This also seem to be the case for the quadratic model without an intercept.

**Residual vs. Leverage Plot***

The residual vs. leverage plot helps to determine whether or not outliers that may be present are influential. In other words, if an outlier is not influential, its presence or lack thereof will not effect the regression results. An outlier in this graph that appears in the upper or lower right corners is an indication that an outlier is influential to a regression and thus should not be excluded from the analysis. In the linear fit, it looks like we are a few influential outliers in the top right-hand corner. In the quadratic without intercept fit, it looks as though there are a few influential outliers in the bottom right-hand corner.

**Model Selection - BIC**

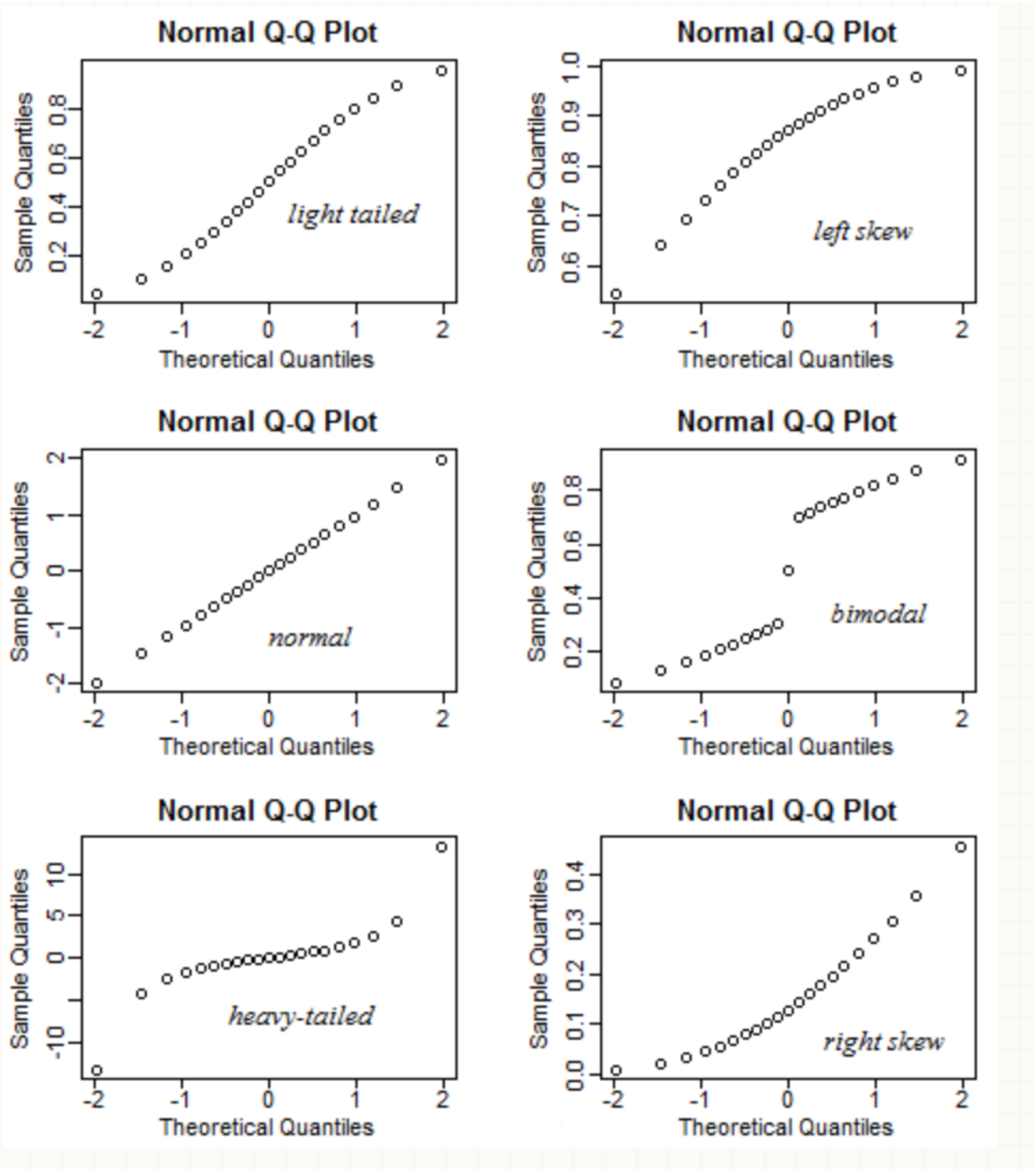



Figure 48: Different Q-Q plot senarios.(https://stats.stackexchange.com/questions/101274/how-to-interpret-a-qq-plot)

We can select a good model using the Bayesian information criterion (BIC). This criterion takes into account both goodness of fit and number of model parameters. The model with the lowest BIC is the best model. We can find this statistic using the AIC() function.The BIC for a model is typically written as:

$2 \log L + kp$

where p is the number of parameters and L is the likelihood function. For BIC, $k = \log(n)$. You can specify the use of BIC by setting $k$ accordingly.

```
AIC(fit.lin, fit.quad, fit.quad2,
    fit.cube,k=log(length(PLR)))
```

```
##           df       AIC
## fit.lin    3 -21.67141
## fit.quad   4 -53.62257
## fit.quad2  3 -53.93326
## fit.cube   5 -52.60726
```

In this case, the **quadratic model without an intercept** has the lowest BIC and is therefore the best model.

### Model Selection - AIC

Akaike information criterion (AIC) is a very similar statistic which is also used for model selection. AIC uses the same function as BIC. One of the main differences however, is the value of $k$. For AIC $k = 2$. Thus AIC differs from BIC in that BIC penalizes more for retaining more parameters.

```
AIC(fit.lin, fit.quad, fit.quad2,
    fit.cube, k=2)
```

```
##           df       AIC
## fit.lin    3 -22.26308
## fit.quad   4 -54.41147
## fit.quad2  3 -54.52494
## fit.cube   5 -53.59338
```

Again, the best model has the smallest AIC. The **quadratic model without an intercept** is also found to be the best fit using this method. Model selection will be discussed in greater detail below.

**Confidence Intervals** It is clear now that the quadratic model without an intercept is the best fit for the data. Now that we have selected our best model, we can determine the confidence intervals:

```
confint(fit.quad2)
```

```
##                2.5 %    97.5 %
## PLR       0.05974664 0.1534698
## I(PLR^2) 0.84859492 0.9639874
```

We can also look at the confidence intervals for the mean response of each estimate:

```
predict(fit.quad2, interval="confidence")
```

```
##          fit        lwr       upr
## 1 0.05757329 0.05041628 0.0647303
## 2 0.11354867 0.10443578 0.1226615
## 3 0.18764987 0.17759846 0.1977013
## 4 0.27987689 0.26979294 0.2899609
## 5 0.39022974 0.38076905 0.3996904
## 6 0.51870842 0.50993519 0.5274816
## 7 0.66531291 0.65611764 0.6745082
## 8 0.83004323 0.81818081 0.8419057
```

```
## 9 1.01289937 0.99610912 1.0296896
```

**Prediction Intervals**

The prediction intervals for each individual prediction can be found using the predict() function with the argument interval"prediction":

```
predict(fit.quad2, interval="prediction")
```

```
##          fit        lwr        upr
## 1 0.05757329 0.03398211 0.08116446
## 2 0.11354867 0.08929242 0.13780491
## 3 0.18764987 0.16302566 0.21227407
## 4 0.27987689 0.25523938 0.30451440
## 5 0.39022974 0.36584070 0.41461878
## 6 0.51870842 0.49457773 0.54283910
## 7 0.66531291 0.64102560 0.68960023
## 8 0.83004323 0.80462596 0.85546050
## 9 1.01289937 0.98484170 1.04095705
```

We can then take the confidence interval and plot the confidence bands:

```
pred.frame <- data.frame(PLR=seq(0,1,0.05))
pc <- predict(fit.quad2, int="c", newdata=pred.frame)
pp <- predict(fit.quad2, int="p", newdata=pred.frame)
require(graphics)
plot(FFLP ~ PLR , ylim=range(PLR , pp, na.rm=T))
pred.PLR <- pred.frame$PLR
matlines(pred.PLR, pc, lty=c(1,2,2), lwd=1.5)
matlines(pred.PLR, pp, lty=c(1,3,3), lwd=1.5)
```

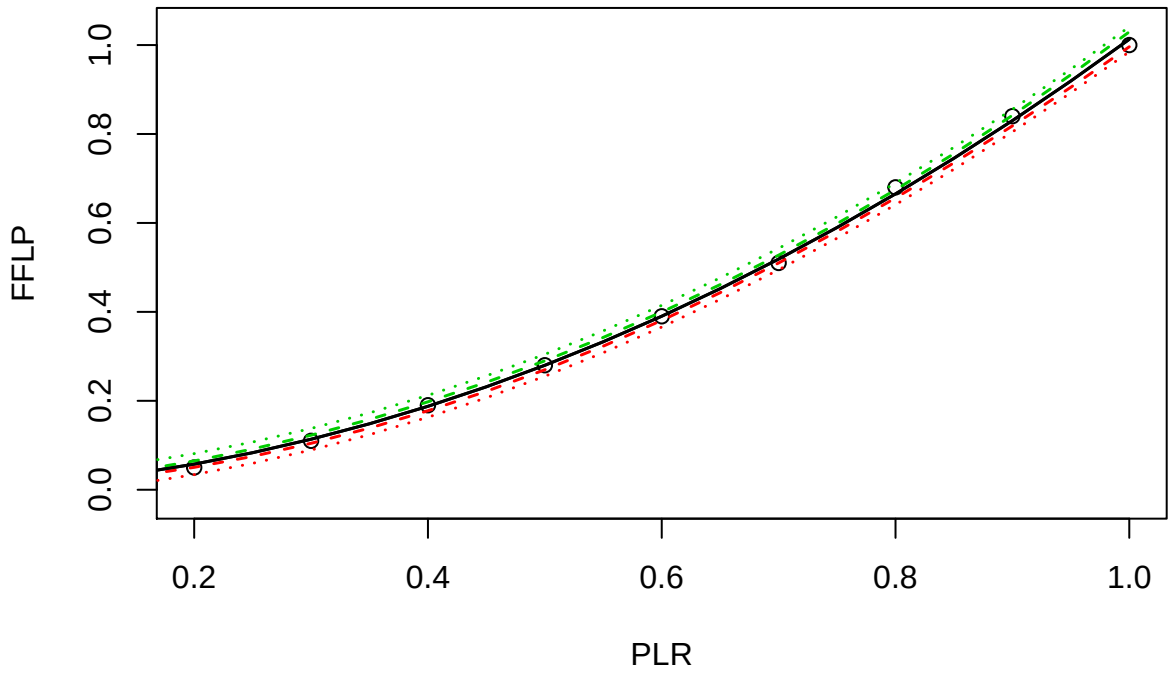



Figure 49: Plot of the quadratic fit without an intercept with confidence bands

**Example 2: Multiple Linear Regression**

Now consider the same data set with 13 parameters. We will look at the NREL RSF (Research Support Facility) data which has 13 variables: time, cooling, mechanical, lighting, plug loads, data center needs, PV generation, whole building loads, whole building net, global irradiance, irradiance on the south facade, and dry bulb temperature, and attempt to predict the heating loads. We will just look at the first 1000 observations.

To regress Heating onto just two other variable, we can write:

```
library(EMBA)
data=rsf2011[1:1000,]
fit=lm(Heating~Lighting+Plug_Loads,data=data)
summary(fit)
```

```
##
## Call:
## lm(formula = Heating ~ Lighting + Plug_Loads, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -407.60 -208.12  -73.88  123.82 1272.84
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 386.3317    34.8298  11.092  < 2e-16 ***
## Lighting     -1.9939     0.7257  -2.747  0.00612 **
## Plug_Loads    0.7427     0.9854   0.754  0.45122
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 300.4 on 997 degrees of freedom
## Multiple R-squared:  0.01012,    Adjusted R-squared:  0.008131
## F-statistic: 5.095 on 2 and 997 DF,  p-value: 0.006288
```

Using a . instead of a parameter name, we can consider Heating against all of the parameters in the data frame:

```
fit1=lm(Heating~.,data=data)
summary(fit1)
```

```
##
## Call:
## lm(formula = Heating ~ ., data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7653 -0.1791 -0.0026  0.1830  2.4095
##
## Coefficients: (1 not defined because of singularities)
##                            Estimate Std. Error  t value Pr(>|t|)
## (Intercept)              5.960e+01  5.346e+01    1.115   0.2651
## Time                    -4.925e-08  4.190e-08   -1.175   0.2402
## Cooling                 -9.253e-01  2.685e-02  -34.461   <2e-16 ***
## Mechanical              -1.005e+00  7.889e-03 -127.402   <2e-16 ***
## Lighting                -1.001e+00  3.935e-03 -254.384   <2e-16 ***
## Plug_Loads              -9.808e-01  8.273e-03 -118.552   <2e-16 ***
## Data_Center             -9.450e-01  3.054e-02  -30.946   <2e-16 ***
```

```
## PV_Generation            4.594e-04  5.154e-04    0.891   0.3730
## Whole_Building_Load      1.000e+00  2.281e-04 4383.121   <2e-16 ***
## Whole_Building_Net              NA         NA       NA       NA
## Dry_Bulb_Temp           -6.801e-03  6.989e-03   -0.973   0.3308
## Irradiance_Global       -1.430e-03  8.323e-04   -1.718   0.0861 .
## Irradiance_South_Facade  1.718e-04  4.214e-04    0.408   0.6836
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.204 on 988 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 5.699e+06 on 11 and 988 DF,  p-value: < 2.2e-16
```

From the above summary, you can see that using all of the parameters has increased the $R^2$ value and deceased the residual standard error. It is also apparent that Time and Cooling are not significant.

We are able to plot the residuals and the Q-Q plot to additionally analyze the model fit.

```
par(mfrow=c(2,2))
plot(fit1)
```

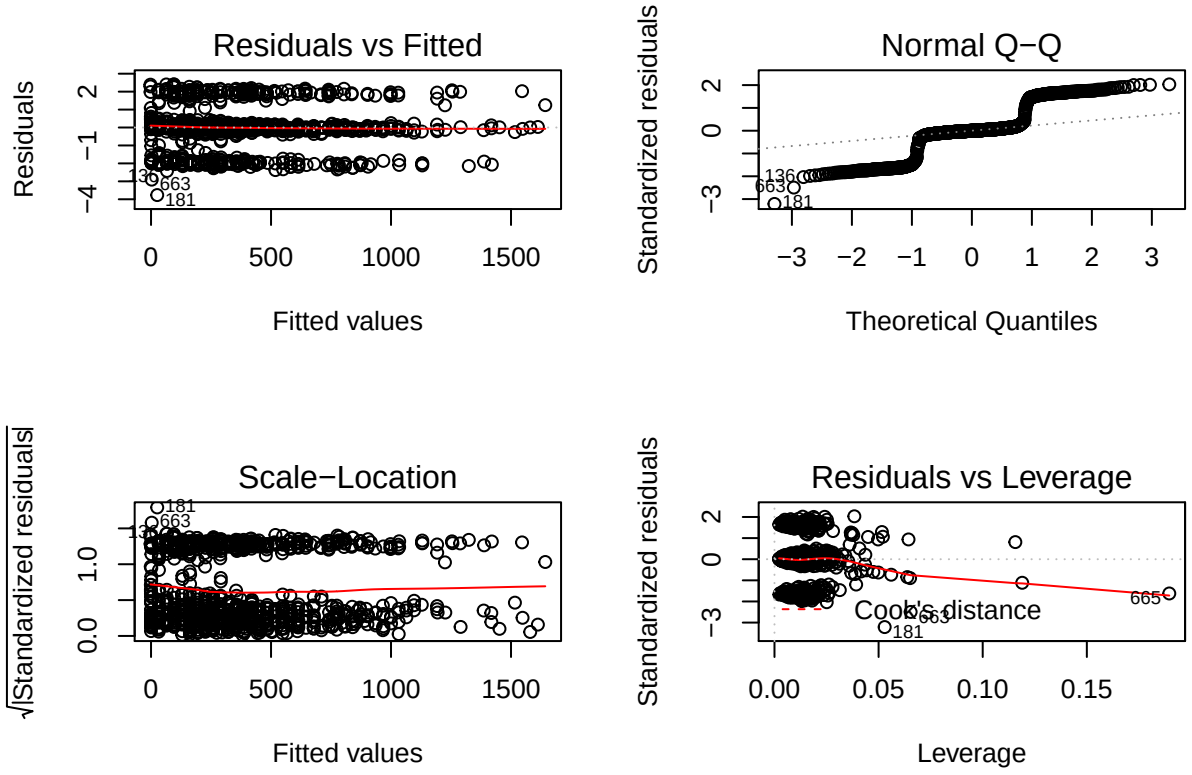
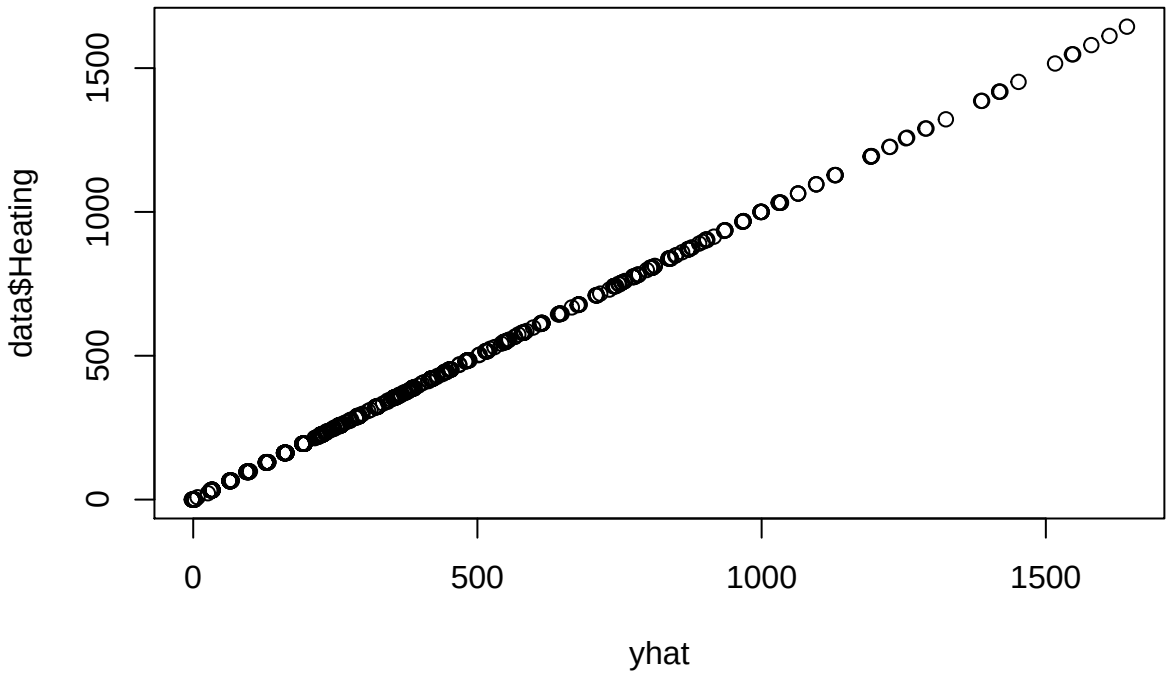



Figure 50: Residual diagnosis of the linear with all parameters excluding time and cooling

Despite unusual residual plots (possibly caused by discrete data), we can see from the $R^2$ value and the parity plot below, that this model is nearly a perfect fit to the data.

```
yhat=predict(fit1)
plot(yhat, data$Heating)
```




**Interaction Terms** It is easy to include interaction terms in a linear model using the lm() function:

- The syntax Plug_Loads:Lighting tells R to include an interaction term between Lighting and Plug_Loads
- The syntax Plug_Loads∗Lighting simultaneously includes lstat, age and the interaction term Plug_Loads∗Lighting as predictors.
- The syntax Plug_Loads+Lighting+Plug_Loads∗Lighting is the long form for Lighting∗Plug_Loads

### 2.1.4 Mitigating Heteroscedasticity

Adapted from https://datascienceplus.com/how-to-detect-heteroscedasticity-and-rectify-it/.

It is common practice to check for heteroscedasticity, or non-constant variance of residuals, once you build the linear regression model. If heteroscedasticity exists, the model may be unable to explain some patterns in the response variable $Y$.

For this example we will use the mtcars data set from the from the data sets package in R. This data set has 50 observations and 2 variables, speed and stopping distance.

Let's first build a linear model using the lm() function. We will preform a linear regression of distance onto speed:

```
fit.lm <- lm(dist~speed, data=cars)
```

Now that the model is ready there are two ways to test for heteroscedasticity, lets look at the residuals:

```
par(mfrow=c(2,2))
plot(fit.lm)
```

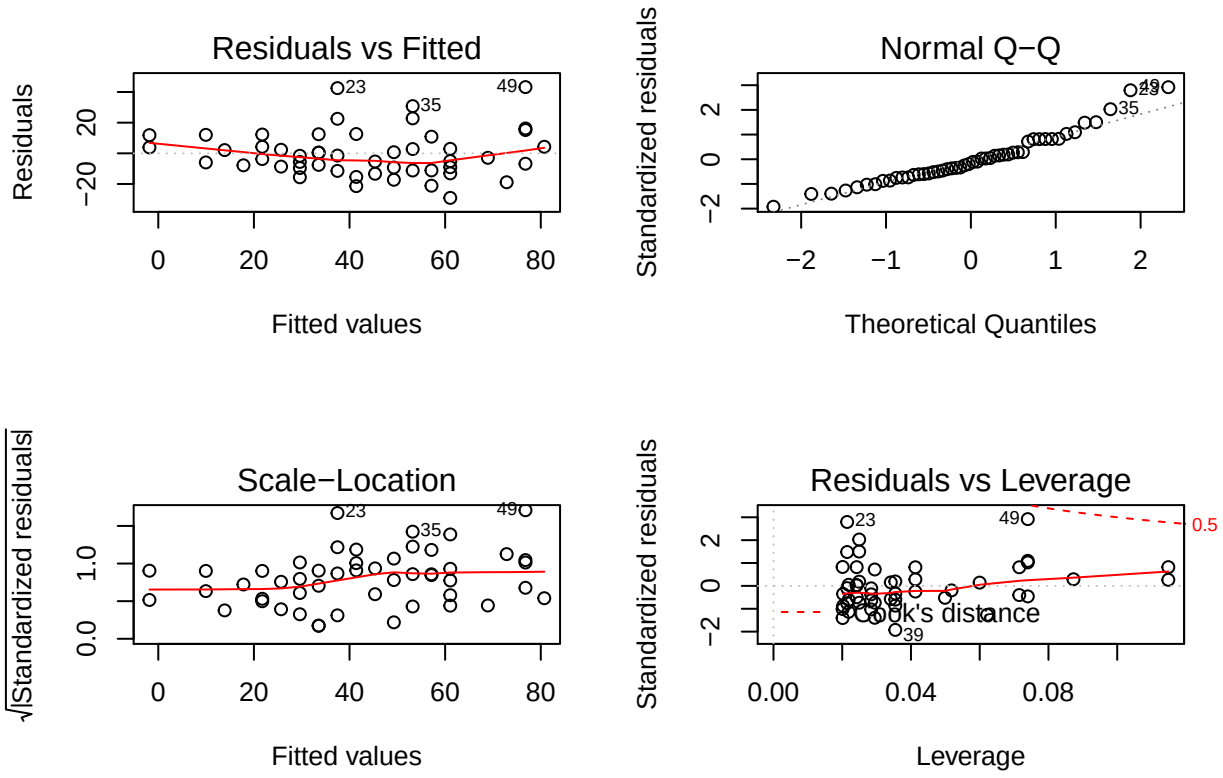



Figure 51: Residual diagnosis of the linear fit to cars data

We are mostly interested in the top-left and bottom-left plots. If there is no heteroscedasticity, you should see a completely random, equal distribution of points throughout the range of $x$-axis and a flat red line.

In this case you can see from the top-left plot that the red line is slightly curved and the residuals tend to increase with the fitted Y values. Thus, there is heteroscedasticity.

**Testing for Heteroscedasticity**

The **Breusch Pagan Test** tests for heteroscedasticity using the following method:

1. We have $y = \beta_0 + \beta_1 x_1 + ... + \beta_k x_k + \epsilon$ and the model is estimated using OLS.
2. The estimated error, or $\hat{\epsilon}$, is then used in an auxiliary regression on the predictor variables.
   - $\hat{\epsilon}^2 = \delta_0 + \delta_1 x_1 + ... + \delta_k x_k$
   - If any of the coefficients do not equal 0, then heteroscedasticity exists. i.e. the residuals depend on the the predictors variables.
3. Find $R^2 n$ where $n$ is the number of points in the sample and $R^2$ is the $R^2$ value of the auxiliary regression.
   - If $R^2 n$ is high, the variation in residual can be explained by the predictor variables, which is indicative of heteroscedasticity.

```
lmtest::bptest(fit.lm)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  fit.lm
## BP = 3.2149, df = 1, p-value = 0.07297
```

The null hypothesis is that homoscedasticity, i.e. the absence of heteroscedasticity:

- $H_0 : \delta_1 = \delta_2 = ... = \delta_k = 0$
- $H_1 : \delta_i \neq 0$

As we know $p$ values evaluate the significance of a hypothesis test. Low $p$ values, e.g. less than 0.05, indicate that the null hypothesis can be rejected. Here the $p$ value is 0.07297, i.e., fairly low and thus we can reject the null hypothesis and suspect that heteroscedasticity is present.

**Rectifying Heteroscedasticity**

Heteroscedasticity can be fixed using data transformations. Here we will use a Box Cox transformation of the response variable instead of the variable itself. The Box Cox transformation is defined by:

$Y(\lambda) = \begin{cases} (Y^\lambda - 1)/\lambda & \text{if } \lambda \neq 0 \\ \log Y & \text{if } \lambda = 0 \end{cases}$ - where $Y$ is the response, and $\lambda$ is the transformation parameter, $-5 \leq \lambda \leq 5$.

All values of $\lambda$ are considered and the optimal value for the data is selected. (The "optimal value" is the one which results in the best approximation of a normal distribution curve.) This value is chosen using MLE or Bayesian estimation methods.

- First, we create a Box Cox transformed response variable:

```
distBCmod <- caret::BoxCoxTrans(cars$dist)
print(distBCmod)
```

```
## Box-Cox Transformation
##
## 50 data points used to estimate Lambda
##
## Input data summary:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00   26.00   36.00   42.98   56.00  120.00
##
## Largest/Smallest: 60
## Sample Skewness: 0.759
##
## Estimated Lambda: 0.5
```

- We now can apply it on cars$dist and append it to a new data frame

```
cars<-cbind(cars,dist_new=predict(distBCmod, cars$dist))
head(cars)
```

```
##   speed dist  dist_new
## 1     4    2 0.8284271
## 2     4   10 4.3245553
## 3     7    4 2.0000000
## 4     7   22 7.3808315
## 5     8   16 6.0000000
## 6     9   10 4.3245553
```

- The transformed data for our new regression model is ready. Let's build the model and check for heteroscedasticity.

```
fit.lm_bc <- lm(dist_new~speed, data=cars)
bptest(fit.lm_bc)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  fit.lm_bc
## BP = 0.011192, df = 1, p-value = 0.9157
```

- With a $p$-value of 0.91, we fail to reject the null hypothesis of homoscedasticity and therefore can assume there is no heteroscedasticity. In fact, we now have a straight red line in the top left plot and can infer that there is no heteroscedasticity.

```
par(mfrow=c(2,2))
plot(fit.lm_bc)
```

Graphically you can see that there is a much flatter line and the residuals are much more evenly distributed.

### 2.1.5 Model Selection

The model selection techniques discussed in this section aim to reduce the number of parameters in the model. There are two important factors that make reducing the number of variables beneficial when building a model.

- Prediction accuracy: If the number of observations is much larger then the the number of variables, then the OLS estimates will have a low variance and small test prediction error. If the number of observations is only slight larger than the number of parameters, then the opposite will be true. Therefore, by reducing the number of parameters we are able to reduce the variance.
- Model interpretability: It is not uncommon for some of the parameters in a multidimensional data set to not be related to the response. Including these variables in the model can lead to a unnecessarily complex model. Thus, by eliminating irrelevant variables, we can help with the interpretability of the model.

#### 2.1.5.1 Stepwise Model Selection

The following is an example of forward stepwise selection with AIC and BIC. Forward stepwise selection starts with a model containing no predictors and then adds them one at a time until all of the predictors are in the model. At each step, the parameter that improves the model is selected.

This test will help decide how many parameters are optimal to use in the model. Again, we will do this using the rsf2011 data from the EMBA package to try to predict the heating load using the first 1000 observations.

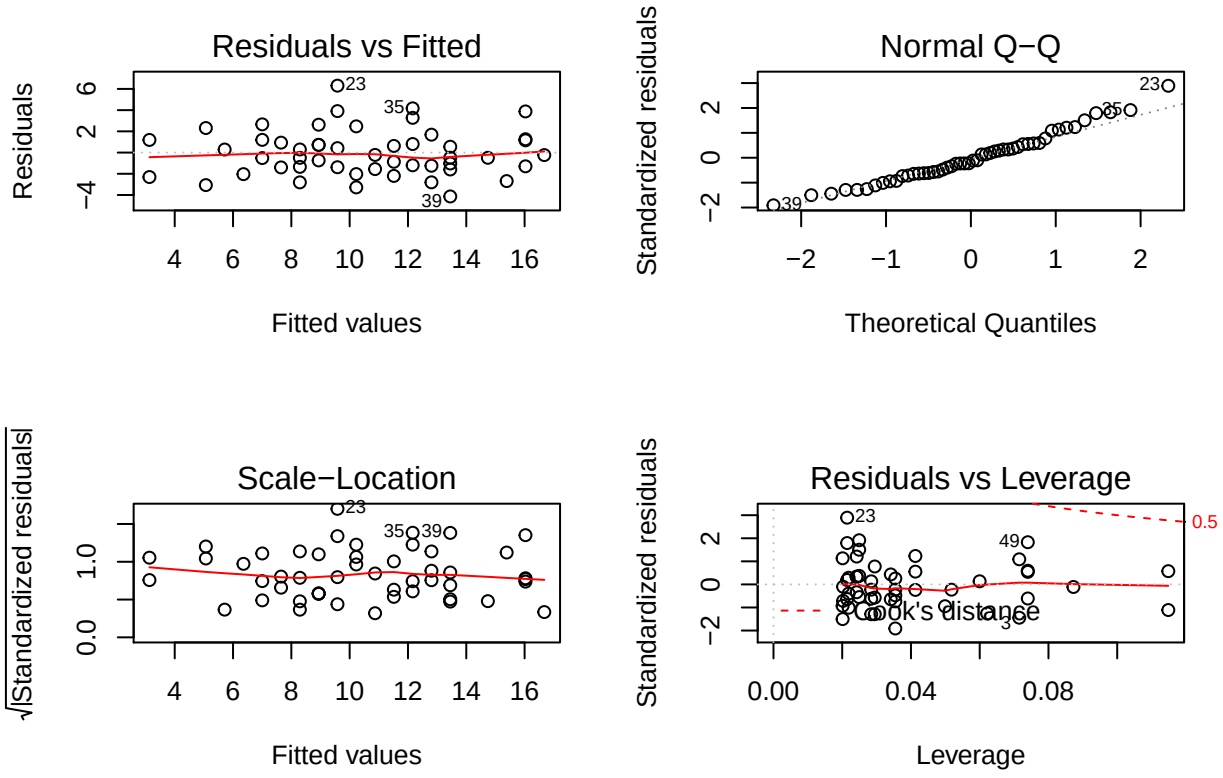



Figure 52: Residual diagnosis of the linear fit to cars data after box cox tranformation has been preformed on the response.

```
library(EMBA)
summary(rsf2011)
```

```
##       Time                         Cooling          Heating
##  Min.   :2011-01-01 01:00:00   Min.   : 0.000   Min.   :   0.00
##  1st Qu.:2011-04-02 06:30:00   1st Qu.: 0.000   1st Qu.:   0.00
##  Median :2011-07-02 12:00:00   Median : 0.000   Median :   0.00
##  Mean   :2011-07-02 12:00:00   Mean   : 3.077   Mean   :  86.74
##  3rd Qu.:2011-10-01 17:30:00   3rd Qu.: 2.000   3rd Qu.: 118.00
##  Max.   :2011-12-31 23:00:00   Max.   :42.000   Max.   :1644.00
##    Mechanical       Lighting        Plug_Loads     Data_Center
##  Min.   : 0.00   Min.   :  0.00   Min.   : 0.00   Min.   : 92.0
##  1st Qu.: 4.00   1st Qu.:  6.00   1st Qu.:32.00   1st Qu.:100.0
##  Median : 6.00   Median : 12.00   Median :36.00   Median :102.0
##  Mean   :14.33   Mean   : 20.32   Mean   :41.53   Mean   :102.5
##  3rd Qu.:28.00   3rd Qu.: 34.00   3rd Qu.:50.00   3rd Qu.:106.0
##  Max.   :56.00   Max.   :116.00   Max.   :98.00   Max.   :122.0
##  PV_Generation    Whole_Building_Load Whole_Building_Net Dry_Bulb_Temp
##  Min.   :  -6.0   Min.   :  94.0      Min.   :-798.00    Min.   :-26.52
##  1st Qu.:   0.0   1st Qu.: 154.0      1st Qu.:  15.87    1st Qu.:  3.56
##  Median :   0.0   Median : 226.0      Median : 150.00    Median : 10.62
##  Mean   : 136.1   Mean   : 267.8      Mean   : 131.71    Mean   : 10.93
##  3rd Qu.: 230.0   3rd Qu.: 298.0      3rd Qu.: 250.00    3rd Qu.: 19.11
##  Max.   :1142.0   Max.   :1852.0      Max.   :1852.00    Max.   : 34.22
##  Irradiance_Global  Irradiance_South_Facade
```

```
##  Min.   :   0.000   Min.   :   0.000
##  1st Qu.:   4.935   1st Qu.:   0.000
##  Median :  14.979   Median :   4.694
##  Mean   : 201.288   Mean   : 163.903
##  3rd Qu.: 357.039   3rd Qu.: 245.847
##  Max.   :1060.963   Max.   :1260.548
```

There are several ways that we can preform model selection. Let's first use the regsubsets() function in the leaps package to select the best model. We can use forward and backward stepwise selection by writing method="forward"or "backward". In this example we will use forward stepwise selection.

```
library(leaps)
data=rsf2011[1:1000,]
regfit.fwd=regsubsets(Heating~., data=data, nvmax=12,method="forward")
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 1 linear dependencies found
```

```
## Reordering variables and trying again:
```

```
reg.summary=summary(regfit.fwd)
summary(regfit.fwd)
```

```
## Subset selection object
## Call: regsubsets.formula(Heating ~ ., data = data, nvmax = 12, method = "forward")
## 12 Variables  (and intercept)
##                          Forced in Forced out
## Time                         FALSE      FALSE
## Cooling                      FALSE      FALSE
## Mechanical                   FALSE      FALSE
## Lighting                     FALSE      FALSE
## Plug_Loads                   FALSE      FALSE
## Data_Center                  FALSE      FALSE
## PV_Generation                FALSE      FALSE
## Whole_Building_Load          FALSE      FALSE
## Dry_Bulb_Temp                FALSE      FALSE
## Irradiance_Global            FALSE      FALSE
## Irradiance_South_Facade      FALSE      FALSE
## Whole_Building_Net           FALSE      FALSE
## 1 subsets of each size up to 11
## Selection Algorithm: forward
##           Time Cooling Mechanical Lighting Plug_Loads Data_Center
## 1  ( 1 )  " "  " "     " "        " "      " "        " "
## 2  ( 1 )  " "  " "     "*"        " "      " "        " "
## 3  ( 1 )  " "  " "     "*"        "*"      " "        " "
## 4  ( 1 )  " "  " "     "*"        "*"      "*"        " "
## 5  ( 1 )  " "  "*"     "*"        "*"      "*"        " "
## 6  ( 1 )  " "  "*"     "*"        "*"      "*"        "*"
## 7  ( 1 )  " "  "*"     "*"        "*"      "*"        "*"
## 8  ( 1 )  "*"  "*"     "*"        "*"      "*"        "*"
## 9  ( 1 )  "*"  "*"     "*"        "*"      "*"        "*"
## 10  ( 1 ) "*"  "*"     "*"        "*"      "*"        "*"
## 11  ( 1 ) "*"  "*"     "*"        "*"      "*"        "*"
##           PV_Generation Whole_Building_Load Whole_Building_Net
## 1  ( 1 )  " "           "*"                 " "
## 2  ( 1 )  " "           "*"                 " "
```

```
## 3  ( 1 )  " "           "*"                 " "
## 4  ( 1 )  " "           "*"                 " "
## 5  ( 1 )  " "           "*"                 " "
## 6  ( 1 )  " "           "*"                 " "
## 7  ( 1 )  " "           "*"                 " "
## 8  ( 1 )  " "           "*"                 " "
## 9  ( 1 )  " "           "*"                 " "
## 10  ( 1 ) "*"           "*"                 " "
## 11  ( 1 ) "*"           "*"                 " "
##           Dry_Bulb_Temp Irradiance_Global Irradiance_South_Facade
## 1  ( 1 )  " "           " "               " "
## 2  ( 1 )  " "           " "               " "
## 3  ( 1 )  " "           " "               " "
## 4  ( 1 )  " "           " "               " "
## 5  ( 1 )  " "           " "               " "
## 6  ( 1 )  " "           " "               " "
## 7  ( 1 )  " "           "*"               " "
## 8  ( 1 )  " "           "*"               " "
## 9  ( 1 )  "*"           "*"               " "
## 10  ( 1 ) "*"           "*"               " "
## 11  ( 1 ) "*"           "*"               "*"
```

The output shows you how the the model was built, indicated by the number of "*" for each variable.

We now find the best model using BIC. First we can look at the summary of how the BIC changes as more variables are added. You can see that the BIC goes from -3908.943 with one variable to -10975.220 with all 11 variables. We want to the lowest BIC possible. In this example we can see that the lowest BIC occurs when there are 6 parameters. We can also find this using the which.min() function. Additionally, the lowest BIC can be found by plotting all the BICs for the data. Here we have marked the lowest point using the points() function.

```
reg.summary$bic
```

```
##  [1]  -3908.943  -6210.136  -7774.703  -9701.188 -10285.048 -11000.807
##  [7] -10999.479 -10994.399 -10988.091 -10981.960 -10975.220
```

```
which.min(reg.summary$bic)
```

```
## [1] 6
```

```
plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type='l')
points(6,reg.summary$bic[6], col="red", cex=2, pch=20)
```

We can also use this R function to look at the $R^2$ (r2) values, and the AIC statistics, to help choose the best model.

To see the best models regression equation using forward stepwise selection and BIC we can get the equation coefficients using the coef() function and 6 parameters:

```
coef(regfit.fwd, 6)
```

```
##         (Intercept)             Cooling          Mechanical
##          -3.2867102          -0.9185423          -1.0095068
##            Lighting          Plug_Loads         Data_Center
##          -0.9952297          -0.9922048          -0.9521892
## Whole_Building_Load
##           1.0002548
```

We can also achieve similar results using the stepAIC() function from the MASS package to preform model

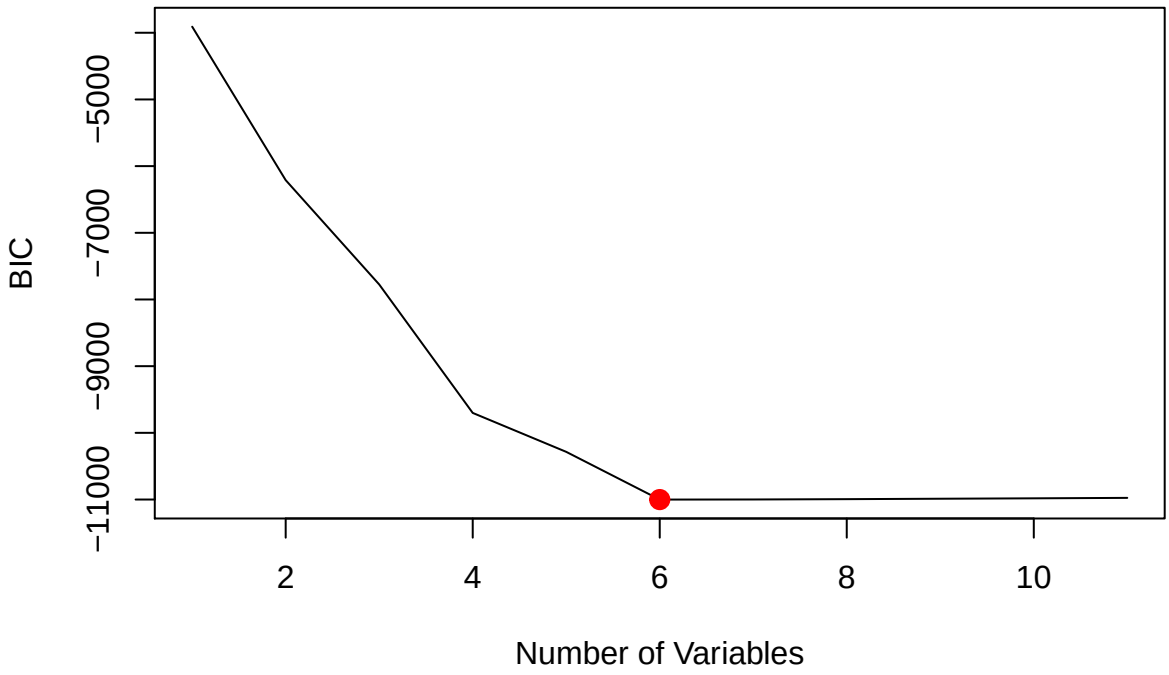



Figure 53: Different values of BIC for different nnumbers of variables in the model. The red dot located at 6 represents the best model.

selection. as before,BIC and AIC can be specified by the value of $k$. For comparison, we will use stepwise selection with BIC.

```
library(MASS)
fit.lm=lm(Heating~., data=data)
k1=log(19)
final.mod <- stepAIC(fit.lm, k=k1)
```

```
## Start:  AIC=394.85
## Heating ~ Time + Cooling + Mechanical + Lighting + Plug_Loads +
##     Data_Center + PV_Generation + Whole_Building_Load + Whole_Building_Net +
##     Dry_Bulb_Temp + Irradiance_Global + Irradiance_South_Facade
##
##
## Step:  AIC=394.85
## Heating ~ Time + Cooling + Mechanical + Lighting + Plug_Loads +
##     Data_Center + PV_Generation + Whole_Building_Load + Dry_Bulb_Temp +
##     Irradiance_Global + Irradiance_South_Facade
##
##                            Df Sum of Sq      RSS     AIC
## - Irradiance_South_Facade  1         0     1433   392.1
## - PV_Generation            1         1     1434   392.7
## - Dry_Bulb_Temp            1         1     1434   392.9
## - Time                     1         2     1435   393.3
## <none>                                     1433   394.8
## - Irradiance_Global        1         4     1437   394.9
## - Data_Center              1      1389     2821  1069.6
## - Cooling                  1      1722     3155  1181.3
## - Plug_Loads               1     20380    21812  3114.9
```

```
## - Mechanical               1     23536    24969  3250.0
## - Lighting                 1     93834    95266  4589.1
## - Whole_Building_Load      1  27857707 27859140 10267.3
##
## Step:  AIC=392.07
## Heating ~ Time + Cooling + Mechanical + Lighting + Plug_Loads +
##     Data_Center + PV_Generation + Whole_Building_Load + Dry_Bulb_Temp +
##     Irradiance_Global
##
##                       Df Sum of Sq      RSS     AIC
## - PV_Generation        1         1     1434   389.9
## - Dry_Bulb_Temp        1         1     1434   390.1
## - Time                 1         2     1435   390.8
## <none>                             1433   392.1
## - Irradiance_Global    1         9     1442   395.7
## - Data_Center          1      1393     2825  1068.1
## - Cooling              1      1757     3190  1189.5
## - Plug_Loads           1     20595    22028  3121.7
## - Mechanical           1     23695    25128  3253.4
## - Lighting             1     95756    97189  4606.1
## - Whole_Building_Load  1  27875208 27876641 10265.0
##
## Step:  AIC=389.9
## Heating ~ Time + Cooling + Mechanical + Lighting + Plug_Loads +
##     Data_Center + Whole_Building_Load + Dry_Bulb_Temp + Irradiance_Global
##
##                       Df Sum of Sq      RSS     AIC
## - Dry_Bulb_Temp        1         1     1435   387.6
## - Time                 1         3     1437   389.0
## <none>                             1434   389.9
## - Irradiance_Global    1         9     1443   393.3
## - Data_Center          1      1391     2825  1065.2
## - Cooling              1      1757     3191  1186.7
## - Plug_Loads           1     20794    22228  3127.8
## - Mechanical           1     23868    25302  3257.4
## - Lighting             1     96964    98398  4615.5
## - Whole_Building_Load  1  28277961 28279395 10276.4
##
## Step:  AIC=387.56
## Heating ~ Time + Cooling + Mechanical + Lighting + Plug_Loads +
##     Data_Center + Whole_Building_Load + Irradiance_Global
##
##                       Df Sum of Sq      RSS     AIC
## - Time                 1         3     1437   386.4
## <none>                             1435   387.6
## - Irradiance_Global    1         9     1444   390.7
## - Data_Center          1      1395     2830  1063.9
## - Cooling              1      1758     3193  1184.4
## - Plug_Loads           1     21286    22721  3146.8
## - Mechanical           1     26945    28380  3369.2
## - Lighting             1     98821   100256  4631.3
## - Whole_Building_Load  1  60285665 60287100 11030.4
##
## Step:  AIC=386.44
```

```
## Heating ~ Cooling + Mechanical + Lighting + Plug_Loads + Data_Center +
##     Whole_Building_Load + Irradiance_Global
##
##                        Df Sum of Sq      RSS     AIC
## <none>                                  1437   386.4
## - Irradiance_Global    1         8     1446   389.1
## - Data_Center          1      1539     2977  1111.4
## - Cooling              1      1772     3209  1186.6
## - Plug_Loads           1     22569    24006  3198.9
## - Mechanical           1     27266    28704  3377.6
## - Lighting             1     99461   100898  4634.7
## - Whole_Building_Load  1  61921705 61923143 11054.3
```

```
final.mod
```

```
##
## Call:
## lm(formula = Heating ~ Cooling + Mechanical + Lighting + Plug_Loads +
##     Data_Center + Whole_Building_Load + Irradiance_Global, data = data)
##
## Coefficients:
##         (Intercept)              Cooling           Mechanical
##           -3.1464287           -0.9179850           -1.0061615
##             Lighting           Plug_Loads          Data_Center
##           -1.0000556           -0.9825921           -0.9553679
## Whole_Building_Load    Irradiance_Global
##            1.0001044           -0.0008547
```

This function gave a slightly different model. Here, using seven (7) parameters results in the lowest BIC and therefore the best model.

#### 2.1.5.2 The Dredge Function

Lastly, the dredge() function from the MuMIn package can be used to choose the best model. Unlike the the stepwise selection method, dredge finds all possible combinations of a model, or $2^p$ different models where p is the number of paramters. It then ranks them using the AIC statistic.

Using the same data from above we can use dredge as follows. Note that the dredge function does not allow for any NAs in any of the data. Therefore, a new data set needs to be created which contains no NA values.

```
library(MuMIn)
```

```
##
## Attaching package: 'MuMIn'
```

```
## The following object is masked from 'package:pls':
##
##     stdize
```

```
## The following object is masked from 'package:ExtDist':
##
##     AICc
```

```
data=rsf2011[1:1000,]
options(na.action = "na.fail")
data2 <- na.omit(data)
lm.fit  <- lm(Heating~., data=data2)
d2 <- dredge(lm.fit)
```

```
## Fixed term is "(Intercept)"
```

```
sel.table<-as.data.frame(d2[1:5])
sel.table
```

```
##      (Intercept)    Cooling Data_Center Dry_Bulb_Temp Irradiance_Global
## 1260   -3.146429 -0.9179850  -0.9553679            NA     -0.0008546514
## 1772   65.733297 -0.9228105  -0.9443716            NA     -0.0008936435
## 2540   -3.236444 -0.9195901  -0.9549488            NA     -0.0010705567
## 1516   -3.236444 -0.9195901  -0.9549488            NA     -0.0010705567
## 3564   -3.236444 -0.9195901  -0.9549488            NA     -0.0010705567
##      Irradiance_South_Facade  Lighting Mechanical Plug_Loads PV_Generation
## 1260                      NA -1.000056  -1.006162 -0.9825921            NA
## 1772                      NA -1.000496  -1.007342 -0.9801085            NA
## 2540                      NA -1.000354  -1.006370 -0.9814024  1.0005481565
## 1516                      NA -1.000354  -1.006370 -0.9814024  0.0004252873
## 3564                      NA -1.000354  -1.006370 -0.9814024  0.0004252873
##               Time Whole_Building_Load Whole_Building_Net df    logLik
## 1260            NA            1.000104                 NA  9 -1600.382
## 1772 -5.409424e-08            1.000138                 NA 10 -1599.468
## 2540            NA                  NA           1.000123 10 -1600.001
## 1516            NA            1.000123                 NA 10 -1600.001
## 3564            NA            1.000123                 NA 10 -1600.001
##          AICc     delta    weight
## 1260 3218.946 0.0000000 0.2871582
## 1772 3219.159 0.2129179 0.2581587
## 2540 3220.224 1.2780902 0.1515610
## 1516 3220.224 1.2780902 0.1515610
## 3564 3220.224 1.2780902 0.1515610
```

The output lists all of the possible combinations and ranks them by their AICs. Since there are 12 parameters, dredge calculates the AIC for 4096 different models! For convienence, here we have only printed the top 5 models. You can see which parameters are used for each model, thier coefficient estimates, and different model statistics such as the AIC.

## 2.2 Piecewise Models

Piecewise or spline functions are a very important class of function. They allow for a range of locally different behavior to be explained within a single framework. It does this by allowing non-linear functions to be decomposed into simpler, more local patterns. The point at which the local pattern changes to another is called a change point, $x_c$.

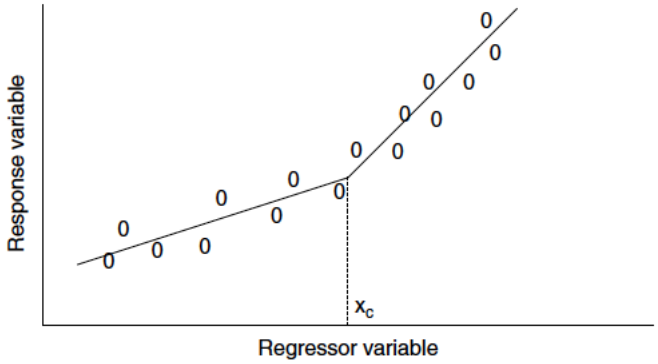



**Fig. 5.25** Piece-wise linear model or first-order spline fit with hinge point at $x_c$. Such models are referred to as *change point models* in building energy modeling terminology

There are two cases:

**Case 1)** The change point in known and the model will have the following form:

- $y = \beta_0 + \beta_1 x + \beta_2 (x - x_c) I$

where the indicator I:

$$I = \begin{cases} 1 & \text{if } x > x_c \\ 0 & \text{otherwise} \end{cases}$$

Hence, for the region $x \leq x_c$, the model is

- $y = \beta_0 + \beta_1 x$, where the slope is $\beta_1$

and for the region $x > x_c$

- $y = (\beta_0 - \beta_2 x_c) + (\beta_1 + \beta_2) x$, where the slope is $\beta_1 + \beta_2$

**Case 2)** The change point in unknown. The approach in this case is to estimate the change point by looking at the data, and then perform a number of fits until the best fit is identified.

**Example 5.7.1**

Electricity utility bills of a residence in Houston, TX have been normalized by the number of days in the month and assembled in the following table. The data is presented along with the corresponding month and monthly mean outdoor temperature values for Houston.

| Month | Mean Outdoor Temp (C) | Monthly Mean Electric Use ($kWh/mˆ2/day) | $x$ (C) |
|---|---|---|---|
| Jan | 11 | 0.1669 | 11 |
| Feb | 13 | 0.1866 | 13 |
| Mar | 16 | 0.1988 | 16 |

| Month | Mean Outdoor Temp (C) | Monthly Mean Electric Use ($kWh/m^2/day) | $x$ (C) |
|---|---|---|---|
| Apr | 21 | 0.2575 | 21 |
| May | 24 | 0.3152 | 24 |
| Jun | 27 | 0.3518 | 27 |
| Jul | 29 | 0.3898 | 29 |
| Aug | 29 | 0.3872 | 29 |
| Sept | 26 | 0.3315 | 26 |
| Oct | 22 | 0.2789 | 22 |
| Nov | 16 | 0.2051 | 16 |
| Dec | 13 | 0.1790 | 13 |

Here the change point is unknown but from visual inspection we assume it to be in the range of $17 - 19°C$ so we will assume the change point of $x_c = 17°C$, the indicator variable is defined as:

$$I = \begin{cases} 1 & \text{if } x > 17C \\ 0 & \text{otherwise} \end{cases}$$

We can then add another column to our data table:

| Month | Mean Outdoor Temp (C) | Monthly Mean Electric Use ($kWh/m^2/day) | $x$ (C) | I (C) |
|---|---|---|---|---|
| Jan | 11 | 0.1669 | 11 | 0 |
| Feb | 13 | 0.1866 | 13 | 0 |
| Mar | 16 | 0.1988 | 16 | 0 |
| Apr | 21 | 0.2575 | 21 | 4 |
| May | 24 | 0.3152 | 24 | 7 |
| Jun | 27 | 0.3518 | 27 | 10 |
| Jul | 29 | 0.3898 | 29 | 12 |
| Aug | 29 | 0.3872 | 29 | 12 |
| Sept | 26 | 0.3315 | 26 | 9 |
| Oct | 22 | 0.2789 | 22 | 5 |
| Nov | 16 | 0.2051 | 16 | 0 |
| Dec | 13 | 0.1790 | 13 | 0 |

Let's try this problem in R. First we need to import the data set. There are a variety of file types that can be imported into R including txt, csv, xlsx files. Here we will load in the data using a csv file.

```
library(SiZer)
```

```
## Loading required package: splines
```

```
## Loading required package: boot
```

```
mydata=read.csv("mydata.csv")
data=as.data.frame(mydata)
data
```

```
##     X V1     V2 V3
## 1   1 11 0.1669  0
## 2   2 13 0.1866  0
## 3   3 16 0.1988  0
## 4   4 21 0.2575  1
## 5   5 24 0.3152  1
```

```
## 6   6 27 0.3518  1
## 7   7 29 0.3898  1
## 8   8 29 0.3872  1
## 9   9 26 0.3315  1
## 10 10 22 0.2789  1
## 11 11 16 0.2051  0
## 12 12 13 0.1790  0
```

Where $X$ is the month, $V1$ is the mean outdoor temperature, $V2$ is the monthly mean daily electric use, and $V3$ is the indicator variable indicating whether the temperature is above (1) or below (0) 17°C.

Let's start by first assessing our estimate of the change point, $x_c$. By looking at the data we can see that a change point of $x_c = 17$°C is reasonable.

```
plot( V2 ~ V1, cex=0.5, data=data, xlab="Monthly Outdoor Temp (C)",
      ylab="Monthly Mean Electric Use (kWh/m^2/day)")
```

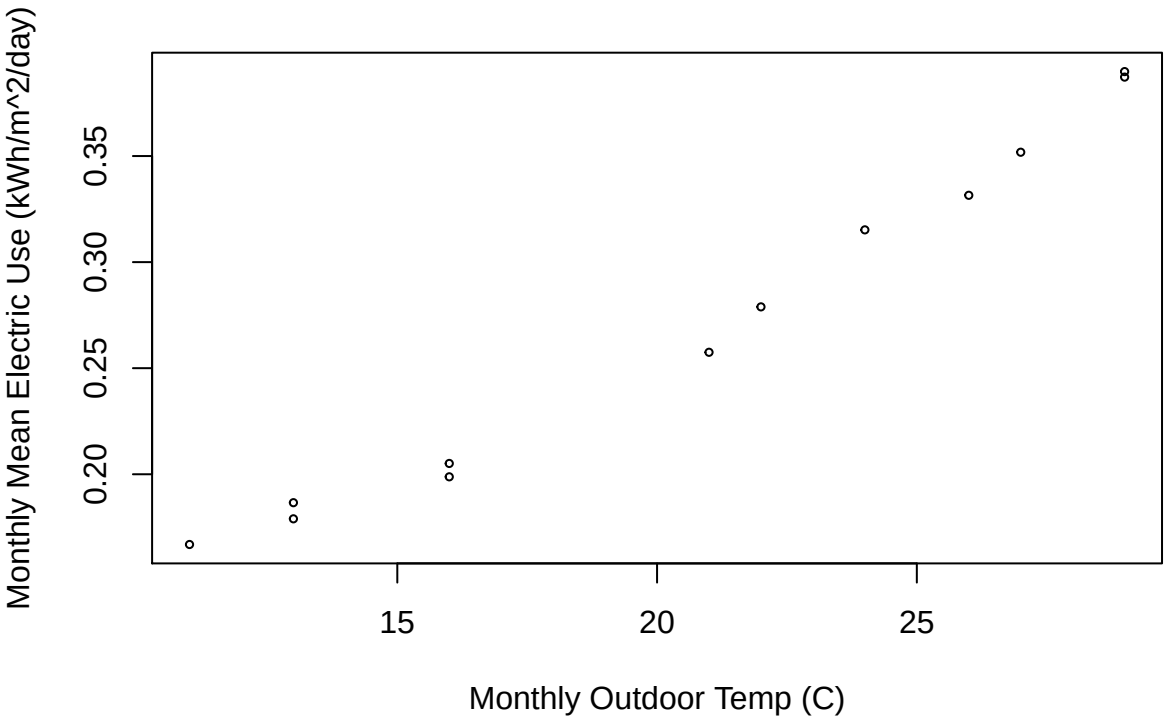



Figure 54: Mean outdoor temperature vs. monthly mean temperature

With this approach we can use the lm() function. Using the equation:

- $y = \beta_0 + \beta_1(V1) + \beta_2(V1 - CP)I$

The only unknown parameter is (V1-CP)I. To find this in R, let (V1-CP)I = V4. We can then find the linear regression of the monthly mean daily electric use (V2) onto the too parameters; the mean outdoor temperature (V1) and V4.

```
CP=17
V4 = (data$V1-CP)*data$V3
lmind = lm( V2 ~ V1+V4, data=data)
```

A linear multiple regression yields:

- $y = 0.1046 + 0.005904x + 0.00905(x - 17)I$

With $R^2$=0.996 and RMSE=0.0055

```
summary(lmind)
```

```
##
## Call:
## lm(formula = V2 ~ V1 + V4, data = data)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0081208 -0.0027602 -0.0006085  0.0051967  0.0059787
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.104828   0.015556   6.739 8.47e-05 ***
## V1          0.005893   0.001081   5.449 0.000406 ***
## V4          0.009063   0.001437   6.306 0.000140 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.005478 on 9 degrees of freedom
## Multiple R-squared:  0.9964, Adjusted R-squared:  0.9957
## F-statistic:  1262 on 2 and 9 DF,  p-value: 9.512e-12
```

Recall that when $x \leq x_c$, $y = \beta_0 + \beta_1 x$

You can see when we plot this that the regression line closely fits the points on the left, but not the points on the right. This line only takes into account the first two coefficients, ie the intercept ($\beta_0$) and the first parameter ($\beta_1$).

```
plot( V2 ~ V1, cex=0.5, data=data,  xlab="Monthly Outdoor Temp (C)", ylab="Monthly Mean Electric Use (k
lmin.coef = coef(lmind)
abline(a = lmin.coef[1], b = lmin.coef[2],
       col ="blue")
```

When $x > x_c$, $y = (\beta_0 - \beta_2 x_c) + (\beta_1 + \beta_2)x$. This line accounts for the data on the right. The two lines intercept exactly at the change point.

```
par(mgp=c(2,1,0))
plot( V2 ~ V1, data=data,  xlab="Monthly Outdoor Temp (C)", ylab="Monthly Mean Electric Use (kWh/m^2/da
lmin.coef = coef(lmind)
abline(a = lmin.coef[1], b = lmin.coef[2],
       col ="blue")
abline(a = lmin.coef[1]-CP*lmin.coef[3],
       b = lmin.coef[2]+lmin.coef[3],
       col = "red")
```

We can also use the data to find an unknown change point for a linear model in R using the SiZer package.

```
pwlm = piecewise.linear(data$V1, data$V2)
pwlm$change.point
```

```
## [1] 18.29124
```

This results in a slightly higher change point of 18.29°C

```
pwlm$model
```

```
##
```

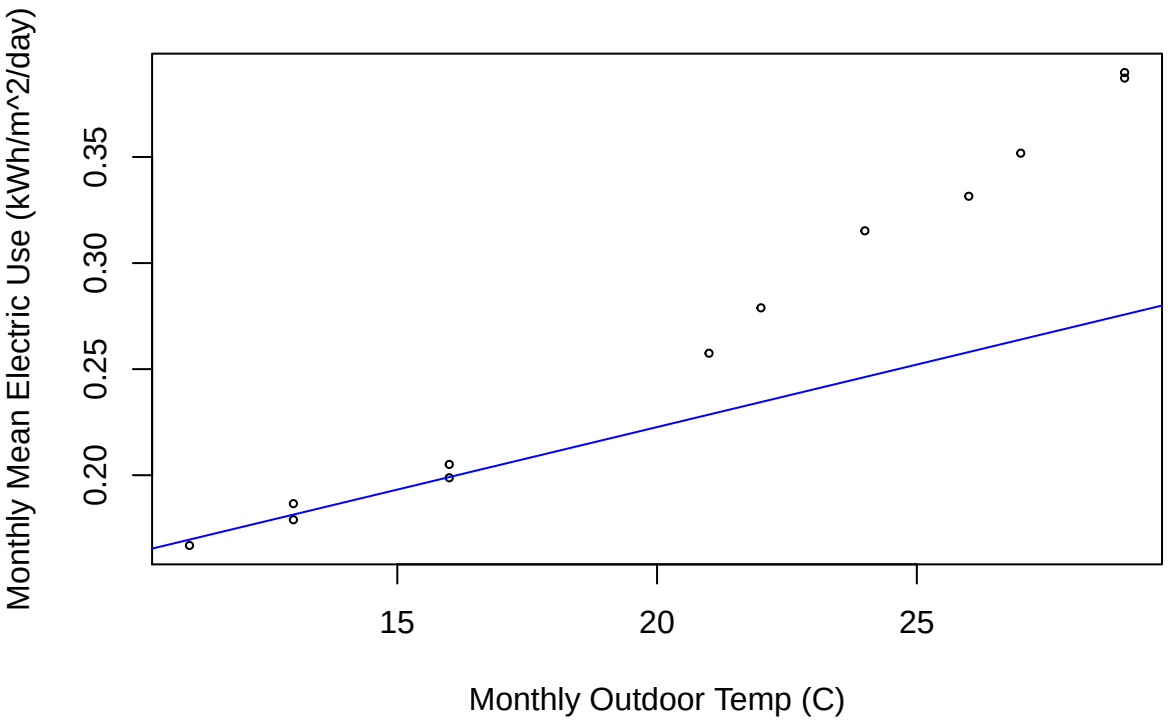



Figure 55: The peicewise regression line for all values less than the change point, 17C

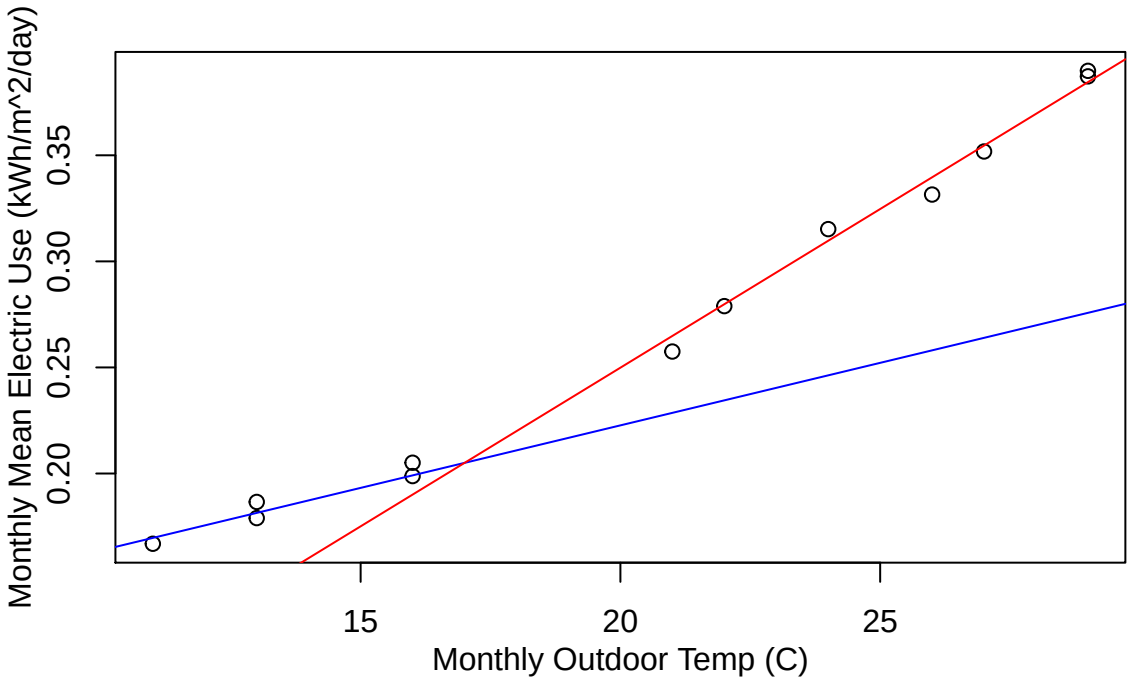



Figure 56: Both peicewise regression lines

```
## Call:
## lm(formula = y ~ x + w)
##
## Coefficients:
## (Intercept)            x            w
##    0.092750     0.006850     0.008874
```

It produces a linear multiple regression equation:

- $y = 0.09275 + 0.00685 * V1 + 0.00874 * V4$

```
plot(V2 ~ V1, data=data, xlab="Monthly Outdoor Temp (C)", ylab="Monthly Mean Electric Use (kWh/m^2/day)")
lines(pwlm, col="blue",lwd=4)
```

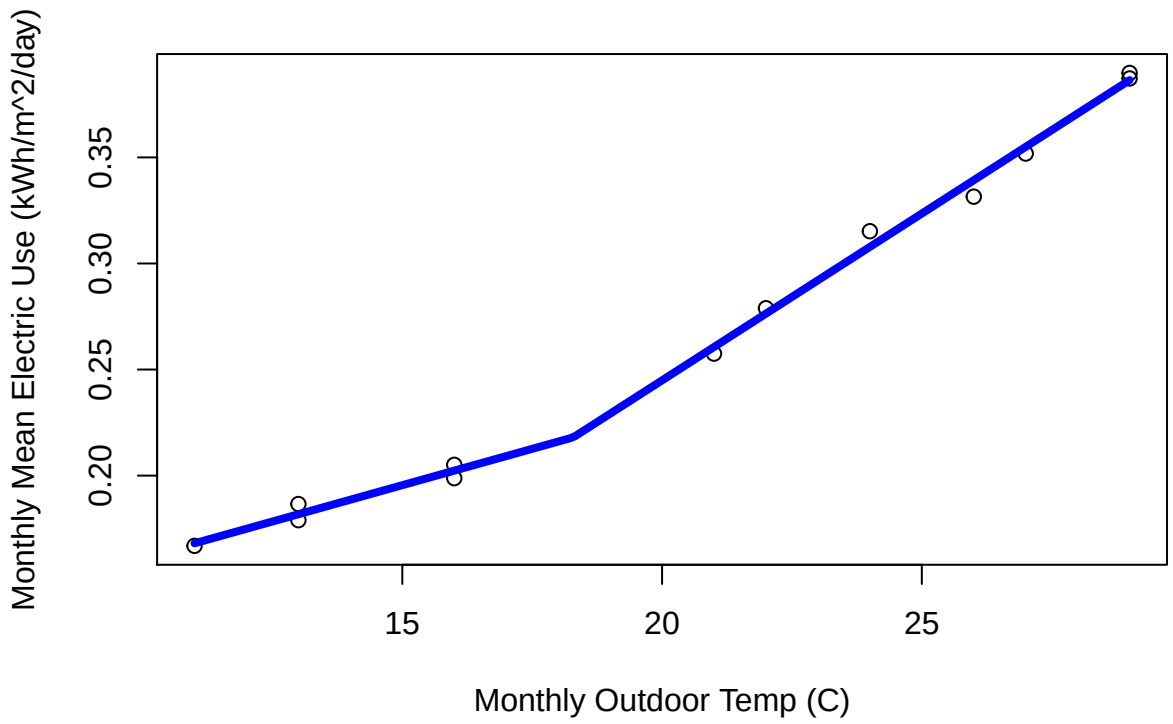



Figure 57: Thre resulting peicewise regression line from the SiZer package.

## 2.3 Local Regression

As we have seen there are clearly many advantages to linear regression. Linear regression is easy to fit and interpret. It also allows one to easily perform tests of significance and ensure that the the model is in fact an appropriate fit for the data.

There are also some disadvantages that come along with linear regression. Linear regression is a parametric approach because it assumes a linear functional form for $f(X)$. It is these assumptions about the form of $f(X)$ that is its biggest disadvantage. For instance, if we assume a linear relationship between $X$ and $Y$ but the true relationship is far from linear, then the resulting model will provide a poor fit to the data, and any conclusions drawn from it will be suspect.

Non-parametric approaches can be more beneficial for non-linear data sets. One of the most common non-parametric methods is **K-nearest neighbors regression**.

The $K$-nearest neighbors (KNN) method is a conceptually simple pattern recognition approach that is widely used for classification and regression.

- $K$-nearest neighbors regression does not make assumptions about the form of $f(X)$, and is therefore more flexible than linear regression.
- It is based on the distance measure and requires training data of observations from different groups

KNN can be used for both regression and classification. For classification, if a future object needs to be predicted, one determines the point closest to this new object and simply assigns the new object to the group to which the closest point belongs.

For regression, given a value of neighborhoods and a prediction point $x_0$, KNN regression first identifies the $K$ observations that are closest to $x_0$, represented by $N_0$. It then estimates $f(x_0)$ using the average of all the responses in $N_0$:

- $\hat{f}(x_0) = \frac{1}{K} \sum_{x_i \in N_i(x)} y_i$

### 2.3.1 K-Nearest Neighbors Regression - Choosing a $K$ value

One of the challenges to performing KNN is choosing how many neighborhoods, $K$, to use. The goal is to create a model which has low variance and is unbiased. KNN regression and classification gets closer to the actual fit of the model as more neighborhoods, $K$, are used. The down side to this is that the model will fit the training data so well that it will not have adequate prediction accuracy for a test set or new data. In other words, the more neighborhoods that are used, the more the model will overfit the data. Lower values of $K$ seem to mitigate overfitting, but they tend to have a high variance from the true observations.Therefore, the optimal value for $K$ ultimately depends of the *bias-variance tradeoff*.

- Small $K$-values provide a more flexible fit, with low bias but high variance
- Large $K$-values provide a smoother and less variable fit but may be more biased

Let's look at some data. We will analyze 1000 observations from the rsf measured data. We will try to predict the whole building energy consumption from the dry bulb temperature.

```
data=rsf2011[1:1000,]
plot(Whole_Building_Load~Dry_Bulb_Temp, data=data, cex=0.75, xlab="Dry Bulb Temp, C",
     ylab="Whole Building Energy Use (kW)")
```

After some visual inspection, it is clear that the data is not linear, so local regression methods, such as KNN, may be better suited in this case. We begin by setting up a training set containing 70% of the observations.

```
set.seed(111)
N = nrow(data)
index=1:N
```

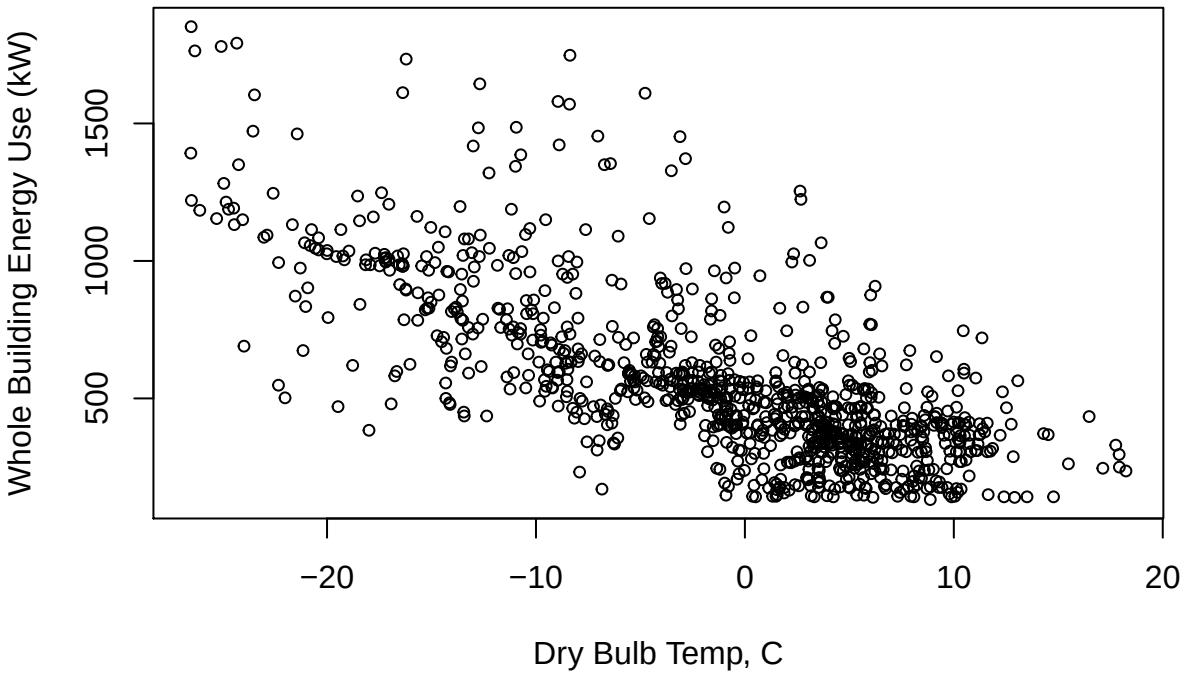



Figure 58: Full data set for 160 observations.

```
NT=round(0.7*N)
trainind = sample(index,NT,replace=FALSE)
testind = setdiff(index,trainind)
test=data[testind,]
train =data[trainind,]
x=train$Dry_Bulb_Temp
y=train$Whole_Building_Load
x0=test$Dry_Bulb_Temp
y0=test$Whole_Building_Load

TRO=train[order(train$Dry_Bulb_Temp, train$Whole_Building_Load ),]
TEO=test[order(test$Dry_Bulb_Temp, test$Whole_Building_Load),]
```

For comparison, let's first develop a linear model on the training data.

```
lm.fit=lm(TRO$Whole_Building_Load~TRO$Dry_Bulb_Temp)
plot(TRO$Whole_Building_Load~TRO$Dry_Bulb_Temp, cex=0.75, xlab="Dry Bulb Temp, C",
ylab="Whole Building Energy Use (kW)")
abline(lm.fit, col="red")
```

```
summary(lm.fit)
```

```
##
## Call:
## lm(formula = TRO$Whole_Building_Load ~ TRO$Dry_Bulb_Temp)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -597.04 -118.77  -32.32   82.42 1011.93
##
```

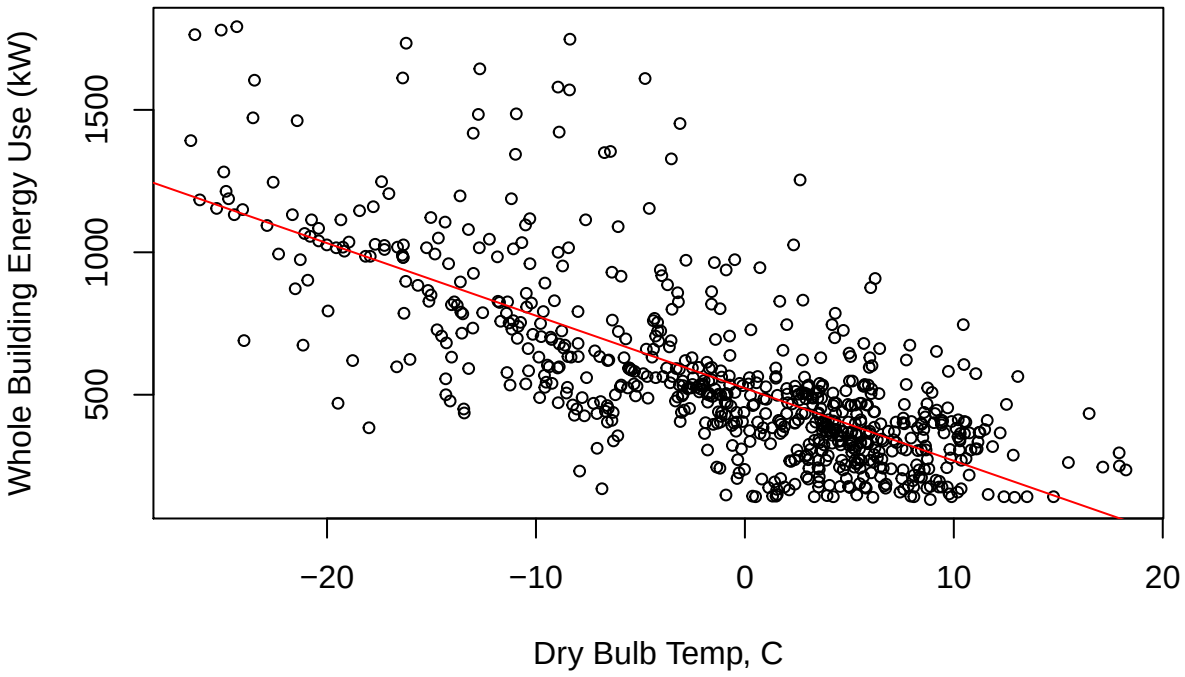



Figure 59: Linear regression on the training dataset

```
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       522.8072     8.0517   64.93   <2e-16 ***
## TRO$Dry_Bulb_Temp -25.4689     0.8899  -28.62   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 211.4 on 698 degrees of freedom
## Multiple R-squared:  0.5399, Adjusted R-squared:  0.5392
## F-statistic: 819.1 on 1 and 698 DF,  p-value: < 2.2e-16
```

```
anova(lm.fit)
```

```
## Analysis of Variance Table
##
## Response: TRO$Whole_Building_Load
##                    Df   Sum Sq  Mean Sq F value    Pr(>F)
## TRO$Dry_Bulb_Temp   1 36598734 36598734  819.08 < 2.2e-16 ***
## Residuals         698 31188696    44683
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As expected, linear model is not flexible enough and does not do an adequate job of explaining the variation in the model. Let's try KNN. We will use the knn.reg() function from the FNN package to do this. First, $K = 15$ is plotted.

```
library(FNN)
knn.train2=knn.reg(TRO$Dry_Bulb_Temp, y=TRO$Whole_Building_Load, k=15)
plot(TRO$Whole_Building_Load~TRO$Dry_Bulb_Temp, cex=0.75, xlab="Dry Bulb Temp, C",
ylab= "Whole Building Energy Use (kW)")
abline(lm.fit, col="red")
```

```
lines(TRO$Dry_Bulb_Temp, knn.train2$pred, col="blue")
```

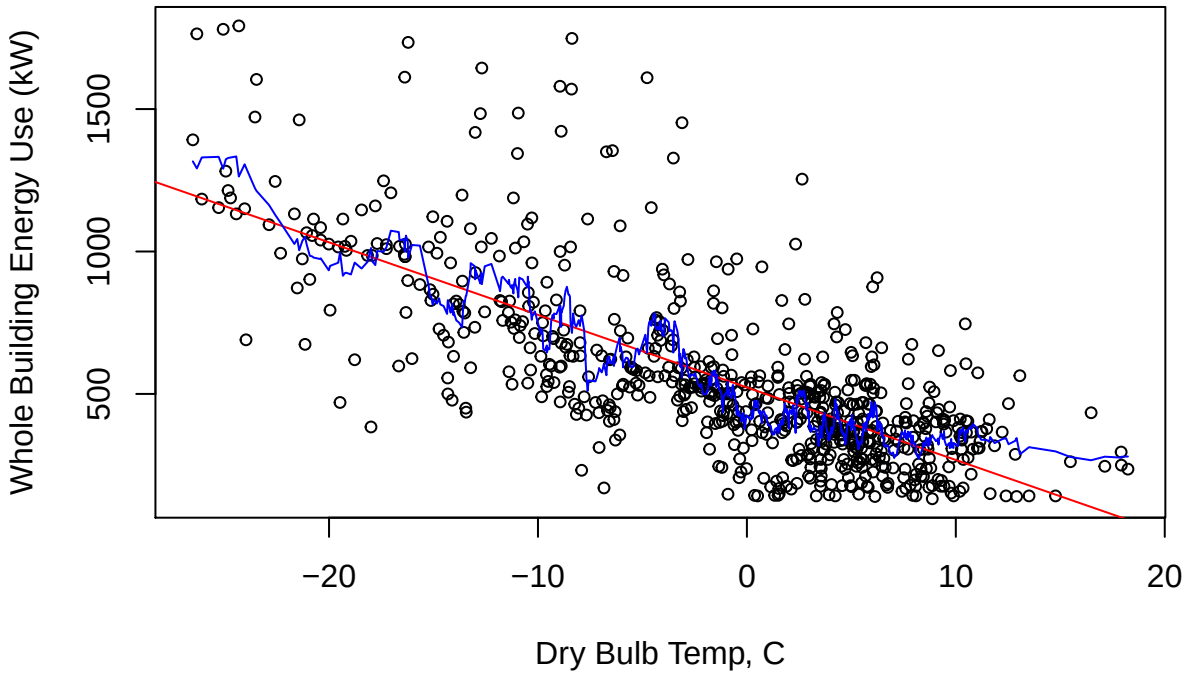



Figure 60: Linear regression (red) and KNN regression for K=15 (blue)

The KNN model seems to do better than the linear model. However, you can see that there is a bit of variance from the regression line to the observations. Let's now compare this with $K = 1$. IN this case, we make little to no mistakes in predictions.

```
knn.train=knn.reg(TRO$Dry_Bulb_Temp, y=TRO$Whole_Building_Load, k=1)
plot(TRO$Whole_Building_Load~TRO$Dry_Bulb_Temp, cex=0.75, xlab="Dry Bulb Temp, C",
ylab= "Whole Building Energy Use (kW)")
abline(lm.fit, col="red")
lines(TRO$Dry_Bulb_Temp, knn.train2$pred, col="blue")
lines(TRO$Dry_Bulb_Temp, knn.train$pred, col="orange")
```

You can see that the larger value for $K$ has the smoothest fit to the data, but has a larger variance than when $K = 1$. For $K = 1$ the model is the most flexible with reduced variance, but now the model is overfit and risks bad prediction accuracy on the test set.

Figure 62 depicts the predictions from the trained model for the test data set. You can see that although the model for $K = 1$ perfectly fits the training data, it does not very accurately predict the test data. Therefore, $K = 1$ does not yield good results for prediction.

```
plot(TEO$Whole_Building_Load~TEO$Dry_Bulb_Temp,
ylab= "Whole Building Energy Use (kW)", xlab="Dry_Bulb_Temp")
abline(lm.fit, col="red")
lines(TRO$Dry_Bulb_Temp, knn.train$pred, col="orange")
lines(TRO$Dry_Bulb_Temp, knn.train2$pred, col="blue")
```

We can choose a appropriate K value in R using the PRESS function. This returns the sum of squares of the predicted residuals. The linear model is indicated by the red line.

```
library(ModelMetrics)
nk=60
```

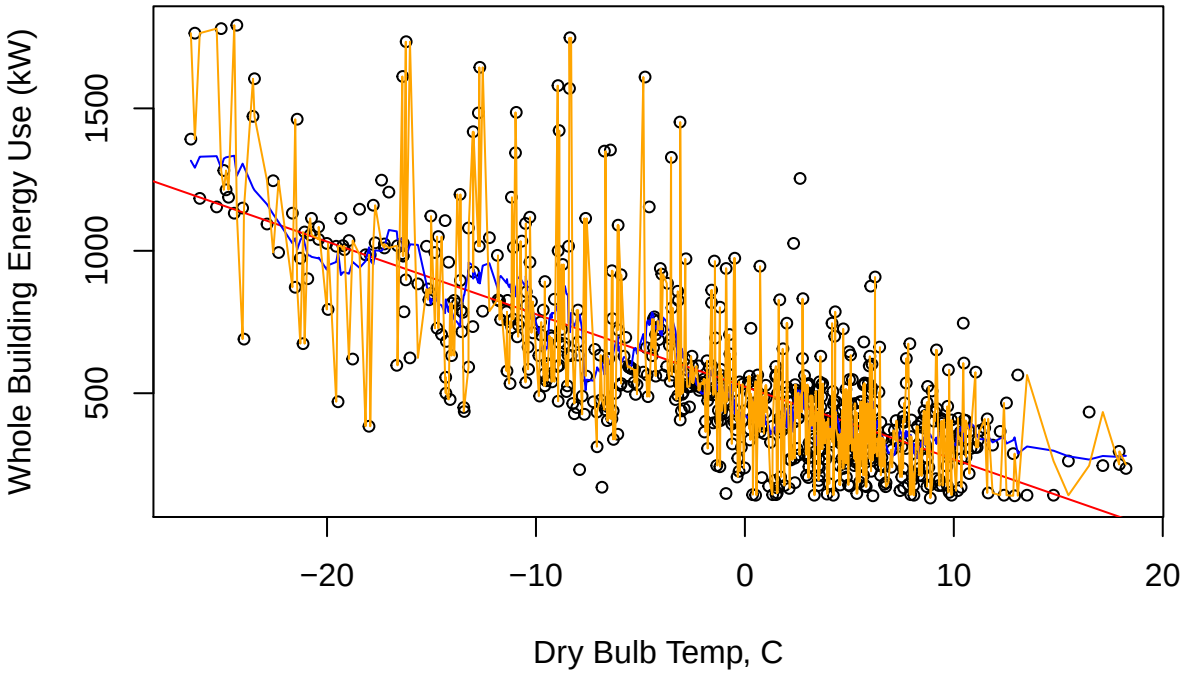



Figure 61: Linear regression (red) and KNN regression for K=15 (blue), and KNN regression for K=1 (orange)

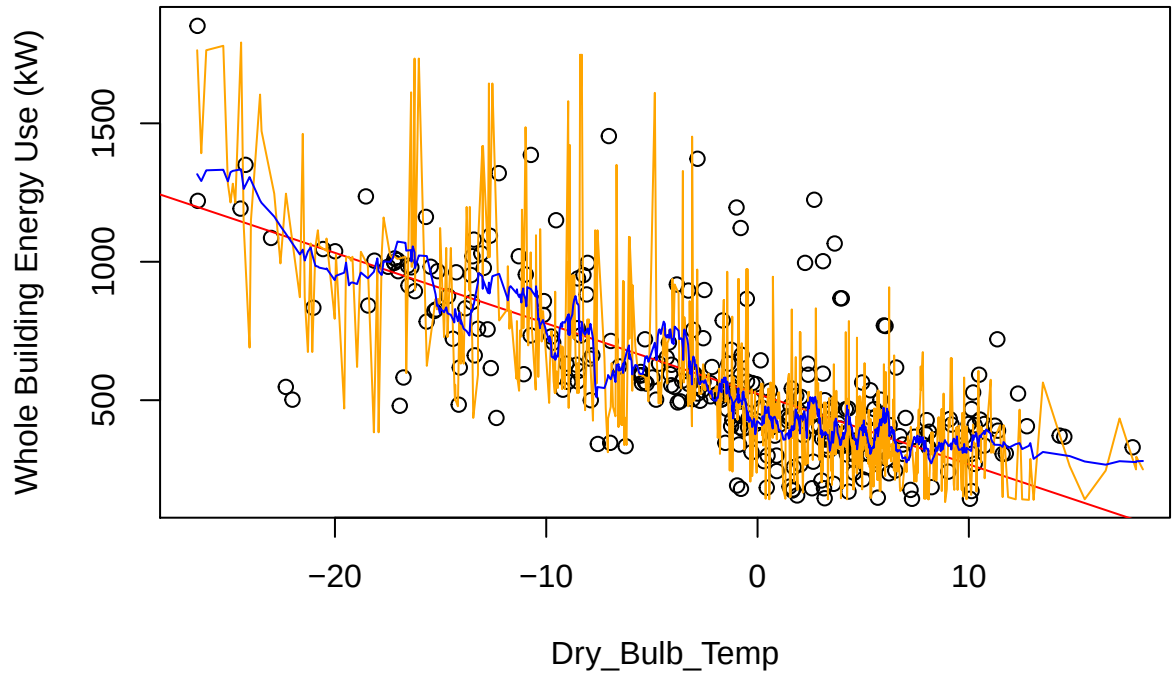



Figure 62: All of the models plotted with the test dataset

```
ks=1:nk
M=numeric(nk)
for(i in 1:nk) {
knn.mod=knn.reg(TRO$Dry_Bulb_Temp, y=TRO$Whole_Building_Load,k=ks[i])
M[i]<-knn.mod$PRESS
}
plot(ks,M, xlab="k", ylab="Residual Square Error", type="l")
abline(h=31188696, col="red")
```

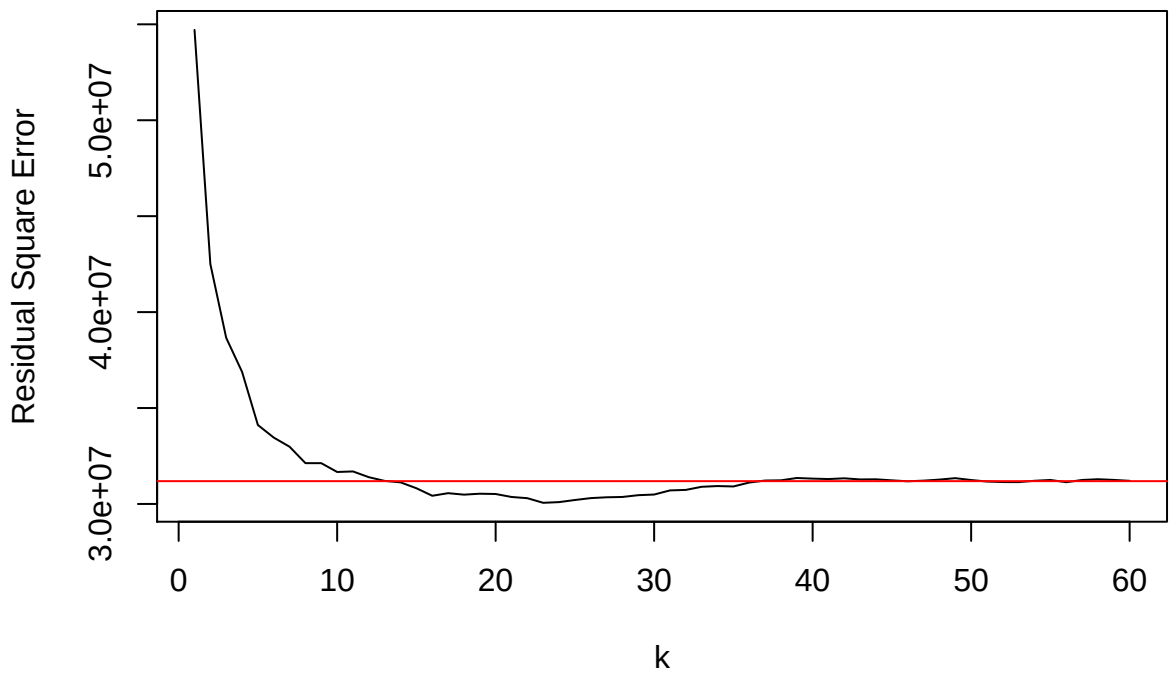



Figure 63: Residual squared error for the prediction residuals for values of k between 1 and 60

Values of $K$ between 13 and 37 are an improvement to the linear model with a minimum residual square error around $K = 22$.

We can also look at the $R^2$ value. Similarly, we can see that the KNN model is superior to the linear model for values of $K$ between 15 and 38 with a maximum $R^2$ at around $K = 22$

```
 for(i in 1:nk) {
knn.mod=knn.reg(TRO$Dry_Bulb_Temp, y=TRO$Whole_Building_Load,k=ks[i])
M[i]<-knn.mod$R2Pred
}
plot(ks,M, xlab="k", ylab="R2 Value", type="l")
abline(h=0.5399, col="red")
```

### 2.3.2 K-Nearest Neighbors and Linear Regression

So far in this section we have seen the benefits of using KNN. However, it is important to recognize at which instances linear regression in more appropriate than KNN.

a) Linear regression is a more appropriate modeling approach when the true regression and the model regression are close.

b) The more linear the relationship in the data, the better linear regression performs.

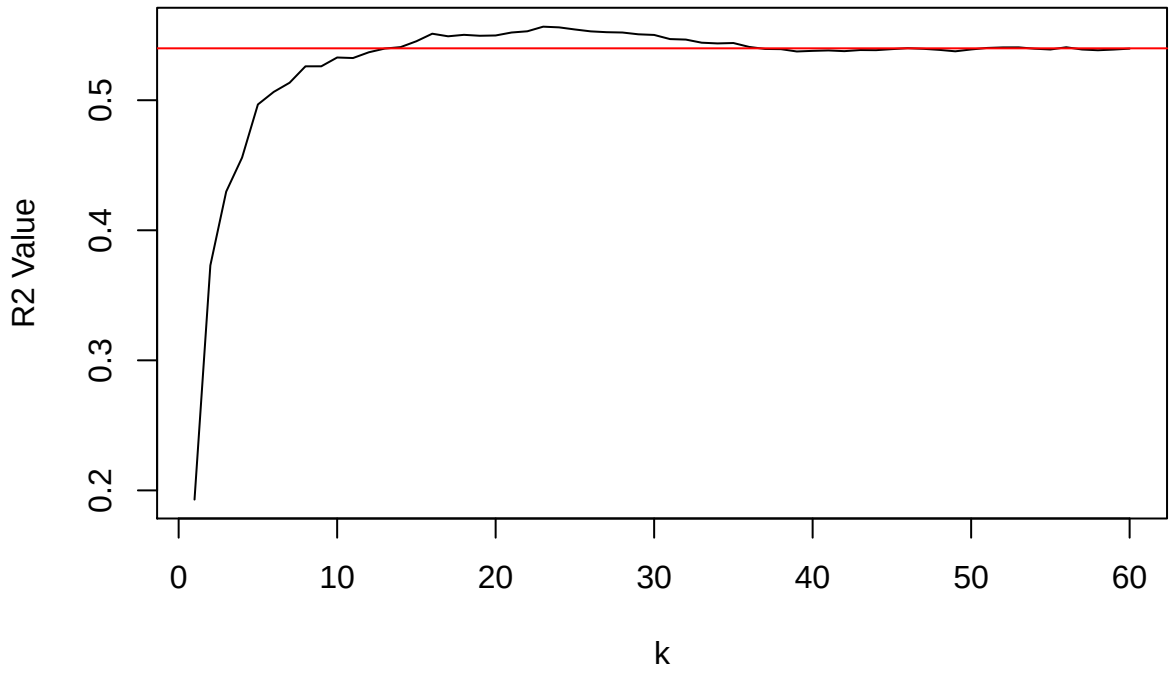



Figure 64: R2 value for the prediction residuals for values of k between 1 and 60

c) As the number of predictors in the model increases, the accuracy of the KNN fit decreases.

# 3 Beyond OLS

The majority of the material and examples provided in this section were adapted from the following sources. Additional sources are listed in the references.

- Ford, Clay. (2015, September 20). Getting Started with Quantile Regression. Retrieved from http://data.library.virginia.edu/getting-started-with-quantile-regression/.
- The Pennsylvania State University. (2017). 6.1-Introduction to Generalized Linear Models. Retrieved from https://onlinecourses.science.psu.edu/stat504/node/216

*By the completion of this section, the reader should be able to*

- *understand the basics of quantile regression*
- *understand the basics of generalized linear models*
- *be able to identify how and when quantile regression and generalized linear models may be superior approach to a linear model*
- *be able to carry out quantile regression and generalized linear models in R*
- *use R to select the best model approach for the given data*

## 3.1 Quantile Regression (QR)

As we have seen, least squares regression captures how the mean of e $Y$ changes $X$. However, sometimes a single mean curve is not informative enough. In this case, quantile functions can provide a more complete view of the relationship in data. Just as linear regression allows us to estimate models for the conditional **mean** function, the QR method allows one to estimate models for the conditional **median** function by modeling the relation between a set of predictor variables and specific percentiles (or quantiles) of the response variable.

While least squares estimation asks how the conditional mean of $Y$ depends on the covariates $X$, quantile regression asks this same question at each quantile of the conditional distribution, enabling one to obtain a more complete description of how the conditional distribution of $Y$ given $X$ depends on the observations.

**Example**

Adapted from http://data.library.virginia.edu/getting-started-with-quantile-regression/.

For this example we will use data from PVWatts. We will try to make predictions about hourly energy production (kWh) from the amount of exposure of solar global horizontal radiation ($kW/m^2$). This data set contains 8760 observations.

```
library(EMBA)
data=rsf2011
attach(data)
plot(Irradiance_Global,PV_Generation,cex=0.05,type="n",
xlab="Global Irradiance [kW/m^2]", ylab="PV Power [kW]", cex.axis=0.5, cex.lab=0.5)
points(Irradiance_Global,PV_Generation,cex=.5,col="darkgray")
```

You can see that there is increasing variability as global horizontal radiation increases. This violates the assumption of linear regression that requires normal error with constant variance. This limits the usefulness of OLS in this case.

The red line represents a linear model fit to the data.

We can now consider quantile regression starting with the median regression. To do this in R we use the rq(formula, tau, data) function from the quantreg package

```
library(quantreg)
```

```
## Loading required package: SparseM
```

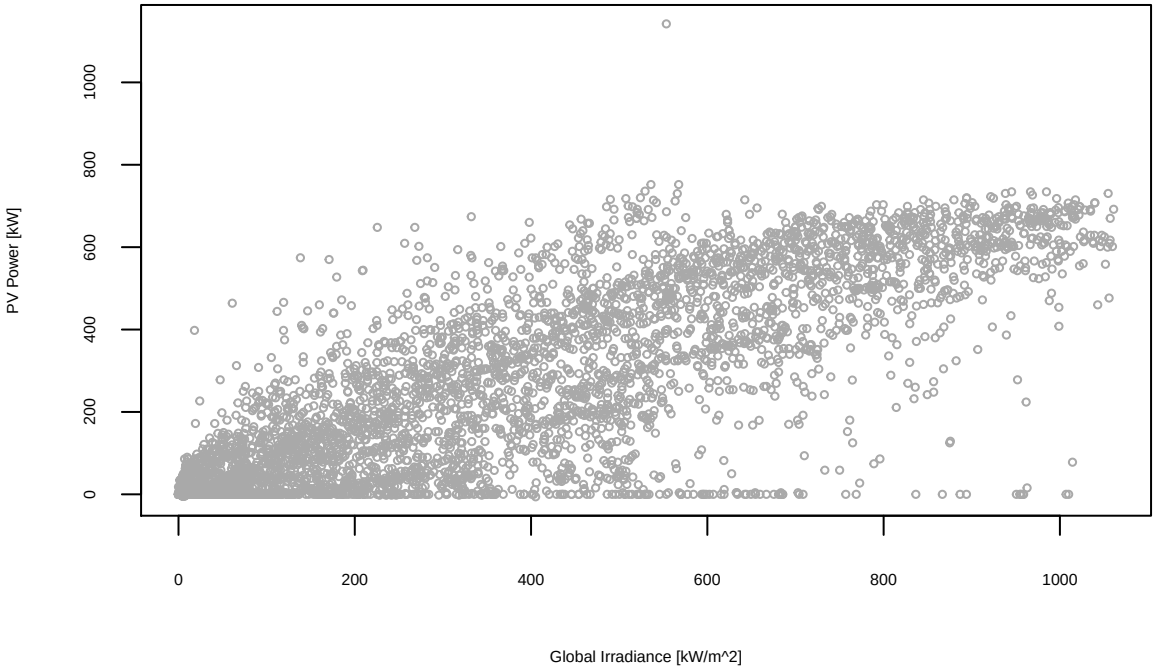



Figure 65: PVWatts dataset

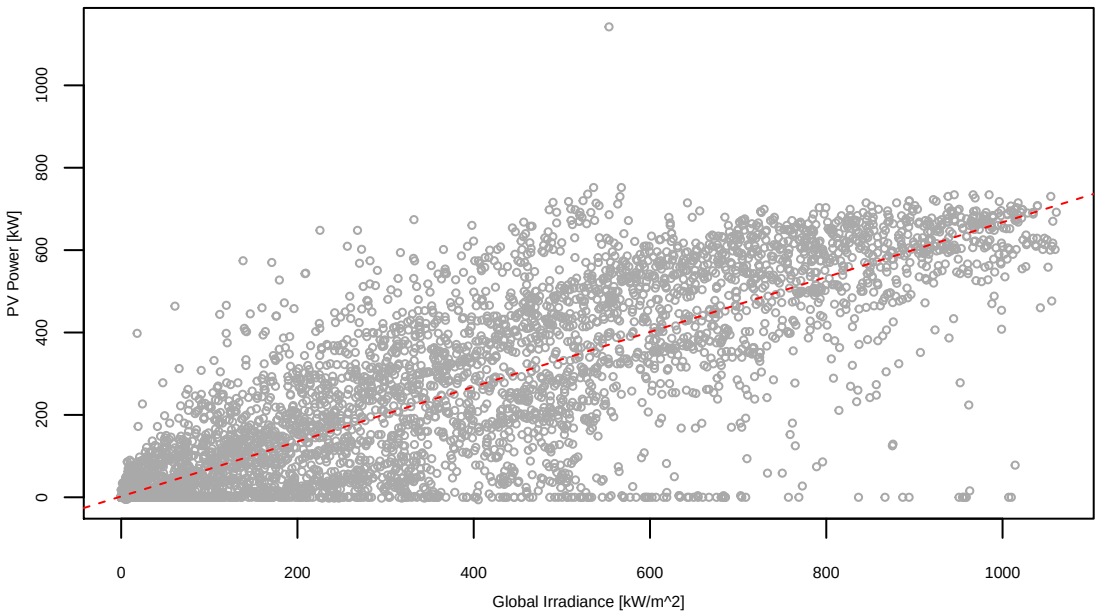



Figure 66: linear regression of PVWatts data

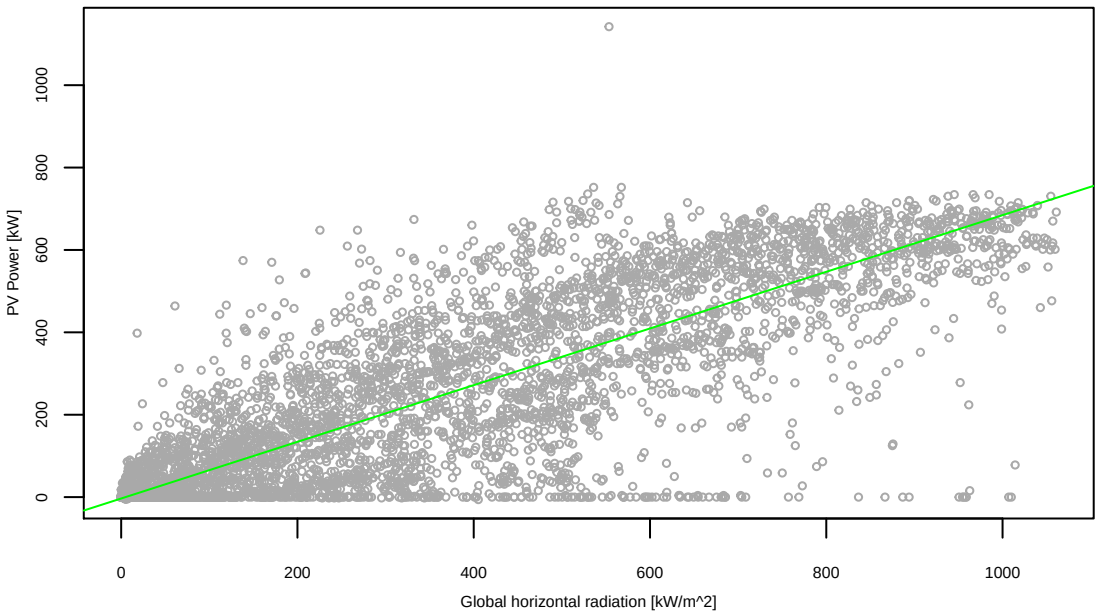



Figure 67: Median quantile regression of PVWatts data

```
## Warning: package 'SparseM' was built under R version 3.3.2
```

```
##
## Attaching package: 'SparseM'
```

```
## The following object is masked from 'package:base':
##
##     backsolve
```

```
fit.med<-rq(PV_Generation~Irradiance_Global, tau=0.5)
par(mgp=c(1.5,0.75,0))
plot(Irradiance_Global,PV_Generation,cex=0.05,type="n",xlab="Global horizontal radiation [kW/m^2]", ylab
points(Irradiance_Global,PV_Generation,cex=.5,col="darkgray")
abline(fit.med, col="green")
```

The green line represent the median quantile fitted to the data.

```
## The following objects are masked from data (pos = 5):
##
##     Cooling, Data_Center, Dry_Bulb_Temp, Heating,
##     Irradiance_Global, Irradiance_South_Facade, Lighting,
##     Mechanical, Plug_Loads, PV_Generation, Time,
##     Whole_Building_Load, Whole_Building_Net
```

Adding the rest of the quantiles in dark green and the linear model as the red dotted line, you can see that the quantile regression is able to more thoroughly describe the data.

The output of the rq() function gives the estimated coefficients and some information about the model.

The following are details on the median quantile.

```
fit.med
```

```
## Call:
## rq(formula = PV_Generation ~ Irradiance_Global, tau = 0.5)
##
```

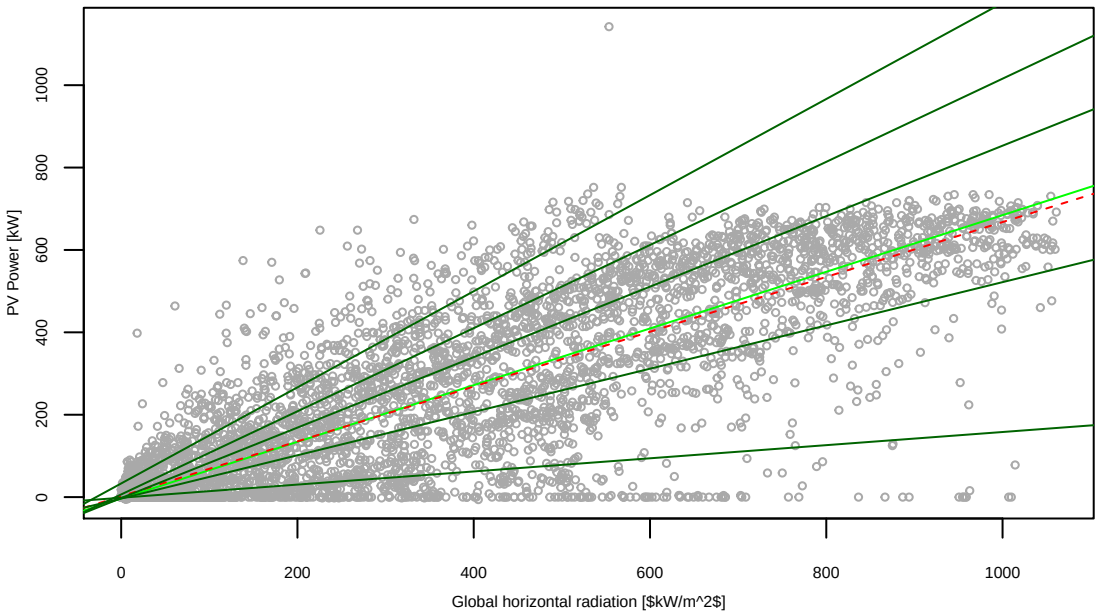



Figure 68: The 10th, 25th, 75th, 90th, and 95th quantile's regression lines on PVWatts data

```
## Coefficients:
##       (Intercept) Irradiance_Global
##        -3.3408753         0.6878617
##
## Degrees of freedom: 8759 total; 8757 residual
```

To obtain more detailed information we can use the summary() function which gives confidence intervals along with the estimated coefficients.

```
summary(fit.med)
```

```
##
## Call: rq(formula = PV_Generation ~ Irradiance_Global, tau = 0.5)
##
## tau: [1] 0.5
##
## Coefficients:
##                   Value     Std. Error t value   Pr(>|t|)
## (Intercept)        -3.34088   0.04548  -73.46080   0.00000
## Irradiance_Global   0.68786   0.00467  147.21789   0.00000
```

We can now predict values at different quantiles. Here we are predicting the hourly energy production from a global horizontal of 0.5 at the median quartile.

```
predict.rq(fit.med, newdata=data.frame(Irradiance_Global=0.5),
           data=data)
```

```
##         1
## -2.996944
```

In R you are able to run simultaneous quantile regressions, which outputs information on multiple quantiles at once. In this example we will print out the 10th and 95th quantile details.

```
fit.qr <- rq(PV_Generation~Irradiance_Global, tau=c(0.1, 0.95), data=data)
summary(fit.qr)
```

```
## Warning in summary.rq(xi, U = U, ...): 1967 non-positive fis
##
## Call: rq(formula = PV_Generation ~ Irradiance_Global, tau = c(0.1,
##     0.95), data = data)
##
## tau: [1] 0.1
##
## Coefficients:
##                   Value     Std. Error t value   Pr(>|t|)
## (Intercept)        -1.53184   0.10122  -15.13361   0.00000
## Irradiance_Global   0.15959   0.01996    7.99453   0.00000
##
## Call: rq(formula = PV_Generation ~ Irradiance_Global, tau = c(0.1,
##     0.95), data = data)
##
## tau: [1] 0.95
##
## Coefficients:
##                   Value    Std. Error t value  Pr(>|t|)
## (Intercept)       33.14872  2.56541   12.92141  0.00000
## Irradiance_Global  1.16660  0.02388   48.86011  0.00000
```

## 3.2 Generalized Linear Model

We know that a linear model specifies the linear relationship between a response variable $y$ and a set of predictor variables $x_i$.

- $y = \beta_0 + \beta_1 x_1 + ... + \beta_i x_i$'

However, there are many relationships that cannot be properly described by a linear equation.

1. The dependent variable may have a non-continuous distribution, and thus the predicted values should also follow the respective distribution
2. The relation between predictors and dependent variables may not be linear, i.e. the **link** between predictors and response variables is nonlinear

Generalized linear models (GLMs) are an extension of linear regression that allows for non-normal error distributions or heteroscedasticity.

Generalized linear models address these issues by:

1. allowing response variables $y_i$ to follow an **exponential family distribution** i.e. Poisson, binomial, Gamma, and normal.
2. transforming the linear functions of the predictor variables, $f(x)$ using a **link** function to relate to the nonlinear response variables

- Rather than modeling the response $y$ directly as in ordinary linear regression, GLMs allow the magnitude of the variance of each response to be a function of its predicted value.

**Three Main Components of GLM:**

- **Random Component** - refers to the probability distribution of the response variable $(y)$.
- **Systematic Component** - specifies the predictor variables $(x_1, x_2, ..., x_k)$ in the model, more specifically their linear combination in creating the so called linear predictor; e.g., $\beta_0 + \beta_1 x_1 + \beta_2 x_2$ as we have seen in linear regression, or $\frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}}$ as we will see in a logistic regression in this lesson.
- **Link Function, $\eta$ or $g(\mu)$** - specifies the link between random and systematic components. It says how the expected value of the response relates to the linear predictor e.g., $\eta = g(E(y_i)) = E(y_i)$ for linear regression.

The following table depicts some examples of the different components and their corresponding links:

To help make the process clear, let's look at a trivial example of logistic regression.

### 3.2.1 Generalized Linear Model - Logistic Regression

Adapted from https://onlinecourses.science.psu.edu/stat504/node/216.

For binary outcomes, logistic regression is used. Let Y be a binary outcome. Instead of modeling $Y$ directly, logistic regression models the probability that $Y$ belongs to a certain category. We want to model the conditional probability of $p(x) = Pr(Y = 1|X = x)$ as a function of x. To achieve this, the logistic, or logit, function is used. The monotone transformation called the **logit** transformation of $p(x)$ is defined as:

- $\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$

Solving for p gives the systematic component:

- $p(x) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}}$

Here we have transformed a linear model to guarantee that we always get a probability out, regardless of the values of $\beta_0, \beta_1$ or $x$. We still have a linear function but we are now modeling the probabilities on a nonlinear scale.

| Model | Random | Link | Systematic |
|---|---|---|---|
| Linear Regression | Normal | Identity | Continuous |
| ANOVA | Normal | Identity | Categorical |
| ANCOVA | Normal | Identity | Mixed |
| Logistic Regression | Binomial | Logit | Mixed |
| Loglinear | Poisson | Log | Categorical |
| Poisson Regression | Poisson | Log | Mixed |
| Multinomial response | Multinomial | Generalized Logit | Mixed |

Figure 69: A table of different model types and their link functions and random and systematic components.(https://onlinecourses.science.psu.edu/stat504/node/216)

- Maximum Likelihood estimation is then used to to estimate parameters
- After parameter estimation is complete, the estimated probability, $\hat{p}(x)$, can be found by plugging in the values for the prediction variables $x$.

Multivariate logistic regression is modeled similarly.

- $\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p$
- $p(X) = \frac{e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}}$

### 3.2.2 Example - R Applications

Adapted from Reddy, Agami. T. (2013)

In this example we will use data from an analysis for chiller fault detection. There are four parameters. Class classifies each chiller as either having a fault (1) or not having a fault (0). $T_{cd,sub}$ represents the amount of refrigerant sub cooling in the condenser in degrees Celsius. $T_{cd,app}$ condenser approach temperature in degrees Celsius. Lastly the COP, or the coefficient of performance of each is listed.

First we need to load the data

```
library(MASS)
data = read.csv("CondenserData.csv")
```

Next we create a training set with 42 of the 54 observations; the remainder of the observations will make up the cross validation set.

```
train = data[1:42,]
valid = data[43:54,]
```

Use logistic regression and stepAIC to find a best model to fit the observations in the training set.

Modeling GLMs in R is fairly easy. We simply apply the glm() function as we would the lm() function. The difference is that the random component or family needs to be specified i.e. glm(formula, family, data,... ).

For this example we apply the glm() function to a formula that describes the condition of the chiller, Class, based on its COP, refrigerant sub cooling, $T_{cd,sub}$ and its condenser approach temperature, $T_{cd,app}$. By specifying the family as binomial, this creates a generalized linear model in the binomial family.

```
zz = glm(train[,1]~.,data=train[,2:4],family=binomial(link="logit"))
bestzz = stepAIC(zz,TRACE=TRUE,steps=1000,direction="both")
```

```
## Start:  AIC=8
## train[, 1] ~ COP + Tcd.sub + Tcd.app
##
##           Df Deviance    AIC
## - Tcd.sub  1   0.0000  6.000
## - COP      1   0.0000  6.000
## <none>         0.0000  8.000
## - Tcd.app  1   4.1235 10.123
##
## Step:  AIC=6
## train[, 1] ~ COP + Tcd.app
##
##           Df Deviance    AIC
## <none>          0.000  6.000
## + Tcd.sub  1    0.000  8.000
## - COP      1   22.176 26.176
## - Tcd.app  1   56.388 60.388
```

We can also look at the predictions for the test set.

```
preds = round(predict.glm(bestzz,newdata=valid[,2:4],type="response"))
preds
```

```
## 43 44 45 46 47 48 49 50 51 52 53 54
##  0  0  0  0  0  0  1  1  1  1  1  1
```

The important thing to note here is the difference in the predict function from what we have seen so far. The "response" type gives back the prediction in the original scale. Otherwise the solutions will be returned in the scale of the link function.

We can now get a sense of the test prediction errors.

```
errors = as.numeric(preds != valid$Class)
err = sum(errors)/length(errors)
err
```

```
## [1] 0
```

There are zero errors when predicting the test set for this model!

# 4 Unsupervised Learning

The majority of the material and examples provided in this section were adapted from the following sources. Additional sources are listed in the references.

- Reddy, T.Agami. Applied Data Analysis and Modeling for Energy Engineers and Scientists. Springer, 2011.
- James, Witten, Hastie, and Tibshirani. (2013). An Introduction to Statistical Learning. New York: Springer
- Smith, L. S. (2002, February 26,). A tutorial on Principal Components Analysis. Retrieved from http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf.

*By the completion of this section, the reader should be able to*

- *understand the distinction of unsupervised and supervised learning*
- *understand the benefits and limitations of PCA and PCR*
- *understand why and when clustering methods may work well for some data*
- *be able to use R to find the ideal number of PCs for use in PCR*
- *know how to find patterns in data using clustering methods in R*

Up until now we have discussed statistical learning in the context of **supervised learning**. In **supervised learning** we observe a set of features $X_1, X_2, ..., X_p$ with a response variable Y with the *goal* of predicting Y using $X_1, X_2, ..., X_p$. We have introduced many methods for regression and classification.

Starting in this section, we will begin to discuss methods in the context of **unsupervised learing**. In this type of learning, we observe only the features $X_1, X_2, ..., X_p$ with no associated response variable Y. Unlike supervised learning, unsupervised learning has no single, simple goal. Using these methods we can evaluate the data, identify patterns, highlight similarities and difference, etc...

We will discuss two methods:

1. **Pricipal Component Analysis**
2. **Clustering**

## 4.1 Principal Component Analysis (PCA)

One of the biggest issues with multivariate regression is the assumption that the variables are independent when in reality they are often related (collinear) to some extent. This can cause regression coefficients to be biased or to have the wrong sign.

PCA is an unsupervised method and is very useful for removing the adverse effects of collinearity, while summarizing the main aspects of the variation in the regressor set. It should be noted that PCA is not a method that leads to a decision on a hypothesis. Instead it is a method for identifying which parameters are collinear and reduces the dimensionality of data. This in turns leads to more robust model building and useful insights into the behavior of the data. Since patterns in data of high dimensions can be difficult to find since no graphical representation is available, PCA is powerful tool for analyzing data.

The premise in PCA is that the variance in the collinear multidimensional data comprising of the regressor variable vector $X$ can be re-framed in terms of a set of orthogonal transformed variable vectors $U$. These vectors will then provide a means of retaining only a subset of variables which explains most of the variability in the data. Once the vectors have been identified, the data can be compressed without much loss of the information contained in the original data set.

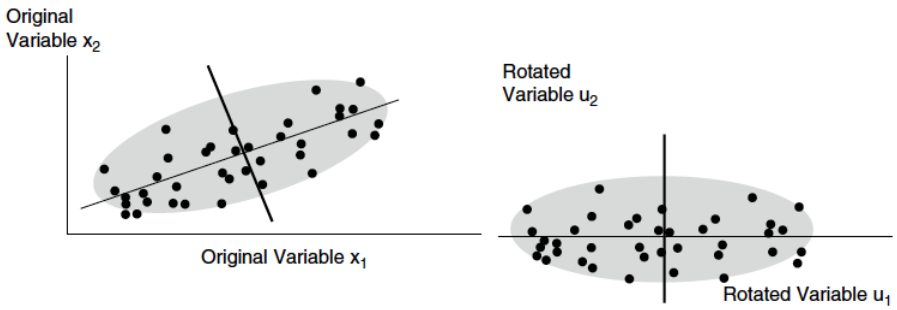



Figure 70: Visual representation of the rotation that takes place during PCA. (Figure 10.5, Reddy 2011)

### 4.1.1 Mathematics Review

Because PCA requires some understanding of basic mathematics principles, lets do a quick review of the main concepts before we go into how the components are found.

**Covariance** is the degree to which the dimensions vary from the mean with respect to each other.

Variance: one dimension:

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

Covariance: two dimensions

- $cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$ Sign of covariance gives info about data:
- (+): the dimensions increase together
- (-): As one dimension increases, the other decreases

The covariance between two variables gives you the variance of that variable from the mean.

**The covariance matrix** is used when there are more than 2 dimensions. Is it a matrix that contains all of the covariance values between every dimension.

- $C^{nxn} = (c_{i,j}, c_{i,c} = cov(Dim_i, Dim_j))$

where $C^{nxn}$ is a matrix with $n$ rows and $n$ columns. If you have an $n$-dimensional data set, then the matrix has $n$ rows and columns and each entry int he matrix is the result of calculating the covariance between two separate dimensions. Ex: the 2nd row, 3rd column, is the covariance value calculated between the 2nd and 3rd dimension.

Ex: Imaginary three-dimensional data set $x, y$, and $z$.

- $C = \begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix}$

Note: the main diagonal are the variances for each dimension. Everything is symmetrical across this diagonal, $cov(a,b) = cov(b,a)$.

**Eigenvectors** $A$ and **eigenvalues** $\lambda$ of a matrix $Z$ are defined by:

- $AZ = \lambda Z$

The eigenvalues are the solutions of the determinant of the covariance matrix.

- $|Z'Z - \lambda I| = 0$

Properties of Eigenvectors

- Eigenvectors can only be found for square matrices
- Given an $n \times n$ matrix, there are $n$ eigenvectors
- Eigenvectors of a matrix are perpendicular

### 4.1.2 PCA - The Basics

PCA analysis is typically done with standardized variables $Z$ instead of the original variables $X$. Variables $Z$ have a zero mean and unit variance. This is achieved by subtracting each dimension by the mean.

PCA is especially useful for data with high dimensions. In such cases one needs to have some mathematical means of ascertaining the degree of variation in the multi-variate data along different dimensions. PCA uses eigenvectors and eigenvalues of the covariance matrix to combine uncorrelated features with maximum variance.

- To find the eigenvalues the covariance matrix of the variables $Z$ needs to be found.
- The eigenvectors and eigenvalues can then be found from the covariance matrix. Since eigenvectors are orthogonal, each of them are uncorrelated. The eigenvalue is the length of the axis, while the eigenvector represents the direction of rotation.

In order to better visualize PCA, let's look at a two dimensional example.

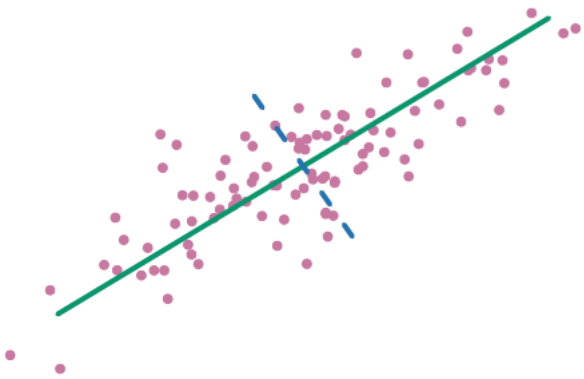


Figure 71: The first principle component (green) describes most of the data while the second PC describes the rest. (Figure 6.14, James, Witten, Hastie, & Harlow, 2013)

Consider the above figure. The green line represents the first principle component, while the blue dotted line represents the second principle component. You can see that the blue line represents most of the variability in the data. It is also the longer of the two lines which will produce the largest eigenvalue and therefore will be identified as the first principle component. The blue line represent less of the variability and is therefore shorter and the second PC. Figure 60 represents the data after rotation of the principle components.

On of the important properties of eigenvalues that was not discussed above is that their sum is equal to the trace of the covariance matrix, i.e. $\lambda_1 + \lambda_2 + ... + \lambda_p = p$, where $p$ is the number of variables. This stems from the fact that the diagonal of the covariance matrix sums to unity.

The eigenvalues are ranked such that the first has the largest numerical value (explains the most variance), the second has the next largest value, and so on. The corresponding eigenvector represents the coefficient of

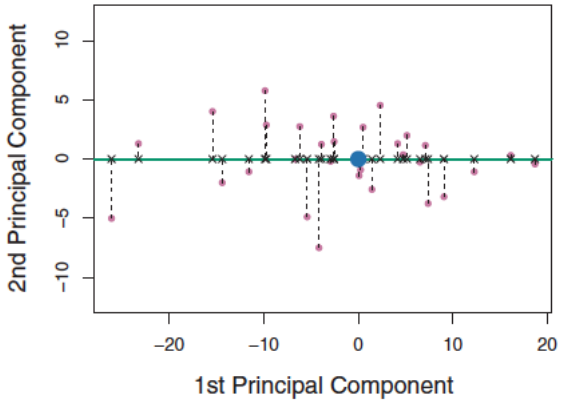



Figure 72: Rotated data during PCA (Figure 6.15, James, Witten, Hastie, & Harlow, 2013)

the **principle component** (PCs). Thus, the linearized transformation for the PC from the original vector of standardized variables Z can be represented by:

- $PC1 : u_1 = a_{11}z_1 + a_{12}z_2 + ... + a_{1p}z_p$ subject to $a_{11}^2 + a_{12}^2 + .... + a_{1p}^2 = 1$
- $PC2 : u_2 = a_{21}z_1 + a_{22}z_2 + ... + a_{2p}z_p$ subject to $a_{21}^2 + a_{22}^2 + .... + a_{2p}^2 = 1$ .
- $PCp : u_p = a_{p1}z_1 + a_{p2}z_2 + ... + a_{pp}z_p$ subject to $a_{p1}^2 + a_{p2}^2 + .... + a_{pp}^2 = 1$ where $a_{ii}$ are the **component weights** and are the scaled elements of the corresponding eigenvectors.

Thus the covariance matrix for the standardized and rotated variable is now transformed into

- $C = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & ... & 0 \\ 0 & 0 & \lambda_2 \end{pmatrix}$

where $\lambda_1 > \lambda_2 > ... > \lambda_p$

The off-diagonal terms of the new covariance matrix are zero because the variable vector U is orthogonal. It is clear from this matrix that the eigenvalues represent the variability of the data along the principle components.

From this point the dimensionality of the data can be reduced. Without this reduction, not much is gained from PCA. This compression of the data is done by rejecting the principle components that explain small portions of the variance. Recall that the eigenvalues are already ranked from the first explaining the most variance to the last explain little to no variance. A typical rule of thumb to decide what PCs to leave out is to drop any that explain less than $1/p$ of the variability where $p$ is the original dimensionality of the data.

### 4.1.3 The Principal Component Regression Approach

The principle components regression (PCR) approach involves choosing the $M$ principle components, $Z_1, ..., Z_M$, and then using these components as the predictors in a linear regression model that is fit using least squares. The key idea is that often times a smaller number of principal components can describe the data sufficiently. In other words we assume that the directions in which $X_1, ..., X_p$ show the most variation are the directions that are associated with $Y$. If this assumption holds that fitting a model to $Z_1, ...., Z_M$ will lead to better results than fitting a least squares model to $X_1, ..., X_p$.

**PCR Example**

```
library(pls)
library(EMBA)
```

```
set.seed(2)
data=na.omit(rsf2011)
data1=data[,-9]
```

The pcr() function is similar to the lm() function with a few differences. The main different is the scale argument. In order to standardize the predictors, we set the scale argument equal to true

```
pcr.fit=pcr(Whole_Building_Load~., data=data, scale=T)
summary(pcr.fit)
```

```
## Data:    X dimension: 8759 12
##  Y dimension: 8759 1
## Fit method: svdpc
## Number of components considered: 12
## TRAINING: % variance explained
##                       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## X                      39.475     60.4    71.93    82.36    88.84    92.56
## Whole_Building_Load     4.322     69.8    70.07    71.88    95.99    96.28
##                       7 comps  8 comps  9 comps  10 comps  11 comps
## X                       94.93    96.82    98.11     99.12       100
## Whole_Building_Load     97.29    99.28    99.32     99.34       100
##                       12 comps
## X                          100
## Whole_Building_Load        100
```

The $X$ represents the percentage of the variance explained. You can also look at the performance of the model using different numbers of PCs by looking at the $R^2$ and the mean square error of prediction (MSEP).

```
validationplot(pcr.fit, val.type="MSEP")
```

```
validationplot(pcr.fit, val.type="R2")
```

Using five components explains over 90% of the variance, while maintaining a reasonable $R^2$ and MSEP value.

Let's now use PCR on a training and test set to evaluate the performance of our model. Here we will use only 5 PCs.

```
set.seed(1)

train=data[1:5000,]
ytest=data[5000:8759,8]
test=data1[5000:8759,]

pcr.fit=pcr(Whole_Building_Load~., data=train, scale=T)

pcr.pred=predict(pcr.fit,test,ncomp=5)
mean((pcr.pred-ytest)^2)
```

```
## [1] 50983.37
```

For comparison, lets look at the linear model.

```
lm.fit=lm(Whole_Building_Load~., data=train)
lm.pred=predict(lm.fit,test)
mean((lm.pred-ytest)^2)
```

```
## [1] 70844.53
```

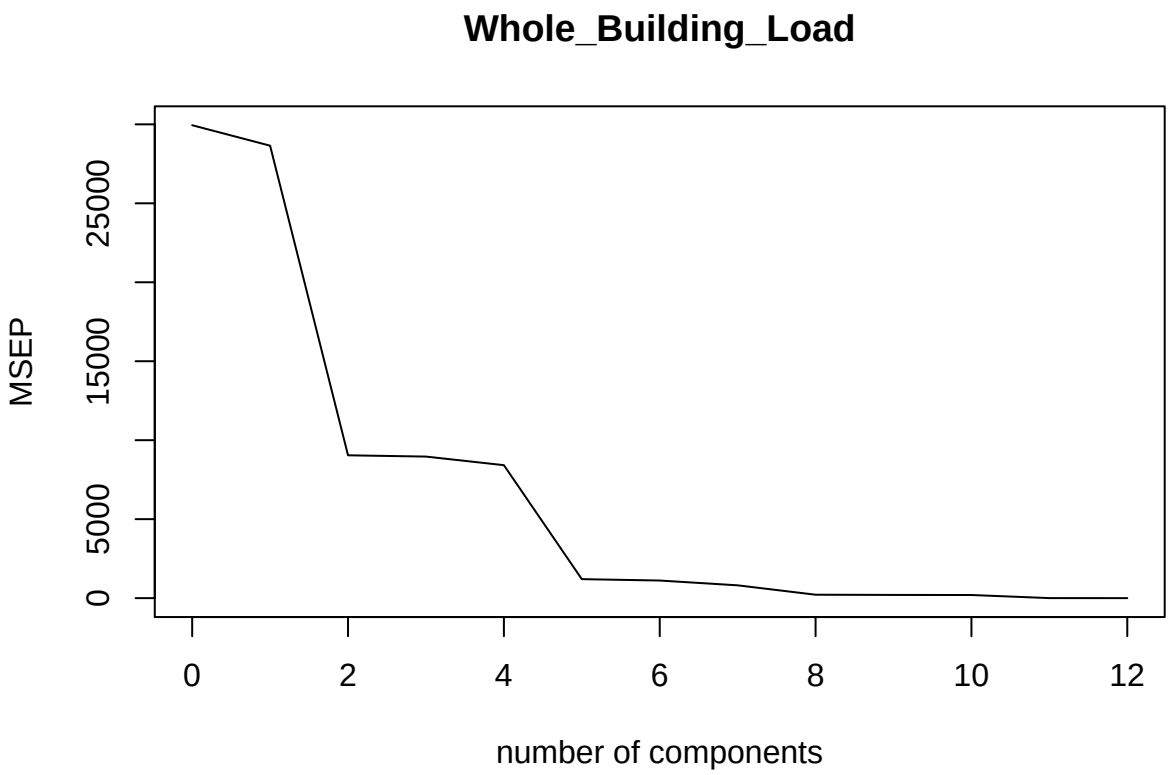



Figure 73: Comparison of the mean square error and the number of components.

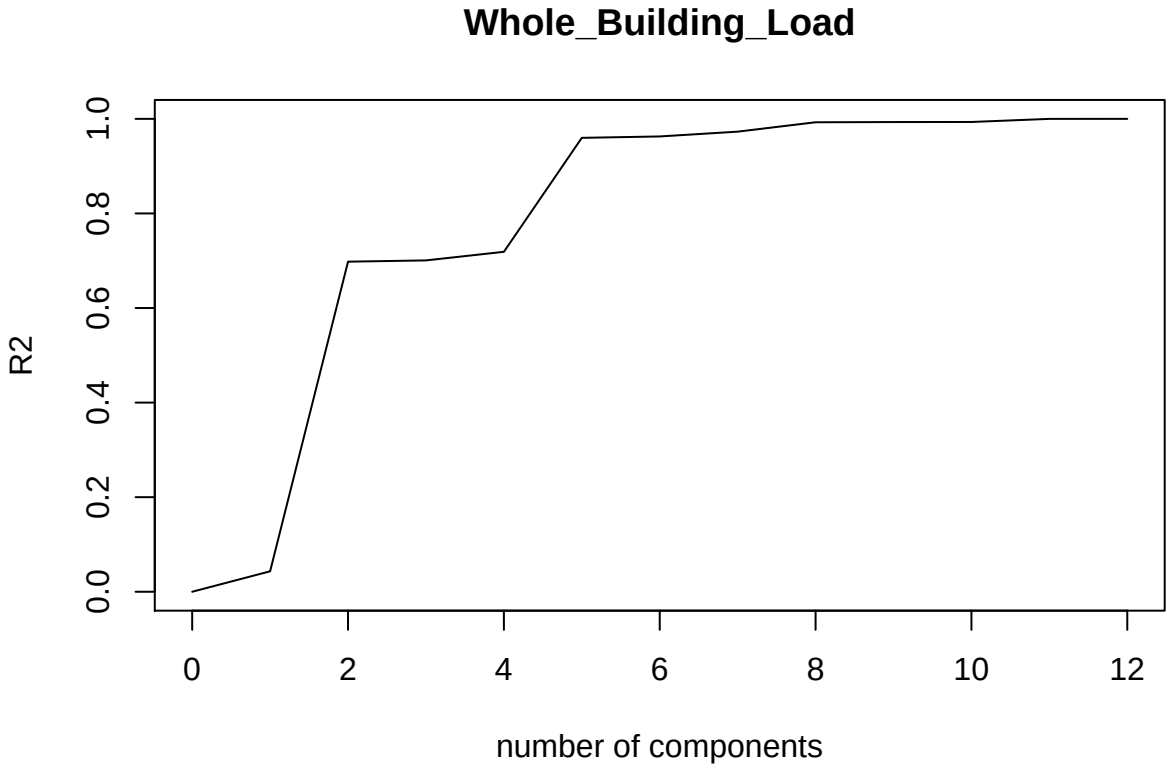



Figure 74: Comparison of the R2 value with the number of components.

The PCR approach is an improvement to the linear model.

What happens if we use select the amount of regressors that create the best linear model and compare it to the same number of PCs? Let's use the stepAIC() function to find the ideal number of regressors.

```
library(MASS)
lm.fit=lm(Whole_Building_Load~., data=data)
k1=2
final.mod <- stepAIC(lm.fit, k=k1)
```

```
## Start:  AIC=-484800.3
## Whole_Building_Load ~ Time + Cooling + Heating + Mechanical +
##     Lighting + Plug_Loads + Data_Center + PV_Generation + Whole_Building_Net +
##     Dry_Bulb_Temp + Irradiance_Global + Irradiance_South_Facade
##
##                            Df Sum of Sq   RSS     AIC
## - Mechanical                1         0     0 -487689
## - Plug_Loads                1         0     0 -485574
## - Data_Center               1         0     0 -485353
## - Lighting                  1         0     0 -485073
## - Irradiance_Global         1         0     0 -484802
## - Irradiance_South_Facade   1         0     0 -484801
## - Dry_Bulb_Temp             1         0     0 -484801
## <none>                                    0 -484800
## - Heating                   1         0     0 -482906
## - Cooling                   1         0     0 -476828
## - Time                      1         0     0 -442332
## - PV_Generation             1     15795 15795    5188
## - Whole_Building_Net        1     15816 15816    5200
##
## Step:  AIC=-487690.1
## Whole_Building_Load ~ Time + Cooling + Heating + Lighting + Plug_Loads +
##     Data_Center + PV_Generation + Whole_Building_Net + Dry_Bulb_Temp +
##     Irradiance_Global + Irradiance_South_Facade
##
##                            Df Sum of Sq    RSS     AIC
## - Heating                   1         0      0 -489059
## - Plug_Loads                1         0      0 -488802
## - Irradiance_Global         1         0      0 -487691
## - Irradiance_South_Facade   1         0      0 -487691
## <none>                                     0 -487690
## - Dry_Bulb_Temp             1         0      0 -487690
## - Lighting                  1         0      0 -487092
## - Cooling                   1         0      0 -481015
## - Data_Center               1         0      0 -480799
## - Time                      1         0      0 -442308
## - PV_Generation             1    281633 281633   30420
## - Whole_Building_Net        1    283607 283607   30481
##
## Step:  AIC=-489007.5
## Whole_Building_Load ~ Time + Cooling + Lighting + Plug_Loads +
##     Data_Center + PV_Generation + Whole_Building_Net + Dry_Bulb_Temp +
##     Irradiance_Global + Irradiance_South_Facade
##
##                            Df Sum of Sq       RSS     AIC
```

```
## - Data_Center               1         0         0 -490066
## - Irradiance_Global         1         0         0 -489009
## - Irradiance_South_Facade   1         0         0 -489009
## - Dry_Bulb_Temp             1         0         0 -489009
## <none>                                          0 -489007
## - Plug_Loads                1         0         0 -483450
## - Lighting                  1         0         0 -482614
## - Cooling                   1         0         0 -476596
## - Time                      1         0         0 -458490
## - PV_Generation             1  45453874  45453874   74948
## - Whole_Building_Net        1 127944151 127944151   84012
##
## Step:  AIC=-490066.1
## Whole_Building_Load ~ Time + Cooling + Lighting + Plug_Loads +
##     PV_Generation + Whole_Building_Net + Dry_Bulb_Temp + Irradiance_Global +
##     Irradiance_South_Facade
##
##                            Df Sum of Sq       RSS     AIC
## <none>                                           0 -490066
## - Irradiance_South_Facade   1         0         0 -489998
## - Irradiance_Global         1         0         0 -489998
## - Dry_Bulb_Temp             1         0         0 -489995
## - Plug_Loads                1         0         0 -489284
## - Lighting                  1         0         0 -486239
## - Cooling                   1         0         0 -486220
## - Time                      1         0         0 -457894
## - PV_Generation             1  45523288  45523288   74959
## - Whole_Building_Net        1 127953313 127953313   84011
```

```
final.mod
```

```
##
## Call:
## lm(formula = Whole_Building_Load ~ Time + Cooling + Lighting +
##     Plug_Loads + PV_Generation + Whole_Building_Net + Dry_Bulb_Temp +
##     Irradiance_Global + Irradiance_South_Facade, data = data)
##
## Coefficients:
##              (Intercept)                      Time                   Cooling
##                3.597e-11                -2.704e-20                -6.284e-14
##                 Lighting                Plug_Loads             PV_Generation
##                -2.597e-15                -4.654e-15                 1.000e+00
##      Whole_Building_Net            Dry_Bulb_Temp         Irradiance_Global
##                 1.000e+00                 2.175e-15                -1.955e-17
## Irradiance_South_Facade
##                -5.939e-18
```

Using stepwise regression we find that the model with the lowest AIC is a model containing 9 out of the original 12 regressors. We can evaluate the model accuracy on the same test set as before.

```
lm.fit1=lm(formula = Whole_Building_Load ~ Time + Cooling + Lighting +
Plug_Loads + PV_Generation + Whole_Building_Net + Dry_Bulb_Temp +
Irradiance_Global + Irradiance_South_Facade, data = train)
lm.pred1=predict(lm.fit1,test)
mean((lm.pred1-ytest)^2)
```

```
## [1] 70844.53
```

The mean square error is the same in this model as it was using all of the regressors. This indicates an excellent fit has been achieved using all of the regressors, and that reducing the number of regressors has done little to improve the prediction accuracy linear model.

We will now use 9 PC's in a PCR approach. Here we have found that even when 9 PCs are used, there is still a improvement in model performance.

```
pcr.fit=pcr(Whole_Building_Load~., data=train, scale=T)

pcr.pred1=predict(pcr.fit,test,ncomp=9)
mean((pcr.pred1-ytest)^2)
```

```
## [1] 66548.79
```

## 4.2 Clustering Methods

The majority of the material and examples provided in this section were adapted from the following sources. Additional sources are listed in the references.

- James, Witten, Hastie, and Tibshirani. (2013). An Introduction to Statistical Learning. New York: Springer

Another unsupervised learning technique is called **clustering**.

Clustering refers to a set of methods for finding subgroups or **clusters** in a data set. The main goal in this method is to partition observations into distinct groups.

Two clustering Methods:

- $K$-means clustering
- hierarchical clustering

### 4.2.1 $K$-Means Clustering

$K$-means clustering is a simple approach that attempts to divide data into distinct, not-overlapping clusters.

Let $C_1, ..., C_K$ be sets contacting the indices of the observations in each cluster. These sets satisfy two properties:

- $C_1 \bigcup C_2 \bigcup ... \bigcup C_k = 1, ..., K$. Each observation belongs to one of $K$ clusters.
- $C_k \bigcap C_{k'} = \emptyset$ for all $k \neq k'$. No observation belongs to more than one cluster.

In $K$-means clustering, we want the variation within the cluster to be as small as possible. This variance, $W(C_k)$, is the measure by which the observations within a cluster differ for each other. Hence we want to solve:

- $\min_{C_1,...,C_K} \left\{ \sum_{k=1}^{K} W(C_k) \right\}$

In other words, we want to partition the observations into $K$ clusters such that the total within-cluster variation, summed over all $K$ clusters, is minimized.

The most common way to define the variance within the cluster involves the squared Euclidean distance:

- $W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$

where $|C_k|$ denotes the number of observations in the kth cluster. This formula says that the variation for the $k$-th cluster is the sum of all of the pairwise squared Euclidean distances between observations in the kth cluster, divided by the total number of observations in the kth cluster.

Combing these two formulas we get:

- $\min_{C_1,...,C_K} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}$

The following is a simple algorithm which provides a local optimum to the $K$-means optimization problem.

- STEP 1: **Randomly** assign a number from 1 to $K$, to each of the observations. This will be the initial cluster assignments.
- STEP 2: Reiterate until the cluster assignments stop changing:
  - For each cluster, compute the centroid. This is the vector of the $p$ feature means for the observations in the $k$-th cluster.
  - Assign each observation to the cluster whose centroid is closest.

As the minimization formula is ran, the variance within the cluster will continue to decrease. When the result no longer changes, a local optimum has been reached.

- **Note** that the results will depend on the initial random clusters assignment of each observation in Step 1 of the algorithm. Therefore, the algorithm should be run multiple times from different random initial configurations. The best (lowest within-cluster variance) should be selected

**Example**

Let's create some data to analyze.

```
set.seed(101)
x=matrix(rnorm(100*2),100,2)
xmean=matrix(rnorm(8,sd=4),4,2)
which=sample(1:4,100,replace=T)
x=x+xmean[which,]
plot(x,col=which,pch=19)
```

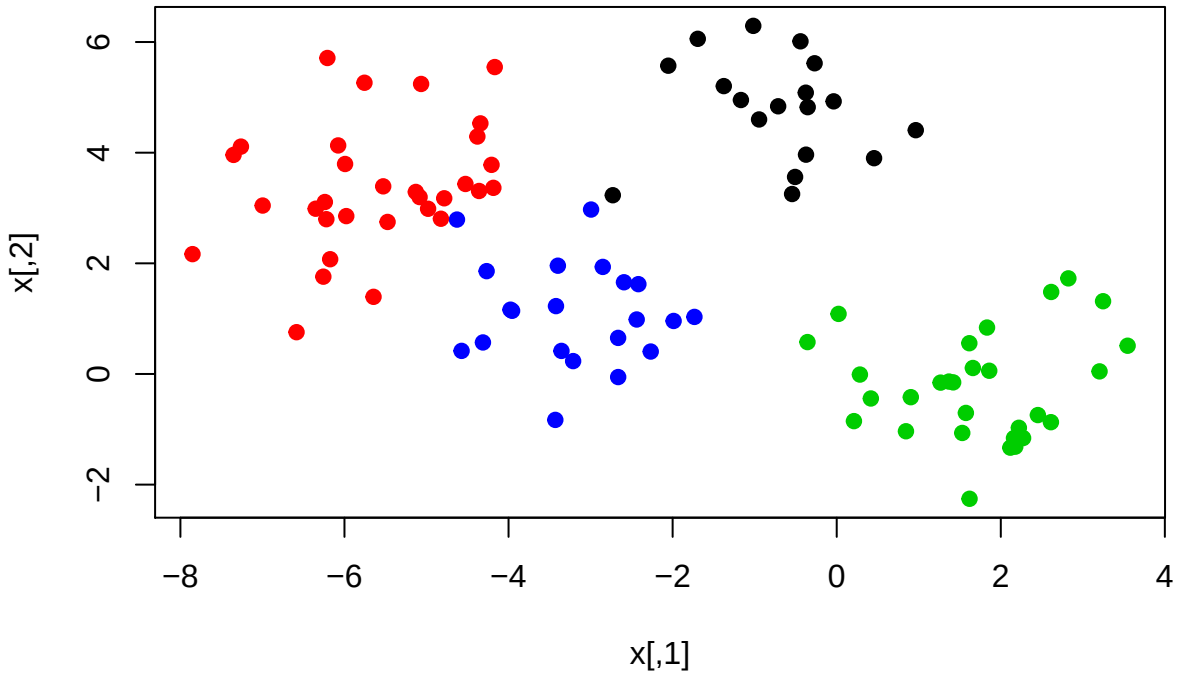



Figure 75: Initial random assignment of data to K means clusters.

The different colors represent the initial cluster assignments.

In order to find the $K$-means clusters we will use the kmeans() function.

```
km.out=kmeans(x,4,nstart=15)
km.out
```

```
## K-means clustering with 4 clusters of sizes 21, 30, 32, 17
##
## Cluster means:
##         [,1]       [,2]
## 1 -3.1068542  1.1213302
## 2  1.7226318 -0.2584919
## 3 -5.5818142  3.3684991
## 4 -0.6148368  4.8861032
```

```
##
## Clustering vector:
##   [1] 2 3 3 4 1 1 4 3 2 3 2 1 1 3 1 1 2 3 3 2 2 3 1 3 1 1 2 2 3 1 1 4 3 1 3
##  [36] 3 1 2 2 3 2 2 3 3 1 3 1 3 4 2 1 2 2 4 3 3 2 2 3 2 1 2 3 4 2 4 3 4 4 2
##  [71] 2 4 3 2 3 4 4 2 2 1 2 4 4 3 3 2 3 3 1 2 3 2 4 4 4 2 3 3 1 1
##
## Within cluster sum of squares by cluster:
## [1] 30.82790 54.48008 71.98228 21.04952
##  (between_SS / total_SS =  87.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

The "between_SS/ total_SS" output is measure of fit. It is similar to that of the $R^2$ value for linear regression in that values closer to 1 indicate a good fit. Between refers to the deviance between clusters ($\sum(y_i - y_i)^2$), and total refers to the sum of the deviance between and within the clusters. In general, the goal is to obtain a model that yields segregation from cluster to cluster and small point to point distances within clusters. In this case, the between_SS/total_SS values is acceptable at 87.6%.

```
plot(x,col=km.out$cluster,cex=2,pch=1,lwd=2)
points(x,col=which,pch=19)
points(x,col=c(4,3,2,1)[which],pch=19)
```

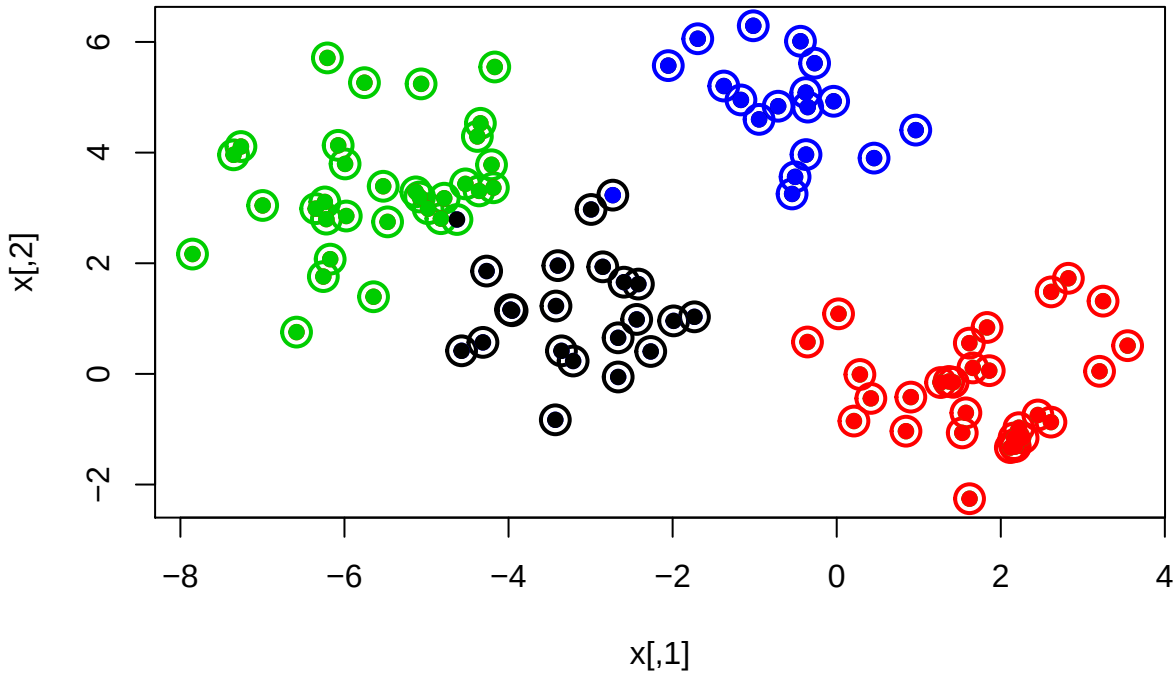



Figure 76: The final K means cluster assignments are represented by the circles. The function in R has reassigned some observations as appropriate.

The outer rings is the fitted $K$-means clustering. The inner dots are the original assignments. You can see that some of them have changed clusters.

### 4.2.2 Hierarchical Clustering

Unlike $K$-means clustering, hierarchical clustering does not require the number of clusters, $K$, to be initially specified. This method also has the advantage of producing a **dendrogram**, a tree based representation of the observations making it easy to interpret.

We will discuss *bottom-up* clustering. This is one of the most common type of hierarchical clustering and refers to starting with individual observations and taking a combinatorial approach.

Hierarchical clustering is particularly useful when the underlying physical data is known to have nested, tree-like relationships.

**Interpreting Dendrograms**

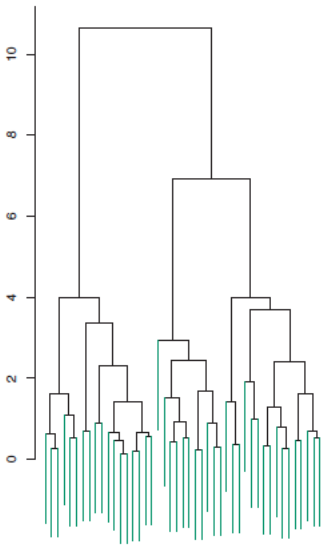



Figure 77: An example of a dendrogram. (Figure 10.9, James, Witten, Hastie, & Harlow, 2013)

Each green lines or 'leaf' of the dendrogram represents an individual observation. As we move up the tree, some of the leaves begin to combine together to make a new branch. These represent the combination of two observations that are most similar to each other. The earlier the combination occurs, the more similar the groups of observations are to each other. For any two observations, we can look at the point in the tree that they first combine. The height, as measured by the vertical axis, indicates how different or similar the observations are. Note that we cannot assume anything about the similarity of two observations based on their proximity along the horizontal axis.

Lets now look at how to identify clusters on the basis of a dendrogram.

position of horizontal cuts form the clusters.

- cutting at height of 9 gives 2 clusters
- cutting at height of 5 gives 3 clusters
- any number of clusters can be achieved between 1 and the number of observations, n.

**Ward's Algorithm**

STEP 1: Define dissimilarity measure between each pair or observations, most often Euclidean distance, initially treating all observations as individual clusters.

STEP 2:

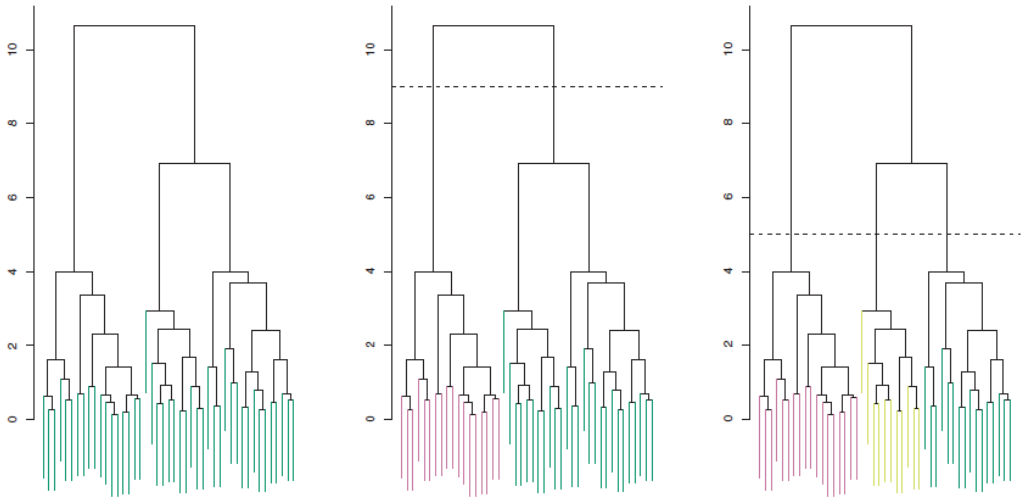



**FIGURE 10.9.** Left: *dendrogram obtained from hierarchically clustering the data from Figure 10.8 with complete linkage and Euclidean distance.* Center: *the dendrogram from the left-hand panel, cut at a height of nine (indicated by the dashed line). This cut results in two distinct clusters, shown in different colors.* Right: *the dendrogram from the left-hand panel, now cut at a height of five. This cut results in three distinct clusters, shown in different colors. Note that the colors were not used in clustering, but are simply used for display purposes in this figure.*

Figure 78: An example of a dendrograms cut at different levels to produce different numbers of clusters. (James, Witten, Hastie, & Harlow, 2013)

- Choose the pair of clusters that is the most similar and combine them. The degree of dissimilarity between these two clusters indicates that the height at which the two observations should fuse.
- Compute the new pairwise inter-cluster dissimilarities among the $i - 1$ remaining clusters.

**Example**

When there are more than one observations combined in a cluster, the concept of dissimilarities needs to be extended to a pair of groups of observations. This concept is know as *linkage* and describes the method for which the clusters are made. The most common of these is Ward's method. This method aims to find compact spherical clusters. This can be specified in R using method argument.

```
hc.ward=hclust(dist(x),method="ward.D")
plot(hc.ward,hang=-1, cex=0.6)
```

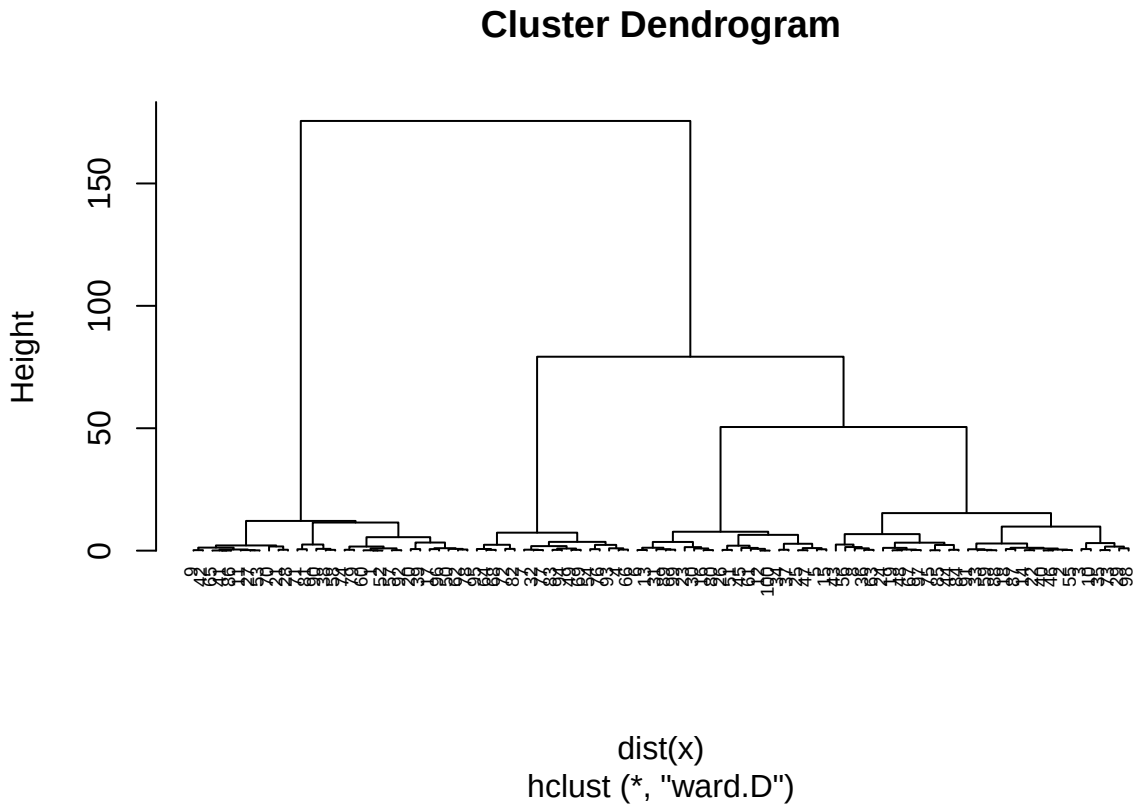



Figure 79: A dendrogram created using Ward's method.

Below is a table of other common linkage methods in hierarchical clustering and how their approach to choosing which observations go into which cluster.

**Example**

The resulting dendrogram typically depends quite strongly on the type of linkage used. For example, compare the dendrogram using Ward's method with the dendrogram using the complete method.

```
hc.complete=hclust(dist(x),method="complete")
plot(hc.complete, hang=-1, cex=0.6)
```

Using the Ward's method based model, lets cut the model to make 4 and 3 clusters.

```
clusterCut <- cutree(hc.ward, 4)
clusterCut1<- cutree(hc.ward, 3)
```

| *Linkage* | *Description* |
|---|---|
| Complete | Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities. |
| Single | Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *smallest* of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time. |
| Average | Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *average* of these dissimilarities. |
| Centroid | Dissimilarity between the centroid for cluster A (a mean vector of length $p$) and the centroid for cluster B. Centroid linkage can result in undesirable *inversions*. |

**TABLE 10.2.** *A summary of the four most commonly-used types of linkage in hierarchical clustering.*

Figure 80: Several linkage types and their descriptions. (Table 10.2 James, Witten, Hastie, & Harlow, 2013)

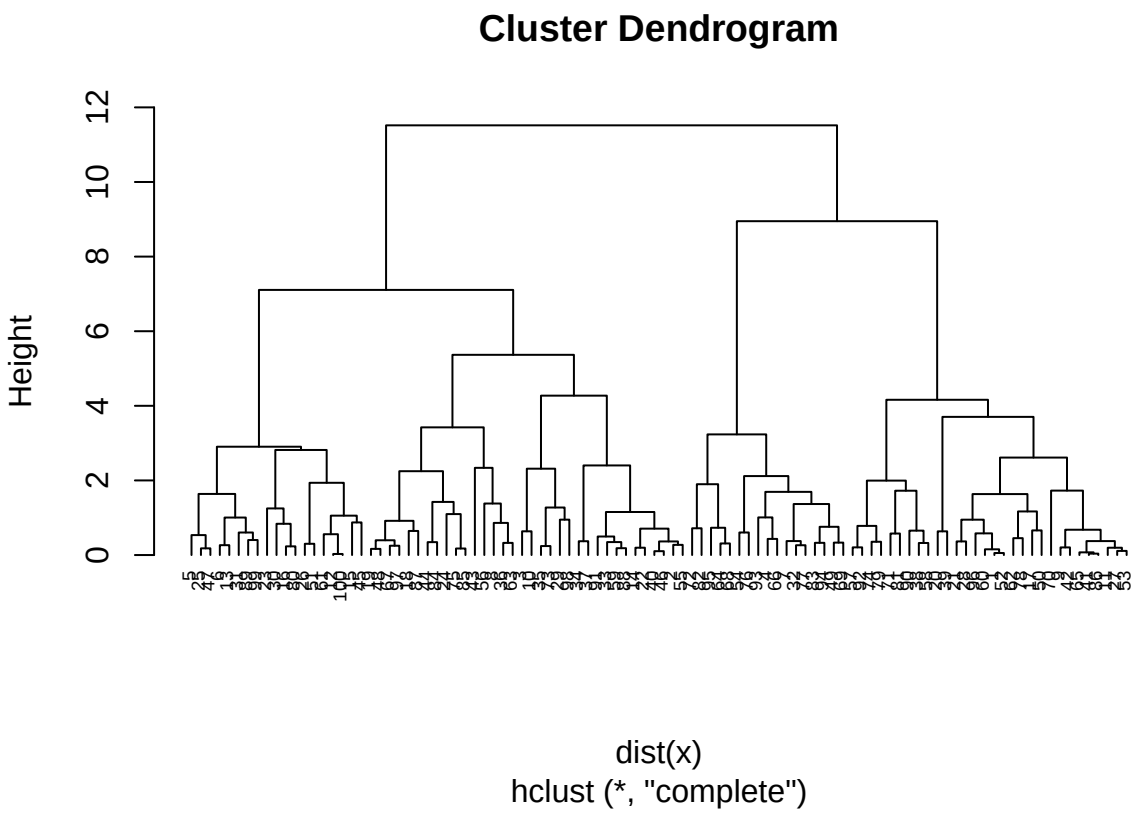



Figure 81: The same dendrogram as in Figure 79 but created using the complete method.

```
plot(hc.ward, hang=-1, cex=0.6)
abline(h = 40, lty = 2, col="red")
abline(h=13, lty=2,col="blue")
```

**Practical Issues in Clustering**

1. In clustering, there are many decision that need to be made that can greatly affect the results. The decisions include:
   - Should features be standardized? (Centered to have zero mean and scaled to have unity standard deviation)
   - In hierarchical clustering we need to decide:
     - Which dissimilarity measure should be used?
     - What type of linkage to use?
     - Where the dendrogram should be cut/how many cluster?
   - In $K$-means clustering: How many clusters should select?
2. Validating Clusters It needs to be determined if the clusters truly represent subgroups. There are several techniques to check this but no consensus on a single best approach.
3. Sometimes not appropriate to cluster - outliers

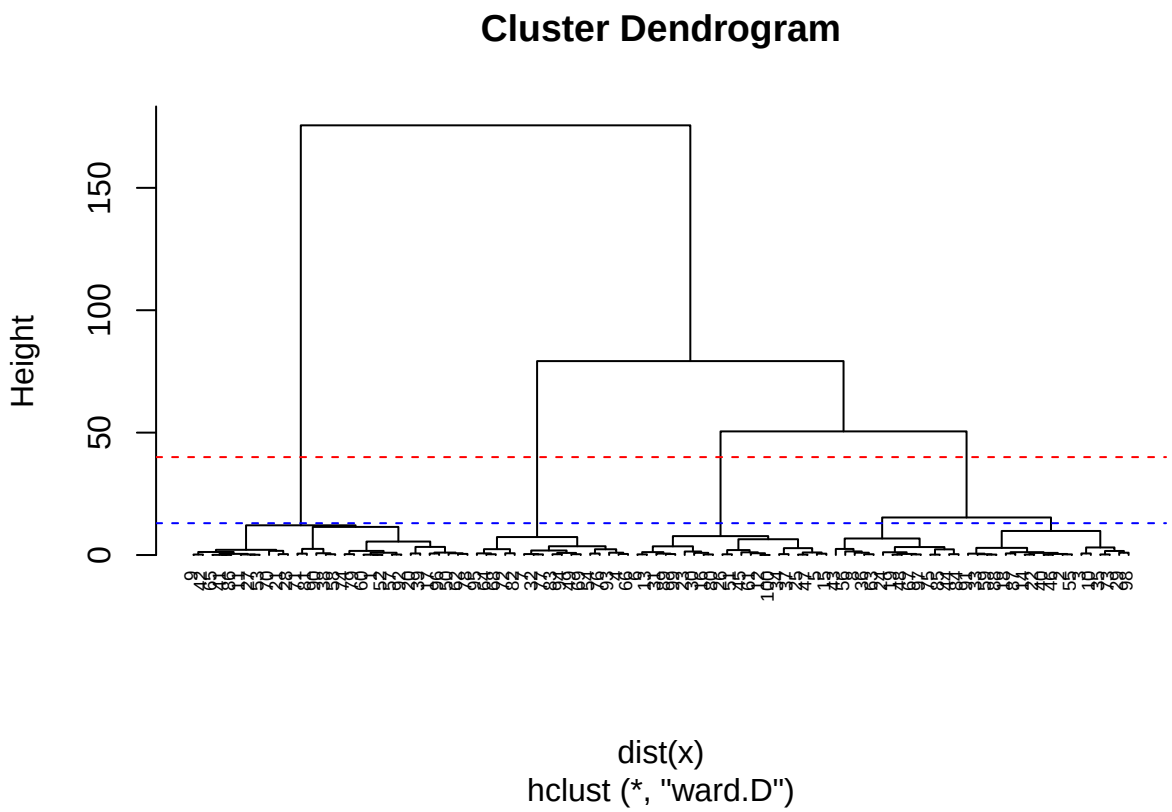



Figure 82: The same tree use in Figure 79 cut at different levels to make four and 5 distinct clusters.

# 5 Tree Based Models

The majority of the material and examples provided in this section were adapted from the following sources. Additional sources are listed in the references.

- James, Witten, Hastie, and Tibshirani. (2013). An Introduction to Statistical Learning. New York: Springer

*By the completion of this section, the reader should be able to*

- *understand CART and its benefits and limitations*
- *understand when to use classification and regression trees for given data*
- *know the benefits of pruning, bagging, and random forests on the prediction accuracy of tree based models*
- *know how to preform tree based model approaches in R effectively.*

## 5.1 Decision Trees - CART

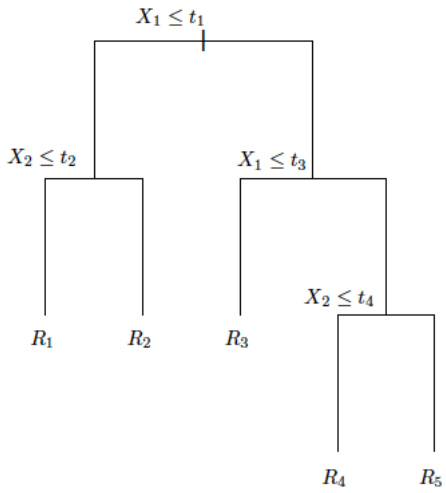
Classification and regression trees (CART) are non-parametric decision tree techniques that can be applied to either classification or regression problems.

- **Classification Trees**: The response variable is *qualitative or categorical* and the tree is used to identify the class within which a response variable would likely fall into.
- **Regression Trees**: The response variable is *numerical and continuous* and the tree is used to predict its value.

CART is a good method for complex data involving nonlinear relationships, high-order interactions, and missing values in the response or the regressor.

**Useful Terminology:**

- Creating decision trees involves segmenting the predictor space into smaller **regions**; $R_1, R_2, R_3$.
- The points along the tree where the predictor space is split are referred to as **internal nodes**.
- The label at the internal nodes is $X_j < t_k$, where $t_k$ are the **splitting condition** values and $X_j$ are the **parameters**.
- The **terminal nodes** are the "leaves" of the tree. These nodes describe the partition of the predictor space.




Most trees are binary decision trees i.e. at each node the tree splits into two branches (although this is not a requirement). Each branch of the tree ends in a terminal node while each observation falls into exactly one terminal node.

The trees are formed by an exhaustive search performed at each node to determine the best split.

- At a given internal node, the label indicates a move to the left-hand branch emanating from that split and the right-hand branch corresponds to the opposite, i.e. $X_j \geq t_k$.
- Each group is characterized by a typical value of the response variable, the number of observations in the group, and the values of the explanatory variables that define it.
- The computation stops when any further split does not improve the classification.

In order to grow a decision tree, the following must be completed:

i) choosing the splitting attributes - the set of important variables to preform splitting
ii) ordering the splitting attributes - ranking them by order of importance in terms of being able to explain the variation in the dependent variable.
iii) deciding on the number of splits of the splitting attributes, which is dictated by the domain or range of variation of that particular attribute.
iv) defining the tree structure i.e. the number of nodes and branches
v) selecting stopping criteria which are a set of pre-defined rules meant to reveal that no further gain is being made in the model that involve a trade off between accuracy of classification
vi) pruning a tree which involves making modifications to the tree constructed using the training data so that it applies well to the testing data.

While CART is a computer intensive method that determines the best tree size and configuration in multivariate data, it is important to understand the process that goes into building these trees.

### 5.1.1 Regression Trees - The Tree Building Process

We first want to **divide the predictor space**, (the set of possible values for $X_1, X_2, X_3$), into $J$ distinct and non-overlapping regions, $R_1, R_2, ...R_J$.

- To do this, we use **recursive binary splitting, also known as the top down greedy approach**: Starting at top with all the data in the data set, the observations are split into two pieces one at a time at each level (hence the name top down). At each step, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step (hence the name greedy).
- Select the predictor $X_i$ and the cut point such that splitting the predictor space into two regions leads to the greatest reduction in the residual sum of squares, a measure of error between the true values of $y_i$, and the predicted values $\hat{y}_i$) ($RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)$). In other words, we want to minimize the total variation of the observations around the mean in each region.
- Repeat this process, looking for best predictor and best cut point in order to split the data further so as to minimize the RSS within each of the resulting regions. Instead of splitting the entire predictor space, we split one of the two previously identified regions, and so on and so forth
- The process continues until a stopping criterion is reached. For example, we may continue until no region contains more than five observations.

For every observation that falls into the region $R_J$, we make the same prediction, which is simply the mean of the response values for the training observations in $R_J$

### 5.1.2 Classification Trees

The method of building classification trees is very similar to that of regression trees. Again we use recursive binary splitting to grow the tree. However, we cannot use RSS here as a criterion for making the binary splits

Instead we use two measures which determine the variance or error in the different classes. Either can be used to gain similar results:

- **Gini index** is a measure of total variance across the $J$ classes. $G = \sum_{k=1}^{K}\hat{p}_{mk}(1 - \hat{p}_{mk})$

where $\hat{p}_m k$ is the proportion of training obervatiosn in the mth regoin that are from the kth class.

The Gini index takes on a small value if all of the $\hat{p_{mk}}$'s are close to zero or one. For this reason, the Gini index is referred to as a measure of node purity - a small value indicates that a node contains predominantly observations from a single class.

- **Cross entropy** is another measure of classification error and behaves very numerically similarly to the Gini index. It is given by

$$D = -\sum_{k=1}^{K}\hat{p}_{mk}log\hat{p}_{mk}$$

### 5.1.3 Pros and Cons of CART

**Pros**:

- Can be easily interpreted and understood.
- Intuitive - trees more closely mirror human decision-making than other regression and classification methods
- Is graphically displayed
- Can handle qualitative predictors without the need to create dummy variables

- While being a fully automatic method, it is **flexible, powerful, and parsimonious**.

**Cons**:

- Risk of over-fitting - **cross validation set** is essential.
- Not competitive with the best supervised learning approaches in terms of prediction accuracy
  - Note: As we will see, this can be made better by aggregating many decision tress together with methods such as bagging and random forest.

**Classification Tree Example - Chiller Fault Detection**

For this example we will use the chiller fault detection data that we used in the GLM example in Section 3. As you may recall, this data set has 4 variables. The fault-free data is designated as a class value of 0 while faulty data is represented by a 1. $T_{cd,sub}$ represents the amount of refrigerant sub-cooling in the condenser in degrees Celsius, $T_{cd,app}$ condenser approach temperature in degrees Celsius. Lastly the COP, or the coefficient of performance of each is listed.

Again, we need to bring in the data (Reddy, 2011). We can plot it in 3D to get a feel for possible regions.

```
library(scatterplot3d)
setwd("~/Dropbox/DSEB/5-Tree Based Models")
data = read.csv("CondenserData.csv")

scatterplot3d(data$COP, y=data$Tcd.sub, z=data$Tcd.app, box=FALSE, xlab="COP", ylab="Tcd_sub", zlab="Tc
```

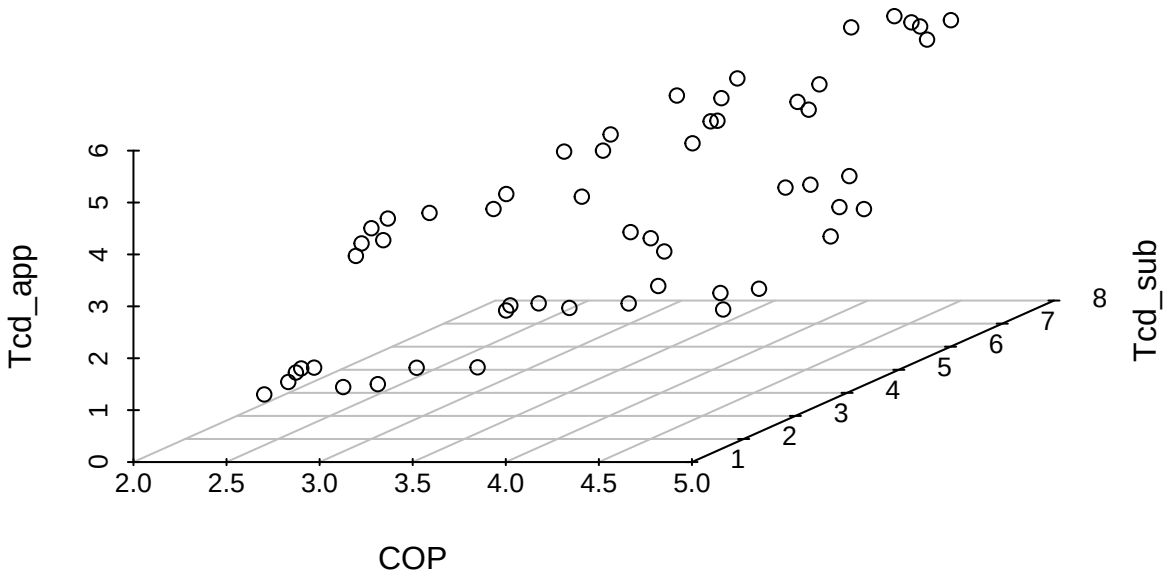



Figure 83: A 3D scatter plot of the condenser data

For this classification problem we use the ifelse() function to create a variable called Class1. This variable will take on a value of "fault"" if Class=0, and "nofault"" if Class=1.

```
library(tree)
Class1=ifelse(data$Class<=0,"nofault", "fault")
Class1
```

```
##  [1] "nofault" "nofault" "nofault" "nofault" "nofault" "nofault" "nofault"
##  [8] "nofault" "nofault" "nofault" "nofault" "nofault" "nofault" "nofault"
## [15] "nofault" "nofault" "nofault" "nofault" "nofault" "nofault" "nofault"
## [22] "fault"   "fault"   "fault"   "fault"   "fault"   "fault"   "fault"
## [29] "fault"   "fault"   "fault"   "fault"   "fault"   "fault"   "fault"
```

```
## [36] "fault"   "fault"   "fault"   "fault"   "fault"   "fault"   "fault"
## [43] "nofault" "nofault" "nofault" "nofault" "nofault" "nofault" "fault"
## [50] "fault"   "fault"   "fault"   "fault"   "fault"
```

We have successfully transformed the data to indicate a faulty and not fault chillers. We then need use the data.frame() function to merge Class1 with the rest of the data set.

```
data.new=data.frame(data,Class1)
data.new
```

```
##    Class   COP Tcd.sub Tcd.app  Class1
## 1      0 3.765   4.911   2.319 nofault
## 2      0 3.405   3.778   1.822 nofault
## 3      0 2.425   2.611   1.009 nofault
## 4      0 4.512   5.800   3.376 nofault
## 5      0 4.748   4.589   2.752 nofault
## 6      0 4.513   3.356   1.892 nofault
## 7      0 3.503   2.244   1.272 nofault
## 8      0 3.593   4.878   2.706 nofault
## 9      0 3.252   3.700   1.720 nofault
## 10     0 2.463   2.578   1.102 nofault
## 11     0 4.274   5.422   3.323 nofault
## 12     0 4.684   4.989   3.140 nofault
## 13     0 4.641   3.589   2.188 nofault
## 14     0 3.038   1.989   1.061 nofault
## 15     0 3.763   4.656   2.687 nofault
## 16     0 3.342   3.456   1.926 nofault
## 17     0 2.526   2.600   1.108 nofault
## 18     0 4.411   5.411   3.383 nofault
## 19     0 4.029   3.844   2.128 nofault
## 20     0 4.443   3.556   2.121 nofault
## 21     0 3.151   2.333   1.224 nofault
## 22     1 3.587   6.656   4.497   fault
## 23     1 3.198   5.767   3.881   fault
## 24     1 2.416   4.333   2.793   fault
## 25     1 2.414   3.811   2.722   fault
## 26     1 4.525   7.256   5.359   fault
## 27     1 4.232   6.022   4.557   fault
## 28     1 3.424   4.544   3.538   fault
## 29     1 3.382   6.533   4.602   fault
## 30     1 3.017   5.667   3.907   fault
## 31     1 3.730   5.933   4.372   fault
## 32     1 2.395   3.989   2.884   fault
## 33     1 4.460   7.356   5.567   fault
## 34     1 4.166   6.044   4.697   fault
## 35     1 2.974   4.456   3.339   fault
## 36     1 3.568   7.033   4.710   fault
## 37     1 3.162   6.044   4.070   fault
## 38     1 2.382   4.544   3.116   fault
## 39     1 4.263   7.567   5.672   fault
## 40     1 3.757   5.967   4.368   fault
## 41     1 4.132   6.589   4.793   fault
## 42     1 2.944   4.811   3.470   fault
## 43     0 3.947   3.567   1.914 nofault
## 44     0 2.434   1.967   0.873 nofault
```

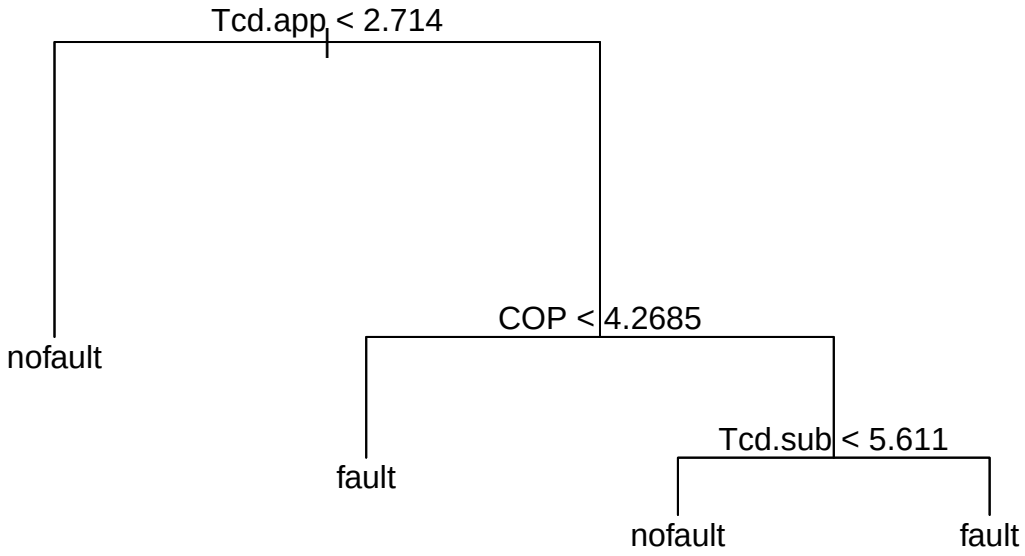



Figure 84: A tree describing important variables in predicting faulty or not faulty condensors

```
## 45     0 3.678   3.389   1.907 nofault
## 46     0 2.517   2.133   1.039 nofault
## 47     0 2.815   2.122   0.946 nofault
## 48     0 4.785   5.100   3.052 nofault
## 49     1 4.330   7.656   5.513   fault
## 50     1 3.716   5.633   4.082   fault
## 51     1 2.309   4.489   2.954   fault
## 52     1 4.059   7.467   5.501   fault
## 53     1 2.615   4.511   3.239   fault
## 54     1 4.539   7.667   5.550   fault
```

You can see that Class1 has been added to the data set data.new as the fifth column.

Now we can use the tree() function to fit a classification tree to this data. With this tree we hope to predict the value of Class1 (fault or nofault) using all variables. Note that the original variable Class must be deleted from the analysis since we are using already using it as the regressor, Class1.

```
tree.data=tree(Class1~.-Class,data.new)
summary(tree.data)
```

```
##
## Classification tree:
## tree(formula = Class1 ~ . - Class, data = data.new)
## Number of terminal nodes:  4
## Residual mean deviance:  0.1001 = 5.004 / 50
## Misclassification error rate: 0.01852 = 1 / 54
```

The summary() function for decision trees gives us the number of terminal nodes, the residual mean deviance, and the misclassification error. Our tree has 4 terminal nodes, a residual mean deviance of 0.1001 and a misclassification error rate of 0.01852.

The trees can be viewed visually using the plot function.

```
plot(tree.data)
text(tree.data,pretty=0)
```

Let's test the prediction accuracy of the tree by creating a training and test set. Our training set will include

42 of the 54 observations.

```
set.seed(2)
train=sample(1:nrow(data.new),42)
test=data.new[-train,]
tree.data=tree(Class1~.-Class,data.new,subset=train)
plot(tree.data);text(tree.data,pretty=0)
```

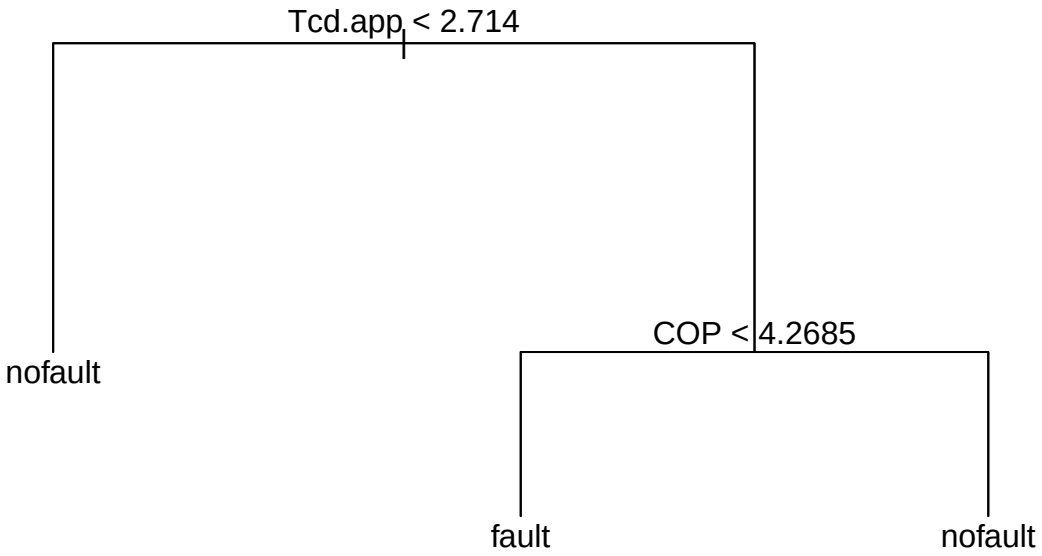



Figure 85: The condensor data desicion tree training set.

After removing the test set, there are now three terminal nodes and $T_{cd,sub}$ is no longer used.

We can look at the misclassification rate of this model with the test set.

```
tree.pred=predict(tree.data,data.new[-train,],type="class")
with(data.new[-train,], table(tree.pred,Class1))
```

```
##          Class1
## tree.pred fault nofault
##   fault       7       0
##   nofault     1       4
```

```
(7+4)/12
```

```
## [1] 0.9166667
```

The tree correctly predicts whether the chiller is faulty or not for 92% of locations in the data set.

## 5.2 Pruning a Decision Tree

### Issues with CART

Once a tree is built, we can predict the test observations, obeying each split, and ending up in a terminal node. From this terminal node, we use the mean of training observations in that region to make a prediction on the observations. However, CART methods may likely overfit a model. For example, if you grow a large tree where each observation has its own terminal node, there will be a training error of zero but you will have poor test set prediction performance and high variance.

Smaller trees with fewer splits and regions might lead to lower variance and better interpretation, at the cost of a little bias. As previously discussed, one possible method to prevent overfitting is to grow the tree only so long as the decrease in the RSS due to each split exceeds some high threshold. This strategy will result in smaller trees, but is too short-sighted. A seemingly worthless split early on in the tree might be followed by a very good split, that is, a split that leads to a large reduction in RSS later on.

Another way to go about growing a decision tree is to grow a very large tree until a stopping criteria is met. Once this large tree is made, we can **prune** the tree from bottom up to produce a subtree.

Selecting a subtree can be difficult. The main objective is to minimize the test prediction error. Since there can be a large number of possible subtrees, estimating the test error for all of them would be too difficult. To do this, we use **cost complexity pruning**, or weakest link pruning.

- $\sum_{m=1}^{|T|} \sum_{x_i \in R_m}^{10} (y_i - \hat{y}_{R_m})^2 + \alpha|T|$

From this we can observe a sequence of subtress all indexed by a tuning parameter $\alpha$. For each value of $\alpha$ there is a corresponding subtree.

- |T| is the number of terminal nodes in each subtree
- $R_m$ is the region corresponding to the $m-$th terminal node
- $\hat{y}_{R_m}$ represents the mean of the training observations in $R_m$
- The tuning parameter $\alpha$ controls a trade-off between the subtrees complexity and its fit to the training data. When $\alpha = 0$ the subtree equals $T_0$. As we increase $\alpha$, the branches get pruned from the tree.
- We select an optimal $\hat{\alpha}$ using cross-validation. We then return to the full data set and obtain the subtree corresponding to $\alpha$.

Again, although this calculation can be easily carried out in programs like R, it is important to know how it works. The following is a summary on how you might prune a decision tree.

1. Using recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of $\alpha$.
3. Use $K$-fold cross-validation to choose $\alpha$. For each $k = 1, \ldots, K$:
   - repeat steps 1 and 2 on the $\frac{k-1}{K}$th fraction of the training data excluding the kth fold.
   - evaluate the mean squared prediction error on the data in the left-out $k$-th fold, as a function of $\alpha$. Average results and pick $\alpha$ to minimize the average error.
4. Return the subtree from step 2 that corresponds to the chosen value of $\alpha$.

## 5.3 Bagging and Random Forests

As previously discussed, the main faults with decision trees are high variance, i.e. the risk of high prediction error. One way to improve this is to build multiple trees and take the average of the predictions from many trees. This is the basis for bagging and random forests.

### 5.3.1 Bagging

**Bagging** is a general-purpose procedure for reducing the variance of a statistical learning method; we introduce it here because it is particularly useful and frequently used in the context of decision trees.

- Recall that given a set of $n$ independent observations $Z_1, Z_2, ..., Z_n$ each with a variance $\sigma^2$, the variance of the mean $\bar{Z}$ of the observations is given by $\sigma^2/n$. In other words, averaging a set of observations reduces variance.
- Therefore, one way to reduce the variance and increase prediction accuracy would be to take multiple training sets of the data, build separate trees from them, and averaging the resulting predictions. Of course, this is not practical because we generally do not have access to multiple training sets.
- Instead we can take repeated samples from the single training data set (this is known as **bootstrapping**). We then make trees with each of these samples and average each prediction.

The approach for bagging regression trees is as follows:

- Generate $B$ different bootstrapped training data sets.
- Train the model on the $b-$th bootstrapped training set in order to get $\hat{f}^{*b}(x)$, the prediction at a point $x$.
- We then average all the predictions to obtained $\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$

Since for classification trees the values are not not numerical, the approach for classification bagging is slightly different.

- In this case, for the values of each observation we record the class predicted by each of the B tress, and take a **majority vote**. In other words, the overall prediction is the *most commonly occuring class* among the B predictions.

### 5.3.2 Random Forests

**Random Forests** provide an improvement over bagged trees by decorrelating the trees. This reduces the variance even further when we average the predictions. For example, consider the presence of a very strong predictor in the data set. This predictor will be chosen in the top split of many or all of the collection of bagged trees. Random forests are used to mitigate this effect.

- As in bagging, a number of decision trees are built on bootstrapped training samples.
- The difference with random forests is that when building these decision trees, each time a split in a tree is considered, a random selection of $m$ predictors is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those $m$ predictors, instead of all the predictors as in bagging.
- A fresh selection of $m$ predictors is taken at each split, and typically we choose $m = \sqrt{p}$. That is the number of predictors considered at each split and is approximately equal to the square root of the total number of predictors.
- This process forces trees to use different predictors at different times, therefore making the average of the resulting trees less variable and more reliable.

**Example - Regression Tree Analysis with Pruning, Bagging, and Random Forest**

For this example we will use the measured data from the RSF building to predict the total building electricity use (kW). We will use 7 variables to determine this: Total Cooling (kW), Total Heating (kW), Total Mechanical (kW), Total Lighting (kW), Total Plug Loads (kW), Total Data Center (kW) and PV (kW).

As usual we need to first load in the data and create a training set. Once we do this we can fit a tree to training data using the tree() function from the tree package.

```
N = nrow(data5)
index=1:N
NT=round(0.7*N)
trainind = sample(index,NT,replace=FALSE)
testind = setdiff(index,trainind)
test=data5[testind,]
train =data5[trainind,]
tree.rsf=tree(Total.Building..kW.~.,data=train )
summary(tree.rsf)
```

```
##
## Regression tree:
## tree(formula = Total.Building..kW. ~ ., data = train)
## Variables actually used in tree construction:
## [1] "Total.Heating..kW."    "Total.Mechanical..kW."
## Number of terminal nodes:  8
## Residual mean deviance:  1437 = 8800000 / 6123
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -242.000  -17.420   -5.416    0.000   18.580  356.000
```

You can see from the summary that only two (2) variables were used to make the tree. In the context of regression, the deviance is simply the sum of squared error for the tree.

Plot the tree

```
plot(tree.rsf)
text(tree.rsf,pretty=0)
```

This trees shows total heating to be the strongest predictor for the total building energy use.

In order to assess if the tree needs to be pruned, we use the cv.tree() function.

```
cv.rsf=cv.tree(tree.rsf)
plot(cv.rsf$size,cv.rsf$dev,type='b')
```

Figure 87 shows the cross validation deviance. We want this number to be as low as possible. It appears as though the most complex tree is chosen from the cross validation with 8 parameters. Therefore no pruning is necessary.

If we wanted to prune the tree, we would use the prune.tree() function. you can see below in figure 87 that if we set the argument best=7, that one internal node is removed.

```
prune.rsf=prune.tree(tree.rsf,best=7)
plot(prune.rsf)
text(prune.rsf,pretty=0)
```

Since our analysis showed that pruning would not be beneficial to our model, we will make predictions on the unpruned tree.

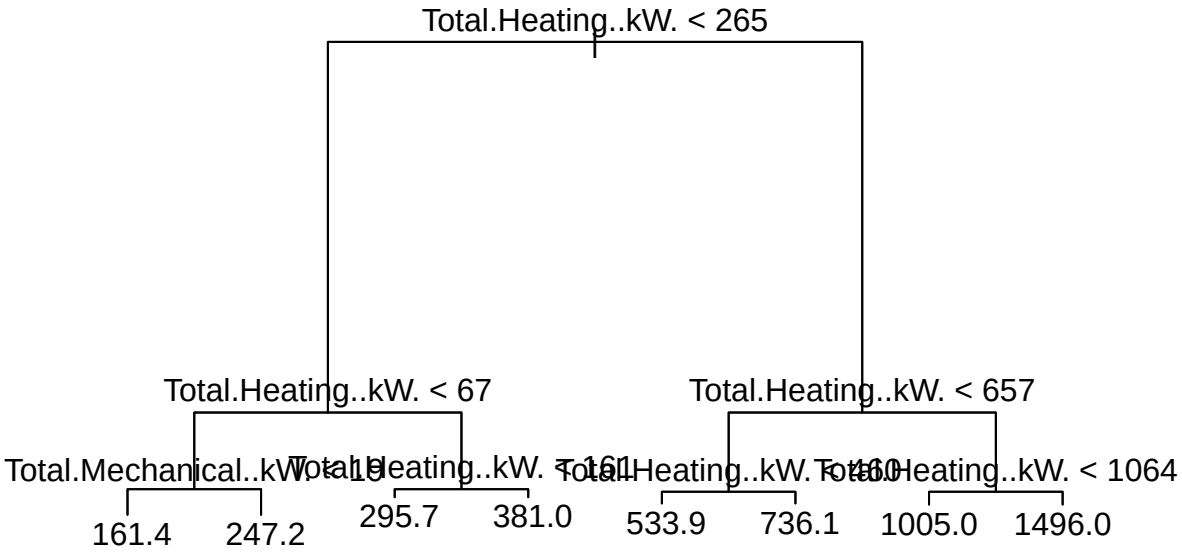



Figure 86: A desicion tree for the rsf2011 dataset. Only total heating energy and total mechanical energy were used to build this tree.

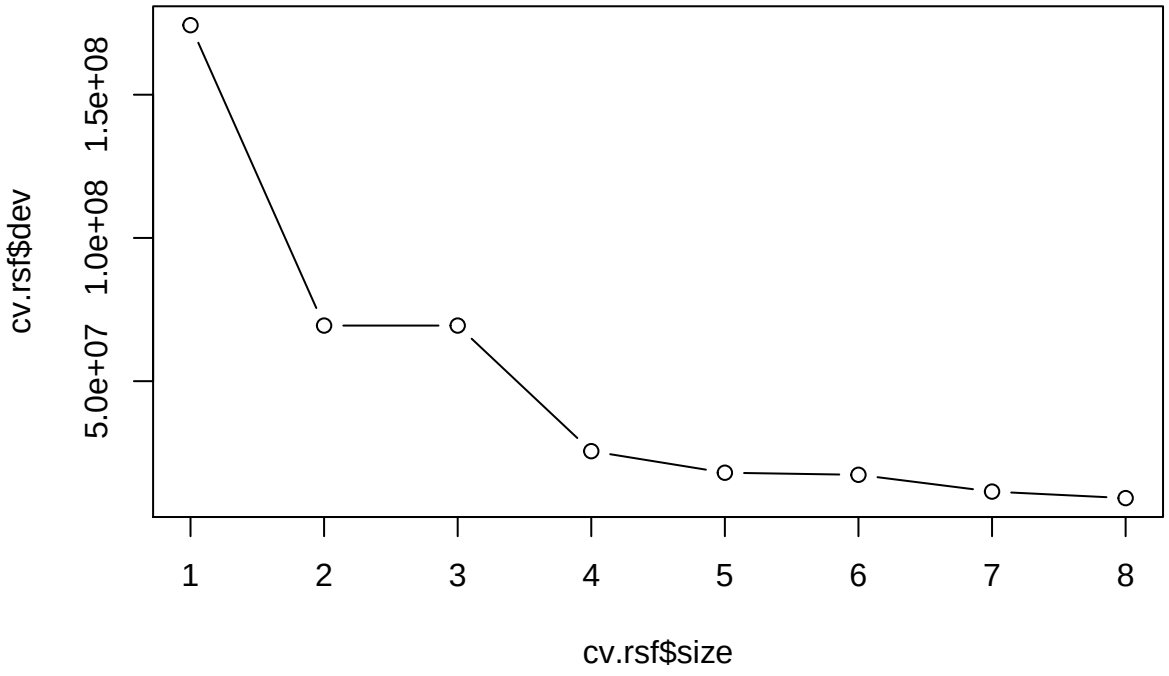



Figure 87: The sum of squared error ploted with the size of the model.

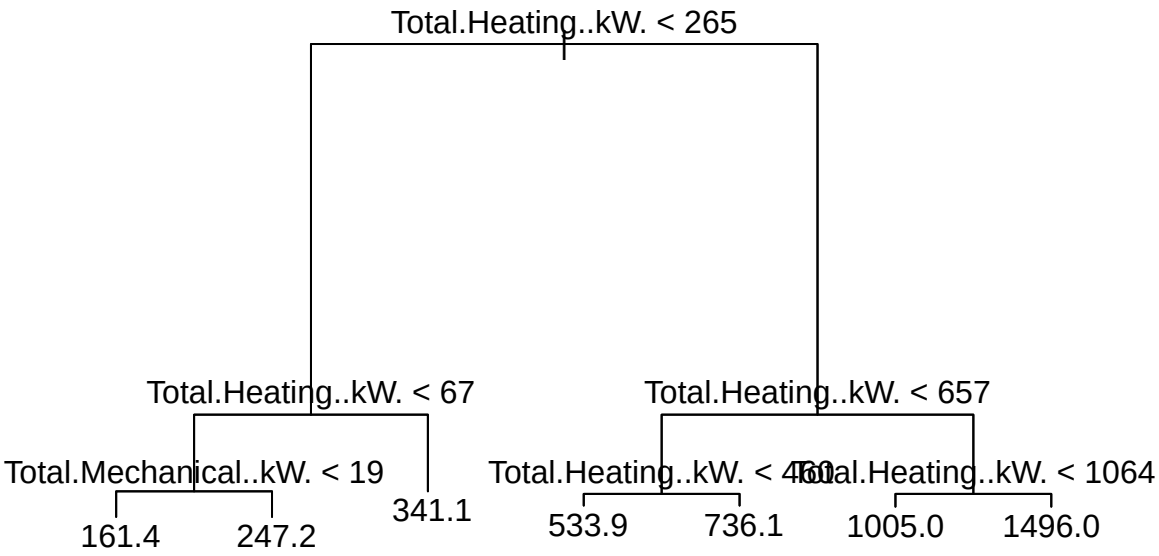



Figure 88: The rsf2011 tree pruned back to 7 terminal nodes.

```
yhat=predict(tree.rsf,test)
rsf.test=test$Total.Building..kW.
plot(yhat,rsf.test)
abline(0,1)
```

```
MSE=mean((yhat-rsf.test)^2)
MSE
```

```
## [1] 1487.449
```

```
RMSE=sqrt(MSE)
RMSE
```

```
## [1] 38.56746
```

In order to apply bagging and random forests to the RSF data set we need to use the randomForest package. Since bagging is a case of random forests, the randomForest() function can be used to perform both bagging and random forests by adjusting the value of $m$, where $m = p$ would represent bagging. This is set using the mtry argument.

**Bagging**: We will start by performing bagging on the model. This is done by setting mtry to the number of parameters, mtyr=7.

```
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:MuMIn':
##
##     importance
```

```
set.seed(1)
bag.rsf = randomForest(Total.Building..kW. ~ ., data = train, mtry = 7, importance = T)
```

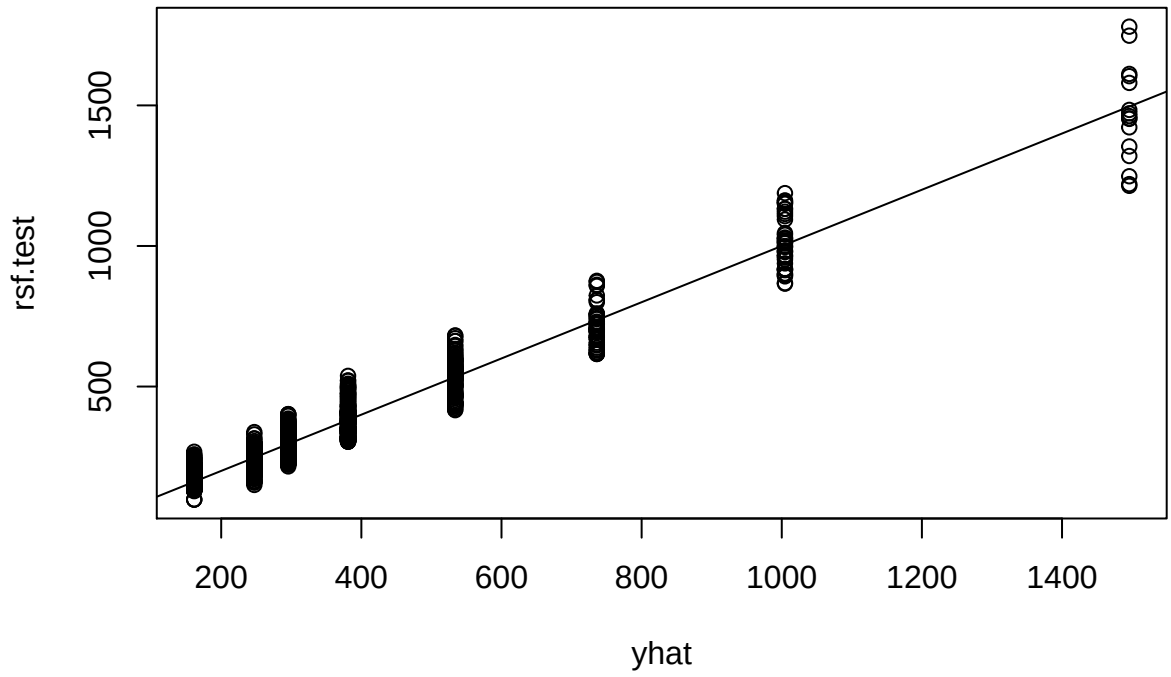



Figure 89: The true, observed values of the test set plotted with the model estimates for these values

```
bag.rsf
```

```
##
## Call:
##  randomForest(formula = Total.Building..kW. ~ ., data = train,      mtry = 7, importance = T)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##           Mean of squared residuals: 36.65616
##                     % Var explained: 99.87
```

We can calculate how well does this model predicts the test data.

```
yhat.bag=predict(bag.rsf,test)
plot(yhat.bag,rsf.test)
abline(0,1)
```

```
MSE=mean((yhat.bag-rsf.test)^2)
MSE
```

```
## [1] 29.99355
```

```
RMSE=sqrt(MSE)
RMSE
```

```
## [1] 5.476637
```

The new MSE and RMSE has been drastically reduced from our original model.

**Random forests**: similarly for random forests we also use the randomForest() function but this time we need to specify a number of parameters $p$. By default, the function used $m = p/3$ for regression models and $\sqrt(p)$ for classification models. In this example we will allow the function to use the default value.

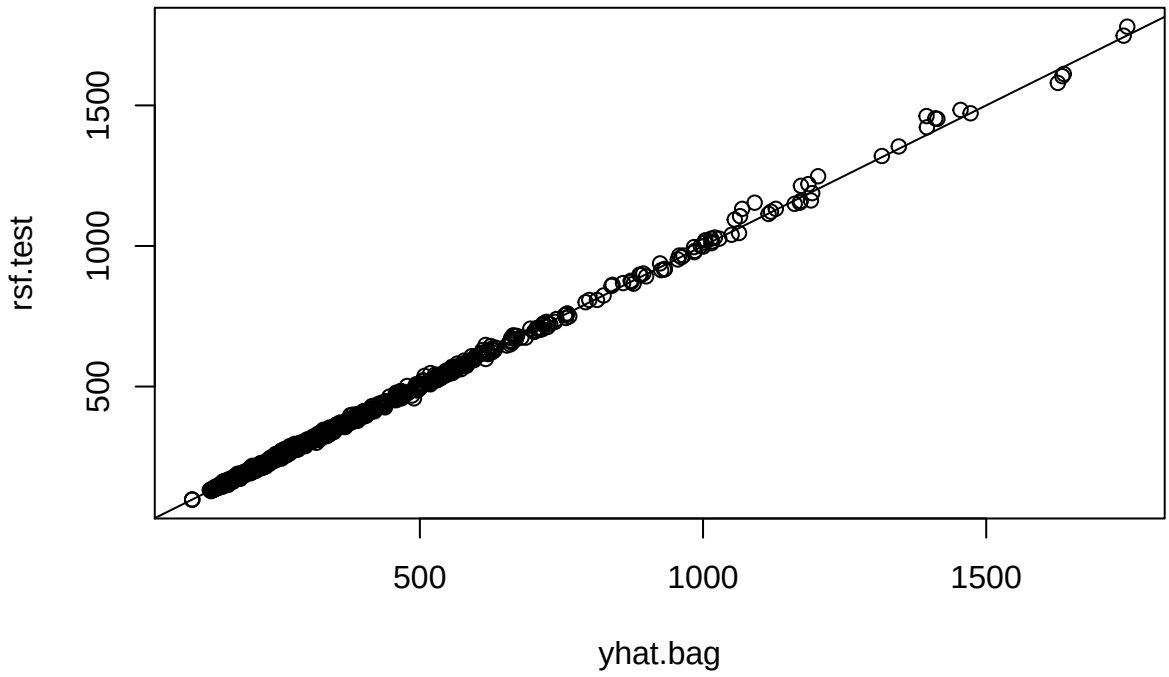



Figure 90: The true, observed values of the test set plotted with the model estimates for these values

```
set.seed(1)
rf.rsf=randomForest(Total.Building..kW.~.,data=train,importance=T)
rf.rsf
```

```
##
## Call:
##  randomForest(formula = Total.Building..kW. ~ ., data = train,      importance = T)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 574.6548
##                     % Var explained: 97.98
```

```
yhat.rf=predict(rf.rsf,test)
MSE=mean((yhat.rf-rsf.test)^2)
MSE
```

```
## [1] 573.857
```

```
RMSE=sqrt(MSE)
RMSE
```

```
## [1] 23.95531
```

In this case, using random forests did not achieve further improvement from bagging. It did however improve from our original tree.

R allows us to view the importance of each variable in both a table and a plot:

```
importance(rf.rsf)
```

```
##                           %IncMSE IncNodePurity
```

```
## Total.Cooling..kW.      15.848680       2955014
## Total.Heating..kW.     107.951705     128195761
## Total.Mechanical..kW.   17.425249      11764130
## Total.Lighting..kW.      9.750733       5575526
## Total.Plug.Loads..kW.   15.487580      10315401
## Total.Data.Center..kW.  20.352093       5961790
## PV..kW.                  8.065513       4648842
```

```
varImpPlot(rf.rsf)
```

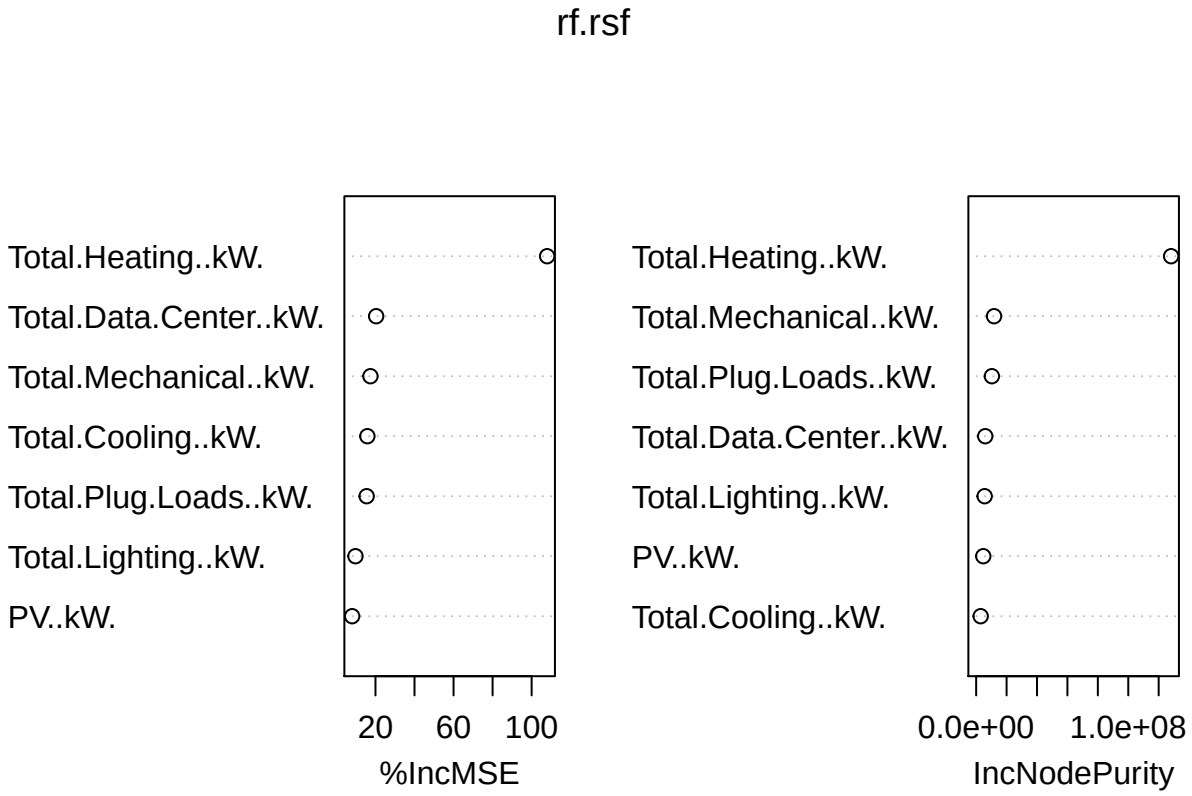



Figure 91: A plot of the important of each variable and the node purity of each variable. Total Heating is shown as the most importnat variable

As expected, Total Heating is by far the most important variable.