Intro_to_R_2016_Code.R

breannechryst

Wed Feb 10 16:38:18 2016

```
## Introduction to R
## Breanne Chryst and Sara Bastomski
## CSSSI StatLab
## Feb 19, 2016

############################################################
## R Basics
############################################################

## R can be used as a calculator, it works as expected:
2+3
```

```
## [1] 5
```

```
exp(2)
```

```
## [1] 7.389056
```

```
5^(2)
```

```
## [1] 25
```

```
3*2+4
```

```
## [1] 10
```

```
(2*4+1)/2
```

```
## [1] 4.5
```

```
## The print command:
print("Hello World")
```

```
## [1] "Hello World"
```

```
print(5*2-11)
```

```
## [1] -1
```

```
## Assigning a variable
x <- 5 # 5 has now been assigned to the variable x

# the "<-" assigns the value on the right to the name on the left. Made by "alt -"

x
```

```
## [1] 5
```

```r
x^2
```

```
## [1] 25
```

```r
## Creating a vector:
y <- c(3,7,5,1,2,3,2,5,5) # "c()" concatenates, creating a vector

## Extracting values of a vector:
y[2]
```

```
## [1] 7
```

```r
3:5 # the whole numbers from 3 to 5
```

```
## [1] 3 4 5
```

```r
y[3:5] # extracts the 3rd to 5th elements in the vector y
```

```
## [1] 5 1 2
```

```r
sub.y <- y[3:5]
sub.y
```

```
## [1] 5 1 2
```

```r
## Other ways to make vectors:
array(data = 0, dim = 3)
```

```
## [1] 0 0 0
```

```r
seq(from = 1, to = 4, by = 1)
```

```
## [1] 1 2 3 4
```

```r
seq(1,4,0.5)
```

```
## [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0
```

```r
rep(x = "cat", times = 3)
```

```
## [1] "cat" "cat" "cat"
```

```r
rep(c("cat", 4, x^(2)), each = 2)
```

```
## [1] "cat" "cat" "4"   "4"   "25"  "25"
```

```r
rep(c("cat", 4, x^(2)), times = 2)
```

```
## [1] "cat" "4"   "25"  "cat" "4"   "25"
```

```r
## "matrix()" creates a matrix from the values entered:
z <- matrix(y, nrow=3) # This is filled by column
z
```

```
##      [,1] [,2] [,3]
## [1,]    3    1    2
## [2,]    7    2    5
## [3,]    5    3    5
```

```r
z <- matrix(y, nrow=3, byrow=T)
# By changing the "byrow" option, we can fill the matrix by row
z
```

```
##      [,1] [,2] [,3]
## [1,]    3    7    5
## [2,]    1    2    3
## [3,]    2    5    5
```

```r
## Extracting values from matrices:
z[2,] # Row
```

```
## [1] 1 2 3
```

```r
z[,3] # Column
```

```
## [1] 5 3 5
```

```r
z[2,3] # Value
```

```
## [1] 3
```

```r
## Other ways to make matrics:
array(data = y, dim = c(3,3))
```

```
##      [,1] [,2] [,3]
## [1,]    3    1    2
## [2,]    7    2    5
## [3,]    5    3    5
```

```r
matrix(c(1,2,3,4,5,6), nrow = 2)
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
## Create Dataframes:
dat <- as.data.frame(z)
names(dat) <- c("cat", "giraffe","bowlingball")
dat
```

```
##   cat giraffe bowlingball
## 1   3       7           5
## 2   1       2           3
## 3   2       5           5
```

```
## R has base functions:
mean(y)
```

```
## [1] 3.666667
```

```
length(y)
```

```
## [1] 9
```

```
sd(y)
```

```
## [1] 1.936492
```

```
var(y)
```

```
## [1] 3.75
```

```
prod(y) # Takes the product of each element in the vector
```

```
## [1] 31500
```

```
apply(z, 2, mean) # Very useful in avoiding for loops, also has useful cousins sapply and lapply
```

```
## [1] 2.000000 4.666667 4.333333
```

```
##########################################################
## Introduction to Statistics with R
##########################################################

## Getting help
# help.start()  # Opens html help in web browser (if installed)
# help(help)    # find help on how to use help
# ?help     # Same as above
# help.search("help")   # Find all functions that include the word 'help'

## Getting help on particular functions:
help(plot) # You can use the help function, the argument is an R function
?plot # or just a question mark in front of the funtion you have questions about


## Reading in your data
getwd()                # What directory are we in?
```

```
## [1] "/Users/breannechryst/Desktop/Consultant"
```

```
# R will read and write files to the working directory, unless otherwise specified
setwd("~/Desktop")  # You can change your working directory

## Read in data
dat <- read.table("http://www.stat.yale.edu/~blc3/IntroR2015/remote_weight.txt",
                  header=T, sep="", row.names=NULL, as.is = TRUE)
# Read data including headers, data separated by spaces, no row names
ls()                        # List all variables stored in memory
```

```
## [1] "dat"   "sub.y" "x"     "y"     "z"
```

```
head(dat)             # Shows the first 6 rows of the data
```

```
##   id remote weight gender
## 1  1      5    151      0
## 2  2      7    152      0
## 3  3      3    153      0
## 4  4      2    165      0
## 5  5      5    138      0
## 6  6      0    149      0
```

```
head(dat, 10)         # the first 10 rows of the data
```

```
##    id remote weight gender
## 1   1      5    151      0
## 2   2      7    152      0
## 3   3      3    153      0
## 4   4      2    165      0
## 5   5      5    138      0
## 6   6      0    149      0
## 7   7      5    142      0
## 8   8      9    139      0
## 9   9      1    140      0
## 10 10      8    138      0
```

```
tail(dat)             # last 6 rows of the data
```

```
##      id remote weight gender
## 95   95     30    167      1
## 96   96     28    154      1
## 97   97     29    181      1
## 98   98     34    172      1
## 99   99     23    159      1
## 100 100     25    177      1
```

```
## Extracting data from the data frame
dim(dat)                # Find out how many rows and columns in the data set
```

```
## [1] 100   4
```

```r
names(dat)          # List all variable names in the dataset
```

```
## [1] "id"     "remote" "weight" "gender"
```

```r
str(dat)            # Look at the structure of your data
```

```
## 'data.frame':    100 obs. of  4 variables:
##  $ id    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ remote: int  5 7 3 2 5 0 5 9 1 8 ...
##  $ weight: int  151 152 153 165 138 149 142 139 140 138 ...
##  $ gender: int  0 0 0 0 0 0 0 0 0 0 ...
```

```r
dat                  # See the data frame on the screen
```

```
##       id remote weight gender
## 1      1      5    151      0
## 2      2      7    152      0
## 3      3      3    153      0
## 4      4      2    165      0
## 5      5      5    138      0
## 6      6      0    149      0
## 7      7      5    142      0
## 8      8      9    139      0
## 9      9      1    140      0
## 10    10      8    138      0
## 11    11      7    137      0
## 12    12      7    119      0
## 13    13      8    140      0
## 14    14      9    145      0
## 15    15      5    126      0
## 16    16      3    142      0
## 17    17      6    127      0
## 18    18     10    135      0
## 19    19      7    149      0
## 20    20      7    134      0
## 21    21      5    157      0
## 22    22      7    141      0
## 23    23      4    146      0
## 24    24      8    127      0
## 25    25      1    143      0
## 26    26      4    151      0
## 27    27      8    132      0
## 28    28      2    149      0
## 29    29      0    144      0
## 30    30      5    148      0
## 31    31      2    119      0
## 32    32      9    129      0
## 33    33      8    138      0
## 34    34      1    165      0
## 35    35     10    142      0
## 36    36      8    138      0
## 37    37      5    137      0
```

```
## 38    38      9    154      0
## 39    39      1    133      0
## 40    40      9    139      0
## 41    41      2    146      0
## 42    42      9    152      0
## 43    43      7    149      0
## 44    44     10    128      0
## 45    45      8    131      0
## 46    46      4    136      0
## 47    47      1    148      0
## 48    48      4    134      0
## 49    49      1    137      0
## 50    50      4    137      0
## 51    51     31    159      1
## 52    52     32    167      1
## 53    53     27    175      1
## 54    54     37    189      1
## 55    55     28    179      1
## 56    56     27    145      1
## 57    57     30    161      1
## 58    58     35    181      1
## 59    59     31    177      1
## 60    60     27    178      1
## 61    61     26    170      1
## 62    62     31    158      1
## 63    63     32    179      1
## 64    64     37    192      1
## 65    65     35    183      1
## 66    66     31    184      1
## 67    67     29    169      1
## 68    68     34    182      1
## 69    69     30    190      1
## 70    70     30    195      1
## 71    71     35    182      1
## 72    72     30    174      1
## 73    73     37    184      1
## 74    74     29    188      1
## 75    75     23    150      1
## 76    76     31    178      1
## 77    77     35    186      1
## 78    78     28    170      1
## 79    79     21    149      1
## 80    80     33    203      1
## 81    81     27    172      1
## 82    82     23    160      1
## 83    83     36    179      1
## 84    84     30    195      1
## 85    85     32    160      1
## 86    86     20    147      1
## 87    87     22    144      1
## 88    88     35    178      1
## 89    89     34    168      1
## 90    90     27    157      1
## 91    91     34    184      1
```

```
## 92   92      18     133       1
## 93   93      31     160       1
## 94   94      29     170       1
## 95   95      30     167       1
## 96   96      28     154       1
## 97   97      29     181       1
## 98   98      34     172       1
## 99   99      23     159       1
## 100 100      25     177       1
```

```
dat[1:10,]           # See the first 10 rows
```

```
##    id remote weight gender
## 1   1      5    151      0
## 2   2      7    152      0
## 3   3      3    153      0
## 4   4      2    165      0
## 5   5      5    138      0
## 6   6      0    149      0
## 7   7      5    142      0
## 8   8      9    139      0
## 9   9      1    140      0
## 10 10      8    138      0
```

```
dat[,"weight"]       # See only the weight column
```

```
##   [1] 151 152 153 165 138 149 142 139 140 138 137 119 140 145 126 142 127
##  [18] 135 149 134 157 141 146 127 143 151 132 149 144 148 119 129 138 165
##  [35] 142 138 137 154 133 139 146 152 149 128 131 136 148 134 137 137 159
##  [52] 167 175 189 179 145 161 181 177 178 170 158 179 192 183 184 169 182
##  [69] 190 195 182 174 184 188 150 178 186 170 149 203 172 160 179 195 160
##  [86] 147 144 178 168 157 184 133 160 170 167 154 181 172 159 177
```

```
dat[,3]              # Same as above
```

```
##   [1] 151 152 153 165 138 149 142 139 140 138 137 119 140 145 126 142 127
##  [18] 135 149 134 157 141 146 127 143 151 132 149 144 148 119 129 138 165
##  [35] 142 138 137 154 133 139 146 152 149 128 131 136 148 134 137 137 159
##  [52] 167 175 189 179 145 161 181 177 178 170 158 179 192 183 184 169 182
##  [69] 190 195 182 174 184 188 150 178 186 170 149 203 172 160 179 195 160
##  [86] 147 144 178 168 157 184 133 160 170 167 154 181 172 159 177
```

```
dat$weight           # Yet another way
```

```
##   [1] 151 152 153 165 138 149 142 139 140 138 137 119 140 145 126 142 127
##  [18] 135 149 134 157 141 146 127 143 151 132 149 144 148 119 129 138 165
##  [35] 142 138 137 154 133 139 146 152 149 128 131 136 148 134 137 137 159
##  [52] 167 175 189 179 145 161 181 177 178 170 158 179 192 183 184 169 182
##  [69] 190 195 182 174 184 188 150 178 186 170 149 203 172 160 179 195 160
##  [86] 147 144 178 168 157 184 133 160 170 167 154 181 172 159 177
```

```r
dat[1:10,"weight"]  # See only the first 10 values of the weight col.
```

```
##  [1] 151 152 153 165 138 149 142 139 140 138
```

```r
dat[,-1]              # See all but the first column of data
```

```
##    remote weight gender
## 1       5    151      0
## 2       7    152      0
## 3       3    153      0
## 4       2    165      0
## 5       5    138      0
## 6       0    149      0
## 7       5    142      0
## 8       9    139      0
## 9       1    140      0
## 10      8    138      0
## 11      7    137      0
## 12      7    119      0
## 13      8    140      0
## 14      9    145      0
## 15      5    126      0
## 16      3    142      0
## 17      6    127      0
## 18     10    135      0
## 19      7    149      0
## 20      7    134      0
## 21      5    157      0
## 22      7    141      0
## 23      4    146      0
## 24      8    127      0
## 25      1    143      0
## 26      4    151      0
## 27      8    132      0
## 28      2    149      0
## 29      0    144      0
## 30      5    148      0
## 31      2    119      0
## 32      9    129      0
## 33      8    138      0
## 34      1    165      0
## 35     10    142      0
## 36      8    138      0
## 37      5    137      0
## 38      9    154      0
## 39      1    133      0
## 40      9    139      0
## 41      2    146      0
## 42      9    152      0
## 43      7    149      0
## 44     10    128      0
## 45      8    131      0
```

```
## 46          4     136          0
## 47          1     148          0
## 48          4     134          0
## 49          1     137          0
## 50          4     137          0
## 51         31     159          1
## 52         32     167          1
## 53         27     175          1
## 54         37     189          1
## 55         28     179          1
## 56         27     145          1
## 57         30     161          1
## 58         35     181          1
## 59         31     177          1
## 60         27     178          1
## 61         26     170          1
## 62         31     158          1
## 63         32     179          1
## 64         37     192          1
## 65         35     183          1
## 66         31     184          1
## 67         29     169          1
## 68         34     182          1
## 69         30     190          1
## 70         30     195          1
## 71         35     182          1
## 72         30     174          1
## 73         37     184          1
## 74         29     188          1
## 75         23     150          1
## 76         31     178          1
## 77         35     186          1
## 78         28     170          1
## 79         21     149          1
## 80         33     203          1
## 81         27     172          1
## 82         23     160          1
## 83         36     179          1
## 84         30     195          1
## 85         32     160          1
## 86         20     147          1
## 87         22     144          1
## 88         35     178          1
## 89         34     168          1
## 90         27     157          1
## 91         34     184          1
## 92         18     133          1
## 93         31     160          1
## 94         29     170          1
## 95         30     167          1
## 96         28     154          1
## 97         29     181          1
## 98         34     172          1
## 99         23     159          1
```

```
## 100     25    177      1
```

```r
dat.o <- dat         # Copy the data frame to a data.frame named dat.O
ls()                       # Now we have 5 variables: 'x', 'y', 'z', 'dat' and 'dat.o'
```

```
## [1] "dat"   "dat.o" "sub.y" "x"     "y"     "z"
```

```r
## Getting familiar with the data
summary(dat)             # Generate summary statistics of data
```

```
##        id             remote          weight          gender   
##  Min.   :  1.00   Min.   : 0.00   Min.   :119.0   Min.   :0.0  
##  1st Qu.: 25.75   1st Qu.: 5.00   1st Qu.:139.8   1st Qu.:0.0  
##  Median : 50.50   Median :14.00   Median :152.0   Median :0.5  
##  Mean   : 50.50   Mean   :17.59   Mean   :156.4   Mean   :0.5  
##  3rd Qu.: 75.25   3rd Qu.:30.00   3rd Qu.:174.2   3rd Qu.:1.0  
##  Max.   :100.00   Max.   :37.00   Max.   :203.0   Max.   :1.0
```

```r
apply(dat, 2, sd)    # Calculate standard deviations of all variables
```

```
##         id     remote     weight     gender 
## 29.0114920 12.8488454 20.0833491  0.5025189
```

```r
var(dat)                # Variance on diagonal, covariance off diagonal
```

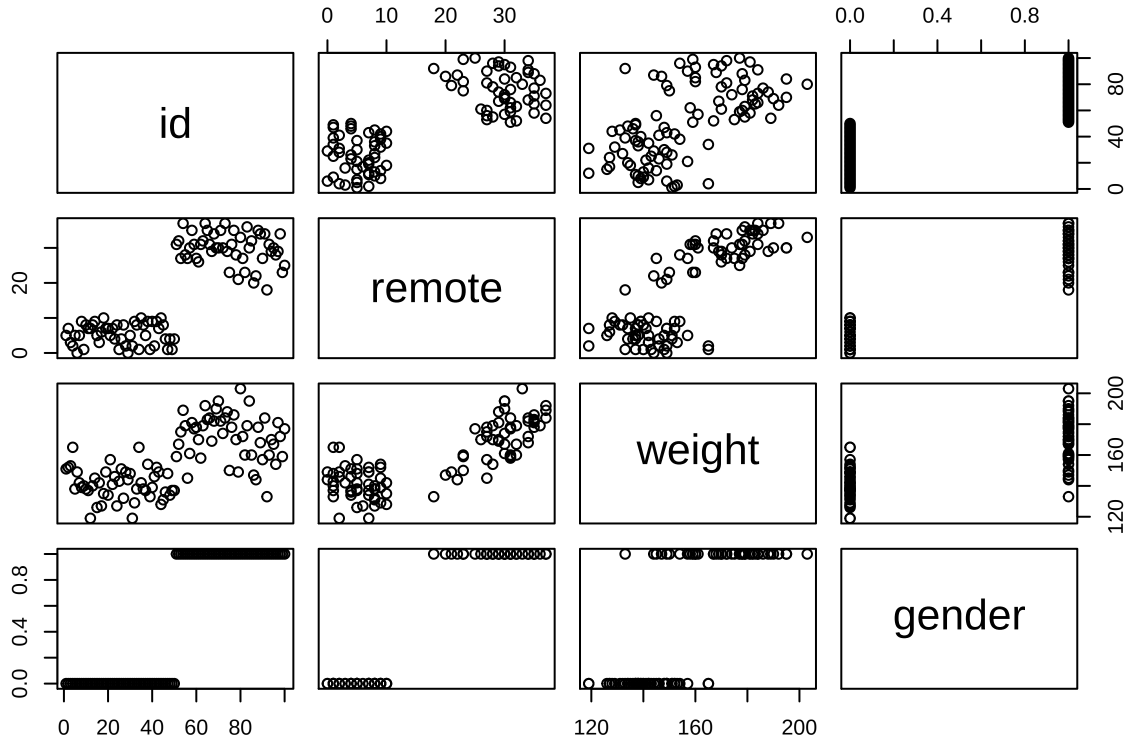
```
##               id     remote     weight     gender
## id     841.66667 299.287879 355.691919 12.6262626
## remote 299.28788 165.092828 209.448990  6.1565657
## weight 355.69192 209.448990 403.340909  7.7929293
## gender  12.62626   6.156566   7.792929  0.2525253
```

```r
mean(dat)              # Calculate the mean of all variables
```

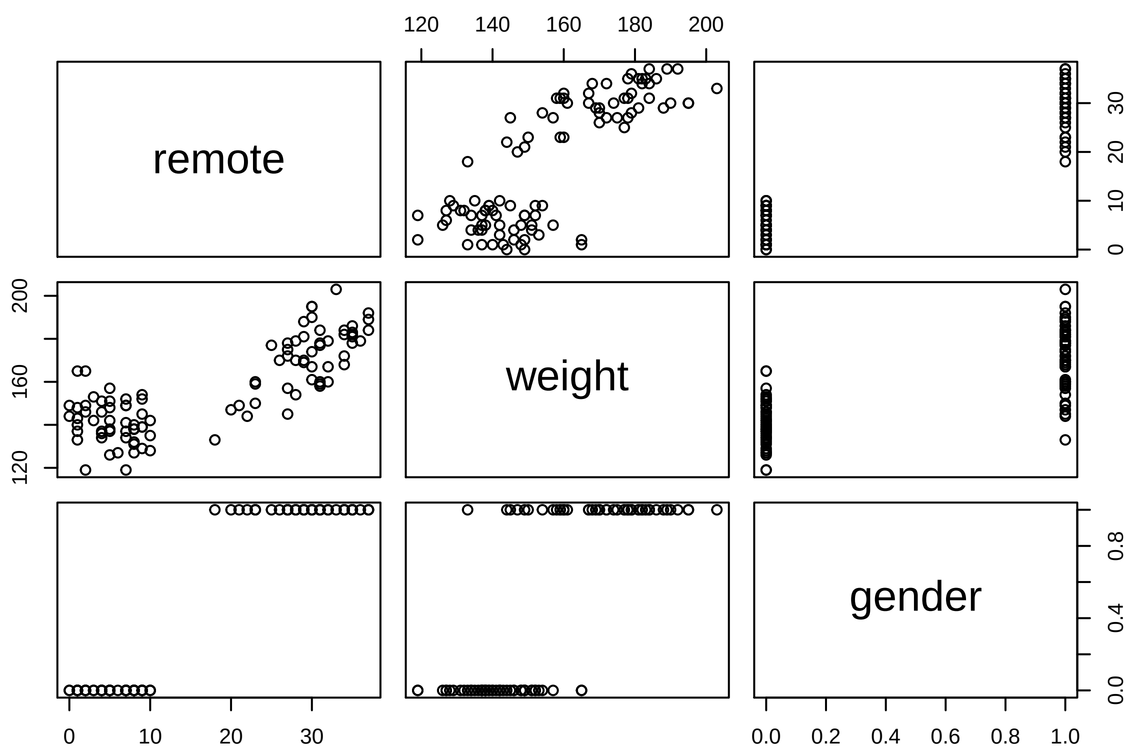
```
## Warning in mean.default(dat): argument is not numeric or logical: returning
## NA
```

```
## [1] NA
```

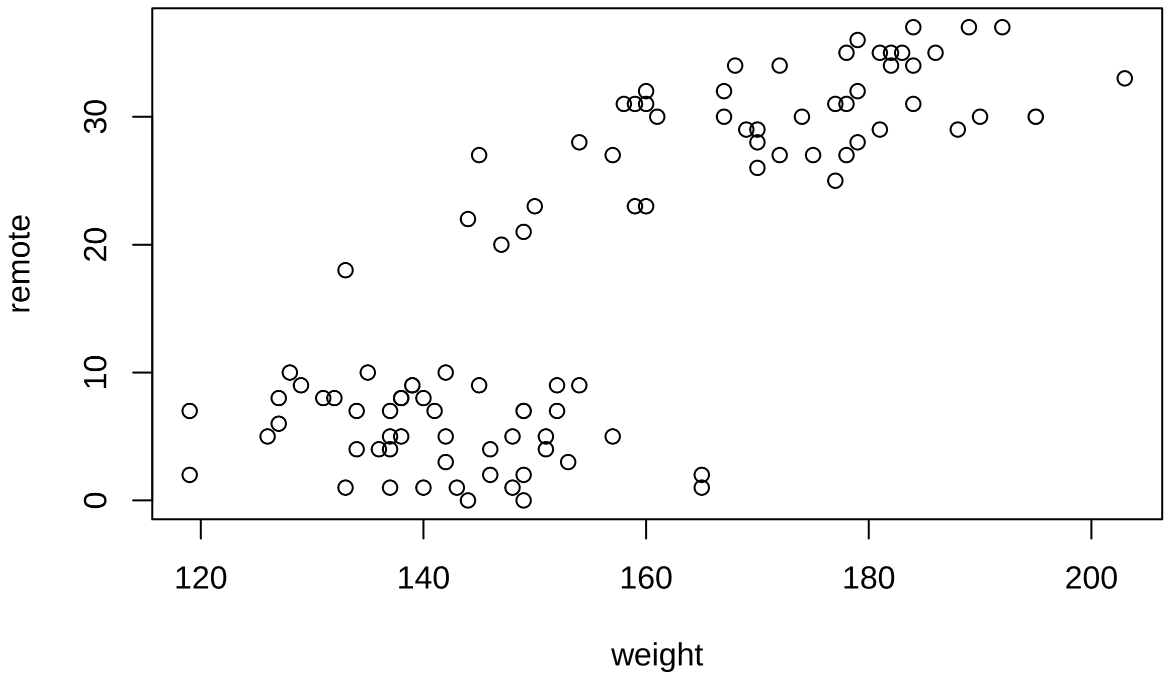
```r
pairs(dat)             # A general view of data through scatter plots
```




```
pairs(dat[,-1])
```




```
# See scatterplots for all pairs of variables except the first ('id') in the data frame
plot(remote ~ weight, data = dat)    # Scatterplot of 'weight' vs. 'remote'
```




```
# Changing data type
class(dat$gender)  # What kind of variable is 'gender'?
```

```
## [1] "integer"
```

```
dat$gender <- factor(dat$gender)    # Converts 'gender' from type integer to factor
class(dat$gender)    # Verify that gender is now indeed of type factor
```

```
## [1] "factor"
```

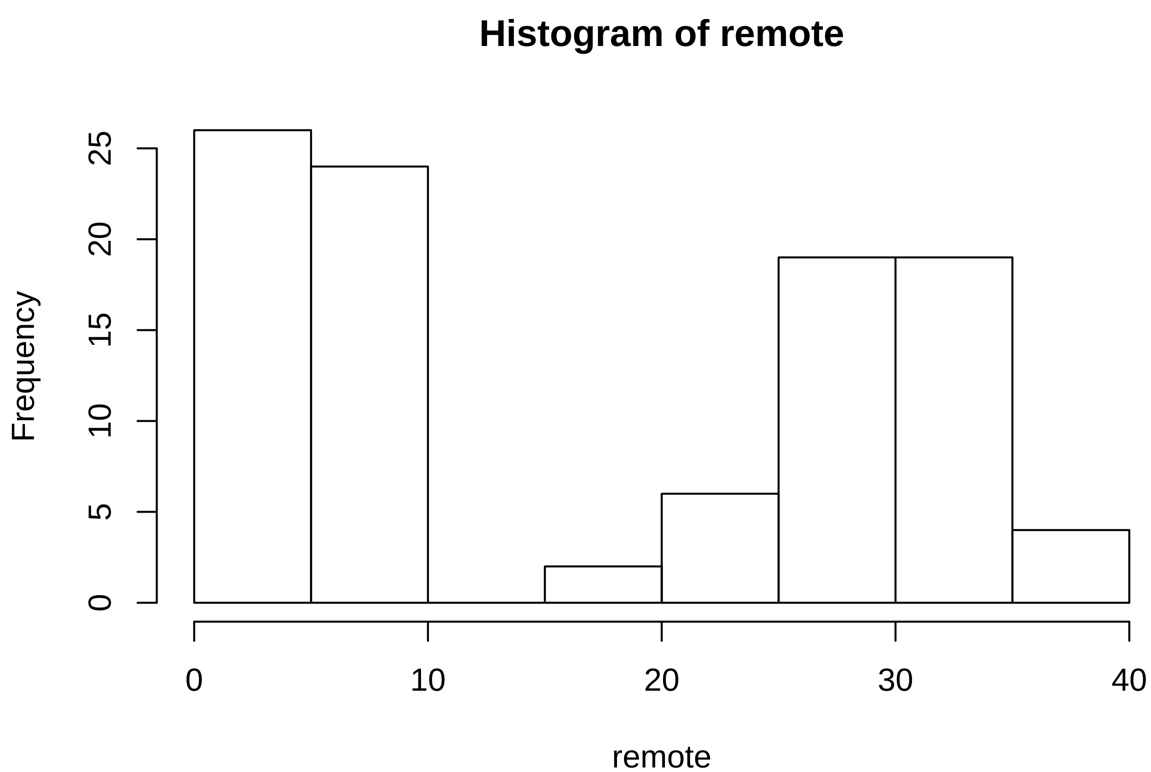
```
dat$gender              # See all data in column 'gender'; note "Levels: 0 1" at the bottom
```

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [36] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [71] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## Levels: 0 1
```

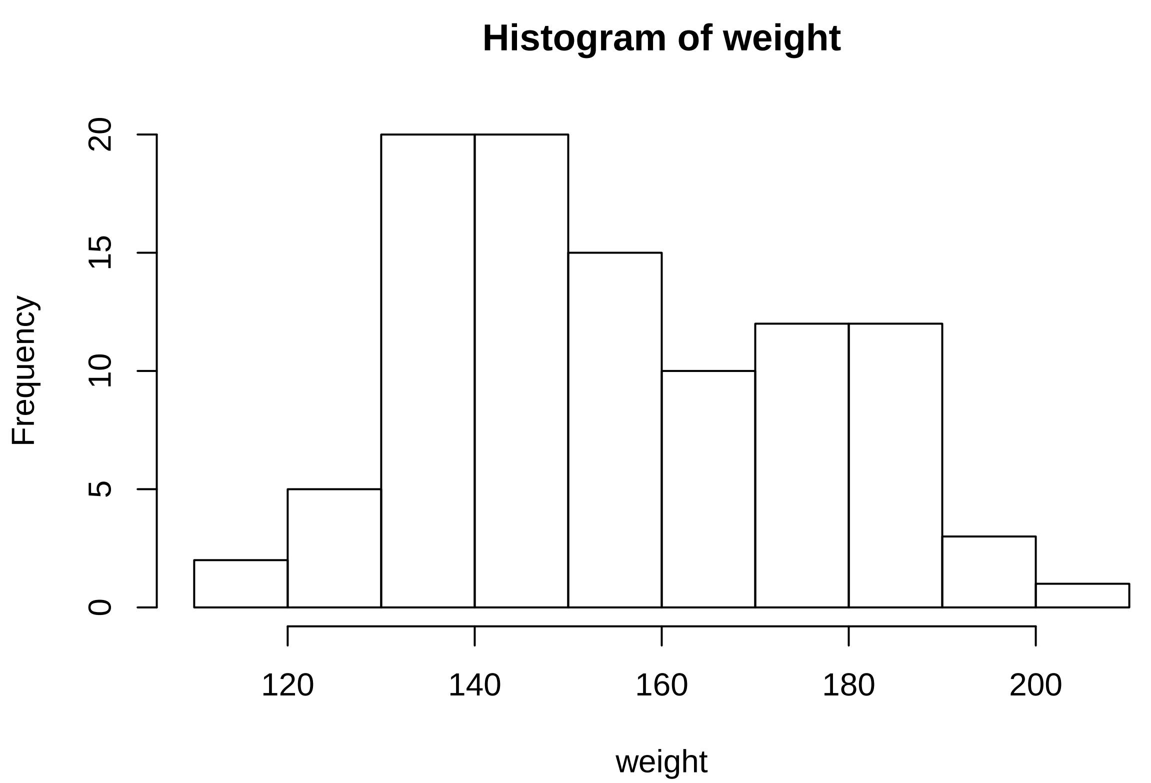
```
# Attaching the data frame
attach(dat)         # Attach the data frame
remote                  # Now we can refer directly to the variable without using $
```

```
##   [1]  5  7  3  2  5  0  5  9  1  8  7  7  8  9  5  3  6 10  7  7  5  7  4
##  [24]  8  1  4  8  2  0  5  2  9  8  1 10  8  5  9  1  9  2  9  7 10  8  4
##  [47]  1  4  1  4 31 32 27 37 28 27 30 35 31 27 26 31 32 37 35 31 29 34 30
##  [70] 30 35 30 37 29 23 31 35 28 21 33 27 23 36 30 32 20 22 35 34 27 34 18
##  [93] 31 29 30 28 29 34 23 25
```

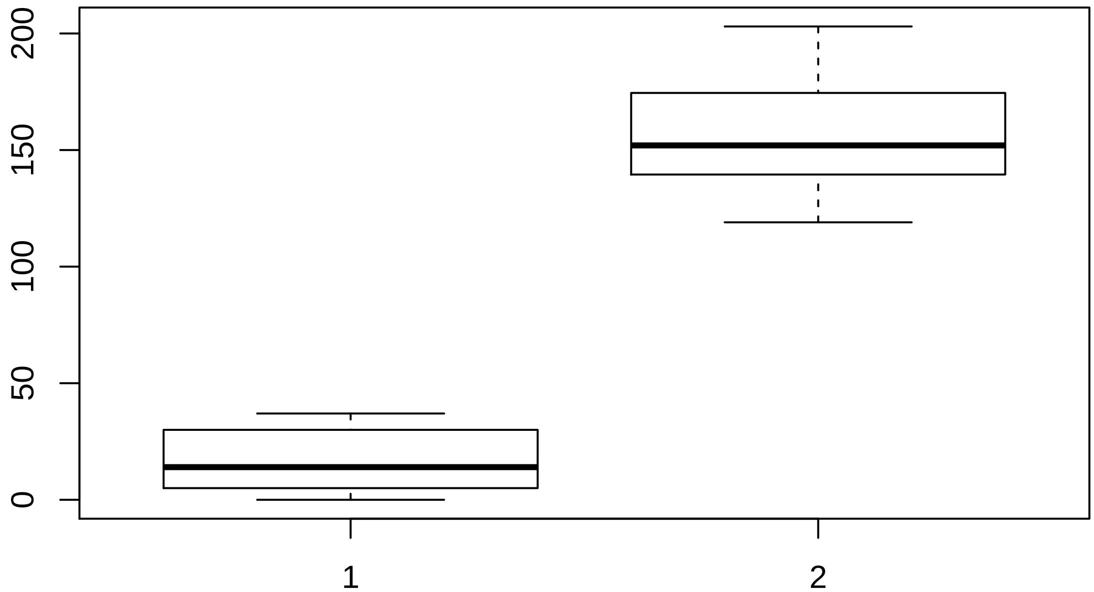
```r
# Basic Graphics
hist(remote)                    # Histogram of 'remote'
```



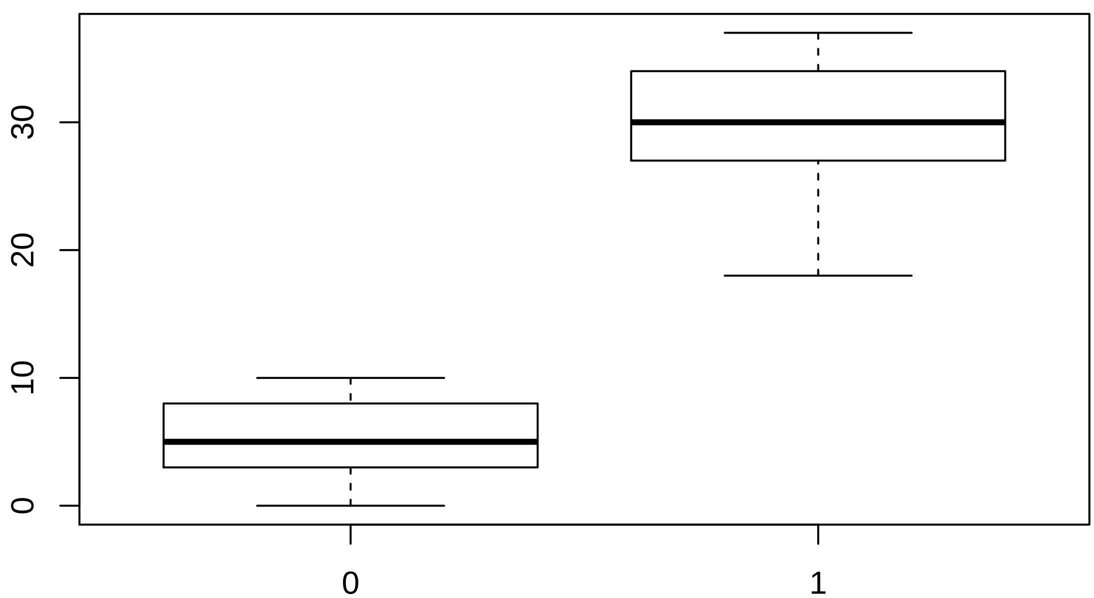

```r
hist(weight)                    # Histogram of 'weight'
```



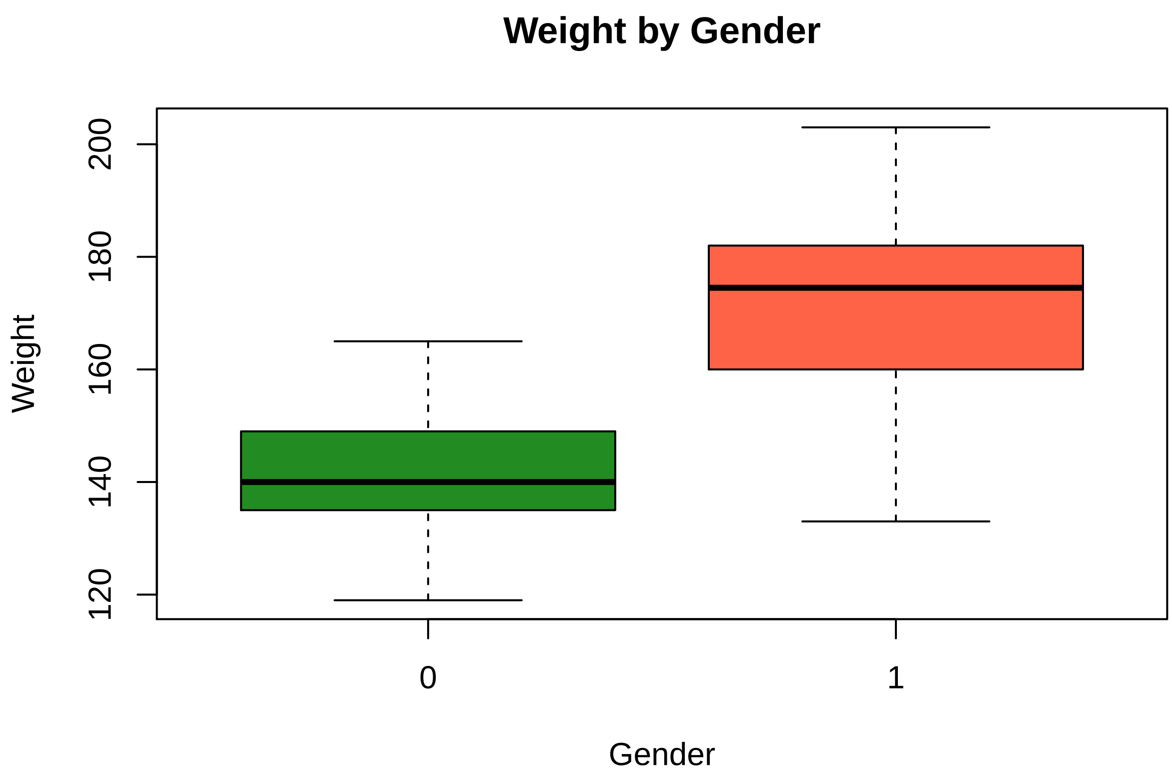
```r
boxplot(remote,weight)        # Boxplot of 'remote' and 'weight'
```




```r
boxplot(remote ~ gender)     # Boxplot of 'remote' conditioned on 'gender'
```




```r
boxplot(weight ~ gender, main = "Weight by Gender",
        xlab = "Gender", ylab = "Weight",
        col = c("forestgreen", "tomato"))    # Boxplot of 'weight' conditioned on 'gender'
```




```r
## Inferential statistics
cor(remote,weight)                # Run correlation coefficient
```

```
## [1] 0.8116673
```

```r
t.test(remote ~ gender)          # Did frequency of remote use differ by gender?
```

```
##
##  Welch Two Sample t-test
##
## data:  remote by gender
## t = -31.32, df = 84.845, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -25.92777 -22.83223
## sample estimates:
## mean in group 0 mean in group 1
##            5.40           29.78
```

```r
rem.t <- t.test(remote ~ gender)# Save results of last analysis
rem.t                        # Display analysis
```

```
##
##  Welch Two Sample t-test
##
## data:  remote by gender
## t = -31.32, df = 84.845, p-value < 2.2e-16
```

```
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -25.92777 -22.83223
## sample estimates:
## mean in group 0 mean in group 1
##            5.40           29.78
```

```r
names(rem.t)                # See the names of variables in the object rem.t
```

```
## [1] "statistic"   "parameter"   "p.value"     "conf.int"    "estimate"
## [6] "null.value"  "alternative" "method"      "data.name"
```

```r
rem.t$statistic         # See the statistics variable in the object rem.t
```

```
##         t
## -31.31951
```

```r
mod1 <- lm(remote ~ gender)     # Linear model, regressing 'remote' on 'gender'
anova(mod1)                # ANOVA table of the previous model
```

```
## Analysis of Variance Table
##
## Response: remote
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## gender     1 14859.6 14859.6  980.91 < 2.2e-16 ***
## Residuals 98  1484.6    15.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
anova(lm(remote ~ gender))      # You can combine the two steps in to one line
```

```
## Analysis of Variance Table
##
## Response: remote
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## gender     1 14859.6 14859.6  980.91 < 2.2e-16 ***
## Residuals 98  1484.6    15.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
mod2 <- lm(remote ~ weight)     # Model 'remote' as a linear function of 'weight'
mod3 <- lm(remote ~ weight + gender)  # Model 'remote' as a linear function of 'weight' & 'gender'
mod4 <- lm(remote ~ weight*gender)
# Equivalent to all main effects and interaction:
# lm(remote ~ weight + gender + weight*gender)
summary(mod3)              # See regression table for model 3 (remote ~ weight + gender)
```

```
##
## Call:
## lm(formula = remote ~ weight + gender)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.2651 -2.6495  0.1842  2.9608  6.1556
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -11.44900    4.02622  -2.844  0.00544 **
## weight        0.11948    0.02832   4.219 5.53e-05 ***
## gender1      20.69286    1.13190  18.282  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.596 on 97 degrees of freedom
## Multiple R-squared:  0.9232, Adjusted R-squared:  0.9217
## F-statistic: 583.4 on 2 and 97 DF,  p-value: < 2.2e-16
```

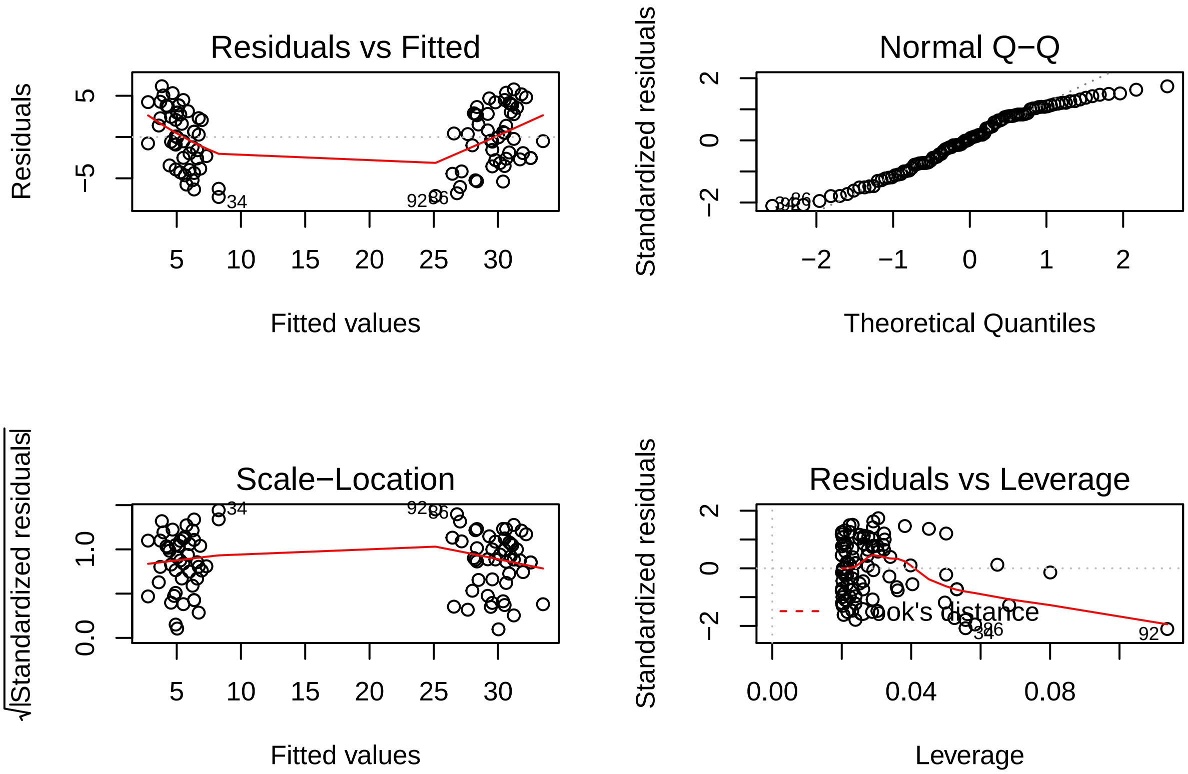
```
summary(mod4)            # See regression table for model 4 (remote ~ weight*gender)
```

```
##
## Call:
## lm(formula = remote ~ weight * gender)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.8530 -2.7707  0.0422  2.5473  5.0332
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     16.72442    6.32943   2.642  0.00962 **
## weight          -0.08030    0.04477  -1.794  0.07601 .
## gender1        -22.96669    8.18338  -2.807  0.00606 **
## weight:gender1   0.28988    0.05393   5.375 5.36e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.169 on 96 degrees of freedom
## Multiple R-squared:  0.941,  Adjusted R-squared:  0.9392
## F-statistic: 510.4 on 3 and 96 DF,  p-value: < 2.2e-16
```

```
anova(mod3,mod4)        # Prints ANOVA table comparing model 3 to model 4 (delta F)
```

```
## Analysis of Variance Table
##
## Model 1: remote ~ weight + gender
## Model 2: remote ~ weight * gender
##   Res.Df     RSS Df Sum of Sq      F    Pr(>F)
## 1     97 1254.43
## 2     96  964.23  1     290.2 28.893 5.359e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Regression diagnostics
mod3.1 <- lm(remote ~ weight + gender)  # Gives you regression diagnostics
par(mfrow=c(2,2))        # Set up plotting region for a 2x2 grid
plot(mod3.1)               # Plot the regression diagnostics (R knows automatically to do this)
```




```r
## Saving the graphs as PDF
pdf("prettygraph.pdf")  # Turn on the PDF device and open a blank file called "prettygraph.ps"
plot(mod3.1)            # Plot the model
dev.off()               # Turn off the postscript device
```

```
## pdf
##   2
```

```r
#############################################################
## Intro to Simulation and Functions with R:
#############################################################

## Create a vector of 12 random numbers drawn
## from a uniform distribution over the
## interval between 0 and 1:
z <- runif(12) # Generates 12 observations from Unif(0,1)

z
```

```
##  [1] 0.4556848 0.9854712 0.4200237 0.8291504 0.4703348 0.2857762 0.6584719
##  [8] 0.2546510 0.9307130 0.5630982 0.6581636 0.0478349
```

```r
## We can see which of these is less than 0.5 with the expression "z < 0.5"
z < 0.5
```

```
##  [1]  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE FALSE FALSE
## [12]  TRUE
```

```r
## R identifies "True" with "1" and "False" with "0":
as.numeric(z < 0.5)
```

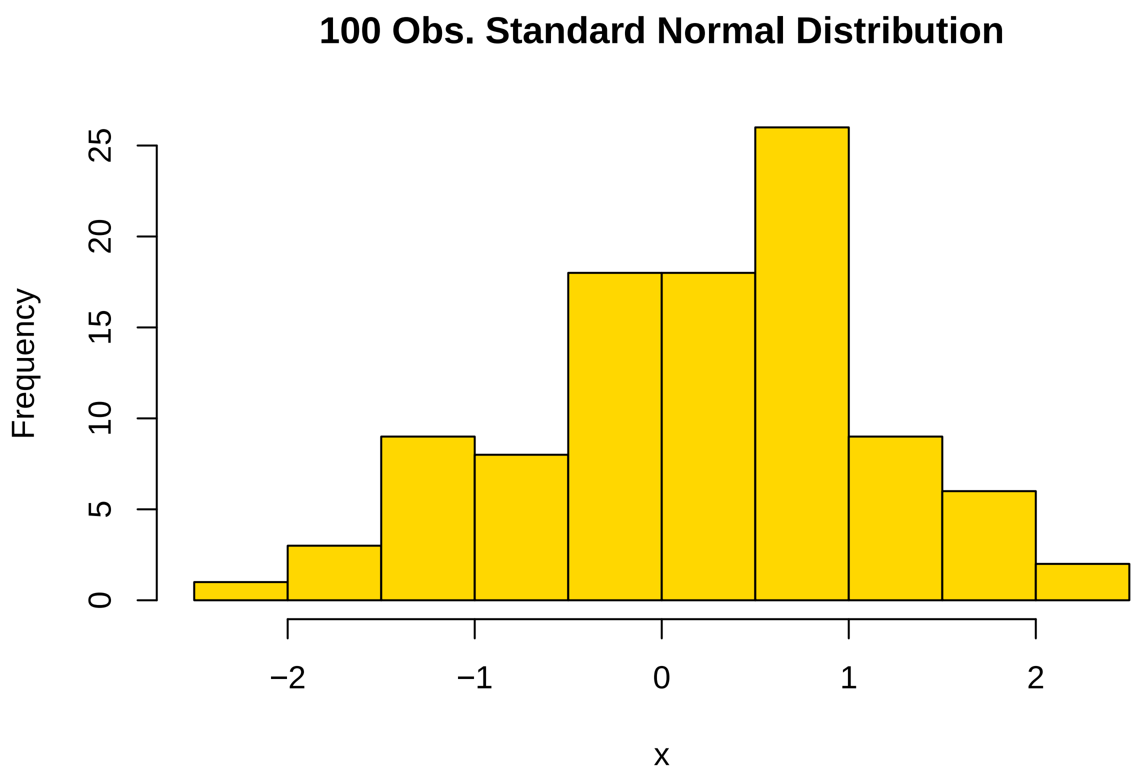
```
##  [1] 1 0 1 0 1 1 0 1 0 0 0 1
```

```r
## Now let x be a vector of 100 random draws from a "Normal" distribution:
x <- rnorm(100) # Generates 100 random normal observations, mean 0 sd 1

x
```

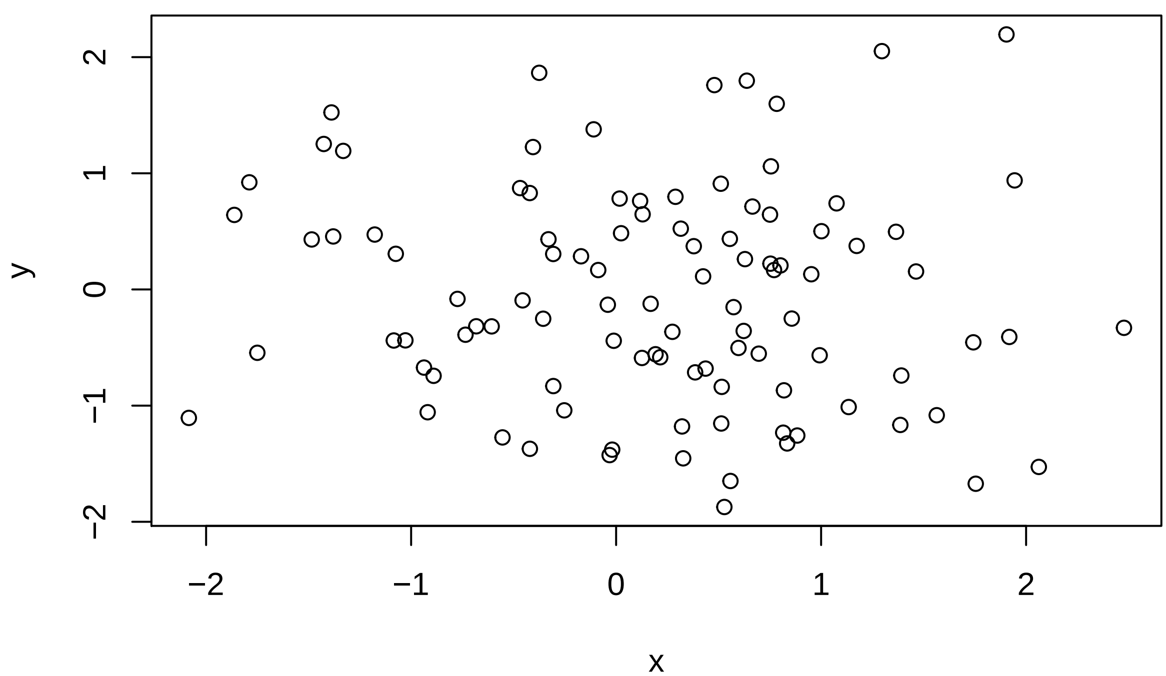
```
##   [1] -0.60707335 -0.33005169  0.62230989  0.38551562  0.31543422
##   [6]  2.47740446 -1.86243466  1.39123147  0.43647982 -0.73471005
##  [11]  0.75510995 -1.38845849  0.78355418  1.13443368  1.90415334
##  [16] -2.08424796  0.63740099  0.57301829  1.07548641  0.16850846
##  [21]  0.37881990  0.95210272  1.56399504  1.94397236  0.62846865
##  [26]  0.42433338 -0.01904825  0.12624928 -0.91928159 -0.25292838
##  [31]  0.51290226 -0.01157962 -1.17750323 -0.89039806  1.46312738
##  [36] -1.42614654  1.17351646  1.91803359 -1.37987164 -0.45621027
##  [41] -0.30672590 -0.10949010  0.51073777  0.80178036  0.19224516
##  [46] -0.04049865  0.59691799 -0.42127898 -0.42071509 -0.30627443
##  [51]  1.75439433  0.55773478  0.75073092 -0.35571323 -0.55421789
##  [56]  2.06204030 -1.02788681  0.32160627 -1.78920299  0.28950207
##  [61] -0.17114089  1.38651570  0.52806083  0.77039366  0.51529084
##  [66]  1.29652106 -0.68216217 -1.08444008 -0.93716872 -0.77377716
##  [71]  0.69587054  0.81513938 -1.07469060  0.12932937  0.11704900
##  [76]  0.75277881 -0.40551496 -0.37512544 -0.46837951  0.21524363
##  [81]  0.27444701 -0.08730682 -1.48489614  0.85683601 -1.33118827
##  [86]  0.83443599  1.36535987  0.01713701  0.81882825 -0.03155508
##  [91]  0.88367911  0.66498246 -1.75013434  0.55482368  0.02422135
##  [96]  0.99295549  1.74252547  0.47925919  1.00161752  0.32721646
```

```r
par(mfrow=c(1,1)) #Resets the graphics to one plot per page
hist(x, main= "100 Obs. Standard Normal Distribution",
     breaks = 10, col = "gold") # Create a histogram of the vector x
```



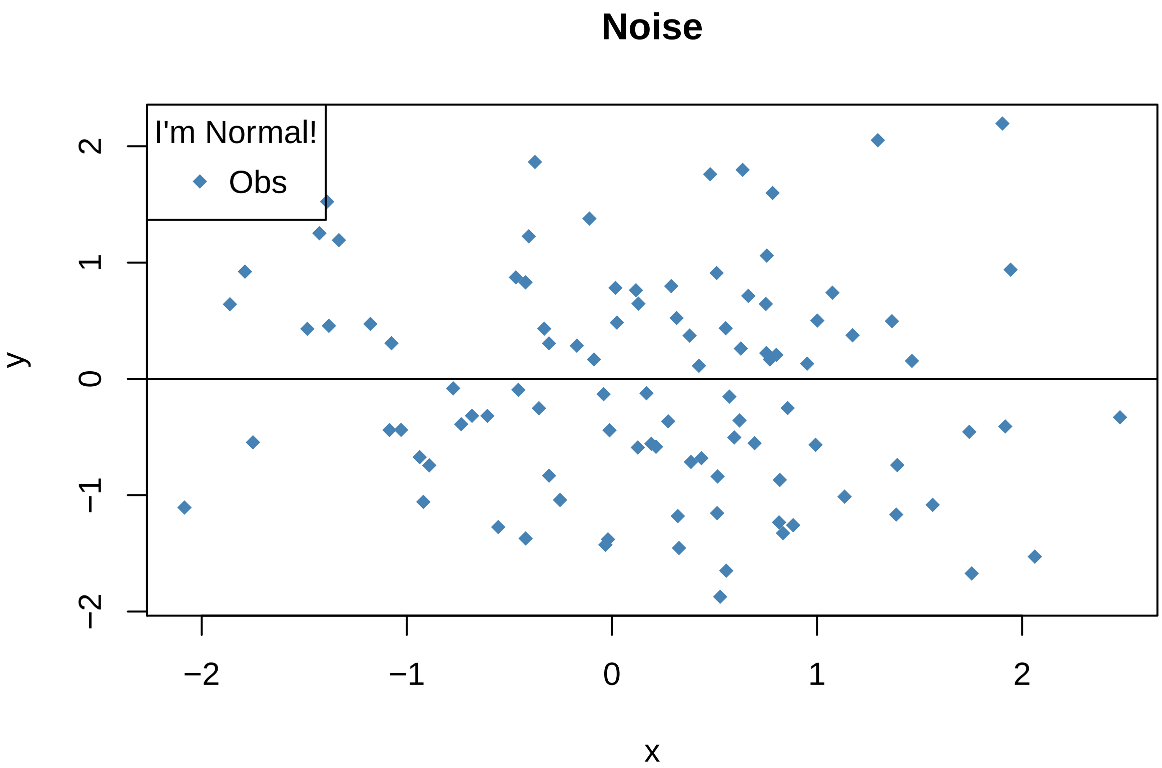

```
y <- rnorm(100) # save 100 random sample from a standard normal distribution to y

plot(x,y)
```

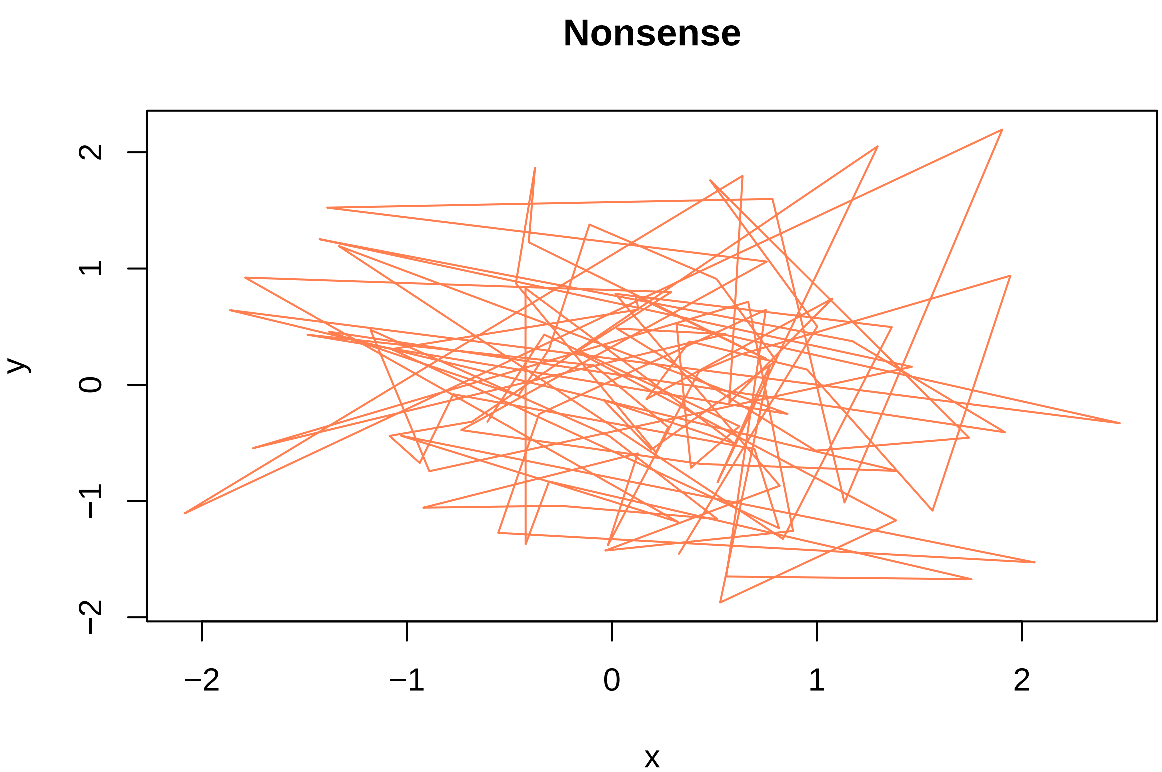



```
plot(x,y,main = "Noise", col = "steelblue", pch = 18)
abline(h=0)
legend("topleft", title = "I'm Normal!", "Obs", pch= 18,
```

```
      col = "steelblue")
```




```
plot(x,y,type="l", main = "Nonsense", col = "coral")
```



```r
#?plot



# Sampling uniformly at random, with replacement
v <- sample(1:10,100,replace=T) # Samples from 1 to 10, 100 times

v
```

```
##   [1] 10  7  1  3  1 10  6  6  9  8  2  5  6  1  3  8  3  8  4  6  4  9  7
##  [24]  7  8  8  9  7 10  4  5  7  6  8  1  5  2  7  4  2  1 10 10  8 10  5
##  [47]  3  9  3  2  8  5  9  1  3  6 10  1 10  5  6  7  5  8  1  6  1  1  2
##  [70]  2  7  9  1  1  3  1  4  7  1  4  2  6  1  7  1  4  7  8  1  4  6  2
##  [93] 10  4  2  2  3  6  3 10
```

```r
table(v)
```

```
## v
##  1  2  3  4  5  6  7  8  9 10
## 17 10  9  9  7 11 11 10  6 10
```

```r
## Functions:
## Numerical calculations for birthday problem:
k <- 40

top <- seq(365,length=k,by=-1) # Creates a vector of 365 to 365-k

bottom <- rep(365,k) # Creates a vector filled with 365 repeated k times

top
```

```
##  [1] 365 364 363 362 361 360 359 358 357 356 355 354 353 352 351 350 349
## [18] 348 347 346 345 344 343 342 341 340 339 338 337 336 335 334 333 332
## [35] 331 330 329 328 327 326
```

```r
bottom
```

```
##  [1] 365 365 365 365 365 365 365 365 365 365 365 365 365 365 365 365 365
## [18] 365 365 365 365 365 365 365 365 365 365 365 365 365 365 365 365 365
## [35] 365 365 365 365 365 365
```

```r
top/bottom
```

```
##  [1] 1.0000000 0.9972603 0.9945205 0.9917808 0.9890411 0.9863014 0.9835616
##  [8] 0.9808219 0.9780822 0.9753425 0.9726027 0.9698630 0.9671233 0.9643836
## [15] 0.9616438 0.9589041 0.9561644 0.9534247 0.9506849 0.9479452 0.9452055
## [22] 0.9424658 0.9397260 0.9369863 0.9342466 0.9315068 0.9287671 0.9260274
## [29] 0.9232877 0.9205479 0.9178082 0.9150685 0.9123288 0.9095890 0.9068493
## [36] 0.9041096 0.9013699 0.8986301 0.8958904 0.8931507
```

```r
prod(top/bottom)  # This is the prob of NO birthday match
```

```
## [1] 0.1087682
```

```r
1 - prod(top/bottom)  # This is the prob of having a birthday match
```

```
## [1] 0.8912318
```

```r
## Let's make a function out of what we just did:

bday <- function(k){ # k is the variable
  top <- seq(365,length=k,by=-1)
  bottom <- rep(365,k)
  return(1-prod(top/bottom))
}

bday(40)
```

```
## [1] 0.8912318
```

```r
##########################################################
## Intro to for loops in R:
##########################################################

s <- 0
for(i in 1:100){
  s <- s+i
}
s
```

```
## [1] 5050
```

```r
## Sometimes you can do the same thing without a loop:
sum(1:100)
```

```
## [1] 5050
```

```r
## You can have more commands in the body of the loop:
s <- 0
for(i in 1:10){
  s <- s+i
  cat("When i = ", i, ", s = ",s, "\n",sep="") # "cat" prints things
}
```

```
## When i = 1, s = 1
## When i = 2, s = 3
## When i = 3, s = 6
## When i = 4, s = 10
## When i = 5, s = 15
## When i = 6, s = 21
```

```
## When i = 7, s = 28
## When i = 8, s = 36
## When i = 9, s = 45
## When i = 10, s = 55
```

```r
s <- 0
for(i in 1:10){
  s <- s+i
  cat("When i = ", i, ", s = ",s, "\n",sep="")
  remainder2 <- (i %% 2)
  twos <- i/2
  if(remainder2 == 0){
    cat("I'm getting", rep("really", twos),"tired!\n")
  }
}
```

```
## When i = 1, s = 1
## When i = 2, s = 3
## I'm getting really tired!
## When i = 3, s = 6
## When i = 4, s = 10
## I'm getting really really tired!
## When i = 5, s = 15
## When i = 6, s = 21
## I'm getting really really really tired!
## When i = 7, s = 28
## When i = 8, s = 36
## I'm getting really really really really tired!
## When i = 9, s = 45
## When i = 10, s = 55
## I'm getting really really really really really tired!
```