# NUMBERS, FUNCTIONS, PERMUTATIONS, GRAPHS

# R is like a scientific calculator

exp(1)         # natural logarithmic base  
log(2)         # logarithm of 2, base e  
pi             # Archimedes' constant

> exp(1)  
[1] 2.718282  
> log(2)  
[1] 0.6931472  
> pi  
[1] 3.141593

getOption("digits")   # single precision display is default but  
# computations always in double precision

options(digits=15)

exp(1)  
log(2)  
pi

> exp(1)  
[1] 2.71828182845905  
> log(2)  
[1] 0.693147180559945  
> pi  
[1] 3.14159265358979

options(digits=7)   # return to default

getOption("scipen")  
options(scipen=-5)   # negative penalty biases toward  
# scientific notation

exp(1)*10^(-17)  
log(2)*10^19  
pi

> exp(1)*10^(-17)  
[1] 2.718282e-17  
> log(2)*10^19  
[1] 6.931472e+18  
> pi  
[1] 3.141593e+00

options(scipen=0)   # return to default
v <- c(exp(1), log(2), pi)  # c means 'concatenate'
v     # vector output
1:12   # a vector with elements 1,2,...,12

> v
[1]  2.71828180 0.6931472 3.1415927
> 1:12
[1]  1  2  3  4  5  6  7  8  9 10 11 12

# how to create a square matrix?
A <- c(1,2,4,2,5,10,0,-1,-1)    # one way
dim(A) <- c(3,3)                # columns are populated in order
A

matrix(c(1,2,4,2,5,10,0,-1,-1),c(3,3))
# another way (done in one line!)

> A
 [,1] [,2] [,3]
[1,]  1  2  0
[2,]  2  5 -1
[3,]  4 10 -1

> matrix(c(1,2,4,2,5,10,0,-1,-1),c(3,3))
 [,1] [,2] [,3]
[1,]  1  2  0
[2,]  2  5 -1
[3,]  4 10 -1

> c(nrow(A),ncol(A))
c(nrow(v),ncol(v))  # a vector is not considered a column matrix
c(NROW(A),NCOL(A))  # NROW same as nrow over matrices;
                   # likewise NCOL same as ncol

> c(NROW(v),NCOL(v))  # NROW and NCOL also work over vectors,
                   # thankfully!

> c(nrow(A),ncol(A))
[1]  3 3
> c(nrow(v),ncol(v))
NULL
> c(NROW(A),NCOL(A))
[1] 3 3
> c(NROW(v),NCOL(v))
[1] 3 1

t(A)       # transpose of a matrix
t(v)       # transpose of a vector yields a row
t(t(v))    # hitting again with t gives a column

> t(A)
 [,1] [,2] [,3]
[1,]  1  2  4
[2,]  2  5 10
[3,]  0 -1 -1
> t(v)
[,1] [,2] [,3] [1,] 2.718282 0.6931472 3.141593
> t(t(v))
   [,1]        [,2]        [,3] [1,] 2.7182818 0.6931472  3.1415927

> t(1:12)
t(t(1:7))

   [2,] 9 10 11 12

B=t(A)
A
B
A%*%B # matrix product
A*B    # products of entries

> A
   [,1] [,2] [,3] [1,] 1  2  0
   [2,] 2  5 -1
   [3,] 4 10 -1
> B
   [,1] [,2] [,3] [1,] 1  2  4
   [2,] 2  5 10
   [3,] 0 -1 -1
> A%*%B
   [,1] [,2] [,3] [1,]  5 12 24
   [2,] 12 30 59
   [3,] 24 59 117
> A*B
   [,1] [,2] [,3] [1,]  1  4  0
   [2,]  4 25 -10
   [3,]  0 -10  1

solve(A) # matrix inverse
A^(-1)   # reciprocals of elements

> solve(A) # matrix inverse
   [,1] [,2] [,3] [1,] 5 2 -2
\[
\begin{bmatrix}
2 & -1 & 1 \\
3 & 0 & -2 & 1 \\
\end{bmatrix}
\]

\[A^{-1}\]  
# reciprocals of elements

\[
\begin{bmatrix}
1,1 & 1.00 & 0.5 & \text{Inf} \\
1,2 & 0.50 & 0.2 & -1 \\
1,3 & 0.25 & 0.1 & -1 \\
\end{bmatrix}
\]

\[\text{solve}(A, \text{diag}(3))\]  
# solve(A) is shorthand for determining X:  
# \[A \times X = \text{diag}(3)\] (the 3x3 identity matrix)

\[\text{matrix}(1,3,3)/A\]  
# another way of finding reciprocals

\[\text{solve}(A, \text{diag}(3))\]

\[
\begin{bmatrix}
1,1 & 5 & 2 & -2 \\
1,2 & -2 & -1 & 1 \\
1,3 & 0 & -2 & 1 \\
\end{bmatrix}
\]

\[\text{matrix}(1,3,3)/A\]

\[
\begin{bmatrix}
1,1 & 1.00 & 0.5 & \text{Inf} \\
1,2 & 0.50 & 0.2 & -1 \\
1,3 & 0.25 & 0.1 & -1 \\
\end{bmatrix}
\]

# a simple one-line function

\[\text{sq} \leftarrow \text{function}(x) \ x^2\]  
# formula to right
\[\text{sq}(-2)\]
\[\text{sq}(3)\]

\[\text{cb} \leftarrow \text{function}(x)\]
\{
\[x^3\]  
# formula inside \{ \}
\}
\[\text{cb}(-2)\]
\[\text{cb}(3)\]

\[\text{sapply}(1:12, \text{sq})\]  
# \text{sq} is evaluated at each integer in the vector  
# (\text{s means 'simplify' - related to \text{lapply}} -  
# \text{sapply does likewise at each variable in a dataframe)}
\[\text{sapply}(-5:5, \text{cb})\]

\[\text{sapply}(1:12, \text{sq})\]

\[\text{[1]}\ 1\ 4\ 9\ 16\ 25\ 36\ 49\ 64\ 81\ 100\ 121\ 144\]

\[\text{sapply}(-5:5, \text{cb})\]

\[\text{[1]}\ -125\ -64\ -27\ -8\ -1\ 0\ 1\ 8\ 27\ 64\ 125\]
sq(1:12)       # this shorthand also works (but not for next example)
cb(-5:5)

> sq(1:12)
[1]   1   4   9  16  25  36  49  64  81 100 121 144
> cb(-5:5)
[1] -125  -64  -27   -8   -1    0    1    8   27   64  125

sq(1:24)       # wrap-around is messy

> sq(1:24)
[1]   1   4   9  16  25  36  49  64  81 100 121 144 169 196 225 256 289 324 361
[20] 400 441 484 529 576

# How can we efficiently generate random objects? (simulation)

sample(10)     # a random permutation on 1:10
sample(10)
sample(10)
sample(10)
sample(10)
sample(10)
sample(10)     # how to do this m times & store in an m-by-10 matrix?

> sample(10)
[1]  5  3  6  8  1  2  9 10  4  7
> sample(10)
[1]  8  3  1  9  5  4 10  6  7  2
> sample(10)
[1]  2  4  8  6  5  3 10  7  9  1
> sample(10)
[1]  3  7  6  4  9  8 10  1  2  5
> sample(10)
[1]  6  3  8  9  7 10  4  5  2  1
> sample(10)
[1]  8  5  7  3  1 10  2  4  9  6
> sample(10)
[1]  5  8  4  1  3  6 10  2  9  7

sapply(1:7,sample)       # doesn't do what we want...

> sapply(1:7,sample)
[[1]]
[1] 1
[[2]]
[1] 1 2
[[3]]
[1] 2 1 3
[[4]]
[1] 4 1 2 3
[[5]]
[1] 5 4 1 2 3
sapply(1:7,sample(10))   # ...nor does this (but error msg is helpful)

t(sapply(1:7, function(o) sample(10)))   # this works!

> sapply(1:7, function(o) sample(10))
Error in match.fun(FUN) : 'sample(10)' is not a function, character or symbol

> t(sapply(1:7, function(o) sample(10)))

# more generally, define:

myrndprm <- function(m,n)
{
  t(sapply(1:m, function(o) sample(n)))  # o is only a dummy variable
}  # t means 'transpose'

myrndprm(7,10)

> myrndprm(7,10)

# Random permutations can be used to approximate the solution
# to the following "matching problem": n letters fall out
# of their envelopes and are replaced at random. What is
# the probability of at least one correctly replaced letter?

# Here is the same, using "for" loops (as in C). In older
# versions of R, "sapply" was substantially faster than # "for". Nowadays, the two approaches take essentially # the same runtime.

```r
myrndprm.alt <- function(m,n)
{
  M <- matrix(0,m,n)   # initially a matrix of zeroes
  for (j in 1:m)
  {
    M[j,] <- sample(n)   # row j of the matrix
  }
  M
}

myrndprm.alt(7,10)
```

```r
> myrndprm.alt(7,10)
[1,]  2  7  1  3  4  5  6  8 10   9
[2,] 10  4  1  8  6  2  3  9  7   5
[3,]  7  2  6  5 10  4  9  8  3   1
[4,]  7  5  1  3  9  6  8  2 10   4
[5,]  7  2  3  9  6  4  8 10  1   5
[6,]  5  7  6  3  9  1  2 10  4   8
[7,] 10  5  2  6  9  3  8  4  1   7
```

```r
time.start <- proc.time( )
Y <- myrndprm.alt(100000,10)
time.used <- proc.time( ) - time.start
cat('User time elapsed:', time.used[1], '
')
```

```r
> User time elapsed: 1.509
```

# When possible, use a "whole-object" approach in R # (avoiding multiple nested "for" loops, whose # entry-by-entry bookkeeping slows everything down)

# We'll see a faster way of generating random permutations # in Matlab (creating a random matrix first, then sorting # & retaining the indices per row via quick built-in).

# Graphing a curve in xy-plane (y is a function of x) is easy!

```r
f <- function(x) 3.5^(-0.5*x)*cos(6*x)
curve(f,from=-2,to=4,col="blue",lwd=2, 
    main="Plot of a Function R -> R", 
    xlab="independent variable", 
    ylab="dependent variable")
```
```r
# Graphing a curve in xyz-space (parametric functions of t)
# can be done using a user-contributed function.
library(scatterplot3d)
t <- seq(0, 6*pi, 0.01)
x <- sin(t)
y <- cos(t)
z <- 5*t
(x11() # keep old plot and open a new plotting window
caret_3d(x, y, z, highlight.3d=TRUE, col.axis="blue",
```)
# Graphing a surface in xyz-space (z is a function of x,y) is
# a little harder.

x <- seq(-2, 2, 0.05)
y <- seq(-2, 2, 0.05)
h <- function(x, y) 100*(y-x^2)^2+(1-x)^2
z <- outer(x, y, h) # outer product, i.e., h applied to each (x,y)
```
x11()
persp(x, y, z, zlim = c(0,1000), theta = 15, phi = 30, expand = 0.5,
    col = c("grey","red"), ticktype = "detailed",
    main = "Plot of a Function R^2 -> R")
```
# How can we solve a single nonlinear equation?
{
  x11()
  curve(x - 2*(1-exp(-x)), from=0, to=2, col="red", lwd=2)
  curve(0*x, from=0, to=2, add=TRUE, col="blue", lty=3)
}

options(digits=15)
k <- function(x) x - 2*(1 - exp(-x))
unroot(k, lower=1.5, upper=1.8, tol=10^(-15))$root
options(digits=7)  # return to default
graphics.off()    # closes all four plots
uniroot(k, lower=1.5, upper=1.8, tol=10^(-15))$root
[1] 1.59362426004004

ls.str()  # listing of variables currently in memory
rm(X,Y)    # delete certain variables
ls.str()

> ls.str()
A :  num [1:3, 1:3] 1 2 4 2 5 10 0 -1 -1
B :  num [1:3, 1:3] 1 2 0 2 5 -1 4 10 -1
cb : function (x)
f : function (x)
h : function (x, y)
k : function (x)
myrndprm : function (m, n)
myrndprm.alt : function (m, n)
sq : function (x)
t :  num [1:1885] 0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 ...
time.start :  num [1:5] 4.35 0.18 225.82 NA NA
time.used :  num [1:5] 2.75 0.00 8.61 NA NA
v :  num [1:3] 2.718 0.693 3.142
x :  num [1:81] -2 -1.95 -1.9 -1.85 -1.8 -1.75 -1.7 -1.65 -1.6 -1.55 ...
X :  int [1:100000, 1:10] 8 4 1 1 6 8 4 3 7 10 ...
y :  num [1:81] -2 -1.95 -1.9 -1.85 -1.8 -1.75 -1.7 -1.65 -1.6 -1.55 ...
Y :  num [1:100000, 1:10] 10 1 9 9 1 4 6 10 9 5 ...
z :  num [1:81, 1:81] 3609 3376 3156 2948 2948 2754 ...
>
> ls.str()
A :  num [1:3, 1:3] 1 2 4 2 5 10 0 -1 -1
B :  num [1:3, 1:3] 1 2 0 2 5 -1 4 10 -1
cb : function (x)
f : function (x)
h : function (x, y)
k : function (x)
myrndprm : function (m, n)
myrndprm.alt : function (m, n)
sq : function (x)
t :  num [1:1885] 0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 ...
time.start :  num [1:5] 4.35 0.18 225.82 NA NA
time.used :  num [1:5] 2.75 0.00 8.61 NA NA
v :  num [1:3] 2.718 0.693 3.142
x :  num [1:81] -2 -1.95 -1.9 -1.85 -1.8 -1.75 -1.7 -1.65 -1.6 -1.55 ...
X :  int [1:100000, 1:10] 8 4 1 1 6 8 4 3 7 10 ...
y :  num [1:81] -2 -1.95 -1.9 -1.85 -1.8 -1.75 -1.7 -1.65 -1.6 -1.55 ...
Y :  num [1:100000, 1:10] 10 1 9 9 1 4 6 10 9 5 ...
z :  num [1:81, 1:81] 3609 3376 3156 2948 2948 2754 ...

# More plots & equations will be examined
# when we talk about Matlab. Let's move away
# from scientific programming for now...

# RUDIMENTARY DATA ANALYSIS & STATISTICS

furnace0 <- read.delim("C:/Users/sfinch/Desktop/furnace.txt", header=TRUE)
# must use forward slash /, not backslash \, when specifying
# pathnames as above

```r
furnace <- subset(furnace0, select = c(CHArea, CHHght, Age, BTUIn, BTUOut, Damper))
furnace  # just one missing datapoint (CHArea in line 24)
names(furnace)  # names of all variables in dataframe...
summary(furnace)  # ...and associated summary statistics
```

```r
> furnace

<table>
<thead>
<tr>
<th>CHArea</th>
<th>CHHght</th>
<th>Age</th>
<th>BTUIn</th>
<th>BTUOut</th>
<th>Damper</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>28</td>
<td>20</td>
<td>8</td>
<td>7.87</td>
<td>8.25</td>
</tr>
<tr>
<td>2</td>
<td>144</td>
<td>26</td>
<td>75</td>
<td>9.43</td>
<td>9.66</td>
</tr>
<tr>
<td>3</td>
<td>80</td>
<td>30</td>
<td>44</td>
<td>7.16</td>
<td>8.33</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>24</td>
<td>75</td>
<td>8.67</td>
<td>8.82</td>
</tr>
<tr>
<td>5</td>
<td>168</td>
<td>35</td>
<td>30</td>
<td>12.31</td>
<td>12.06</td>
</tr>
<tr>
<td>6</td>
<td>28</td>
<td>17</td>
<td>4</td>
<td>9.84</td>
<td>9.67</td>
</tr>
<tr>
<td>7</td>
<td>64</td>
<td>24</td>
<td>45</td>
<td>16.90</td>
<td>17.51</td>
</tr>
<tr>
<td>8</td>
<td>64</td>
<td>18</td>
<td>16</td>
<td>10.04</td>
<td>10.79</td>
</tr>
<tr>
<td>9</td>
<td>96</td>
<td>25</td>
<td>45</td>
<td>12.62</td>
<td>13.59</td>
</tr>
<tr>
<td>10</td>
<td>108</td>
<td>27</td>
<td>40</td>
<td>7.62</td>
<td>7.99</td>
</tr>
<tr>
<td>11</td>
<td>64</td>
<td>16</td>
<td>22</td>
<td>11.12</td>
<td>12.64</td>
</tr>
<tr>
<td>12</td>
<td>63</td>
<td>30</td>
<td>40</td>
<td>13.43</td>
<td>14.42</td>
</tr>
<tr>
<td>13</td>
<td>42</td>
<td>15</td>
<td>13</td>
<td>9.07</td>
<td>9.25</td>
</tr>
<tr>
<td>14</td>
<td>117</td>
<td>25</td>
<td>99</td>
<td>6.94</td>
<td>7.79</td>
</tr>
<tr>
<td>15</td>
<td>64</td>
<td>18</td>
<td>19</td>
<td>10.28</td>
<td>11.29</td>
</tr>
<tr>
<td>16</td>
<td>28</td>
<td>17</td>
<td>30</td>
<td>9.37</td>
<td>10.26</td>
</tr>
<tr>
<td>17</td>
<td>64</td>
<td>28</td>
<td>60</td>
<td>7.93</td>
<td>9.46</td>
</tr>
<tr>
<td>18</td>
<td>64</td>
<td>19</td>
<td>30</td>
<td>13.96</td>
<td>14.77</td>
</tr>
<tr>
<td>19</td>
<td>28</td>
<td>26</td>
<td>10</td>
<td>6.80</td>
<td>7.21</td>
</tr>
<tr>
<td>20</td>
<td>80</td>
<td>27</td>
<td>60</td>
<td>4.00</td>
<td>4.29</td>
</tr>
<tr>
<td>21</td>
<td>28</td>
<td>14</td>
<td>24</td>
<td>8.58</td>
<td>9.81</td>
</tr>
<tr>
<td>22</td>
<td>28</td>
<td>23</td>
<td>70</td>
<td>8.00</td>
<td>8.41</td>
</tr>
<tr>
<td>23</td>
<td>64</td>
<td>17</td>
<td>12</td>
<td>5.98</td>
<td>6.78</td>
</tr>
<tr>
<td>24</td>
<td>NaN</td>
<td>30</td>
<td>60</td>
<td>15.24</td>
<td>16.30</td>
</tr>
<tr>
<td>25</td>
<td>64</td>
<td>27</td>
<td>40</td>
<td>8.54</td>
<td>9.01</td>
</tr>
<tr>
<td>26</td>
<td>64</td>
<td>19</td>
<td>17</td>
<td>11.09</td>
<td>11.41</td>
</tr>
<tr>
<td>27</td>
<td>50</td>
<td>18</td>
<td>15</td>
<td>11.70</td>
<td>12.37</td>
</tr>
<tr>
<td>28</td>
<td>50</td>
<td>18</td>
<td>18</td>
<td>12.71</td>
<td>13.28</td>
</tr>
<tr>
<td>29</td>
<td>50</td>
<td>18</td>
<td>4</td>
<td>6.78</td>
<td>7.24</td>
</tr>
<tr>
<td>30</td>
<td>28</td>
<td>16</td>
<td>5</td>
<td>9.82</td>
<td>10.55</td>
</tr>
<tr>
<td>31</td>
<td>80</td>
<td>26</td>
<td>75</td>
<td>12.91</td>
<td>13.89</td>
</tr>
<tr>
<td>32</td>
<td>50</td>
<td>18</td>
<td>14</td>
<td>10.35</td>
<td>10.72</td>
</tr>
<tr>
<td>33</td>
<td>28</td>
<td>15</td>
<td>8</td>
<td>9.60</td>
<td>9.22</td>
</tr>
<tr>
<td>34</td>
<td>100</td>
<td>31</td>
<td>99</td>
<td>9.58</td>
<td>10.61</td>
</tr>
<tr>
<td>35</td>
<td>28</td>
<td>16</td>
<td>99</td>
<td>9.83</td>
<td>10.04</td>
</tr>
<tr>
<td>36</td>
<td>50</td>
<td>20</td>
<td>34</td>
<td>9.52</td>
<td>10.20</td>
</tr>
<tr>
<td>37</td>
<td>108</td>
<td>25</td>
<td>80</td>
<td>18.26</td>
<td>20.55</td>
</tr>
<tr>
<td>38</td>
<td>64</td>
<td>25</td>
<td>99</td>
<td>10.64</td>
<td>11.75</td>
</tr>
<tr>
<td>39</td>
<td>36</td>
<td>26</td>
<td>99</td>
<td>6.62</td>
<td>7.08</td>
</tr>
<tr>
<td>40</td>
<td>28</td>
<td>16</td>
<td>6</td>
<td>5.20</td>
<td>5.50</td>
</tr>
<tr>
<td>41</td>
<td>49</td>
<td>32</td>
<td>50</td>
<td>12.28</td>
<td>13.07</td>
</tr>
<tr>
<td>42</td>
<td>38</td>
<td>16</td>
<td>10</td>
<td>7.23</td>
<td>7.60</td>
</tr>
<tr>
<td>43</td>
<td>28</td>
<td>18</td>
<td>2</td>
<td>2.97</td>
<td>3.20</td>
</tr>
<tr>
<td>44</td>
<td>64</td>
<td>20</td>
<td>99</td>
<td>8.81</td>
<td>9.28</td>
</tr>
<tr>
<td>45</td>
<td>72</td>
<td>31</td>
<td>15</td>
<td>9.27</td>
<td>9.73</td>
</tr>
<tr>
<td>46</td>
<td>70</td>
<td>39</td>
<td>45</td>
<td>11.29</td>
<td>11.73</td>
</tr>
</tbody>
</table>
```
> names(furnace)
[1] "CHArea" "CHHght" "Age" "BTUIn" "BTUOut" "Damper"

> summary(furnace)

<table>
<thead>
<tr>
<th>CHArea</th>
<th>CHHght</th>
<th>Age</th>
<th>BTUIn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.</td>
<td>28.00</td>
<td>1.00</td>
<td>2.970</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>28.00</td>
<td>12.00</td>
<td>7.947</td>
</tr>
<tr>
<td>Median</td>
<td>64.00</td>
<td>30.00</td>
<td>9.835</td>
</tr>
<tr>
<td>Mean</td>
<td>62.56</td>
<td>21.97</td>
<td>10.038</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>80.00</td>
<td>60.00</td>
<td>12.045</td>
</tr>
<tr>
<td>Max.</td>
<td>168.00</td>
<td>99.00</td>
<td>18.260</td>
</tr>
<tr>
<td>NA's</td>
<td>1.00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

> names(furnace)
[1] "CHArea" "CHHght" "Age" "BTUIn" "BTUOut" "Damper"
```r
# Means, standard deviations & BTU scatterplot

mean(furnace)
mean(furnace, na.rm=TRUE)

sd(furnace, na.rm=TRUE)  # (curiously not given by 'summary'!)

> mean(furnace)
  CHArea    CHHght       Age     BTUIn    BTUOut    Damper
       NA 21.966667 38.566667 10.038444 10.813111  1.555556
> mean(furnace, na.rm=TRUE)
  CHArea     CHHght        Age      BTUIn     BTUOut     Damper
       62.561798  21.966667  38.566667  10.038444  10.813111  1.555556
> sd(furnace, na.rm=TRUE)  # (curiously not given by 'summary'!)
  CHArea     CHHght        Age      BTUIn     BTUOut     Damper
       32.5307390  5.9254735  31.0932089  2.8679903  3.0884073  0.4996878

plot(furnace$BTUOut~furnace$BTUIn,
     xlab="BTUIn", ylab="BTUOut",
     main="BTUIn vs BTUOut for Furnace Data",
     xlim=c(2,20), ylim=c(2,22), col="magenta", pch=5)
```
# BTU scatterplot, stratified by damper type

# Although treated by 'summary' as a numeric variable,  
# "Damper" is actually a categorical (factor) variable

H <- hist(furnace$Damper, breaks=2, plot=FALSE)  
H$counts

> H$counts
[1] 40 50
# We can help R reinterpret "Damper" so that 'summary' gives
# histogram for "Damper", but other stats for other variables

```r
furnace$Damper <- factor(furnace$Damper, levels=1:2)
levels(furnace$Damper)
is(furnace$Damper)[1] # 'is' identifies variable type
is(furnace$CHArea)[1]

> levels(furnace$Damper)
[1] "1" "2"
> is(furnace$Damper)[1] # 'is' identifies variable type
[1] "factor"
> is(furnace$CHArea)[1]
[1] "numeric"

summary(furnace)

> summary(furnace)
  CHArea           CHHght           Age            BTUIn
  Min.   : 28.00   Min.   :14.00   Min.   : 1.00   Min.   : 2.970
  1st Qu.: 28.00   1st Qu.:17.00   1st Qu.:12.00   1st Qu.: 7.947
  Median : 64.00   Median :20.00   Median :30.00   Median : 9.835
  Mean   : 62.56   Mean   :21.97   Mean   :38.57   Mean   :10.038
  3rd Qu.: 80.00   3rd Qu.:27.00   3rd Qu.:60.00   3rd Qu.:12.045
  Max.   :168.00   Max.   :39.00   Max.   :99.00   Max.   :18.260
  NA's   :  1.00
  BTUOut       Damper
  Min.   : 3.200   1:40
  1st Qu.: 8.707   2:50
  Median :10.740
  Mean   :10.813
  3rd Qu.:12.797
  Max.   :20.550

furnace$Damper <- as.numeric(furnace$Damper) # reversal of above
is(furnace$Damper)[1]

> is(furnace$Damper)[1]
[1] "numeric"

# It's inconvenient to be carrying "furnace$"!

search() # objects in the workspace

> search() # objects in the workspace
[1] ".GlobalEnv" "package:scatterplot3d" "package:methods"
[7] "package:utils" "package:datasets" "Autoloads"
[10] "package:base"

attach(furnace)

search() # our dataset now added (no. 2 in search path)

> search() # our dataset now added (no. 2 in search path)
[1] ".GlobalEnv" "furnace" "package:scatterplot3d"
hist(Damper, breaks=2, plot=FALSE)

> hist(Damper, breaks=2, plot=FALSE)

[1] 40 50

{  
x11()  
plot(BTUOut[Damper==1]~BTUIn[Damper==1],  
    xlab="BTUIn", ylab="BTUOut", xaxt="n", yaxt="n",  
    main="BTUIn vs BTUOut for Stratified Furnace Data",  
    xlim=c(2,20), ylim=c(2,22), col="blue", pch=1)
points(BTUOut[Damper==2]~BTUIn[Damper==2], col="red", pch=4)
axis(1, at=seq(2,20,2))
axis(2, at=seq(2,22,2))
legend(15,8,c('EVD','TVD'),pch=c(1,4),col=c("blue","red"))
}
# "points" is for overlay; suppress initial plotting of axes,  
# so as to control tick mark placements subsequently

# "plot" understood by R to be scatterplot; tilde (~) to  
# be explained later

# BTUIn histogram & 90% parameter confidence intervals
x11()
hist(BTUIIn, freq=TRUE, border="darkblue", xaxt="n",
   main="BTUIIn Histogram & Normal Fit",
   xlim=c(2,20), ylim=c(0,30), plot=TRUE)
axis(1, at=seq(2,20,2))
curve(180*dnorm(x, mean(BTUIIn), sd(BTUIIn)),
    from=0, to=20, add=TRUE, col="red", lwd=2)

BTUIIn Histogram & Normal Fit

count <- function(x) sum(!is.na(x))
N <- count(BTUIn)
M <- mean(BTUIn)
S <- sd(BTUIn)

A <- M+S*qt(0.05,N-1)/sqrt(N)  # qt returns specified %-tiles of
B <- M+S*qt(0.95,N-1)/sqrt(N)  # Student t distribution, N-1 dof
cat('90 pct CI about BTUIn mean:', A, M, B, '\n')

> 90 pct CI about BTUIn mean: 9.535954 10.03844 10.54094

A <- S*sqrt((N-1)/qchisq(0.95,N-1))  # qchisq returns specified
B <- S*sqrt((N-1)/qchisq(0.05,N-1))  # %-tiles of Chi square dist
cat('90 pct CI about BTUIn stdv:', A, S, B, '\n')

> 90 pct CI about BTUIn stdv: 2.556354 2.867990 3.275093

# Can obtain histogram & interval estimates
# for BTUOut as well

# One graphical method of comparison: placing
# BTUIn & BTUOut histograms side-by-side

# Simple linear regression

lm(BTUOut~BTUIn)  # very brief output!

> lm(BTUOut~BTUIn)

Call:
lm(formula = BTUOut ~ BTUIn)

Coefficients:
(Intercept)        BTUIn
     0.2074       1.0565

# tilde (~) means here that "BTUOut" is described by "BTUIn"

# much more information is seen via various extractor functions

btu <- lm(BTUOut~BTUIn)  # give the linear model object a name
summary(btu)  # R^2 (square of correlation coefficient)

> summary(btu)

> Call:
lm(formula = BTUOut ~ BTUIn)

Residuals:
    Min     1Q Median     3Q    Max
-1.79450 -0.26337 -0.04183  0.24949  3.36064

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.20742    0.23177   0.895    0.373
BTUIn           1.05651    0.02221  47.571   <2e-16


x11()
plot(BTUOut~BTUIn,
     xlab="BTUIn", ylab="BTUOut",
     main="BTUIn vs BTUOut and Least Squares Regression Line",
     xlim=c(2,20), ylim=c(2,22), col="magenta", pch=5)
abline(btu, col="red")

# our original BTU scatterplot, plus best linear fit
FIT <- fitted(btu)  # y-values we expect for observed
# x-values, given the linear fit

segments(BTUIn,FIT,BTUIn,BTUOut)  # (x1,y1,x2,y2)

# Same scatterplot, plus vertical lines (residuals = obs & fit diff)

# Regression diagnostic plots say even more
(x11())
par(mfrow=c(3,2), mex=0.5)  # establish 3x2 layout, compressed margin
plot(btu,which=1:6)         # Cook's distance: measure of influence of 
                            # each obs on the regression coefficients

par(mfrow=c(1,1), mex=1)    # reset
{
xll()
plot(btu,which=1:6)        # or, if you prefer, visit one plot at a time
}
graphics.off()  # close plot