EDA heart attack analysis prediction

Exploratory Data Analysis on heart attack analysis prediction

By : Vikas Gowda M N

USN : 1NT20SDS18

Github: https://github.com/Vikas-viky/R-Commands.git (https://github.com/Vikas-viky/R-Commands.git)

Data loading

```
df<-read.csv("C:/Users/Vikas/Documents/EDA 1/heart.CSV")
head(df)
```

```
##   age sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall output
## 1  63   1  3    145  233   1       0      150    0     2.3   0   0     1      1
## 2  37   1  2    130  250   0       1      187    0     3.5   0   0     2      1
## 3  41   0  1    130  204   0       0      172    0     1.4   2   0     2      1
## 4  56   1  1    120  236   0       1      178    0     0.8   2   0     2      1
## 5  57   0  0    120  354   0       1      163    1     0.6   2   0     2      1
## 6  57   1  0    140  192   0       1      148    0     0.4   1   0     1      1
```

The type of object can be ascertained using the class() command

```
class(df)
```

```
## [1] "data.frame"
```

```
class(df$age)
```

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

```
class(dimnames(df))
```

```
## [1] "list"
```

Shows the top of the data object and by default shows the first six rows:

```
head(df)
```

```
##   age sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall output
## 1  63   1  3    145  233   1       0      150    0     2.3   0   0     1      1
## 2  37   1  2    130  250   0       1      187    0     3.5   0   0     2      1
## 3  41   0  1    130  204   0       0      172    0     1.4   2   0     2      1
## 4  56   1  1    120  236   0       1      178    0     0.8   2   0     2      1
## 5  57   0  0    120  354   0       1      163    1     0.6   2   0     2      1
## 6  57   1  0    140  192   0       1      148    0     0.4   1   0     1      1
```

Can elect to show a different number of rows using the n = instruction like so:

```
head(df, n=3)
```

```
##   age sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall output
## 1  63   1  3    145  233   1       0      150    0     2.3   0   0     1      1
## 2  37   1  2    130  250   0       1      187    0     3.5   0   0     2      1
## 3  41   0  1    130  204   0       0      172    0     1.4   2   0     2      1
```

can also display the bottom of the data using the tail() command default shows the last six rows:

```
tail(df)
```

```
##     age sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall
## 298  59   1  0    164  176   1       0       90    0     1.0   1   2     1
## 299  57   0  0    140  241   0       1      123    1     0.2   1   0     3
## 300  45   1  3    110  264   0       1      132    0     1.2   1   0     3
## 301  68   1  0    144  193   1       1      141    0     3.4   1   2     3
## 302  57   1  0    130  131   0       1      115    1     1.2   1   1     3
## 303  57   0  1    130  236   0       0      174    0     0.0   1   1     2
##     output
## 298      0
## 299      0
## 300      0
## 301      0
## 302      0
## 303      0
```

Can select to show a different number of rows using the n = instruction like so:

```
tail(df, n=4)
```

```
##     age sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall
## 300  45   1  3    110  264   0       1      132    0     1.2   1   0     3
## 301  68   1  0    144  193   1       1      141    0     3.4   1   2     3
## 302  57   1  0    130  131   0       1      115    1     1.2   1   1     3
## 303  57   0  1    130  236   0       0      174    0     0.0   1   1     2
##     output
## 300      0
## 301      0
## 302      0
## 303      0
```

##To get information about an Data frame and particular columns:

```
summary(df)
```

```
##       age             sex               cp            trtbps     
##  Min.   :29.00   Min.   :0.0000   Min.   :0.000   Min.   : 94.0  
##  1st Qu.:47.50   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:120.0  
##  Median :55.00   Median :1.0000   Median :1.000   Median :130.0  
##  Mean   :54.37   Mean   :0.6832   Mean   :0.967   Mean   :131.6  
##  3rd Qu.:61.00   3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:140.0  
##  Max.   :77.00   Max.   :1.0000   Max.   :3.000   Max.   :200.0  
##       chol            fbs            restecg          thalachh    
##  Min.   :126.0   Min.   :0.0000   Min.   :0.0000   Min.   : 71.0  
##  1st Qu.:211.0   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:133.5  
##  Median :240.0   Median :0.0000   Median :1.0000   Median :153.0  
##  Mean   :246.3   Mean   :0.1485   Mean   :0.5281   Mean   :149.6  
##  3rd Qu.:274.5   3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:166.0  
##  Max.   :564.0   Max.   :1.0000   Max.   :2.0000   Max.   :202.0  
##       exng           oldpeak          slp             caa        
##  Min.   :0.0000   Min.   :0.00   Min.   :0.000   Min.   :0.0000  
##  1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:1.000   1st Qu.:0.0000  
##  Median :0.0000   Median :0.80   Median :1.000   Median :0.0000  
##  Mean   :0.3267   Mean   :1.04   Mean   :1.399   Mean   :0.7294  
##  3rd Qu.:1.0000   3rd Qu.:1.60   3rd Qu.:2.000   3rd Qu.:1.0000  
##  Max.   :1.0000   Max.   :6.20   Max.   :2.000   Max.   :4.0000  
##      thall           output      
##  Min.   :0.000   Min.   :0.0000  
##  1st Qu.:2.000   1st Qu.:0.0000  
##  Median :2.000   Median :1.0000  
##  Mean   :2.314   Mean   :0.5446  
##  3rd Qu.:3.000   3rd Qu.:1.0000  
##  Max.   :3.000   Max.   :1.0000  
```

```
summary(df$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   29.00   47.50   55.00   54.37   61.00   77.00 
```

```
mean(df$age)
```

```
## [1] 54.36634
```

```
median(df$age)
```

```
## [1] 55
```

Shows the median absolute deviation

```
mad(df$trtbps)
```

```
## [1] 14.826
```

```
mode(df$trtbps)
```

```
## [1] "numeric"
```

```
var(df$trtbps)
```

```
## [1] 307.5865
```

```
sd(df$trtbps)
```

```
## [1] 17.53814
```

```
quantile(df$trtbps)
```

```
##   0%  25%  50%  75% 100% 
##   94  120  130  140  200
```

Returns Tukey’s five number summary (minimum, lower-hinge, median, upper-hinge, maximum) for the input data.

```
fivenum(df$trtbps)
```

```
## [1]  94 120 130 140 200
```

F Test to Compare Two Variances

```
var.test(df$age, df$trtbps)
```

```
##
##  F test to compare two variances
##
## data:  df$age and df$trtbps
## F = 0.26817, num df = 302, denom df = 302, p-value < 2.2e-16
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.2139232 0.3361653
## sample estimates:
## ratio of variances
##          0.2681671
```

str() command is useful to see the object structure

```
str(df)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : int  63 37 41 56 57 57 56 44 52 57 ...
##  $ sex     : int  1 1 0 1 0 1 0 1 1 1 ...
##  $ cp      : int  3 2 1 1 0 0 1 1 2 2 ...
##  $ trtbps  : int  145 130 130 120 120 140 140 120 172 150 ...
##  $ chol    : int  233 250 204 236 354 192 294 263 199 168 ...
##  $ fbs     : int  1 0 0 0 0 0 0 0 1 0 ...
##  $ restecg : int  0 1 0 1 1 1 0 1 1 1 ...
##  $ thalachh: int  150 187 172 178 163 148 153 173 162 174 ...
##  $ exng    : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ oldpeak : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
##  $ slp     : int  0 0 2 2 2 1 1 2 2 2 ...
##  $ caa     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ thall   : int  1 2 2 2 2 1 2 3 3 2 ...
##  $ output  : int  1 1 1 1 1 1 1 1 1 1 ...
```

structure returns the given object with further attributes set

```
head(structure(df))
```

```
##   age sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall output
## 1  63   1  3    145  233   1       0      150    0     2.3   0   0     1      1
## 2  37   1  2    130  250   0       1      187    0     3.5   0   0     2      1
## 3  41   0  1    130  204   0       0      172    0     1.4   2   0     2      1
## 4  56   1  1    120  236   0       1      178    0     0.8   2   0     2      1
## 5  57   0  0    120  354   0       1      163    1     0.6   2   0     2      1
## 6  57   1  0    140  192   0       1      148    0     0.4   1   0     1      1
```

can also look at all the named objects you have at once using the ls.str() command

```
ls.str(df)
```

```
## age :  int [1:303] 63 37 41 56 57 57 56 44 52 57 ...
## caa :  int [1:303] 0 0 0 0 0 0 0 0 0 0 ...
## chol :  int [1:303] 233 250 204 236 354 192 294 263 199 168 ...
## cp :  int [1:303] 3 2 1 1 0 0 1 1 2 2 ...
## exng :  int [1:303] 0 0 0 0 1 0 0 0 0 0 ...
## fbs :  int [1:303] 1 0 0 0 0 0 0 0 1 0 ...
## oldpeak :  num [1:303] 2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
## output :  int [1:303] 1 1 1 1 1 1 1 1 1 1 ...
## restecg :  int [1:303] 0 1 0 1 1 1 0 1 1 1 ...
## sex :  int [1:303] 1 1 0 1 0 1 0 1 1 1 ...
## slp :  int [1:303] 0 0 2 2 2 1 1 2 2 2 ...
## thalachh :  int [1:303] 150 187 172 178 163 148 153 173 162 174 ...
## thall :  int [1:303] 1 2 2 2 2 1 2 3 3 2 ...
## trtbps :  int [1:303] 145 130 130 120 120 140 140 120 172 150 ...
```

can use the pattern = instruction to narrow down your focus

```
ls.str(pattern = 'df')
```

```
## df : 'data.frame':   303 obs. of  14 variables:
##  $ age     : int  63 37 41 56 57 57 56 44 52 57 ...
##  $ sex     : int  1 1 0 1 0 1 0 1 1 1 ...
##  $ cp      : int  3 2 1 1 0 0 1 1 2 2 ...
##  $ trtbps  : int  145 130 130 120 120 140 140 120 172 150 ...
##  $ chol    : int  233 250 204 236 354 192 294 263 199 168 ...
##  $ fbs     : int  1 0 0 0 0 0 0 0 1 0 ...
##  $ restecg : int  0 1 0 1 1 1 0 1 1 1 ...
##  $ thalachh: int  150 187 172 178 163 148 153 173 162 174 ...
##  $ exng    : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ oldpeak : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
##  $ slp     : int  0 0 2 2 2 1 1 2 2 2 ...
##  $ caa     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ thall   : int  1 2 2 2 2 1 2 3 3 2 ...
##  $ output  : int  1 1 1 1 1 1 1 1 1 1 ...
```

The most basic command that enables the viewing of column or row is:

```
names(df)
```

```
##  [1] "age"      "sex"      "cp"       "trtbps"   "chol"     "fbs"     
##  [7] "restecg"  "thalachh" "exng"     "oldpeak"  "slp"      "caa"     
## [13] "thall"    "output"
```

```
row.names(df)
```

```
##   [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"   "10"  "11"  "12" 
##  [13] "13"  "14"  "15"  "16"  "17"  "18"  "19"  "20"  "21"  "22"  "23"  "24" 
##  [25] "25"  "26"  "27"  "28"  "29"  "30"  "31"  "32"  "33"  "34"  "35"  "36" 
##  [37] "37"  "38"  "39"  "40"  "41"  "42"  "43"  "44"  "45"  "46"  "47"  "48" 
##  [49] "49"  "50"  "51"  "52"  "53"  "54"  "55"  "56"  "57"  "58"  "59"  "60" 
##  [61] "61"  "62"  "63"  "64"  "65"  "66"  "67"  "68"  "69"  "70"  "71"  "72" 
##  [73] "73"  "74"  "75"  "76"  "77"  "78"  "79"  "80"  "81"  "82"  "83"  "84" 
##  [85] "85"  "86"  "87"  "88"  "89"  "90"  "91"  "92"  "93"  "94"  "95"  "96" 
##  [97] "97"  "98"  "99"  "100" "101" "102" "103" "104" "105" "106" "107" "108"
## [109] "109" "110" "111" "112" "113" "114" "115" "116" "117" "118" "119" "120"
## [121] "121" "122" "123" "124" "125" "126" "127" "128" "129" "130" "131" "132"
## [133] "133" "134" "135" "136" "137" "138" "139" "140" "141" "142" "143" "144"
## [145] "145" "146" "147" "148" "149" "150" "151" "152" "153" "154" "155" "156"
## [157] "157" "158" "159" "160" "161" "162" "163" "164" "165" "166" "167" "168"
## [169] "169" "170" "171" "172" "173" "174" "175" "176" "177" "178" "179" "180"
## [181] "181" "182" "183" "184" "185" "186" "187" "188" "189" "190" "191" "192"
## [193] "193" "194" "195" "196" "197" "198" "199" "200" "201" "202" "203" "204"
## [205] "205" "206" "207" "208" "209" "210" "211" "212" "213" "214" "215" "216"
## [217] "217" "218" "219" "220" "221" "222" "223" "224" "225" "226" "227" "228"
## [229] "229" "230" "231" "232" "233" "234" "235" "236" "237" "238" "239" "240"
## [241] "241" "242" "243" "244" "245" "246" "247" "248" "249" "250" "251" "252"
## [253] "253" "254" "255" "256" "257" "258" "259" "260" "261" "262" "263" "264"
## [265] "265" "266" "267" "268" "269" "270" "271" "272" "273" "274" "275" "276"
## [277] "277" "278" "279" "280" "281" "282" "283" "284" "285" "286" "287" "288"
## [289] "289" "290" "291" "292" "293" "294" "295" "296" "297" "298" "299" "300"
## [301] "301" "302" "303"
```

length() command used to determine the number of items in an object

```
length(df)
```

```
## [1] 14
```

To extract a particular column and particular row values

```
head(df$age)
```

```
## [1] 63 37 41 56 57 57
```

```
df$trtbps[3]
```

```
## [1] 130
```

```
df$age[1:5]
```

```
## [1] 63 37 41 56 57
```

```
tdf = attach(df)
tdf
```

```
## <environment: 0x0000000013362610>
## attr(,"name")
## [1] "df"
```

The max() and min() commands display the largest and smallest values in a numeric object

```
max(df$age)
```

```
## [1] 77
```

```
max(df$trtbps)
```

```
## [1] 200
```

```
min(df$age)
```

```
## [1] 29
```

```
min(df$trtbps)
```

```
## [1] 94
```

```
head(stack(df))
```

```
##   values ind
## 1     63 age
## 2     37 age
## 3     41 age
## 4     56 age
## 5     57 age
## 6     57 age
```

Sorting default is ascending order

```
head(sort(df$age))
```

```
## [1] 29 34 34 35 35 35
```

```
head(sort(df$trtbps))
```

```
## [1]  94  94 100 100 100 100
```

can get an index using the order() command. This uses the same instructions as the sort() command, but tells you the position of each item along the vector:

```
head(order(df$age))
```

```
## [1]  73  59 126  66 158 228
```

The rank() command gives the rank number like order() in a slightly different manner when the values are same the ranks are shared between them which is not in order()

```
head(rank(df$cp), n=10)
```

```
##  [1] 292.0 237.0 168.5 168.5  72.0  72.0 168.5 168.5 237.0 237.0
```

Displaying selected rows & columns

```
df[3,3]
```

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

```
df[3, 1:5]
```

```
##   age sex cp trtbps chol
## 3  41   0  1    130  204
```

To convert from Data frame from matrix

```
head(as.matrix(df))
```

```
##      age sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall
## [1,]  63   1  3    145  233   1       0      150    0     2.3   0   0     1
## [2,]  37   1  2    130  250   0       1      187    0     3.5   0   0     2
## [3,]  41   0  1    130  204   0       0      172    0     1.4   2   0     2
## [4,]  56   1  1    120  236   0       1      178    0     0.8   2   0     2
## [5,]  57   0  0    120  354   0       1      163    1     0.6   2   0     2
## [6,]  57   1  0    140  192   0       1      148    0     0.4   1   0     1
##      output
## [1,]      1
## [2,]      1
## [3,]      1
## [4,]      1
## [5,]      1
## [6,]      1
```

The cbind() and rbind() commands assemble a matrix/data frames, by columns or rows, from several other object

```
head(rbind(df))
```

```
##   age sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall output
## 1  63   1  3    145  233   1       0      150    0     2.3   0   0     1      1
## 2  37   1  2    130  250   0       1      187    0     3.5   0   0     2      1
## 3  41   0  1    130  204   0       0      172    0     1.4   2   0     2      1
## 4  56   1  1    120  236   0       1      178    0     0.8   2   0     2      1
## 5  57   0  0    120  354   0       1      163    1     0.6   2   0     2      1
## 6  57   1  0    140  192   0       1      148    0     0.4   1   0     1      1
```

```
head(cbind(df))
```

```
##   age sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall output
## 1  63   1  3    145  233   1       0      150    0     2.3   0   0     1      1
## 2  37   1  2    130  250   0       1      187    0     3.5   0   0     2      1
## 3  41   0  1    130  204   0       0      172    0     1.4   2   0     2      1
## 4  56   1  1    120  236   0       1      178    0     0.8   2   0     2      1
## 5  57   0  0    120  354   0       1      163    1     0.6   2   0     2      1
## 6  57   1  0    140  192   0       1      148    0     0.4   1   0     1      1
```

Simple Cumulative Commands

```
head(cumsum(df$thall))
```

```
## [1]  1  3  5  7  9 10
```

```
head(cummax(df$thall))
```

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

```
head(cummin(df$thall))
```

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

The cumulative product

```
head(cumprod(df$thall))
```

```
## [1]  1  2  4  8 16 16
```

The table() command enables you to specify which columns of data you want to use to create your contingency table

```
table(df$thall)
```

```
## 
##   0   1   2   3 
##   2  18 166 117
```

```
head(as.table(df$thall))
```

```
## A B C D E F
## 1 2 2 2 2 1
```

To check whether the data extracted is data frame or not

```
is.data.frame(df)
```

```
## [1] TRUE
```

```
is.table(df)
```

```
## [1] FALSE
```

The class() command can form the basis of a logical test by using the if() command in the following manner:

```
if(class(df) == 'data.frame') TRUE else FALSE
```

```
## [1] TRUE
```

The seq_along() command creates a simple index

```
seq_along(df)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14
```

Create a basic stem and leaf plot using the stem() command

```
stem(df$age)
```

```
## 
##   The decimal point is at the |
## 
##   28 | 0
##   30 | 
##   32 | 
##   34 | 000000
##   36 | 00
##   38 | 0000000
##   40 | 0000000000000
##   42 | 0000000000000000
##   44 | 0000000000000000000
##   46 | 000000000000
##   48 | 000000000000
##   50 | 0000000000000000000
##   52 | 000000000000000000000
##   54 | 000000000000000000000000
##   56 | 0000000000000000000000000000
##   58 | 000000000000000000000000000000000
##   60 | 0000000000000000000
##   62 | 00000000000000000000
##   64 | 000000000000000000
##   66 | 0000000000000000
##   68 | 0000000
##   70 | 0000000
##   72 | 
##   74 | 0
##   76 | 00
```

Now increase the number of bins used by adding a scale = 2 instruction

```
stem(df$age, scale = 2)
```

```
##
##   The decimal point is at the |
##
##   29 | 0
##   30 |
##   31 |
##   32 |
##   33 |
##   34 | 00
##   35 | 0000
##   36 |
##   37 | 00
##   38 | 000
##   39 | 0000
##   40 | 000
##   41 | 0000000000
##   42 | 00000000
##   43 | 00000000
##   44 | 00000000000
##   45 | 00000000
##   46 | 0000000
##   47 | 00000
##   48 | 0000000
##   49 | 00000
##   50 | 0000000
##   51 | 000000000000
##   52 | 0000000000000
##   53 | 00000000
##   54 | 0000000000000000
##   55 | 00000000
##   56 | 00000000000
##   57 | 00000000000000000
##   58 | 0000000000000000000
##   59 | 00000000000000
##   60 | 00000000000
##   61 | 00000000
##   62 | 00000000000
##   63 | 000000000
##   64 | 0000000000
##   65 | 00000000
##   66 | 0000000
##   67 | 000000000
##   68 | 0000
##   69 | 000
##   70 | 0000
##   71 | 000
##   72 |
##   73 |
##   74 | 0
##   75 |
##   76 | 0
##   77 | 0
```

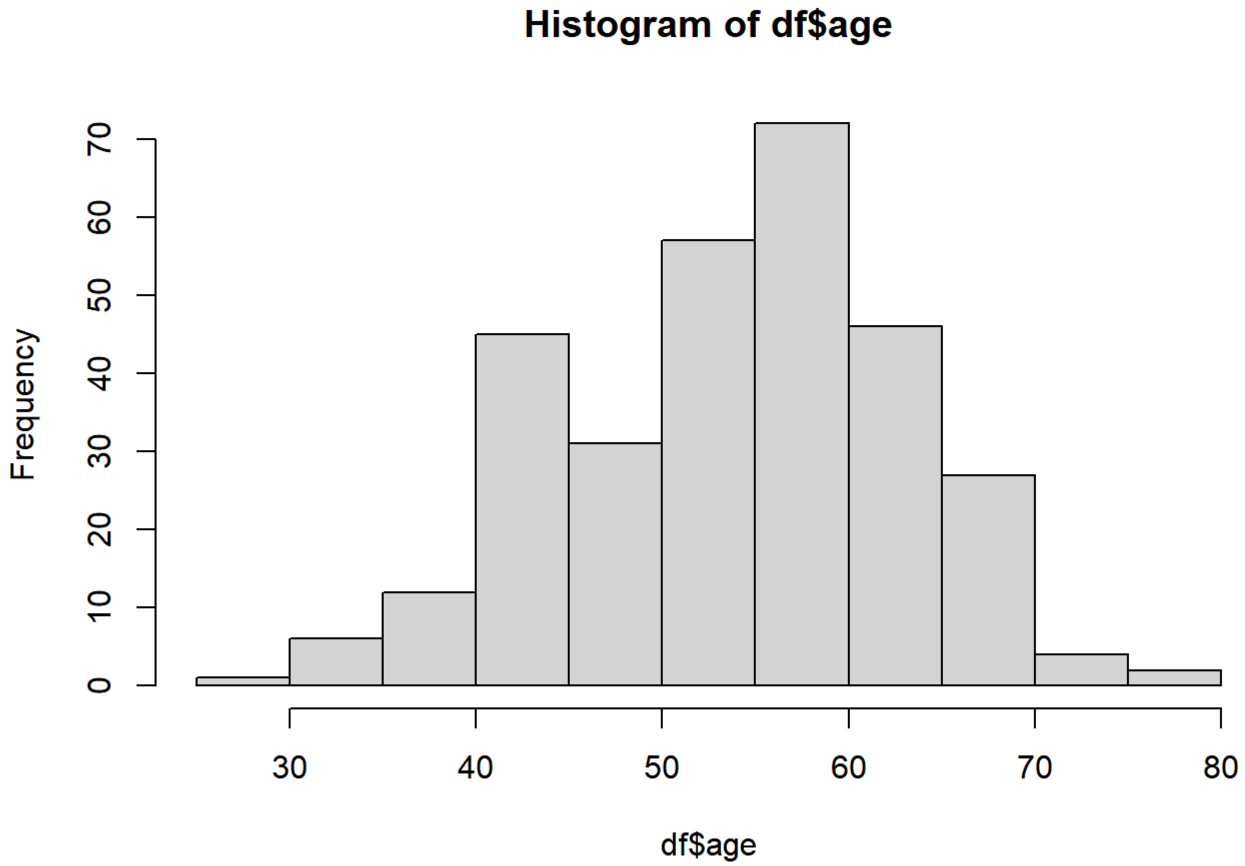
stem with a conditional statement:

```
with(df, stem(df$trtbps[df$sex == 0]))
```

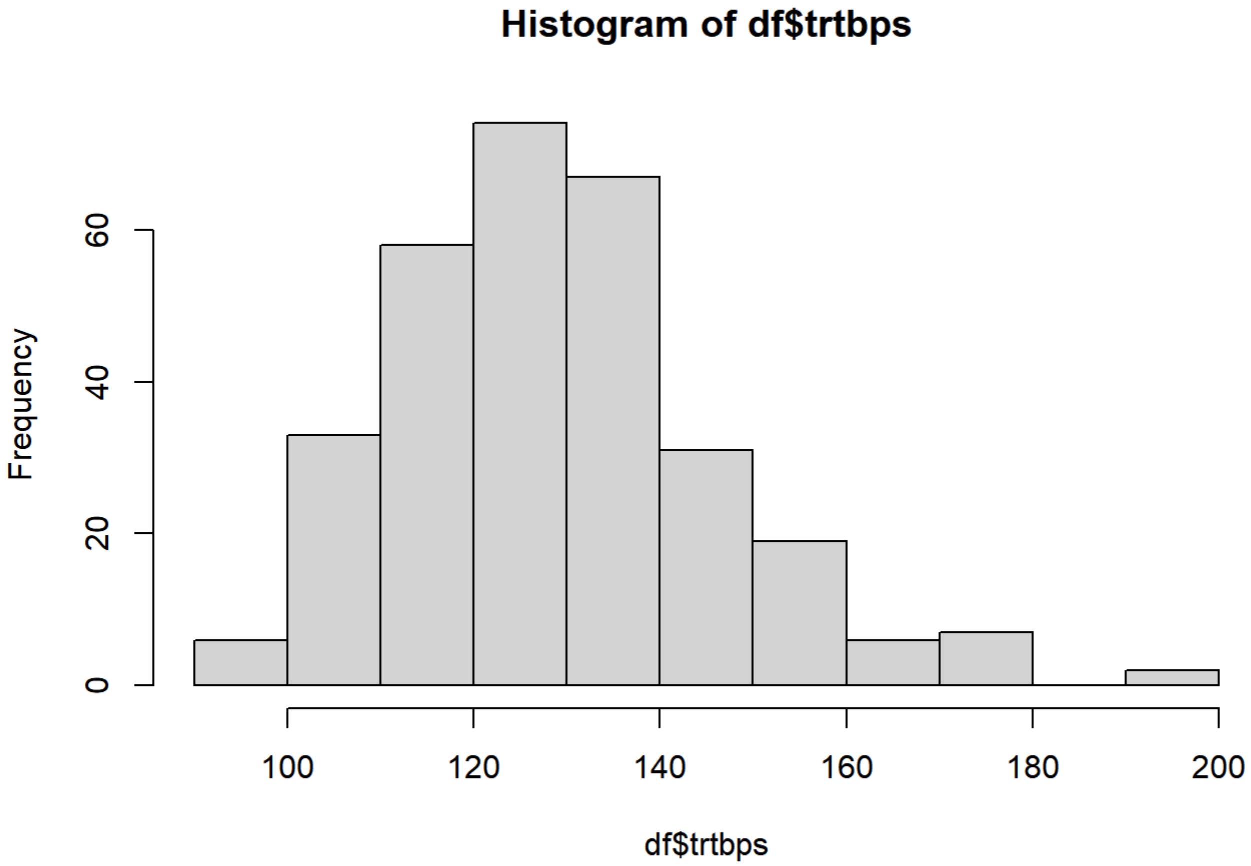
```
##
##   The decimal point is 1 digit(s) to the right of the |
##
##    8 | 4
##   10 | 022556888000222588
##   12 | 00000000002446888000000000000222445556688888 8
##   14 | 0000000000025600000025
##   16 | 0000048
##   18 | 00
##   20 | 0
```

Histogram Plotting:

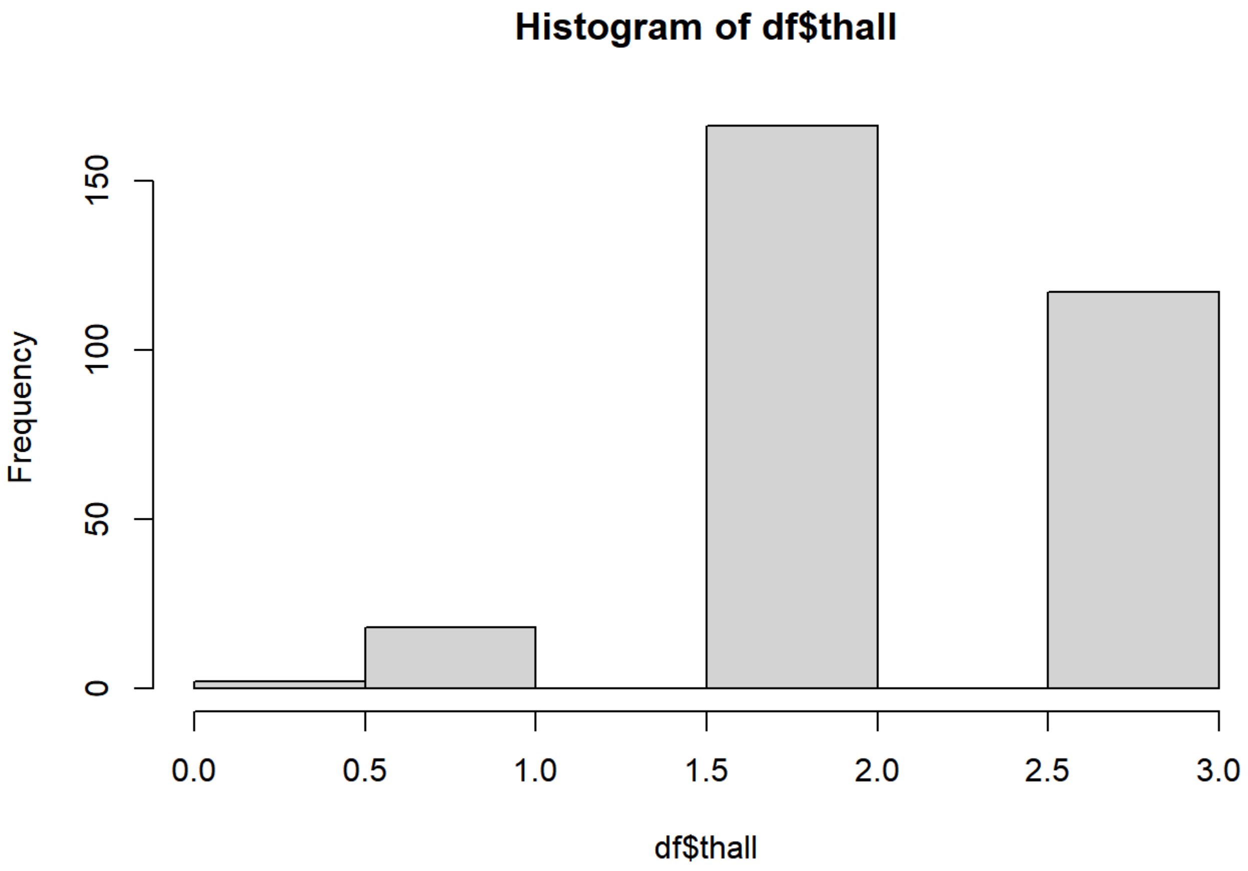
```
hist(df$age)
```



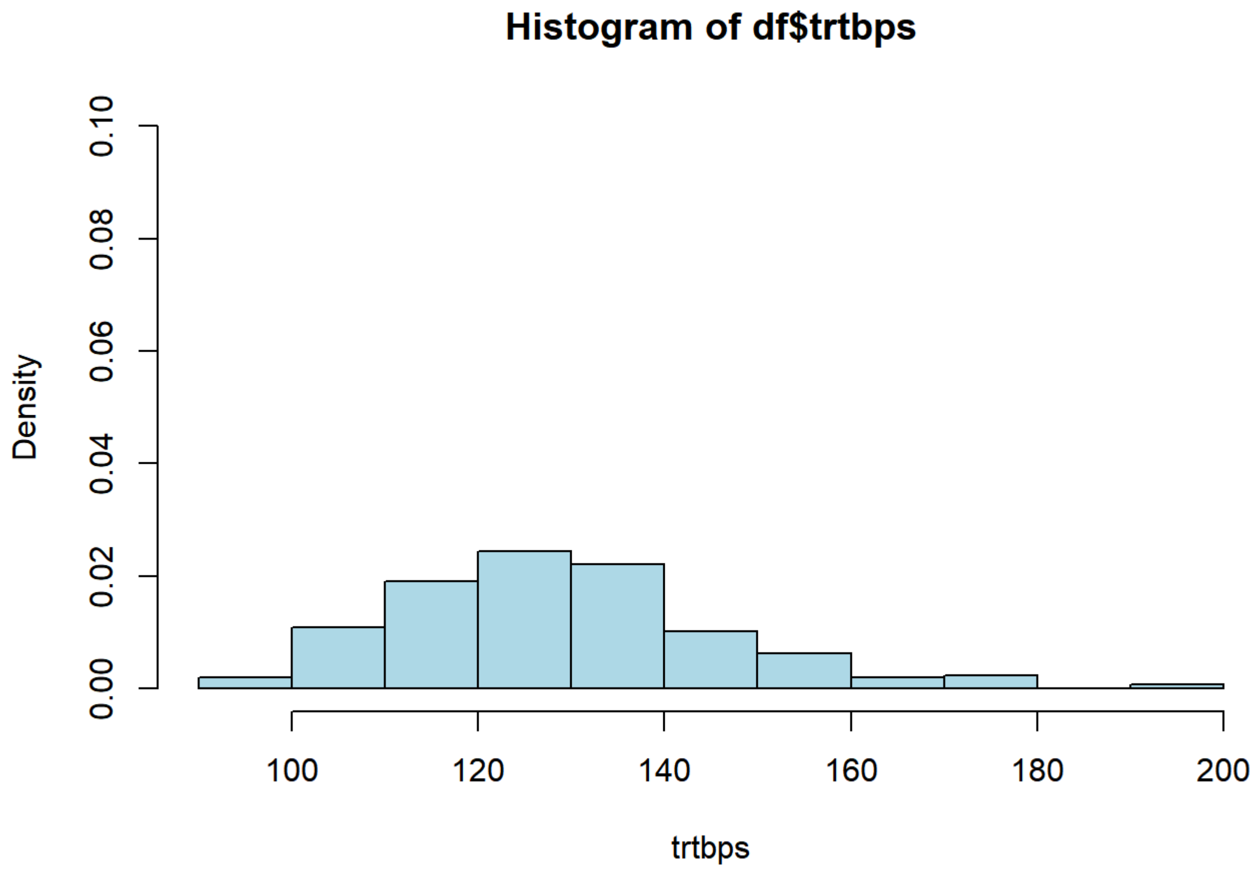
```
hist(df$trtbps)
```



```
hist(df$thall)
```



```
hist(df$trtbps, col = 'lightblue', xlab = 'trtbps', ylim = c(0, 0.1), freq = FALSE)
```



##

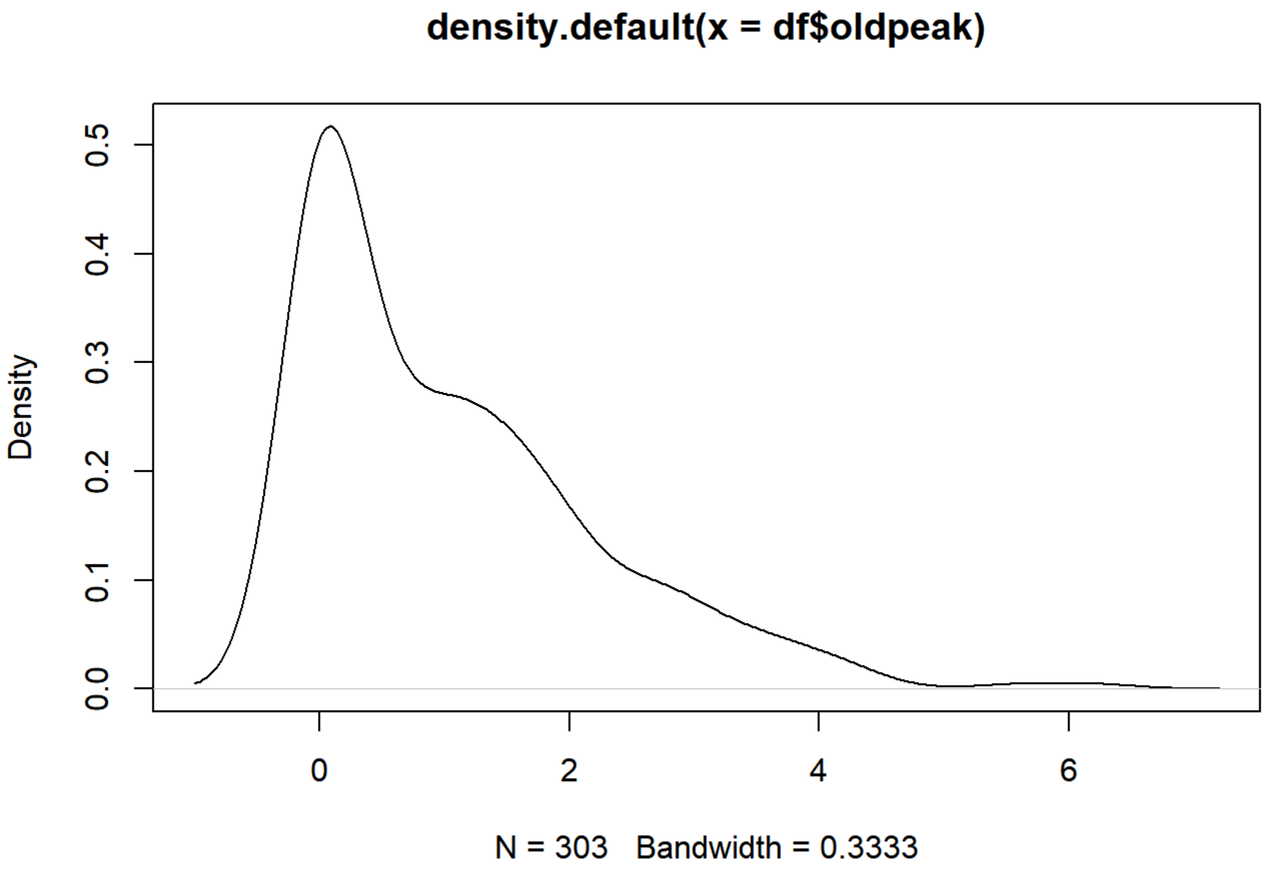
Have seen in drawing a histogram with the hist() command that you can use freq = FALSE to force the y-axis to display the density rather than the frequency of the data. You can also call on the density function directly via the density() command.

```
density(df$thalachh)
```

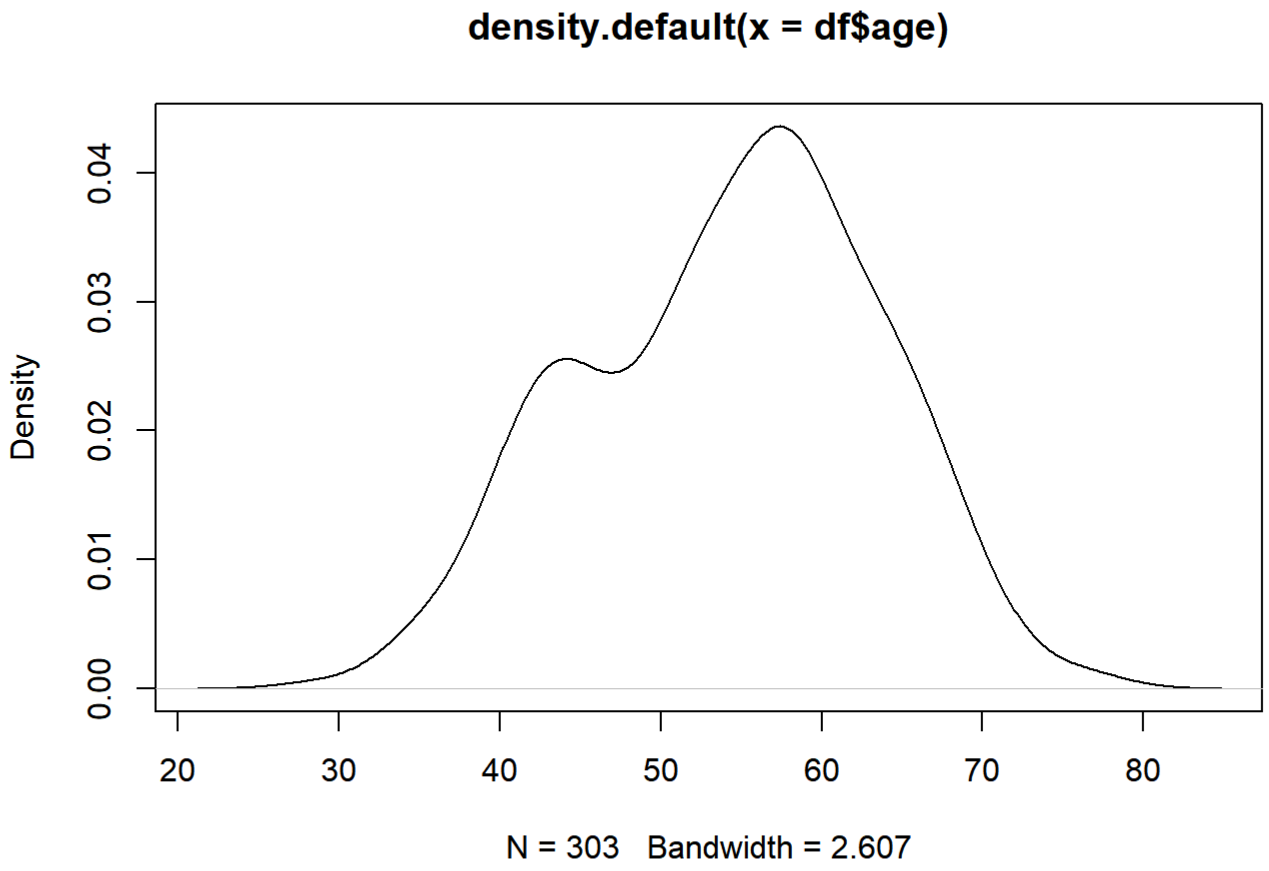
```
##
## Call:
##  density.default(x = df$thalachh)
##
## Data: df$thalachh (303 obs.);    Bandwidth 'bw' = 6.575
##
##        x                 y
##  Min.   : 51.28   Min.   :2.262e-06
##  1st Qu.: 93.89   1st Qu.:2.536e-04
##  Median :136.50   Median :3.488e-03
##  Mean   :136.50   Mean   :5.861e-03
##  3rd Qu.:179.11   3rd Qu.:1.034e-02
##  Max.   :221.72   Max.   :1.792e-02
```

Using the Density Function to Draw a Graph

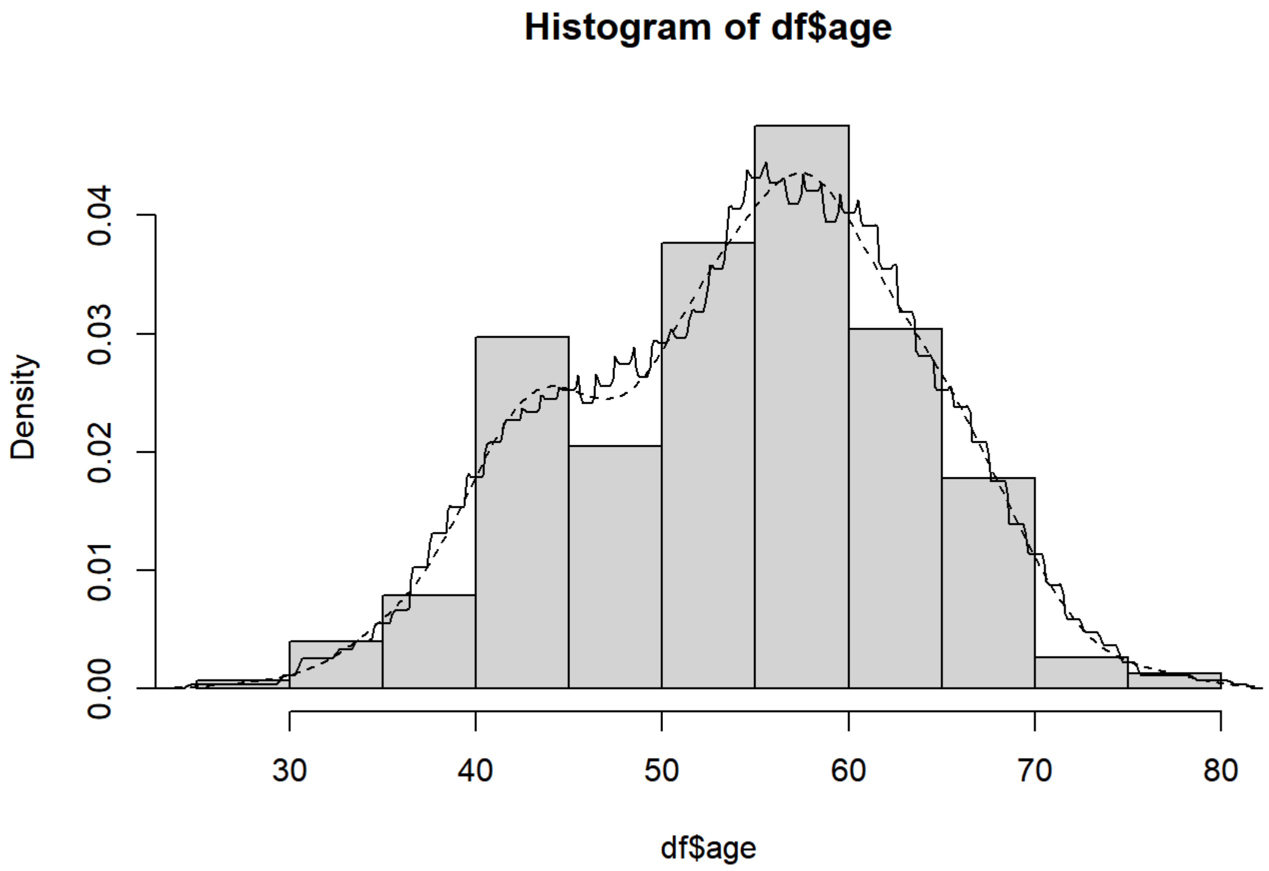
```
plot(density(df$oldpeak))
```



```
plot(density(df$age))
```



```
hist(df$age, freq = F)
lines(density(df$age), lty = 2)
lines(density(df$age, k = 'rectangular'))
```



##

Generates n random numbers from the normal distribution with mean of 0 and standard deviation of 1

```
head(rnorm(df$trtbps, mean = 0, sd = 1))
```

```
## [1] -0.4017413  0.2550076  0.7789583 -0.4123507 -0.2928873 -0.5878548
```

Returns the probability for the quantile q

```
head(pnorm(df$thalachh, mean = 0, sd = 1))
```

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

Returns the quantile for a given probability p

```
head(qnorm(df$oldpeak, mean = 0, sd = 1))
```

```
## Warning in qnorm(df$oldpeak, mean = 0, sd = 1): NaNs produced
```

```
## [1]        NaN        NaN        NaN  0.8416212  0.2533471 -0.2533471
```

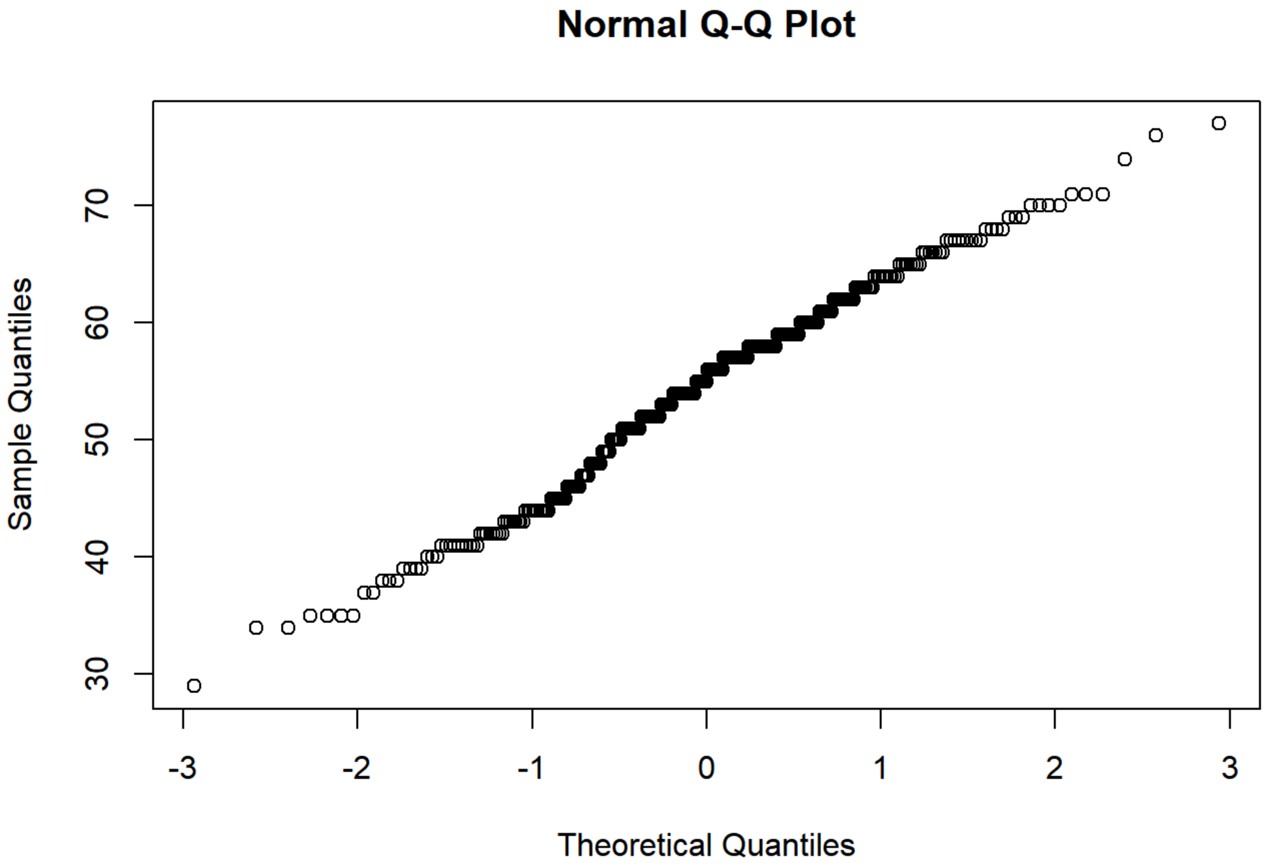
Gives the density function for values x

```
head(dnorm(df$trtbps, mean = 0, sd = 1))
```

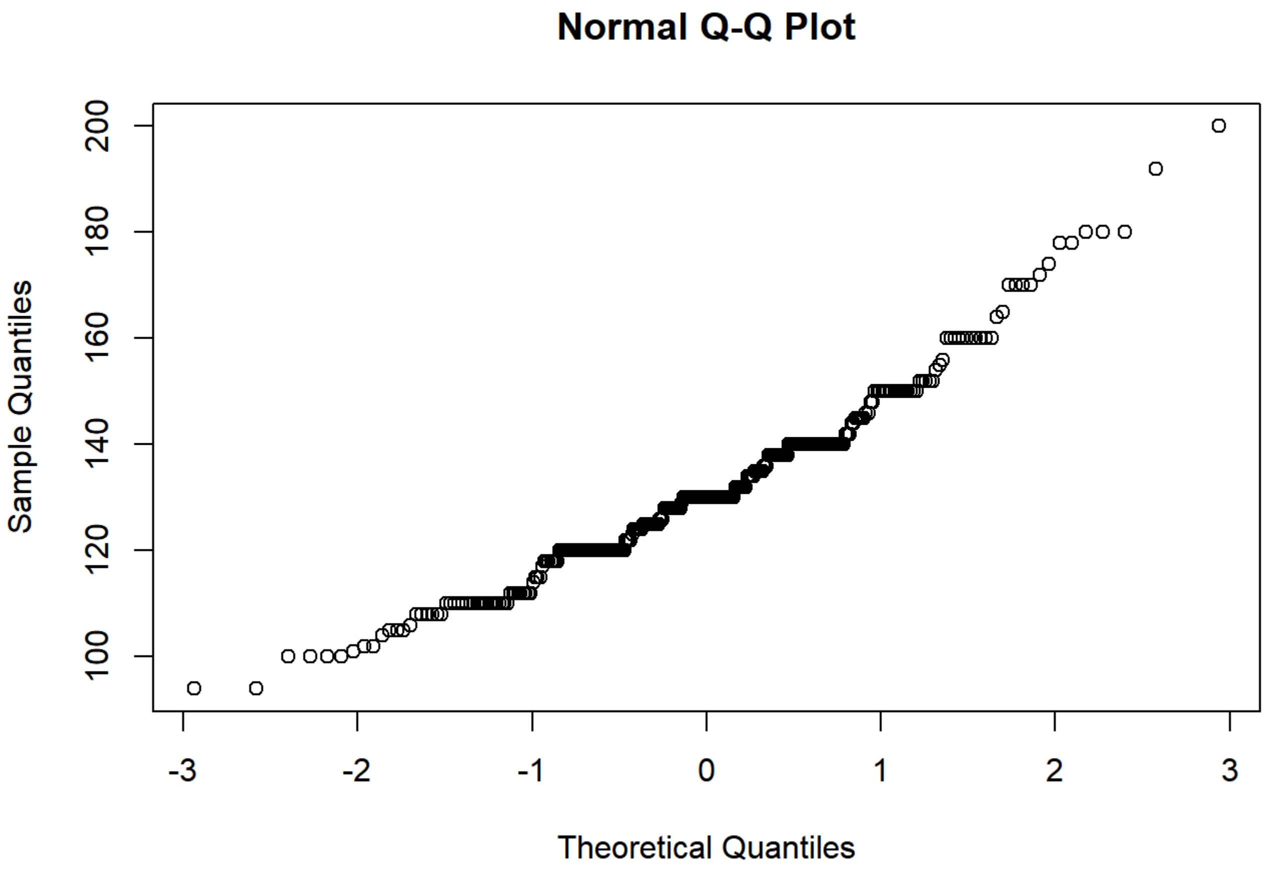
```
## [1] 0 0 0 0 0 0
```

Quantile-Quantile Plot

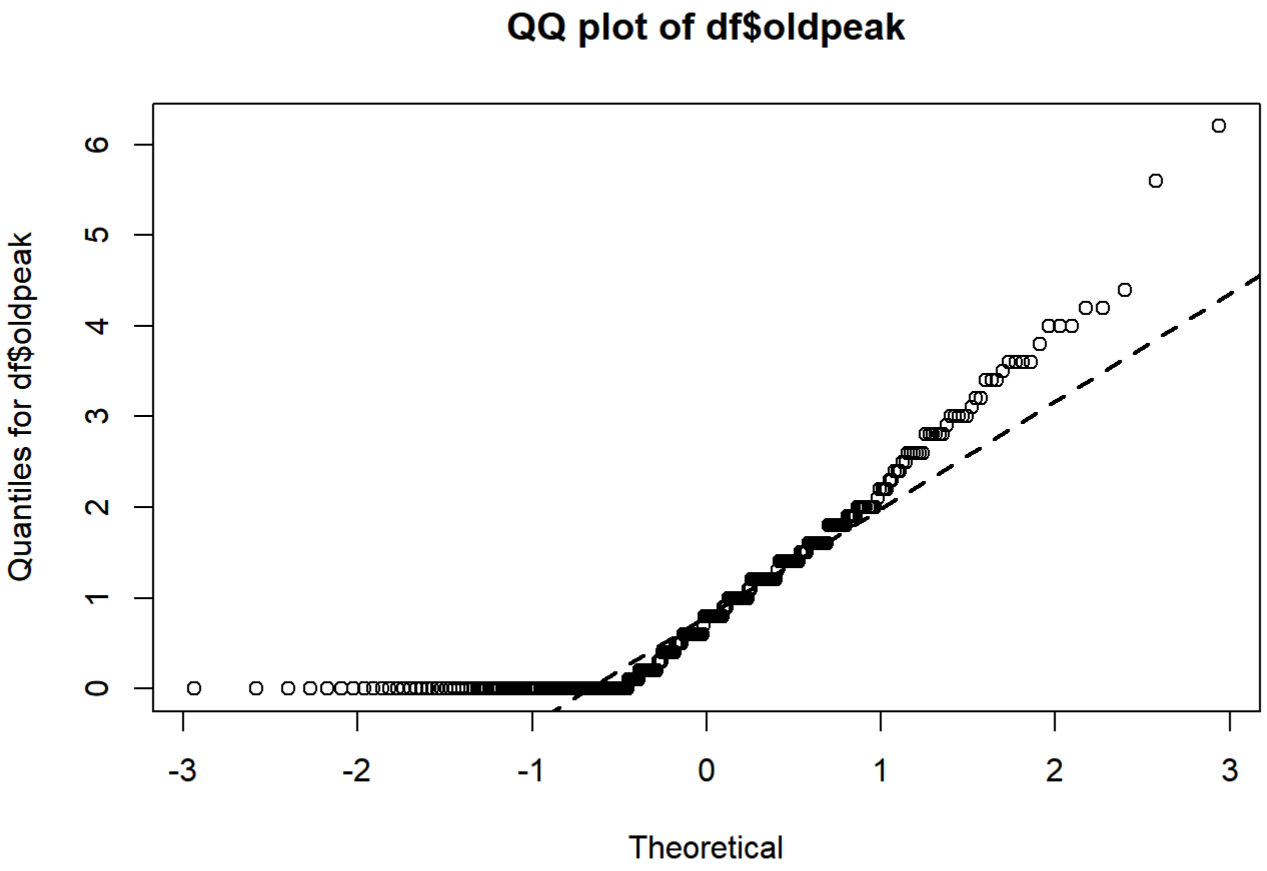
```
qqnorm(df$age)
```



```
qqnorm(df$trtbps)
```



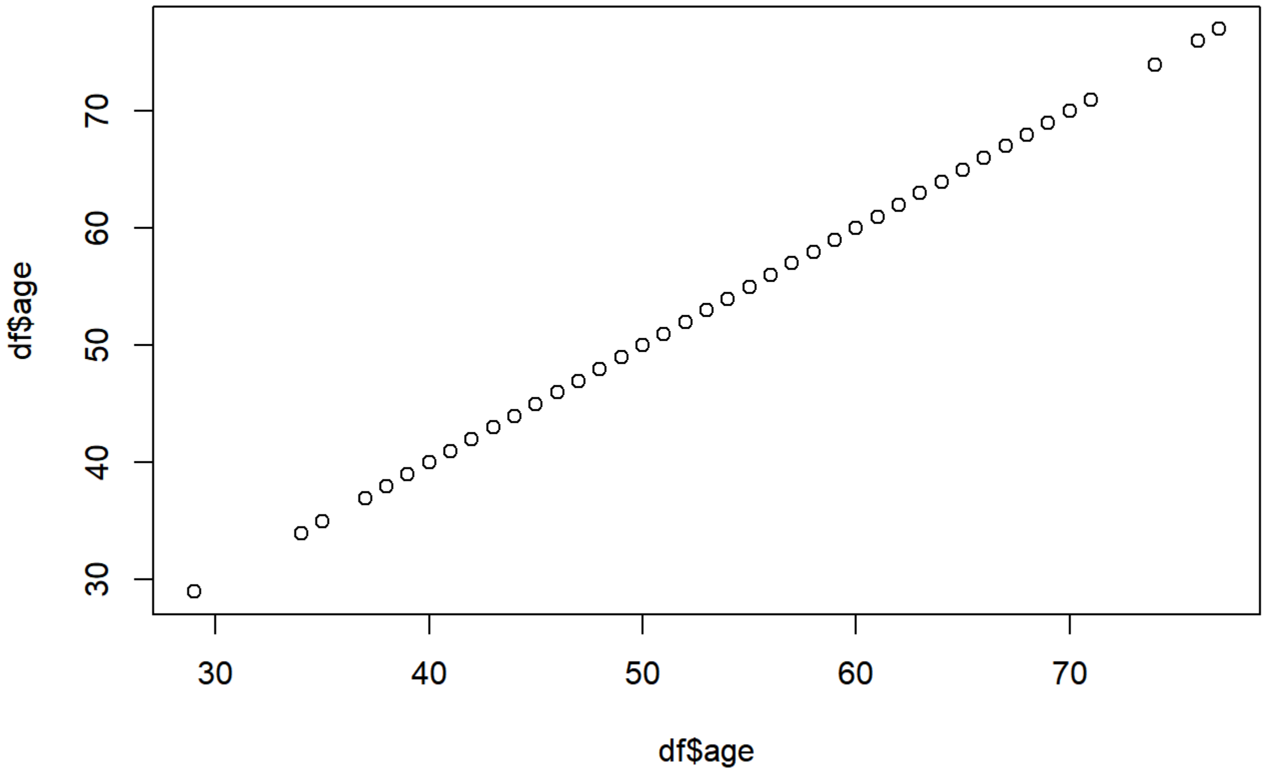
```
qqnorm(df$oldpeak, main = 'QQ plot of df$oldpeak', xlab = 'Theoretical',
ylab = 'Quantiles for df$oldpeak')
qqline(df$oldpeak, lwd = 2, lty = 2)
```



##

can also plot one distribution against another as a quantile-quantile plot using the qqplot() command.

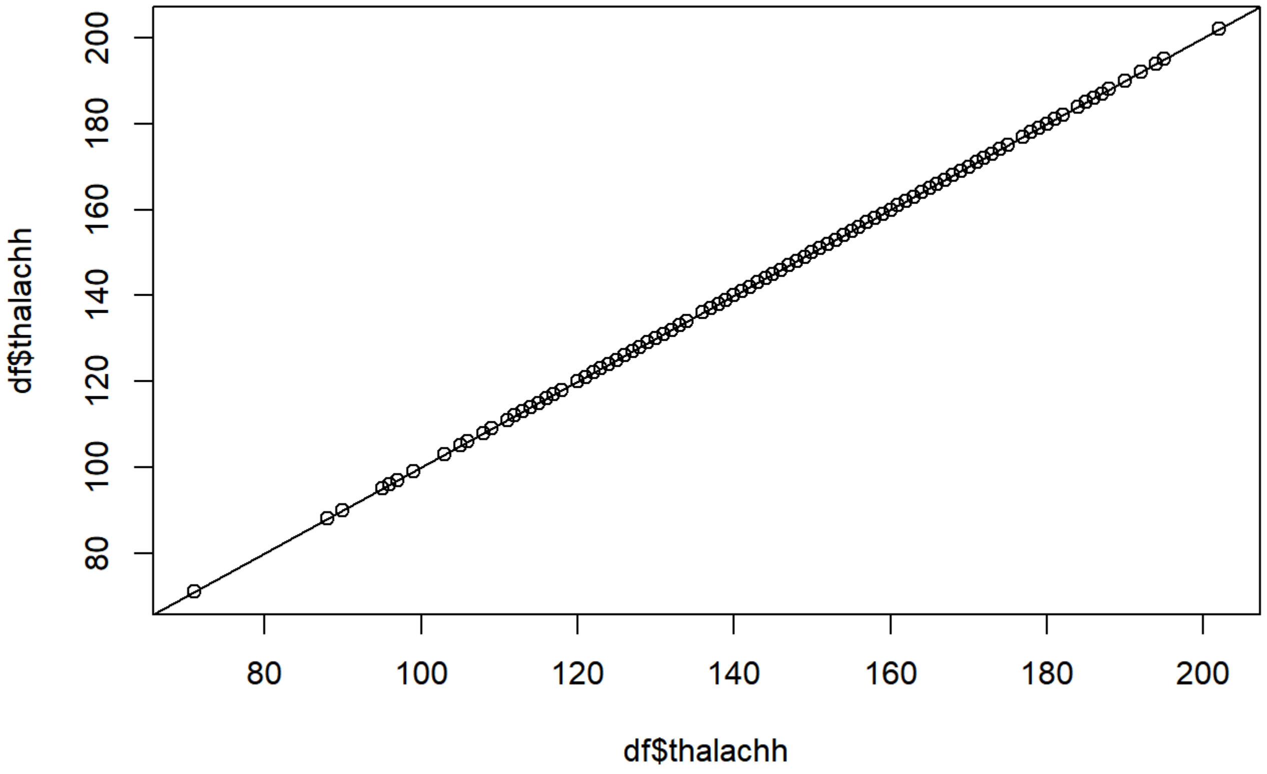
```
qqplot(df$age, df$age)
```



##

Would be useful to draw a straight line on your qqplot() and you can do that using the abline() command. This command uses the properties of a straight line (that is, y = a + bx) to produce a line on an existing plot. The general form of the command is: abline(a = intercept, b = slope) ## Lm(), which carries out linear modeling. This command determines the line of best fit between the x and y values in your qqp object.

```
qqp = qqplot(df$thalachh, df$thalachh)
abline(lm(qqp$y ~ qqp$x))
```



##

The basic method of applying a t-test is to compare two vectors of numeric data

```
t.test(df$age, df$oldpeak)
```

```
##
##  Welch Two Sample t-test
##
## data:  df$age and df$oldpeak
## t = 101.38, df = 311.87, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  52.29178 54.36169
## sample estimates:
## mean of x mean of y
## 54.366337  1.039604
```

Can override the default and use the classic t-test by adding the var.equal = TRUE instruction, which forces the command to assume that the variance of the two samples is equal.

```
t.test(df$age, df$thalachh, var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  df$age and df$thalachh
## t = -67.311, df = 604, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -98.06049 -92.50056
## sample estimates:
## mean of x mean of y
##  54.36634 149.64686
```

can also carry out a one-sample t-test. In this version you supply the name of a single vector and the mean to compare it to (this defaults to 0):

```
t.test(df$age, mu = 5)
```

```
##
##  One Sample t-test
##
## data:  df$age
## t = 94.616, df = 302, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 5
## 95 percent confidence interval:
##  53.33960 55.39307
## sample estimates:
## mean of x
##  54.36634
```

Using Directional Hypotheses

```
t.test(df$age, mu = 5, alternative = 'greater')
```

```
##
##  One Sample t-test
##
## data:  df$age
## t = 94.616, df = 302, p-value < 2.2e-16
## alternative hypothesis: true mean is greater than 5
## 95 percent confidence interval:
##  53.50549      Inf
## sample estimates:
## mean of x
##  54.36634
```

Formula Syntax and Subsetting Samples in the t-Test

```
t.test(df$age ~ df$sex, data = df, subset = df$sex %in% c(0 , 1))
```

```
## 
##  Welch Two Sample t-test
## 
## data:  df$age by df$sex
## t = 1.6805, df = 175.92, p-value = 0.09464
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to
0
## 95 percent confidence interval:
##  -0.3346005  4.1718589
## sample estimates:
## mean in group 0 mean in group 1 
##        55.67708        53.75845
```

Two-Sample U-Test

```
wilcox.test(df$age, df$trtbps)
```

```
## 
##  Wilcoxon rank sum test with continuity correction
## 
## data:  df$age and df$trtbps
## W = 0, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

Carries out a basic correlation between x and y . If x is a matrix or data frame, y can be omitted

```
cor(df$age, df$trtbps)
```

```
## [1] 0.2793509
```

Determines covariance between x and y . If x is a matrix or data frame, y can be omitted

```
x = cov(df$age, df$trtbps)
x
```

```
## [1] 44.4959
```

The cov2cor() command is used to determine the correlation from a matrix of covariance in the following example:

```
v = as.matrix(x)
cov2cor(v)
```

```
##      [,1]
## [1,]    1
```

Significance Testing in Correlation Tests

```
cor.test(df$thall, df$oldpeak)
```

```
## 
##  Pearson's product-moment correlation
## 
## data:  df$thall and df$oldpeak
## t = 3.731, df = 301, p-value = 0.0002279
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.0999335 0.3154492
## sample estimates:
##       cor 
## 0.2102441
```

Chi-squared tests of association can be carried out using the chisq.test() command.

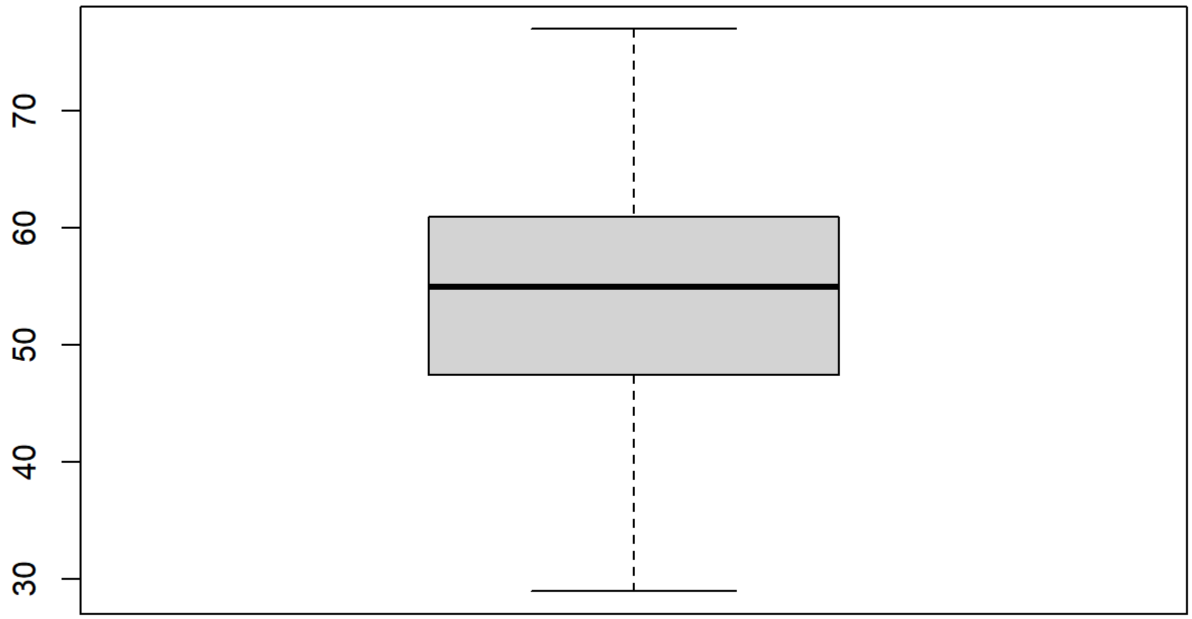
```
chisq.test(df$oldpeak)
```

```
## Warning in chisq.test(df$oldpeak): Chi-squared approximation may be incorrect
```

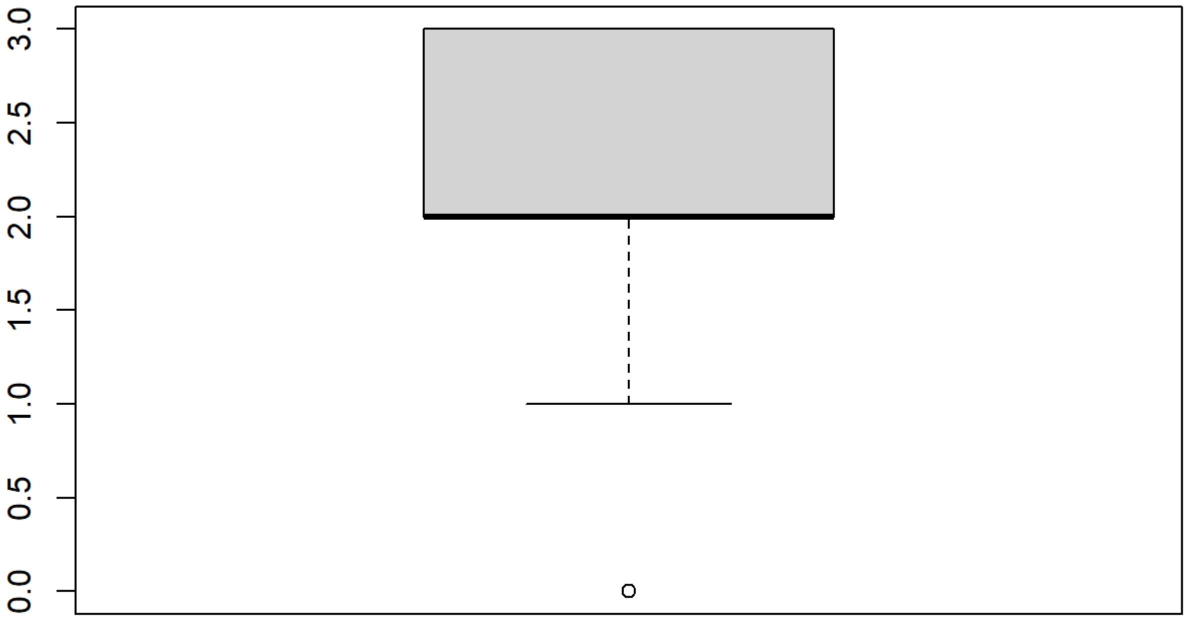
```
## 
##  Chi-squared test for given probabilities
## 
## data:  df$oldpeak
## X-squared = 391.62, df = 302, p-value = 0.0003839
```

BoxPlot’s:

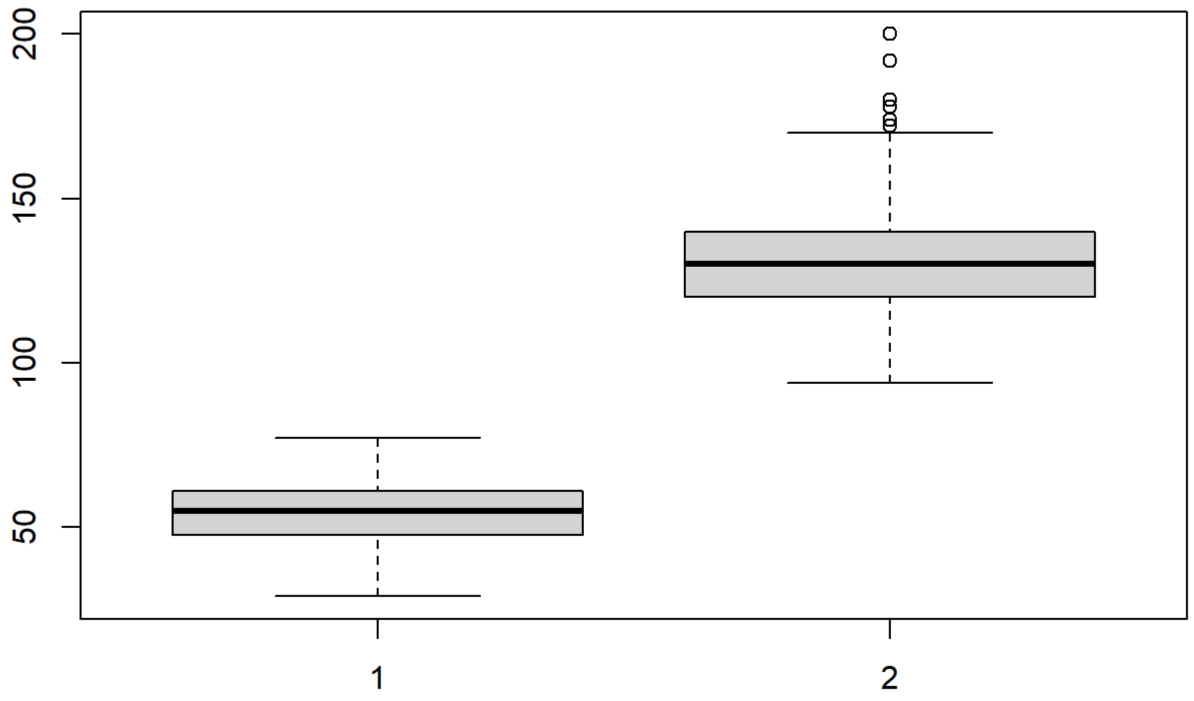
```
boxplot(df$age)
```



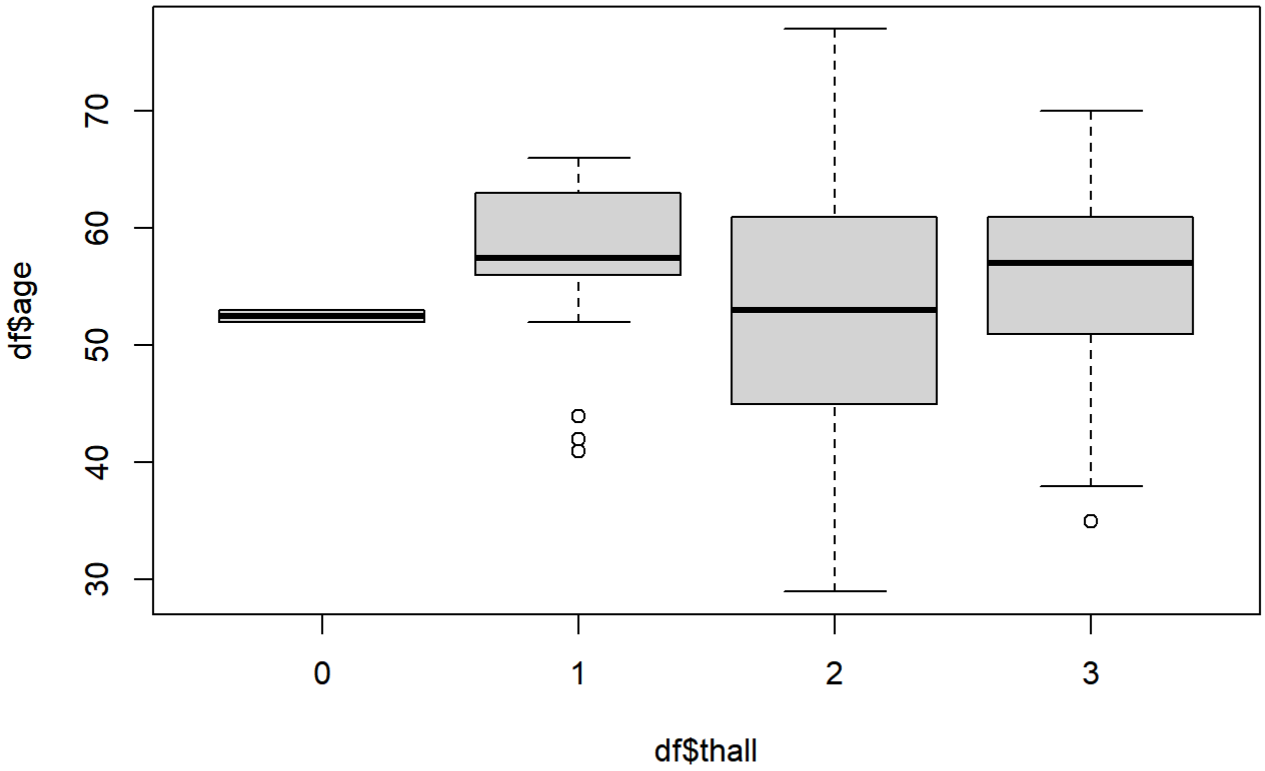
```
boxplot(df$thall)
```



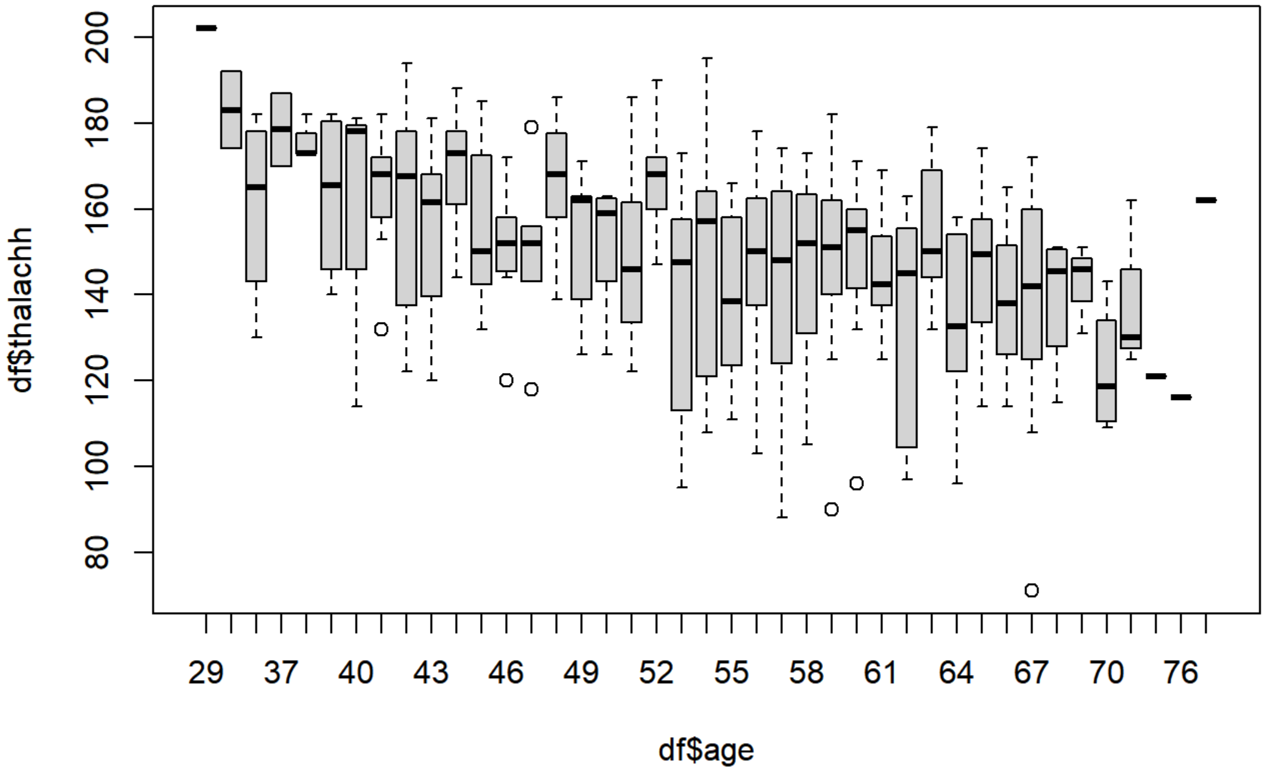
```
boxplot(df$age, df$trtbps)
```



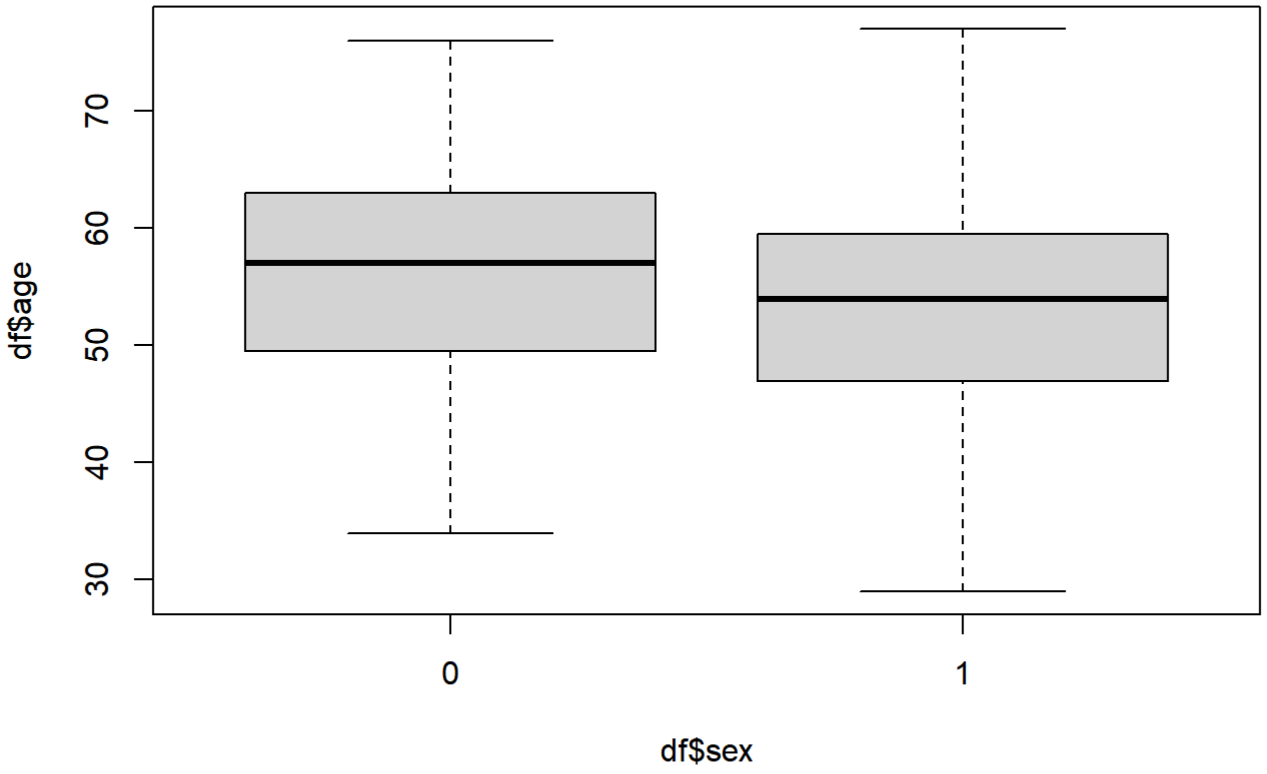
```
boxplot(df$age ~ df$thall, data = df)
```



```
boxplot(df$thalachh ~ df$age, data = df)
```



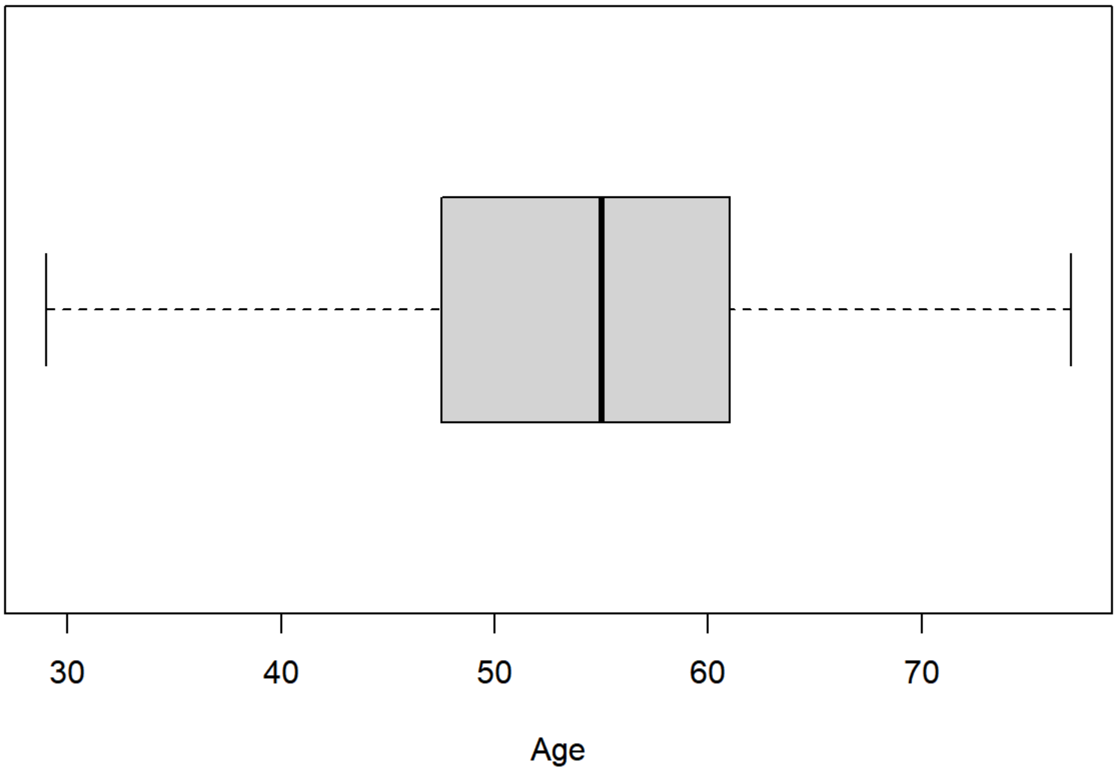
```
boxplot(df$age ~ df$sex, data = df)
```



##

Horizontal Boxplots

```
boxplot(df$age, horizontal = T)
title(xlab = 'Age')
```



##

scatter plot’s

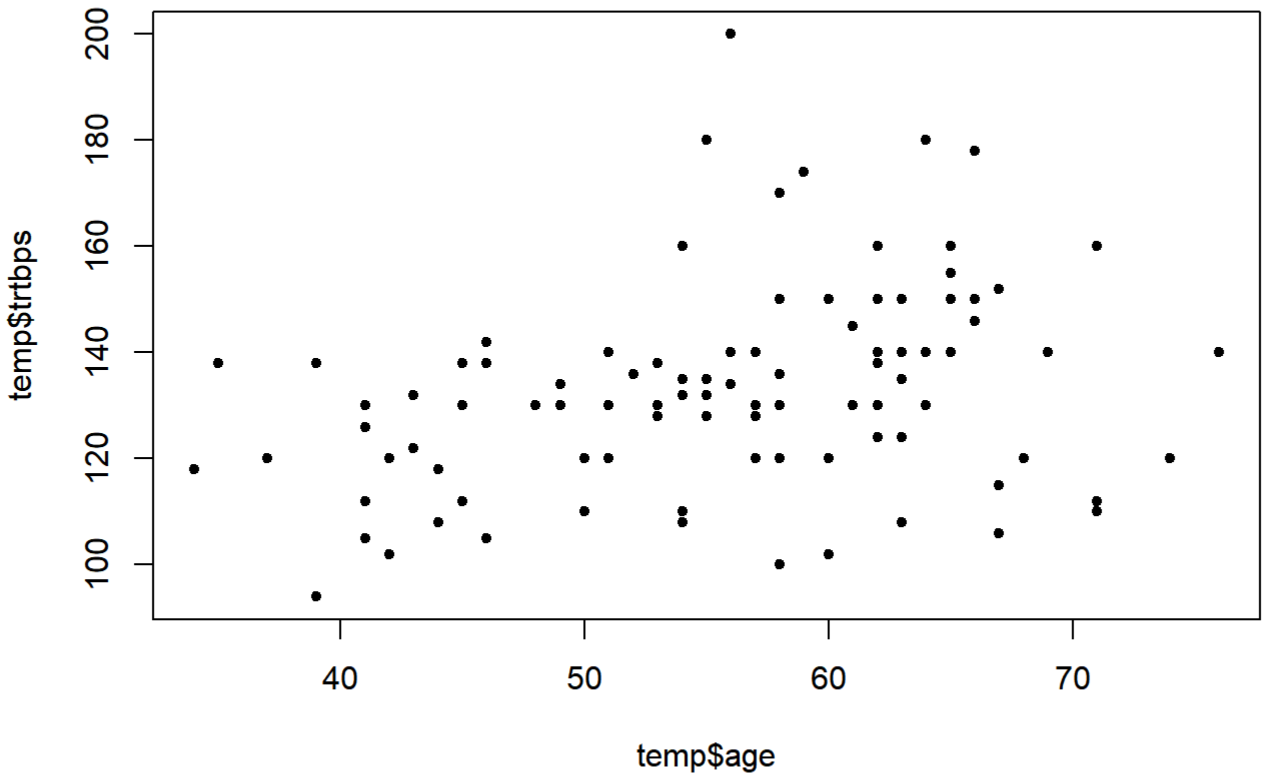
```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

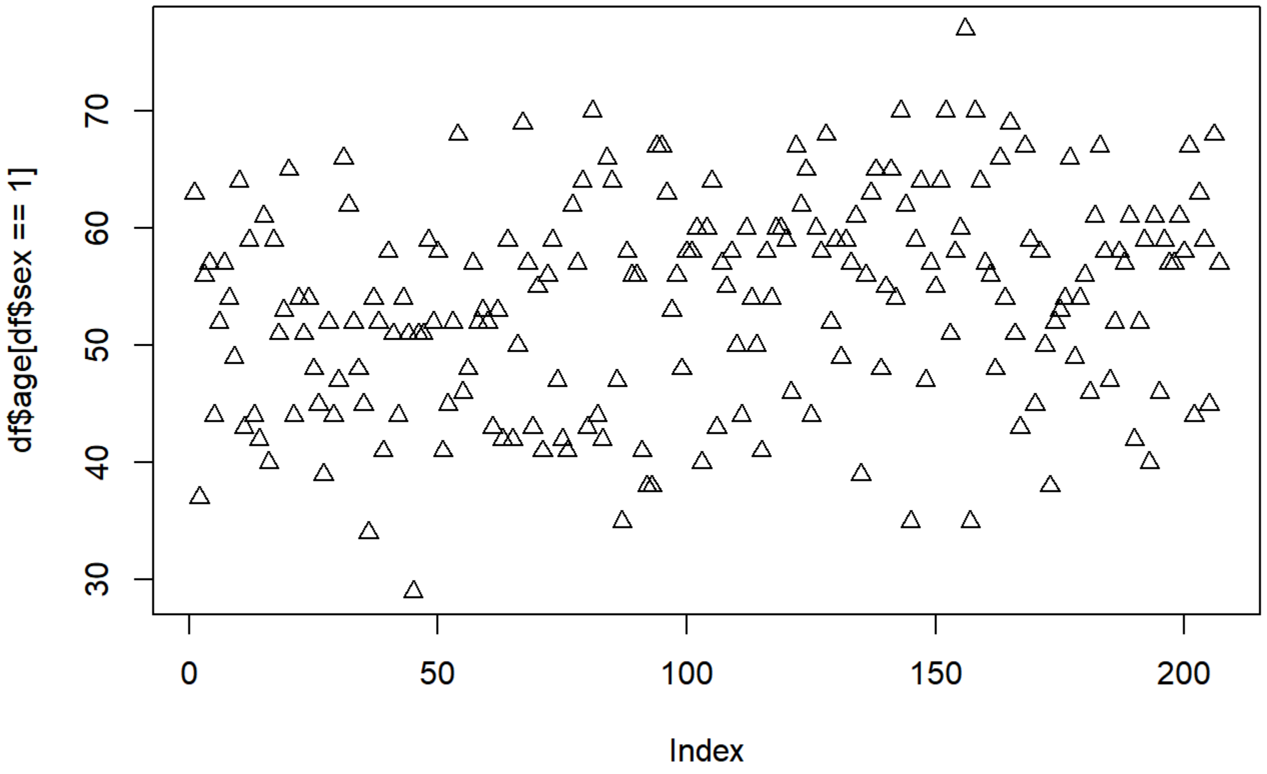
```
temp = df %>% filter(df$sex == 0)
plot(temp$age, temp$trtbps, pch = 20)
```



##

The pch = instruction refers to the plotting character, and can be specified in one of several ways. You can type an integer value and this code will be reflected in the symbol/character produced. For values from 0 to 25

```
with(df, plot(df$age[df$sex == 1], pch = 2))
```



##

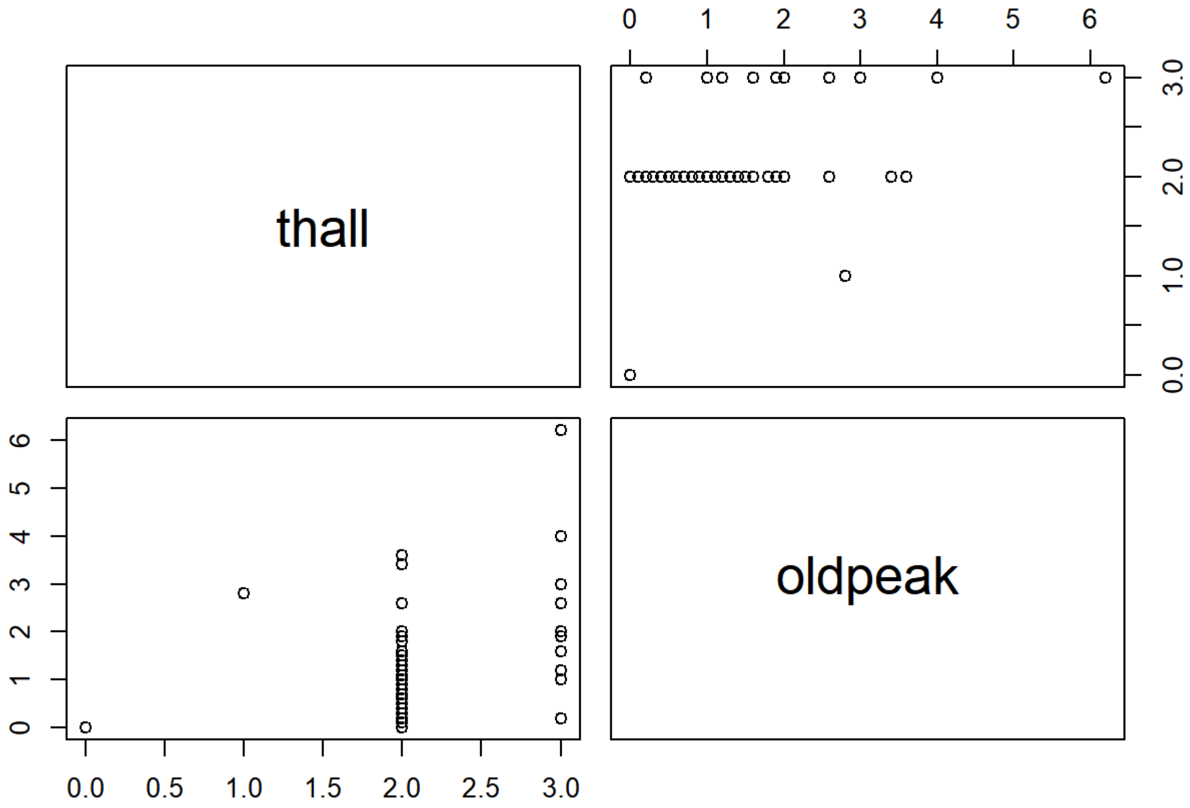
with() command can be used to get the particular values based on row(level) conditions

```
with(df, (df$age[df$sex == 1]))
```

```
##   [1] 63 37 56 57 44 52 57 54 49 64 43 59 44 42 61 40 59 51 53 65 44 54 51 54 48
##  [26] 45 39 52 44 47 66 62 52 48 45 34 54 52 41 58 51 44 54 51 29 51 51 59 52 58
##  [51] 41 45 52 68 46 48 57 52 53 52 43 53 42 59 42 50 69 57 43 55 41 56 59 47 42
##  [76] 41 62 57 64 43 70 44 42 66 64 47 35 58 56 56 41 38 38 67 67 63 53 56 48 58
## [101] 58 60 40 60 64 43 57 55 58 50 44 60 54 50 41 58 54 60 60 59 46 67 62 65 44
## [126] 60 58 68 52 59 49 59 57 61 39 56 63 65 48 55 65 54 70 62 35 59 64 47 57 55
## [151] 64 70 51 58 60 77 35 70 64 57 56 48 66 54 69 51 43 67 59 45 58 50 38 52 53
## [176] 54 66 49 54 56 46 61 67 58 47 52 58 57 61 42 52 59 40 61 46 59 57 57 61 58
## [201] 67 44 63 59 45 68 57
```

the pairs() command takes all the columns in a data frame and creates a matrix of scatter plots.

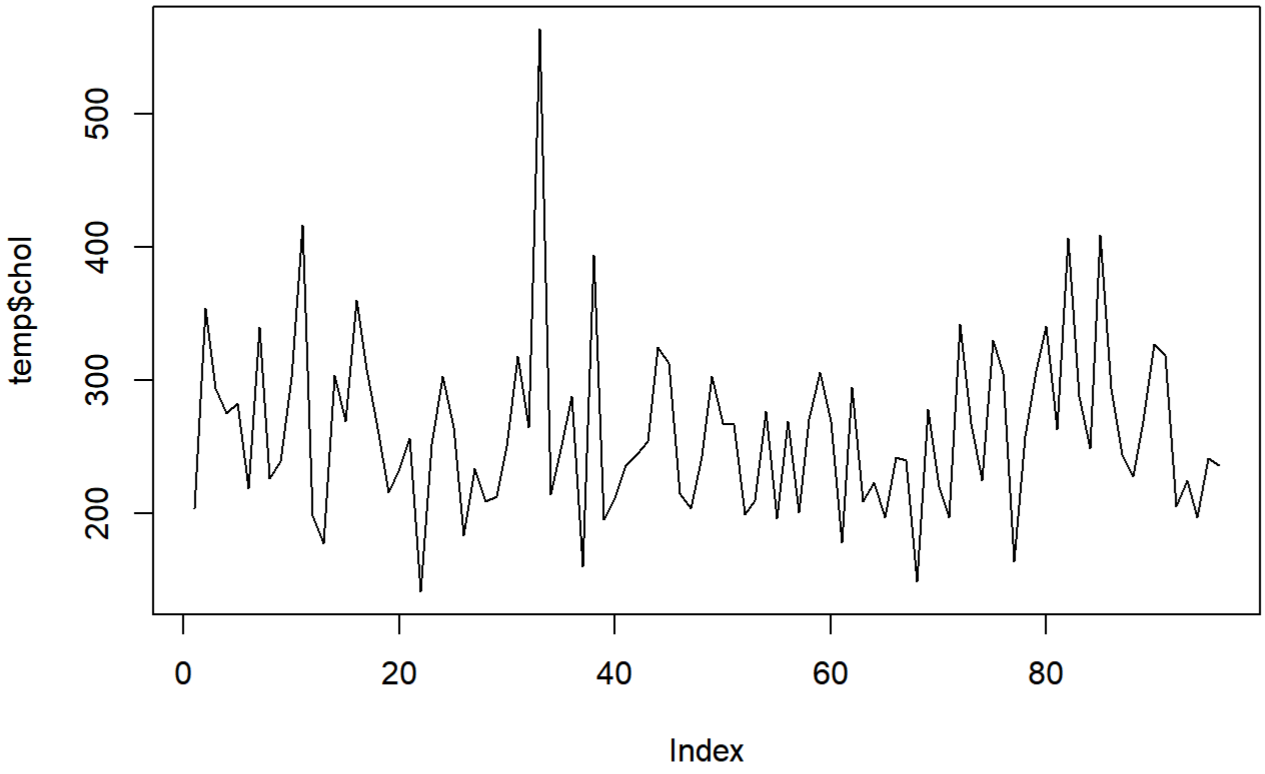
```
pairs(~ thall + oldpeak, data = temp)
```



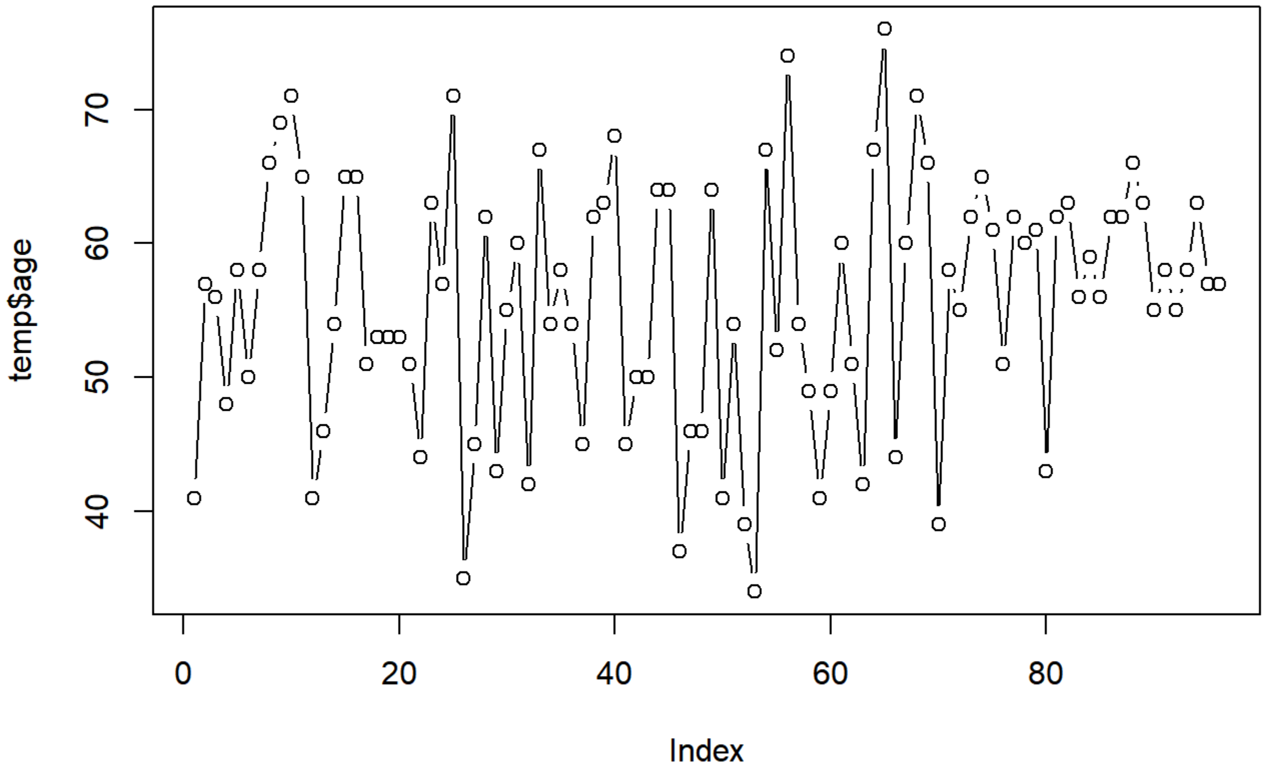
##

Line plot

```
plot(temp$chol, type = 'l' )
```



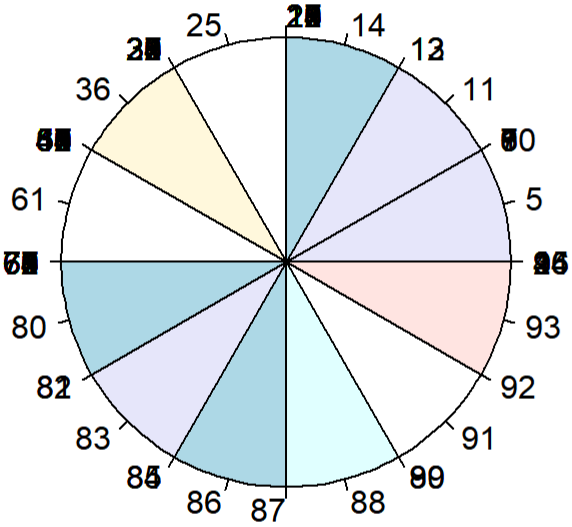
```
plot(temp$age, type = 'b')
```



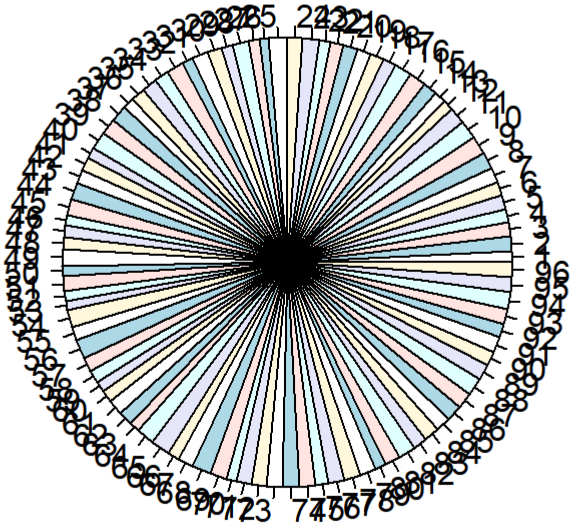
##

Pie chart

```
pie(temp$fbs)
```



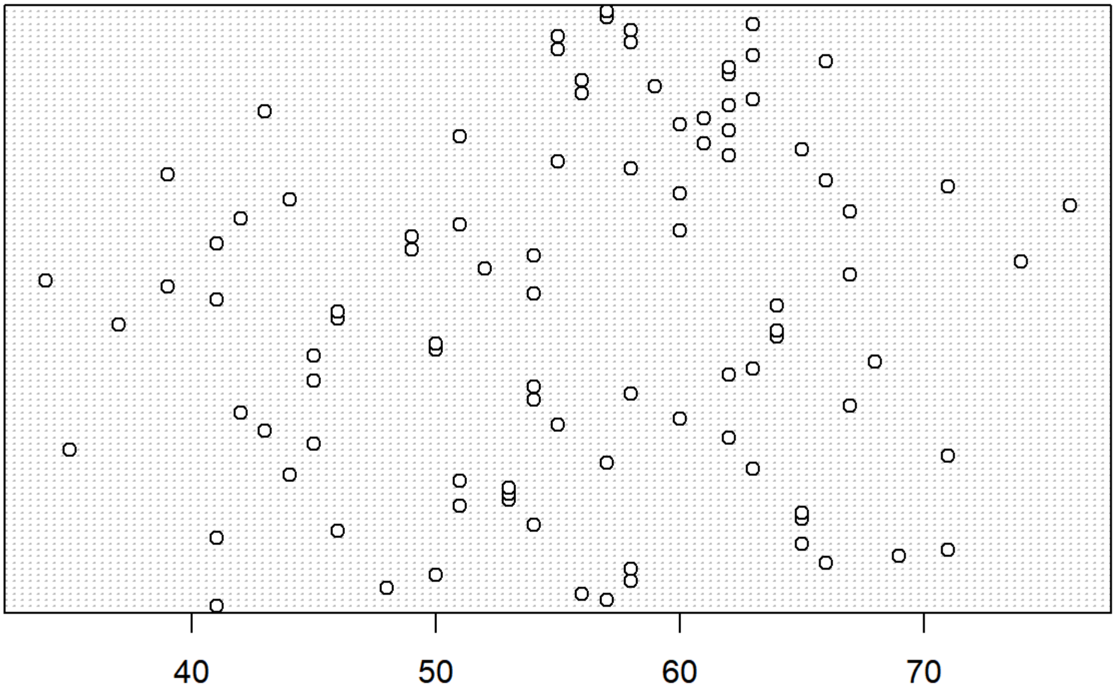
```
pie(temp$age)
```



##

Dot Chart

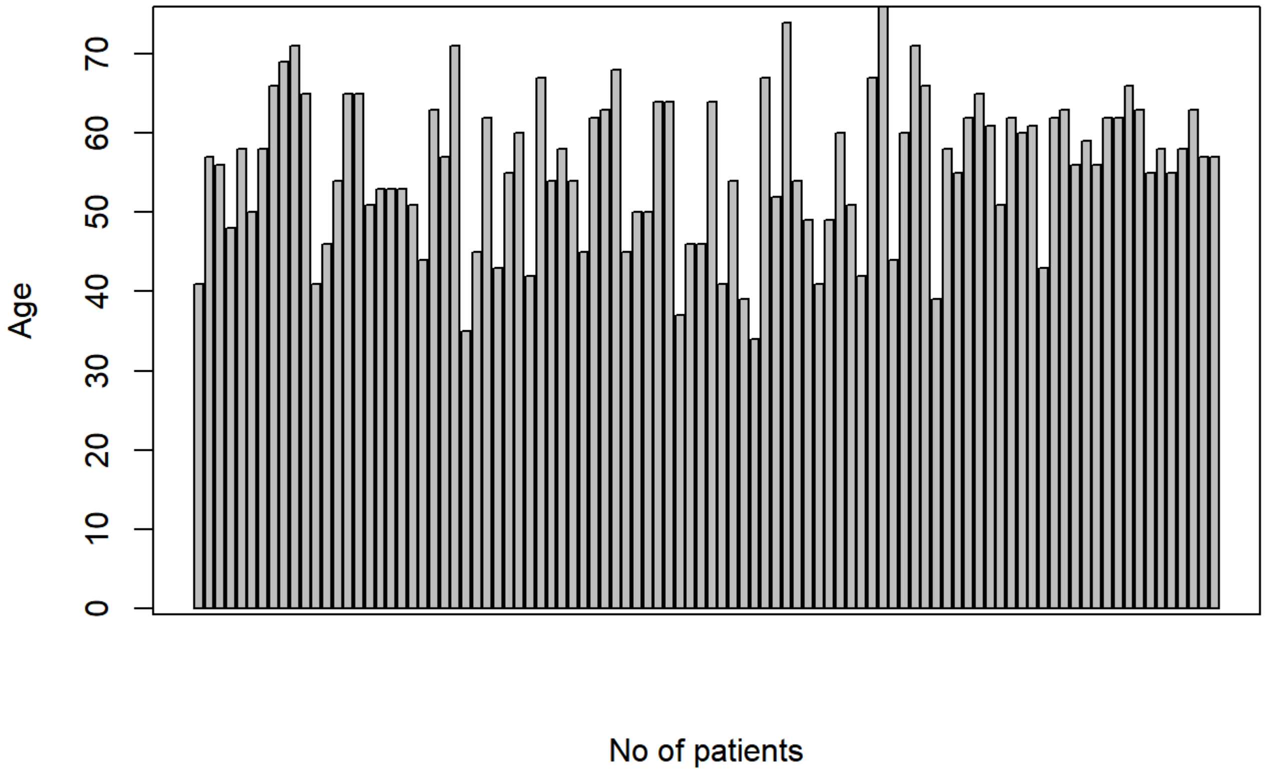
```
dotchart(temp$age)
```



##

Bar plot

```
barplot(temp$age)
title(xlab = "No of patients", ylab = 'Age')
box()
```



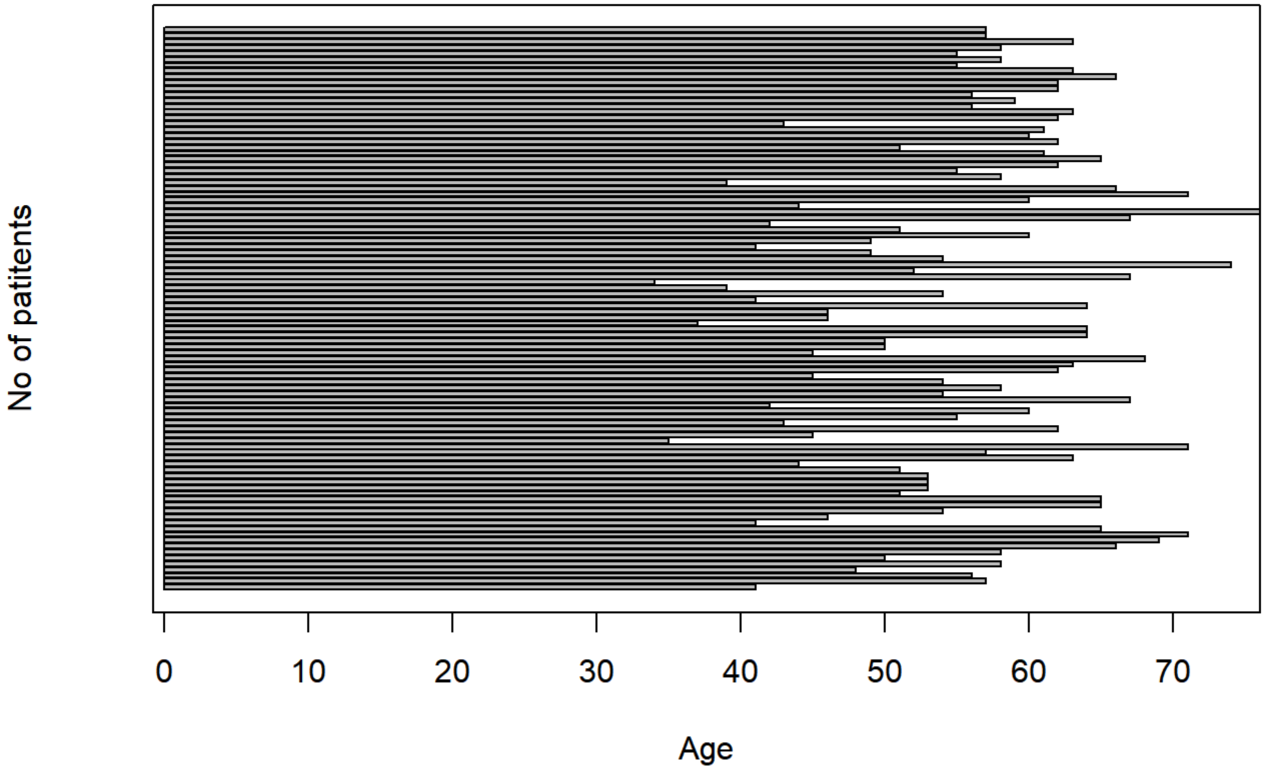
##

table() used to get the column’s diff level(type) counts

```
table(df$age)
```

```
##
## 29 34 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59
##  1  2  4  2  3  4  3 10  8  8 11  8  7  5  7  5  7 12 13  8 16  8 11 17 19 14
## 60 61 62 63 64 65 66 67 68 69 70 71 74 76 77
## 11  8 11  9 10  8  7  9  4  3  4  3  1  1  1
```

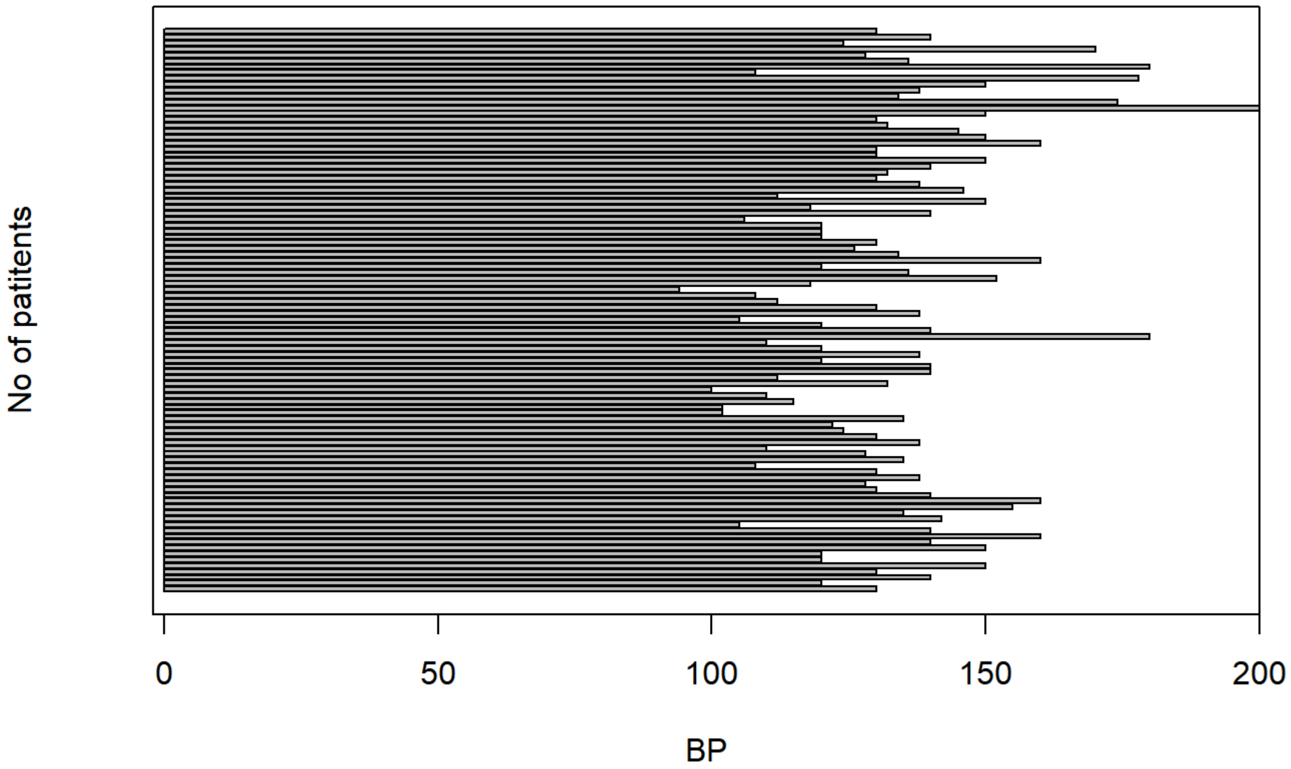
```
barplot(temp$age, horiz = T)
title(xlab = 'Age', ylab = 'No of patitents')
box()
```



```
table(df$age)
```

```
##
## 29 34 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59
##  1  2  4  2  3  4  3 10  8  8 11  8  7  5  7  5  7 12 13  8 16  8 11 17 19 14
## 60 61 62 63 64 65 66 67 68 69 70 71 74 76 77
## 11  8 11  9 10  8  7  9  4  3  4  3  1  1  1
```

```
barplot(temp$trtbps, horiz = T)
title(xlab = 'BP', ylab = 'No of patitents')
box()
```



the analysis of variance using the aov() command

```
temp.aov = na.omit(aov(temp$age ~ temp$trtbps, data = temp))
temp.aov
```

```
## Call:
##    aov(formula = temp$age ~ temp$trtbps, data = temp)
##
## Terms:
##                 temp$trtbps Residuals
## Sum of Squares      829.074  7581.916
## Deg. of Freedom           1        94
##
## Residual standard error: 8.981018
## Estimated effects may be unbalanced
```

```
temp.aov = na.omit(aov(temp$age ~ temp$thalachh, data = temp))
temp.aov
```

```
## Call:
##    aov(formula = temp$age ~ temp$thalachh, data = temp)
##
## Terms:
##                 temp$thalachh Residuals
## Sum of Squares       1382.362  7028.627
## Deg. of Freedom             1        94
##
## Residual standard error: 8.647117
## Estimated effects may be unbalanced
```

To see the classic ANOVA table of results you need to use the summary() command like so

```
summary(temp.aov)
```

```
##               Df Sum Sq Mean Sq F value   Pr(>F)
## temp$thalachh  1   1382  1382.4   18.49 4.18e-05 ***
## Residuals     94   7029    74.8
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Two-way ANOVA

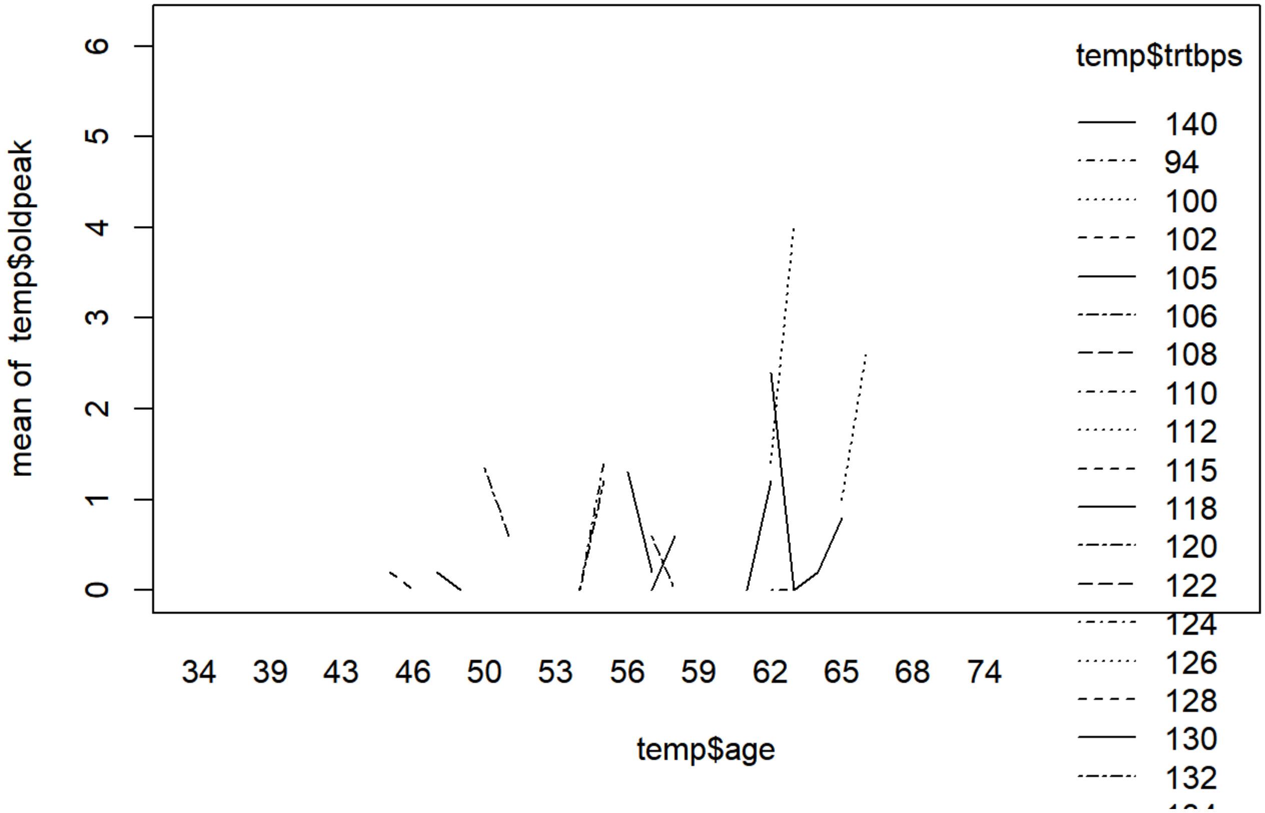
```
toway = aov(age ~ trtbps * oldpeak, data = temp)
toway
```

```
## Call:
##    aov(formula = age ~ trtbps * oldpeak, data = temp)
##
## Terms:
##                   trtbps  oldpeak trtbps:oldpeak Residuals
## Sum of Squares   829.074   51.811        388.873  7141.232
## Deg. of Freedom        1        1              1        92
##
## Residual standard error: 8.81034
## Estimated effects may be unbalanced
```

```
toway = aov(age ~ sex * oldpeak, data = temp)
toway
```

```
## Call:
##    aov(formula = age ~ sex * oldpeak, data = temp)
##
## Terms:
##                  oldpeak Residuals
## Sum of Squares   317.402  8093.588
## Deg. of Freedom        1        94
##
## Residual standard error: 9.279116
## 2 out of 4 effects not estimable
## Estimated effects may be unbalanced
```

```
interaction.plot(temp$age, temp$trtbps, temp$oldpeak)
```



```
factor(df$thalachh_setting)
```

```
## factor(0)
## Levels:
```

creating new testCol/testRow for addition as a new column/row to the df object (using rep() command)

```
testCol = c(rep(df$age - df$sex, length(df)))
head(testCol)
```

```
## [1] 62 36 41 55 57 56
```

Adding Rows or Columns

```
test = data.frame(df, testCol)
```

```
head(test)
```

```
##   age sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall output
## 1  63   1  3    145  233   1       0      150    0     2.3   0   0     1      1
## 2  37   1  2    130  250   0       1      187    0     3.5   0   0     2      1
## 3  41   0  1    130  204   0       0      172    0     1.4   2   0     2      1
## 4  56   1  1    120  236   0       1      178    0     0.8   2   0     2      1
## 5  57   0  0    120  354   0       1      163    1     0.6   2   0     2      1
## 6  57   1  0    140  192   0       1      148    0     0.4   1   0     1      1
##   testCol
## 1      62
## 2      36
## 3      41
## 4      55
## 5      57
## 6      56
```

```
test = df
test["Test",] = NA
tail(test)
```

```
##      age sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall
## 299   57   0  0    140  241   0       1      123    1     0.2   1   0     3
## 300   45   1  3    110  264   0       1      132    0     1.2   1   0     3
## 301   68   1  0    144  193   1       1      141    0     3.4   1   2     3
## 302   57   1  0    130  131   0       1      115    1     1.2   1   1     3
## 303   57   0  1    130  236   0       0      174    0     0.0   1   1     2
## Test  NA  NA NA     NA   NA  NA      NA       NA   NA      NA  NA  NA    NA
##      output
## 299       0
## 300       0
## 301       0
## 302       0
## 303       0
## Test     NA
```

Column Indexes

```
col(temp, as.factor = F)
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
##  [1,]    1    2    3    4    5    6    7    8    9    10    11    12    13
##  [2,]    1    2    3    4    5    6    7    8    9    10    11    12    13
##  [3,]    1    2    3    4    5    6    7    8    9    10    11    12    13
##  [4,]    1    2    3    4    5    6    7    8    9    10    11    12    13
##  [5,]    1    2    3    4    5    6    7    8    9    10    11    12    13
##  [6,]    1    2    3    4    5    6    7    8    9    10    11    12    13
##  [7,]    1    2    3    4    5    6    7    8    9    10    11    12    13
##  [8,]    1    2    3    4    5    6    7    8    9    10    11    12    13
##  [9,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [10,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [11,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [12,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [13,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [14,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [15,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [16,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [17,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [18,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [19,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [20,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [21,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [22,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [23,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [24,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [25,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [26,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [27,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [28,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [29,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [30,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [31,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [32,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [33,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [34,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [35,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [36,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [37,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [38,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [39,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [40,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [41,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [42,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [43,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [44,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [45,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [46,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [47,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [48,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [49,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [50,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [51,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [52,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [53,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [54,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [55,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [56,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [57,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [58,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [59,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [60,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [61,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [62,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [63,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [64,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [65,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [66,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [67,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [68,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [69,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [70,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [71,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [72,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [73,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [74,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [75,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [76,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [77,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [78,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [79,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [80,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [81,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [82,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [83,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [84,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [85,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [86,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [87,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [88,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [89,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [90,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [91,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [92,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [93,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [94,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [95,]    1    2    3    4    5    6    7    8    9    10    11    12    13
## [96,]    1    2    3    4    5    6    7    8    9    10    11    12    13
##       [,14]
##  [1,]    14
##  [2,]    14
##  [3,]    14
##  [4,]    14
##  [5,]    14
##  [6,]    14
##  [7,]    14
##  [8,]    14
##  [9,]    14
## [10,]    14
## [11,]    14
## [12,]    14
## [13,]    14
## [14,]    14
## [15,]    14
## [16,]    14
## [17,]    14
## [18,]    14
## [19,]    14
## [20,]    14
## [21,]    14
## [22,]    14
## [23,]    14
## [24,]    14
## [25,]    14
## [26,]    14
## [27,]    14
## [28,]    14
## [29,]    14
## [30,]    14
## [31,]    14
## [32,]    14
## [33,]    14
## [34,]    14
## [35,]    14
## [36,]    14
## [37,]    14
## [38,]    14
## [39,]    14
## [40,]    14
## [41,]    14
## [42,]    14
## [43,]    14
## [44,]    14
## [45,]    14
## [46,]    14
## [47,]    14
## [48,]    14
## [49,]    14
## [50,]    14
## [51,]    14
## [52,]    14
## [53,]    14
## [54,]    14
## [55,]    14
## [56,]    14
## [57,]    14
## [58,]    14
## [59,]    14
## [60,]    14
## [61,]    14
## [62,]    14
## [63,]    14
## [64,]    14
## [65,]    14
## [66,]    14
## [67,]    14
## [68,]    14
## [69,]    14
## [70,]    14
## [71,]    14
## [72,]    14
## [73,]    14
## [74,]    14
## [75,]    14
## [76,]    14
## [77,]    14
## [78,]    14
## [79,]    14
## [80,]    14
## [81,]    14
## [82,]    14
## [83,]    14
## [84,]    14
## [85,]    14
## [86,]    14
## [87,]    14
## [88,]    14
## [89,]    14
## [90,]    14
## [91,]    14
## [92,]    14
## [93,]    14
## [94,]    14
## [95,]    14
## [96,]    14
```

Extracting two(2) columns from df

```
x = df[ , 6:11]
head(x)
```

```
##   fbs restecg thalachh exng oldpeak slp
## 1   1       0      150    0     2.3   0
## 2   0       1      187    0     3.5   0
## 3   0       0      172    0     1.4   2
## 4   0       1      178    0     0.8   2
## 5   0       1      163    1     0.6   2
## 6   0       1      148    0     0.4   1
```

Simple Column and Row Summaries

```
colMeans(x)
```

```
##         fbs     restecg    thalachh        exng     oldpeak         slp 
##   0.1485149   0.5280528 149.6468647   0.3267327   1.0396040   1.3993399
```

```
rowMeans(x)
```

```
##   [1] 25.55000 31.91667 29.23333 30.30000 27.93333 25.06667 25.88333 29.33333
##   [9] 27.75000 29.76667 27.36667 23.70000 29.10000 24.63333 27.66667 26.93333
##  [17] 29.16667 19.60000 29.25000 25.96667 27.25000 30.56667 30.16667 23.66667
##  [25] 30.56667 27.56667 27.10000 21.10000 26.80000 25.70000 28.50000 23.90000
##  [33] 31.66667 25.41667 21.56667 27.06667 29.00000 28.10000 25.30000 25.63333
##  [41] 24.25000 30.20000 25.50000 24.06667 30.66667 29.20000 30.33333 26.33333
##  [49] 19.50000 27.00000 25.25000 25.40000 24.96667 29.60000 29.00000 26.96667
##  [57] 31.33333 31.16667 29.33333 26.83333 22.16667 26.50000 31.83333 22.33333
##  [65] 28.00000 31.06667 24.53333 29.43333 28.83333 27.66667 24.73333 26.33333
##  [73] 34.00000 31.50000 27.86667 27.23333 28.40000 28.00000 31.33333 26.10000
##  [81] 30.33333 28.66667 27.16667 30.36667 20.60000 27.10000 25.83333 26.66667
##  [89] 26.93333 20.66667 29.83333 28.66667 28.66667 27.16667 23.33333 19.00000
##  [97] 26.53333 25.18333 27.81667 29.33333 30.13333 24.86667 30.33333 32.80000
## [105] 27.66667 19.58333 22.18333 25.70000 27.68333 26.83333 26.33333 29.53333
## [113] 22.70000 27.33333 26.33333 28.83333 28.50000 27.48333 29.16667 25.66667
## [121] 21.00000 30.66667 29.16667 28.16667 30.33333 32.61667 24.35000 29.16667
## [129] 28.35000 20.70000 27.66667 27.33333 27.50000 26.00000 27.66667 27.66667
## [137] 16.66667 23.83333 21.75000 18.03333 26.60000 30.70000 29.16667 24.21667
## [145] 20.01667 24.16667 25.21667 29.15000 28.66667 25.50000 23.71667 21.43333
## [153] 26.10000 25.50000 25.66667 22.26667 30.33333 29.50000 24.40000 27.50000
## [161] 28.33333 28.36667 30.83333 29.33333 29.33333 18.58333 22.26667 27.26667
## [169] 24.90000 26.68333 24.26667 28.33333 27.13333 29.70000 22.73333 19.66667
## [177] 27.73333 26.83333 20.75000 19.10000 22.70000 19.33333 28.50000 28.08333
## [185] 21.93333 25.83333 24.90000 18.86667 27.60000 26.66667 24.36667 22.53333
## [193] 19.40000 24.46667 26.50000 24.06667 25.43333 27.70000 17.30000 26.76667
## [201] 29.83333 24.30000 19.13333 25.76667 25.20000 27.50000 24.20000 26.76667
## [209] 23.83333 27.66667 25.23333 24.43333 23.86667 24.83333 24.70000 23.66667
## [217] 16.70000 22.96667 21.80000 25.66667 26.50000 19.76667 29.56667 23.16667
## [225] 21.96667 21.60000 17.56667 22.43333 26.70000 22.63333 25.83333 21.16667
## [233] 24.63333 16.53333 18.73333 29.76667 28.83333 28.70000 27.50000 26.50000
## [241] 19.65000 24.33333 22.50000 15.36667 18.18333 27.91667 25.65000 20.50000
## [249] 32.83333 24.83333 21.53333 24.35000 18.48333 21.31667 21.16667 24.83333
## [257] 22.50000 21.48333 26.40000 31.46667 28.33333 27.16667 16.66667 28.96667
## [265] 18.33333 22.51667 20.73333 21.46667 20.20000 17.76667 24.46667 24.93333
## [273] 12.33333 26.51667 20.16667 28.66667 18.16667 24.05000 25.83333 21.76667
## [281] 21.63333 26.83333 23.20000 30.66667 23.81667 20.80000 27.63333 27.66667
## [289] 24.83333 22.66667 27.33333 24.40000 25.30000 25.30000 24.80000 25.16667
## [297] 23.16667 15.50000 21.03333 22.53333 24.56667 19.86667 29.16667
```

```
colSums(x)
```

```
##      fbs  restecg thalachh     exng  oldpeak      slp
##       45      160    45343       99      315      424
```

```
rowSums(x)
```

```
##   [1] 153.3 191.5 175.4 181.8 167.6 150.4 155.3 176.0 166.5 178.6 164.2 142.2
##  [13] 174.6 147.8 166.0 161.6 175.0 117.6 175.5 155.8 163.5 183.4 181.0 142.0
##  [25] 183.4 165.4 162.6 126.6 160.8 154.2 171.0 143.4 190.0 152.5 129.4 162.4
##  [37] 174.0 168.6 151.8 153.8 145.5 181.2 153.0 144.4 184.0 175.2 182.0 158.0
##  [49] 117.0 162.0 151.5 152.4 149.8 177.6 174.0 161.8 188.0 187.0 176.0 161.0
##  [61] 133.0 159.0 191.0 134.0 168.0 186.4 147.2 176.6 173.0 166.0 148.4 158.0
##  [73] 204.0 189.0 167.2 163.4 170.4 168.0 188.0 156.6 182.0 172.0 163.0 182.2
##  [85] 123.6 162.6 155.0 160.0 161.6 124.0 179.0 172.0 172.0 163.0 140.0 114.0
##  [97] 159.2 151.1 166.9 176.0 180.8 149.2 182.0 196.8 166.0 117.5 133.1 154.2
## [109] 166.1 161.0 158.0 177.2 136.2 164.0 158.0 173.0 171.0 164.9 175.0 154.0
## [121] 126.0 184.0 175.0 169.0 182.0 195.7 146.1 175.0 170.1 124.2 166.0 164.0
## [133] 165.0 156.0 166.0 166.0 100.0 143.0 130.5 108.2 159.6 184.2 175.0 145.3
## [145] 120.1 145.0 151.3 174.9 172.0 153.0 142.3 128.6 156.6 153.0 154.0 133.6
## [157] 182.0 177.0 146.4 165.0 170.0 170.2 185.0 176.0 176.0 111.5 133.6 163.6
## [169] 149.4 160.1 145.6 170.0 162.8 178.2 136.4 118.0 166.4 161.0 124.5 114.6
## [181] 136.2 116.0 171.0 168.5 131.6 155.0 149.4 113.2 165.6 160.0 146.2 135.2
## [193] 116.4 146.8 159.0 144.4 152.6 166.2 103.8 160.6 179.0 145.8 114.8 154.6
## [205] 151.2 165.0 145.2 160.6 143.0 166.0 151.4 146.6 143.2 149.0 148.2 142.0
## [217] 100.2 137.8 130.8 154.0 159.0 118.6 177.4 139.0 131.8 129.6 105.4 134.6
## [229] 160.2 135.8 155.0 127.0 147.8  99.2 112.4 178.6 173.0 172.2 165.0 159.0
## [241] 117.9 146.0 135.0  92.2 109.1 167.5 153.9 123.0 197.0 149.0 129.2 146.1
## [253] 110.9 127.9 127.0 149.0 135.0 128.9 158.4 188.8 170.0 163.0 100.0 173.8
## [265] 110.0 135.1 124.4 128.8 121.2 106.6 146.8 149.6  74.0 159.1 121.0 172.0
## [277] 109.0 144.3 155.0 130.6 129.8 161.0 139.2 184.0 142.9 124.8 165.8 166.0
## [289] 149.0 136.0 164.0 146.4 151.8 151.8 148.8 151.0 139.0  93.0 126.2 135.2
## [301] 147.4 119.2 175.0
```

The apply() command to apply a function over all the rows or columns of a data frame (or matrix)

```
apply(df[ , 10:11], 2, mean)
```

```
##  oldpeak      slp
## 1.039604 1.399340
```

```
apply(df[ , 10:11], 1, mean)
```

```
##   [1] 1.15 1.75 1.70 1.40 1.30 0.70 1.15 1.00 1.25 1.80 1.60 1.10 1.30 1.40 1.50
##  [16] 1.30 1.00 1.30 1.75 1.90 0.75 1.20 1.00 1.00 1.70 1.20 1.80 1.30 1.40 0.60
##  [31] 1.00 1.20 1.00 0.25 1.70 0.70 1.00 1.80 1.40 1.40 1.75 0.60 2.00 0.70 1.00
##  [46] 1.10 1.00 1.00 1.00 1.00 1.25 0.70 1.40 0.80 1.00 1.40 1.00 1.00 1.00 1.00
##  [61] 1.00 1.00 0.50 0.50 1.00 1.70 1.10 0.80 1.00 1.00 0.70 1.00 1.00 1.00 0.60
##  [76] 1.20 1.70 1.00 1.00 0.80 1.00 1.00 1.00 1.10 0.80 1.30 1.50 1.00 1.30 1.00
##  [91] 1.00 1.00 1.00 1.00 0.50 1.00 1.10 1.05 1.95 1.00 1.40 2.10 1.00 0.40 1.00
## [106] 1.25 0.55 0.60 1.55 1.00 1.00 1.10 1.10 1.00 1.00 1.00 1.50 1.45 1.00 0.50
## [121] 1.50 1.00 1.00 1.00 1.00 1.35 1.05 1.00 0.55 1.10 1.00 0.50 1.00 1.00 1.00
## [136] 1.00 1.00 1.00 1.25 0.60 1.30 1.10 0.50 1.15 1.05 1.00 0.65 1.45 1.00 1.00
## [151] 2.15 1.30 0.80 0.50 0.50 0.80 1.00 1.00 0.70 1.00 0.00 1.60 1.00 1.00 1.00
## [166] 1.25 1.80 1.80 1.20 1.55 0.80 0.50 1.40 2.60 1.70 1.50 1.70 1.00 1.75 0.80
## [181] 1.10 1.00 1.00 1.75 1.80 1.00 1.70 1.60 0.80 1.00 1.10 1.60 1.20 1.90 2.00
## [196] 1.70 2.30 0.60 1.40 1.30 1.00 1.90 1.40 1.30 3.10 1.00 1.10 1.80 1.50 1.00
## [211] 0.70 2.30 1.10 1.00 1.10 2.00 1.10 1.90 1.90 1.00 2.50 2.80 1.20 2.00 1.90
## [226] 1.30 1.20 1.30 0.60 1.40 1.00 1.00 0.90 1.10 1.70 1.80 1.00 1.10 1.00 1.00
## [241] 1.95 0.50 1.50 1.10 1.55 0.75 1.45 0.50 1.00 1.50 2.60 0.55 1.45 0.95 1.00
## [256] 0.50 2.00 0.95 1.20 2.40 1.00 1.00 1.50 1.40 0.50 1.05 2.20 1.40 2.10 0.80
## [271] 1.40 1.80 1.00 1.05 1.00 1.50 1.50 1.15 1.00 2.30 1.40 1.00 1.60 1.00 1.95
## [286] 1.40 1.40 1.00 2.00 1.50 1.00 2.20 1.90 0.90 1.40 3.00 0.50 1.00 0.60 1.10
## [301] 2.20 1.10 0.50
```

```
apply(df[ , 10:11], 2, median)
```

```
## oldpeak     slp
##     0.8     1.0
```

```
apply(df[ , 10:11], 2, var)
```

```
##   oldpeak       slp
## 1.3480952 0.3797347
```

Using tapply() to Summarize Using a Grouping Variable

```
tapply(df$age, df$sex, FUN = mean)
```

```
##        0        1
## 55.67708 53.75845
```

```
tapply(df$age, df$sex, FUN = var)
```

```
##        0        1
## 88.53673 78.92195
```

The aggregate() command enables you to compute summary statistics for subsets of a data frame or matrix; the result comes out as a single matrix rather than an array item, even with multiple grouping factors

```
aggregate(df[ , 10:11], by = list(df$age), FUN = mean)
```

```
##    Group.1   oldpeak       slp
## 1       29 0.0000000 2.0000000
## 2       34 0.3500000 2.0000000
## 3       35 0.7500000 1.7500000
## 4       37 1.7500000 1.0000000
## 5       38 1.2666667 1.6666667
## 6       39 0.3000000 1.5000000
## 7       40 1.1333333 1.6666667
## 8       41 0.3400000 1.8000000
## 9       42 0.5000000 1.3750000
## 10      43 1.3000000 1.3750000
## 11      44 0.3727273 1.6363636
## 12      45 0.6250000 1.2500000
## 13      46 1.0857143 1.2857143
## 14      47 0.2200000 1.8000000
## 15      48 0.2714286 1.4285714
## 16      49 0.6800000 1.6000000
## 17      50 0.9714286 1.4285714
## 18      51 1.2666667 1.6666667
## 19      52 0.3769231 1.6923077
## 20      53 0.8375000 1.2500000
## 21      54 0.9312500 1.4375000
## 22      55 1.9500000 1.1250000
## 23      56 1.4000000 0.9090909
## 24      57 0.7176471 1.4117647
## 25      58 1.3894737 1.3684211
## 26      59 1.0785714 1.2857143
## 27      60 1.6818182 1.4545455
## 28      61 1.7125000 1.3750000
## 29      62 1.8636364 1.0000000
## 30      63 1.7000000 1.3333333
## 31      64 1.0800000 1.2000000
## 32      65 1.0750000 1.6250000
## 33      66 0.9142857 1.1428571
## 34      67 0.9888889 1.2222222
## 35      68 1.8750000 1.2500000
## 36      69 1.3000000 1.3333333
## 37      70 1.9750000 1.0000000
## 38      71 0.6666667 1.6666667
## 39      74 0.2000000 2.0000000
## 40      76 1.1000000 1.0000000
## 41      77 0.0000000 2.0000000
```

```
aggregate(cbind(df$sex, df$age), data = df, by = list(df$trtbps), FUN = mean)
```

```
##    Group.1        V1       V2
## 1       94 0.5000000 45.00000
## 2      100 0.7500000 58.50000
## 3      101 1.0000000 46.00000
## 4      102 0.0000000 51.00000
## 5      104 1.0000000 45.00000
## 6      105 0.3333333 48.33333
## 7      106 0.0000000 67.00000
## 8      108 0.5000000 52.33333
## 9      110 0.8421053 51.84211
## 10     112 0.6666667 51.66667
## 11     114 1.0000000 58.00000
## 12     115 0.6666667 51.66667
## 13     117 1.0000000 60.00000
## 14     118 0.7142857 45.71429
## 15     120 0.7297297 52.94595
## 16     122 0.7500000 45.00000
## 17     123 1.0000000 53.00000
## 18     124 0.6666667 55.33333
## 19     125 1.0000000 58.90909
## 20     126 0.6666667 45.00000
## 21     128 0.7500000 55.41667
## 22     129 1.0000000 50.00000
## 23     130 0.6666667 52.11111
## 24     132 0.6250000 52.62500
## 25     134 0.6000000 55.40000
## 26     135 0.5000000 56.16667
## 27     136 0.3333333 50.66667
## 28     138 0.5384615 49.23077
## 29     140 0.6875000 56.59375
## 30     142 0.6666667 48.00000
## 31     144 1.0000000 59.00000
## 32     145 0.8000000 63.60000
## 33     146 0.5000000 62.00000
## 34     148 1.0000000 51.50000
## 35     150 0.5882353 57.41176
## 36     152 0.8000000 56.60000
## 37     154 1.0000000 57.00000
## 38     155 0.0000000 65.00000
## 39     156 1.0000000 70.00000
## 40     160 0.6363636 64.00000
## 41     164 1.0000000 59.00000
## 42     165 1.0000000 57.00000
## 43     170 0.7500000 60.00000
## 44     172 1.0000000 52.00000
## 45     174 0.0000000 59.00000
## 46     178 0.5000000 62.50000
## 47     180 0.3333333 62.33333
## 48     192 1.0000000 54.00000
## 49     200 0.0000000 56.00000
```

The na.omit() command strips out unwanted NA items from vectors and data frames.

```
head(na.omit(df))
```

```
##   age sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall output
## 1  63   1  3    145  233   1       0      150    0     2.3   0   0     1      1
## 2  37   1  2    130  250   0       1      187    0     3.5   0   0     2      1
## 3  41   0  1    130  204   0       0      172    0     1.4   2   0     2      1
## 4  56   1  1    120  236   0       1      178    0     0.8   2   0     2      1
## 5  57   0  0    120  354   0       1      163    1     0.6   2   0     2      1
## 6  57   1  0    140  192   0       1      148    0     0.4   1   0     1      1
```

ls and objects return a vector of character strings giving the names of the objects in the specified environment

```
objects(df)
```

```
##  [1] "age"      "caa"      "chol"     "cp"       "exng"     "fbs"     
##  [7] "oldpeak"  "output"   "restecg"  "sex"      "slp"      "thalachh"
## [13] "thall"    "trtbps"
```

```
ls(df)
```

```
##  [1] "age"      "caa"      "chol"     "cp"       "exng"     "fbs"     
##  [7] "oldpeak"  "output"   "restecg"  "sex"      "slp"      "thalachh"
## [13] "thall"    "trtbps"
```

Simple lineaR RegReSSion

```
dflm = lm(age ~ trtbps, data = df)
dflm
```

```
## 
## Call:
## lm(formula = age ~ trtbps, data = df)
## 
## Coefficients:
## (Intercept)       trtbps  
##     35.3255       0.1447
```

```
cor.test(~ df$trtbps + df$age, data = df)
```

```
##
##  Pearson's product-moment correlation
##
## data:  df$trtbps and df$age
## t = 5.0475, df = 301, p-value = 7.762e-07
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.1720897 0.3800657
## sample estimates:
##       cor
## 0.2793509
```

Linear Model Results Objects

can extract the coefficients using the coef() command

```
coef(dflm)
```

```
## (Intercept)      trtbps
##  35.3254525   0.1446614
```

can obtain confidence intervals on these coefficients using the confint() command

```
confint(dflm)
```

```
##                  2.5 %     97.5 %
## (Intercept) 27.8365649 42.8143401
## trtbps       0.0882621  0.2010608
```

Fitted Values

```
head(fitted(dflm))
```

```
##        1        2        3        4        5        6
## 56.30136 54.13144 54.13144 52.68483 52.68483 55.57806
```

Residuals

```
head(residuals(dflm))
```

```
##          1          2          3          4          5          6
##   6.698637 -17.131441 -13.131441   3.315174   4.315174   1.421945
```

Formula

```
formula(dflm)
```

```
## age ~ trtbps
```

```
dflm$call
```

```
## lm(formula = age ~ trtbps, data = df)
```

```
formula(df)
```

```
## age ~ sex + cp + trtbps + chol + fbs + restecg + thalachh + exng +
##     oldpeak + slp + caa + thall + output
```

Using the segments() Command for Error Bars

```
df.m = apply(df[,10:11], 2, mean)
df.m
```

```
##  oldpeak      slp
## 1.039604 1.399340
```

```
df.sd = apply(df[ , 10:11], 2, sd)
df.sd
```

```
##   oldpeak       slp
## 1.1610750 0.6162261
```

```
df.s = apply(df[,10:11], 2, sum)
df.s
```

```
## oldpeak     slp
##     315     424
```

```
df.1 = df.s/df.m
df.1
```

```
## oldpeak     slp
##     303     303
```

```
df.se = df.sd / sqrt(df.1)
df.se
```

```
##    oldpeak        slp
## 0.06670202 0.03540127
```

```
df.m + df.se
```

```
##  oldpeak      slp
## 1.106306 1.434741
```

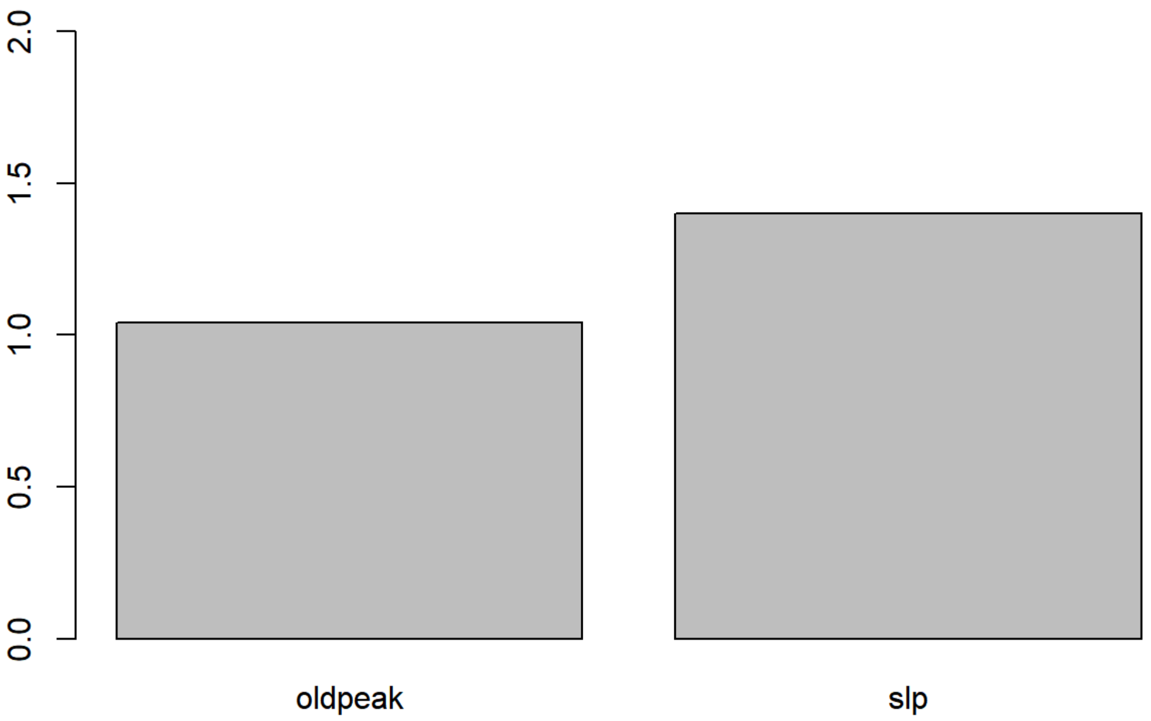
```
max(df.m + df.se)
```

```
## [1] 1.434741
```

```
df.max = round(max(df.m + df.se) + 0.5, 0)
df.max
```

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

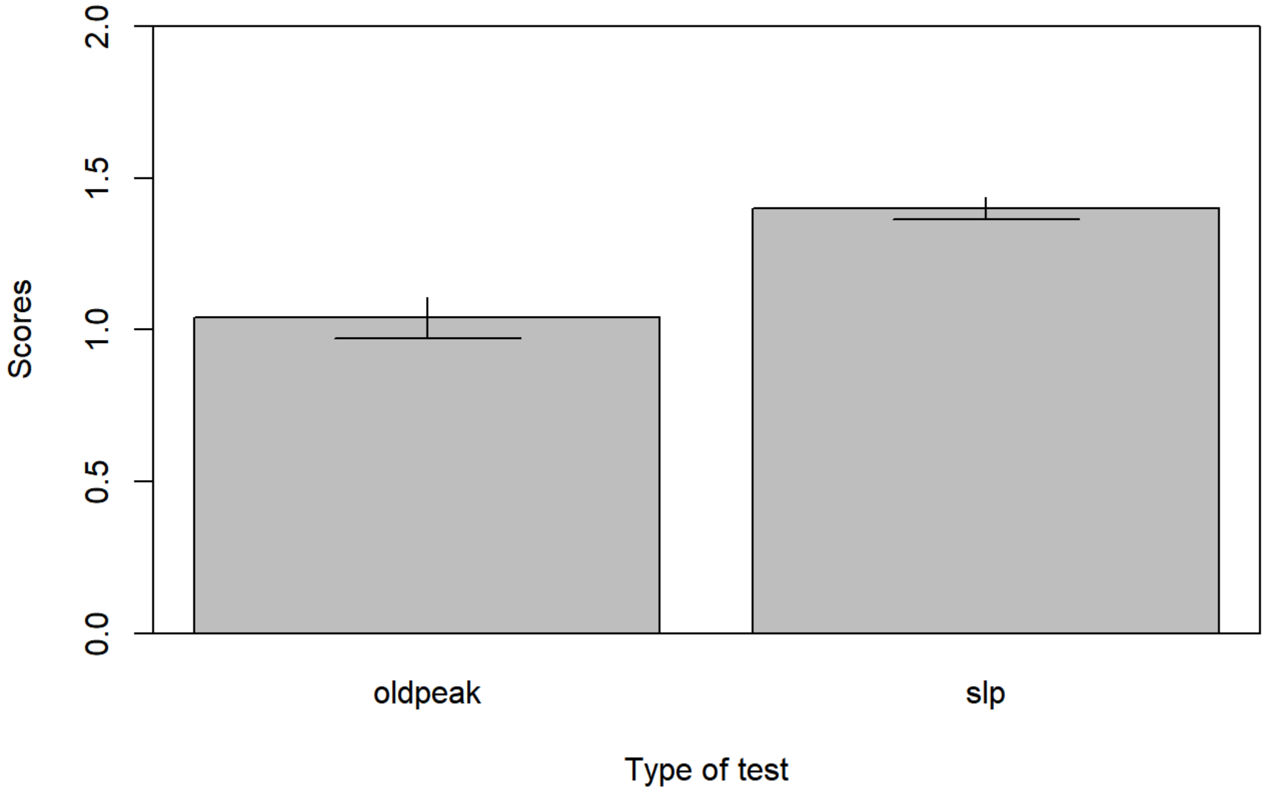
```
bp = barplot(df.m, ylim = c(0, df.max))
```



```
bp
```

```
##      [,1]
## [1,]  0.7
## [2,]  1.9
```

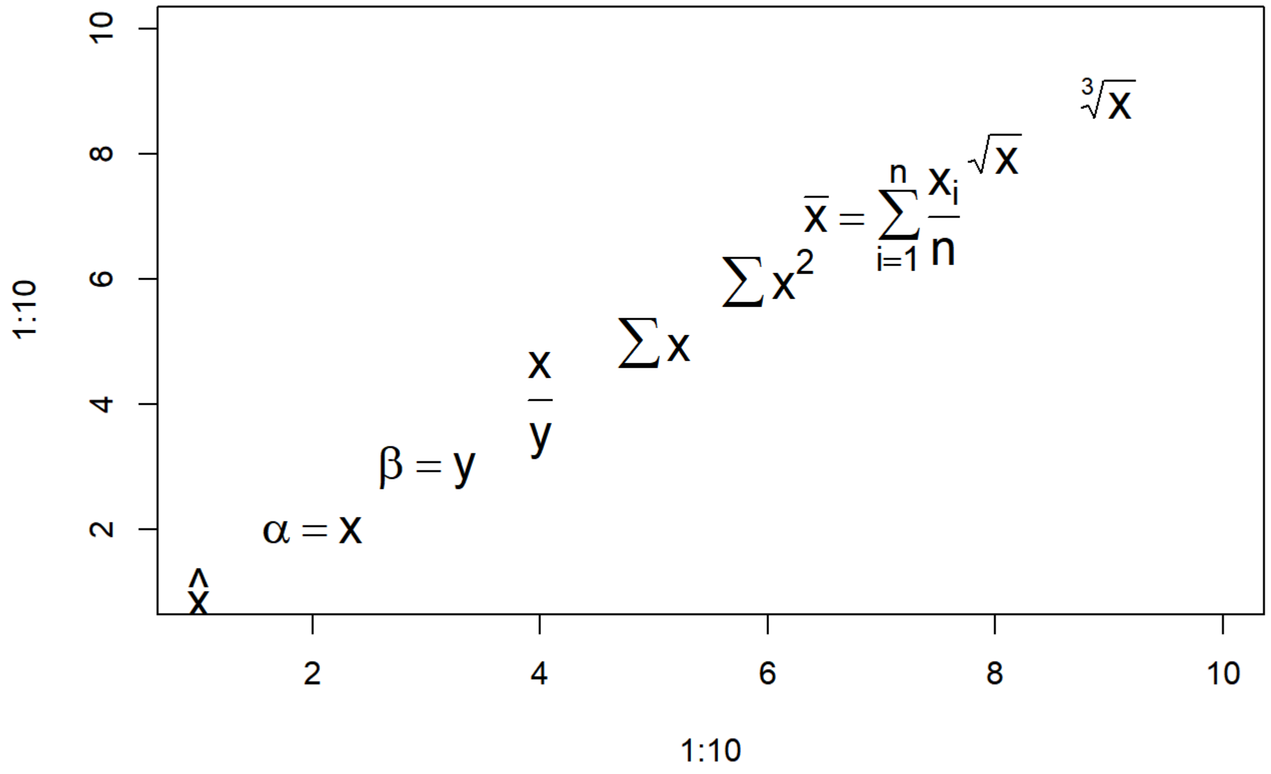
```
bp = barplot(df.m, ylim = c(0, df.max))
segments(bp, df.m + df.se, bp, df.m - df.se)
segments(bp - 0.2, df.m - df.se, bp + 0.2, df.m - df.se)
box()
title(xlab = 'Type of test', ylab = 'Scores')
```



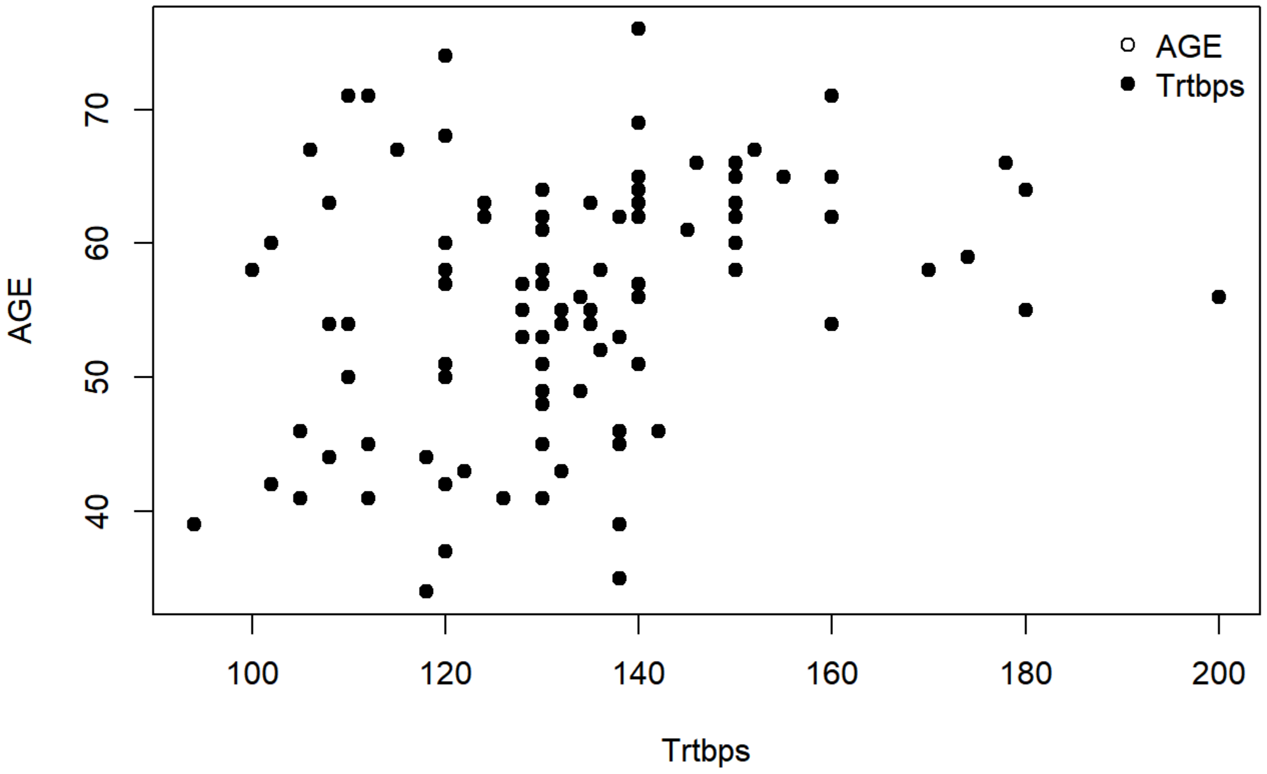
##

Creating Mathematical Expressions

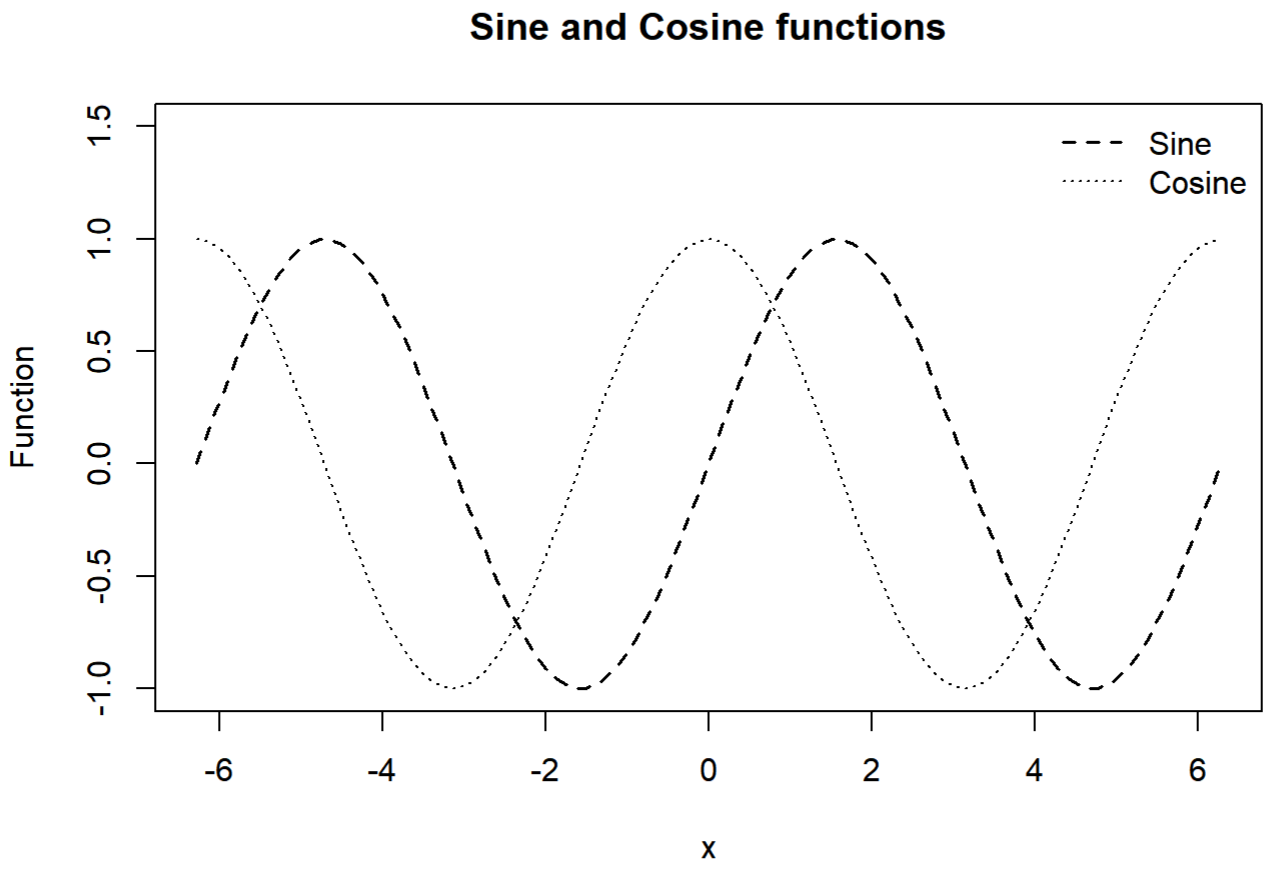
```
plot(1:10, 1:10, type = 'n')
opt = par(cex = 1.5)
text(1, 1, expression(hat(x)))
text(2, 2, expression(alpha==x))
text(3, 3, expression(beta==y))
text(4, 4, expression(frac(x, y)))
text(5, 5, expression(sum(x)))
text(6, 6, expression(sum(x^2)))
text(7, 7, expression(bar(x) == sum(frac(x[i], n), i==1, n)))
text(8, 8, expression(sqrt(x)))
text(9, 9, expression(sqrt(x, 3)))
```



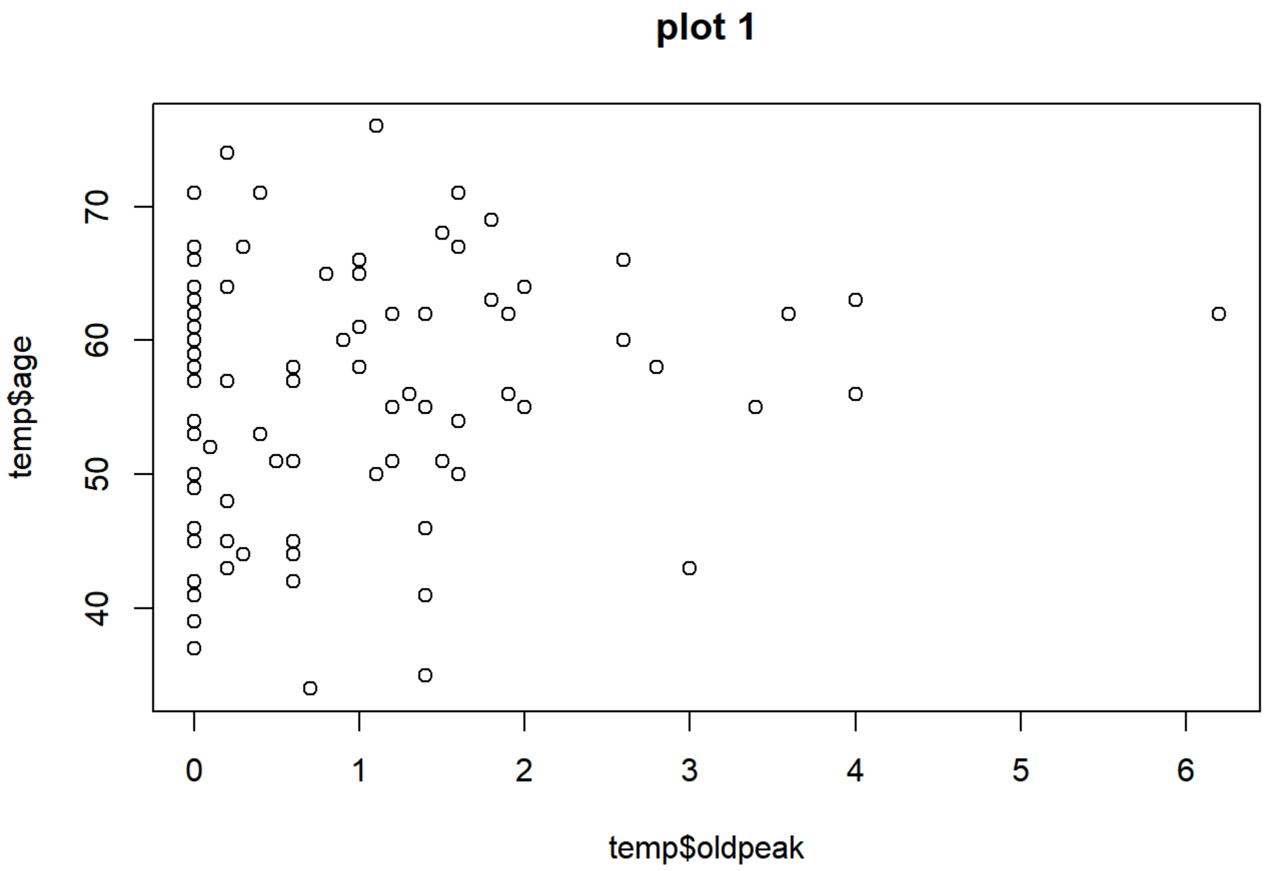
```
plot(temp$age ~ temp$trtbps, data = temp, pch = 21, ylab = 'AGE', xlab = 'Trtbps')
points(temp$age ~ temp$trtbps, data = temp, pch = 19)
legend(x = 'topright', legend = c('AGE', 'Trtbps'), pch = c(21,19), bty =
'n')
```



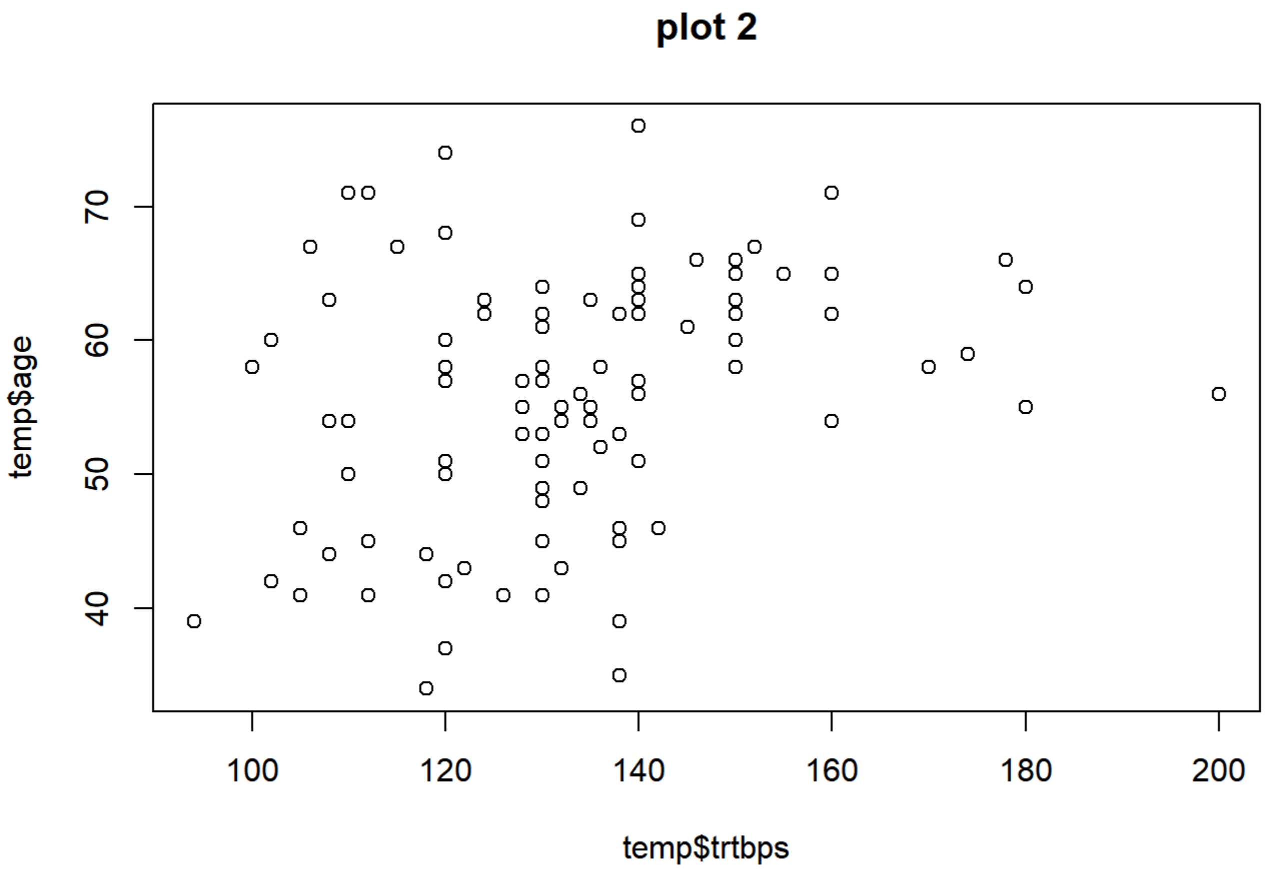
```
curve(sin, -pi*2, pi*2, lty = 2, lwd = 1.5, ylab = 'Function', ylim = c(-1,1.5))
curve(cos, -pi*2, pi*2, lty = 3, lwd = 1, add = TRUE)
legend(x = 'topright', legend = c('Sine', 'Cosine'), lty = c(2, 3),
lwd = c(1.5, 1), bty = 'n')
title(main = 'Sine and Cosine functions')
```



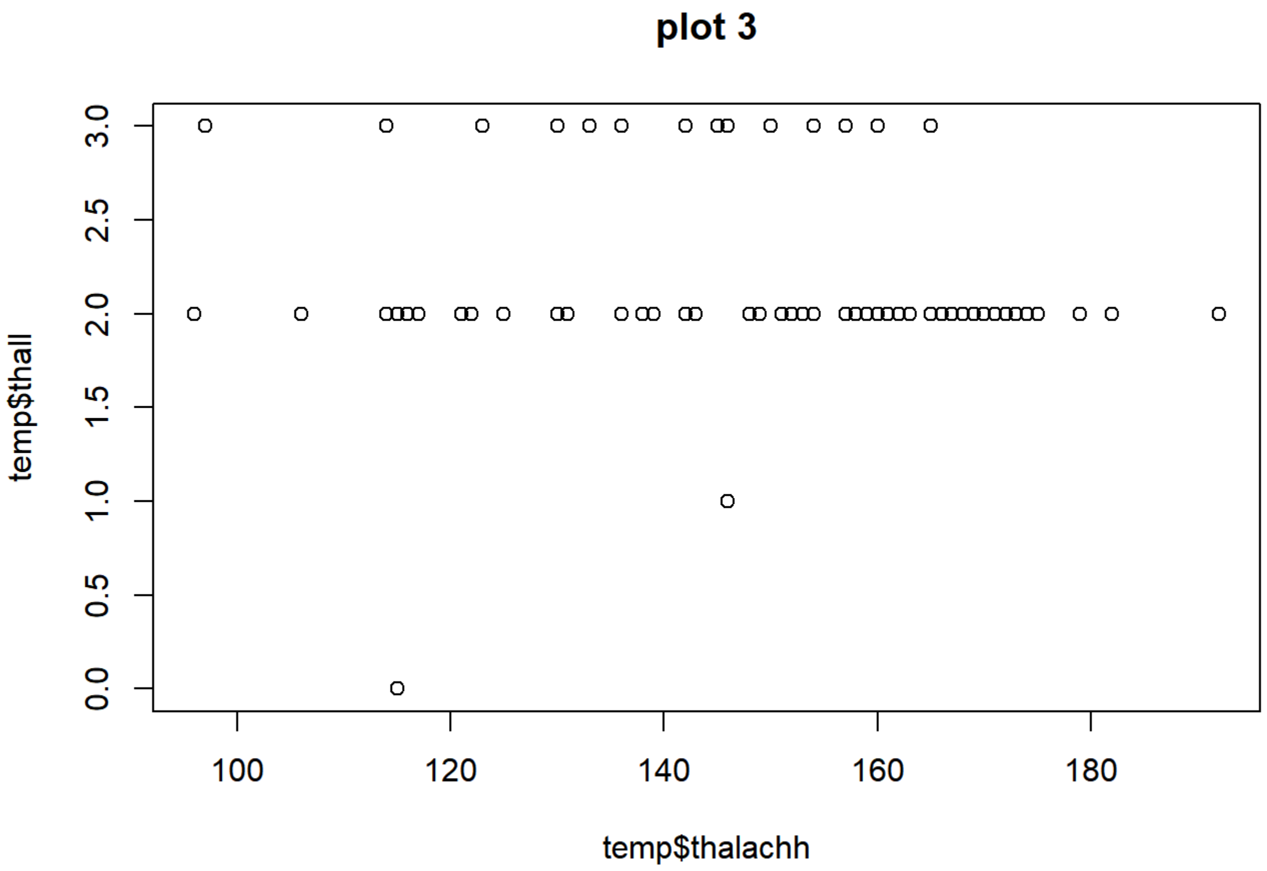
```
plot(temp$age ~ temp$oldpeak, data = df, main = 'plot 1')
```



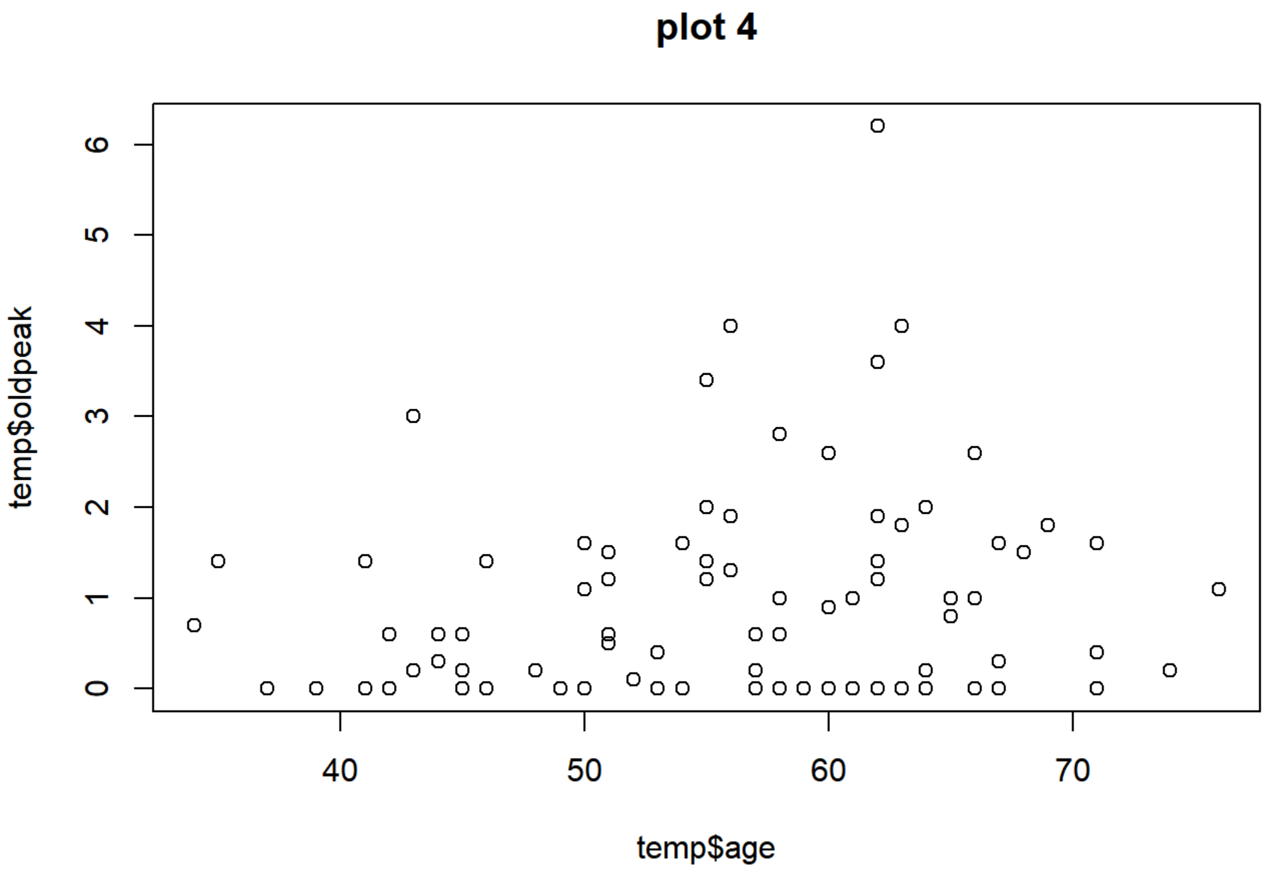
```
plot(temp$age ~ temp$trtbps, data = df, main = 'plot 2')
```



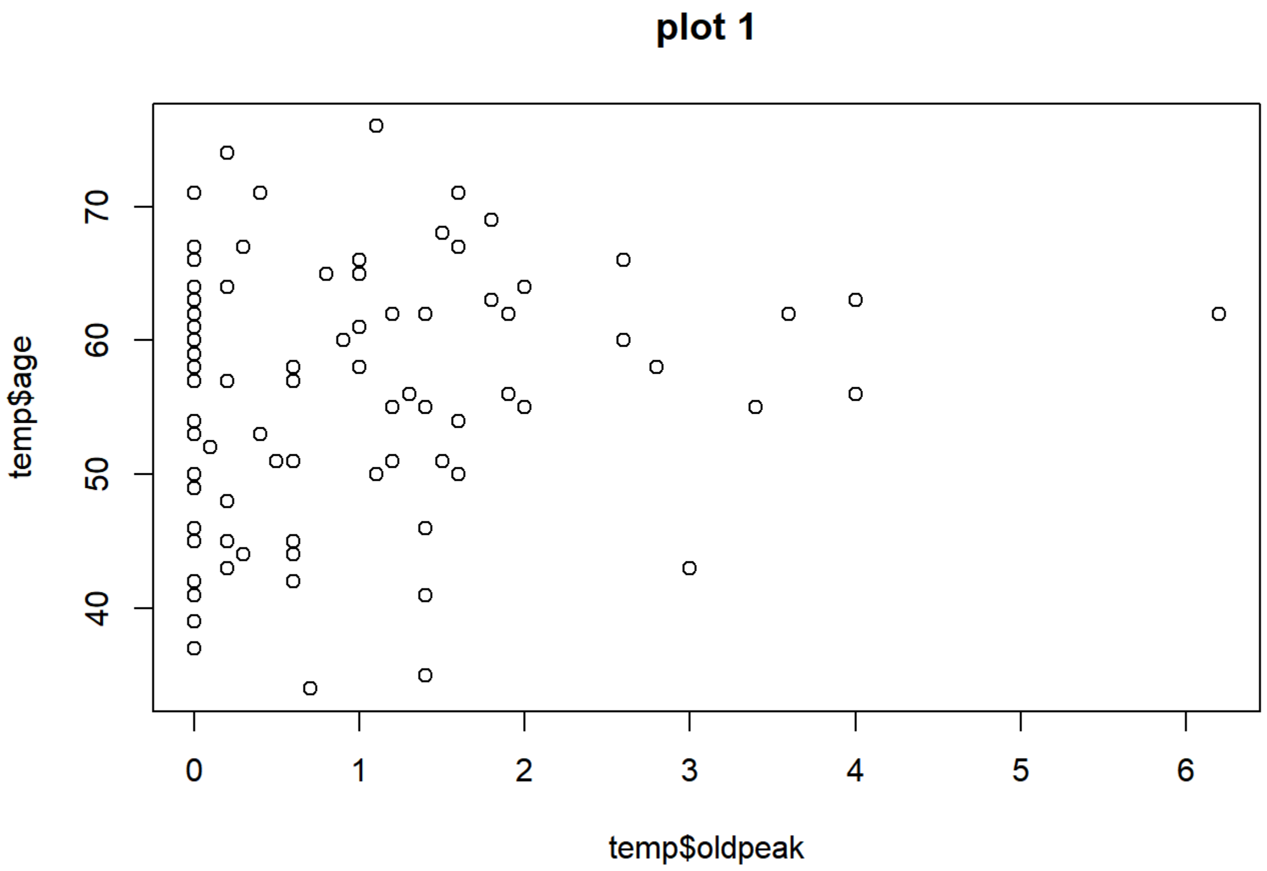
```
plot(temp$thall ~ temp$thalachh, data = df, main = 'plot 3')
```



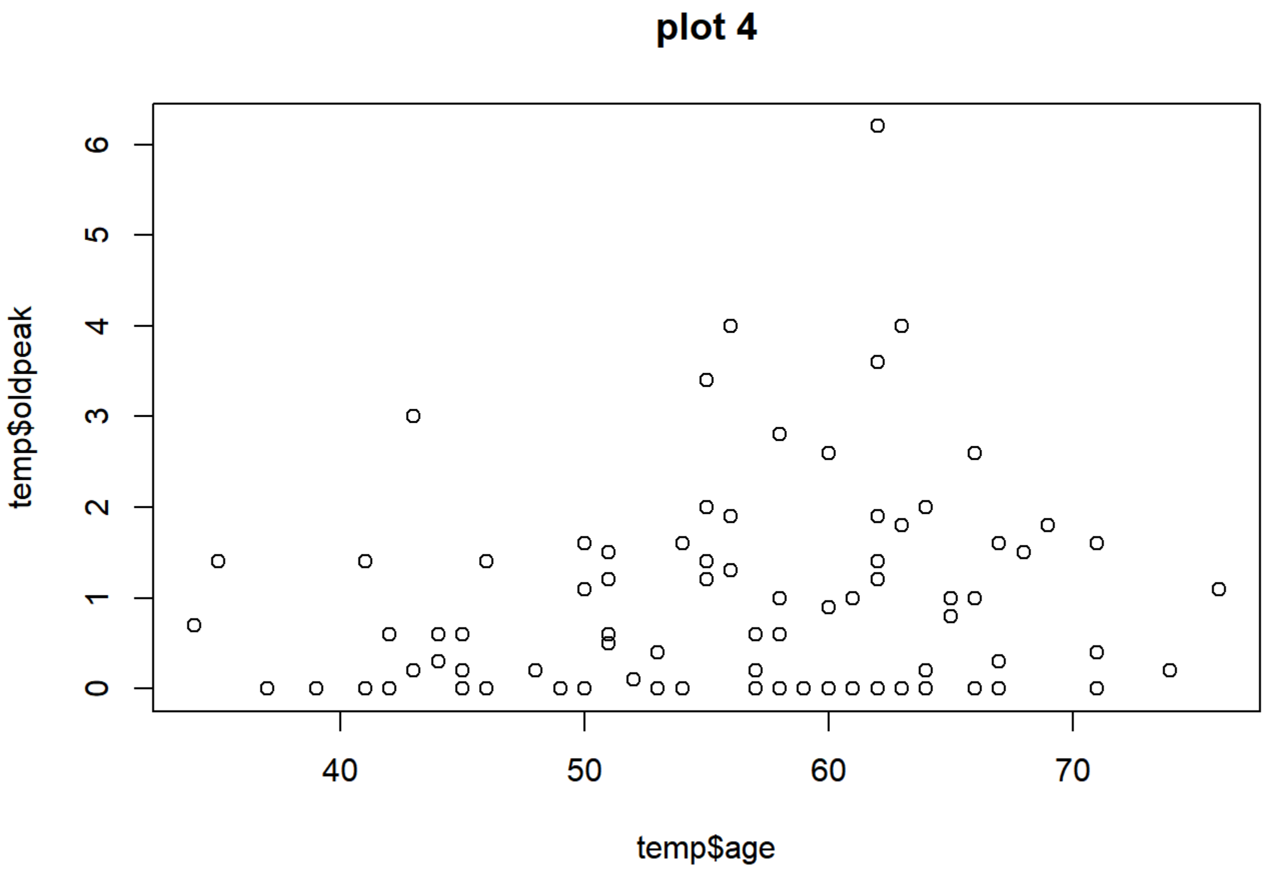
```
plot(temp$oldpeak ~ temp$age, data = df, main = 'plot 4')
```



```
par(opt)
plot(temp$age ~ temp$oldpeak, data = df, main = 'plot 1')
```



```
plot.new()
plot.new()
plot(temp$oldpeak ~ temp$age, data = df, main = 'plot 4')
```



##

Simple customized Functions with multiple lines

```
cummeadian = function(x) {
tmp = seq_along(x)
for(i in 1:length(tmp)) tmp[i] = median(x[1:i])
print(x)}
```

```
cummeadian(temp$age)
```

```
##  [1] 41 57 56 48 58 50 58 66 69 71 65 41 46 54 65 65 51 53 53 53 51 44 63 57 71
## [26] 35 45 62 43 55 60 42 67 54 58 54 45 62 63 68 45 50 50 64 64 37 46 46 64 41
## [51] 54 39 34 67 52 74 54 49 41 49 60 51 42 67 76 44 60 71 66 39 58 55 62 65 61
## [76] 51 62 60 61 43 62 63 56 59 56 62 62 66 63 55 58 55 58 63 57 57
```

One-Line Functions

```
log2 = function(x) log(x, base = 2)
log2
```

```
## function(x) log(x, base = 2)
```

```
log2(temp$thall)
```

```
##  [1] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
##  [9] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
## [17] 1.000000 1.000000     -Inf 1.000000 1.000000 1.000000 1.000000 1.000000
## [25] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
## [33] 1.584963 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
## [41] 1.000000 1.000000 1.000000 1.000000 1.584963 1.000000 1.000000 1.000000
## [49] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
## [57] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
## [65] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
## [73] 1.000000 1.584963 1.000000 1.584963 1.584963 1.584963 1.584963 1.584963
## [81] 1.584963 1.584963 1.584963 1.000000 1.584963 1.000000 1.000000 1.584963
## [89] 1.000000 1.000000 1.000000 1.584963 0.000000 1.000000 1.584963 1.000000
```

Using Default Values in Functions

```
manning = function(radius, gradient, coef=0.1125) (radius^(2/3)*gradient^0.5/coef)
```

```
manning(radius = 1, gradient = 1/500)
```

```
## [1] 0.3975232
```

Chaining Functions Together with %>%, the Pipe Operator

```
df %>% filter(df$sex == 1) %>% summary()
```

```
##       age             sex          cp             trtbps           chol      
##  Min.   :29.00   Min.   :1   Min.   :0.0000   Min.   : 94.0   Min.   :126.0  
##  1st Qu.:47.00   1st Qu.:1   1st Qu.:0.0000   1st Qu.:120.0   1st Qu.:208.0  
##  Median :54.00   Median :1   Median :0.0000   Median :130.0   Median :235.0  
##  Mean   :53.76   Mean   :1   Mean   :0.9324   Mean   :130.9   Mean   :239.3  
##  3rd Qu.:59.50   3rd Qu.:1   3rd Qu.:2.0000   3rd Qu.:140.0   3rd Qu.:268.0  
##  Max.   :77.00   Max.   :1   Max.   :3.0000   Max.   :192.0   Max.   :353.0  
##       fbs            restecg          thalachh        exng      
##  Min.   :0.0000   Min.   :0.0000   Min.   : 71   Min.   :0.000  
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:132   1st Qu.:0.000  
##  Median :0.0000   Median :1.0000   Median :151   Median :0.000  
##  Mean   :0.1594   Mean   :0.5072   Mean   :149   Mean   :0.372  
##  3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:168   3rd Qu.:1.000  
##  Max.   :1.0000   Max.   :2.0000   Max.   :202   Max.   :1.000  
##     oldpeak           slp             caa              thall      
##  Min.   :0.000   Min.   :0.000   Min.   :0.0000   Min.   :0.000  
##  1st Qu.:0.000   1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:2.000  
##  Median :0.800   Median :1.000   Median :0.0000   Median :2.000  
##  Mean   :1.115   Mean   :1.386   Mean   :0.8116   Mean   :2.401  
##  3rd Qu.:1.800   3rd Qu.:2.000   3rd Qu.:1.0000   3rd Qu.:3.000  
##  Max.   :5.600   Max.   :2.000   Max.   :4.0000   Max.   :3.000  
##      output      
##  Min.   :0.0000  
##  1st Qu.:0.0000  
##  Median :0.0000  
##  Mean   :0.4493  
##  3rd Qu.:1.0000  
##  Max.   :1.0000
```

```
df %>% filter(df$sex == 0) %>% summary()
```

```
##       age             sex          cp            trtbps           chol
##  Min.   :34.00   Min.   :0   Min.   :0.000   Min.   : 94.0   Min.   :141.0
##  1st Qu.:49.75   1st Qu.:0   1st Qu.:0.000   1st Qu.:120.0   1st Qu.:214.8
##  Median :57.00   Median :0   Median :1.000   Median :131.0   Median :253.0
##  Mean   :55.68   Mean   :0   Mean   :1.042   Mean   :133.1   Mean   :261.3
##  3rd Qu.:63.00   3rd Qu.:0   3rd Qu.:2.000   3rd Qu.:140.0   3rd Qu.:296.8
##  Max.   :76.00   Max.   :0   Max.   :3.000   Max.   :200.0   Max.   :564.0
##       fbs           restecg          thalachh          exng
##  Min.   :0.000   Min.   :0.0000   Min.   : 96.0   Min.   :0.0000
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:141.2   1st Qu.:0.0000
##  Median :0.000   Median :1.0000   Median :157.0   Median :0.0000
##  Mean   :0.125   Mean   :0.5729   Mean   :151.1   Mean   :0.2292
##  3rd Qu.:0.000   3rd Qu.:1.0000   3rd Qu.:165.0   3rd Qu.:0.0000
##  Max.   :1.000   Max.   :2.0000   Max.   :192.0   Max.   :1.0000
##     oldpeak           slp             caa             thall
##  Min.   :0.000   Min.   :0.000   Min.   :0.0000   Min.   :0.000
##  1st Qu.:0.000   1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:2.000
##  Median :0.600   Median :1.000   Median :0.0000   Median :2.000
##  Mean   :0.876   Mean   :1.427   Mean   :0.5521   Mean   :2.125
##  3rd Qu.:1.400   3rd Qu.:2.000   3rd Qu.:1.0000   3rd Qu.:2.000
##  Max.   :6.200   Max.   :2.000   Max.   :3.0000   Max.   :3.000
##      output
##  Min.   :0.00
##  1st Qu.:0.75
##  Median :1.00
##  Mean   :0.75
##  3rd Qu.:1.00
##  Max.   :1.00
```

Without the pipe operator, the preceding code would be written like this:

```
summary(filter(df, sex == 0))
```

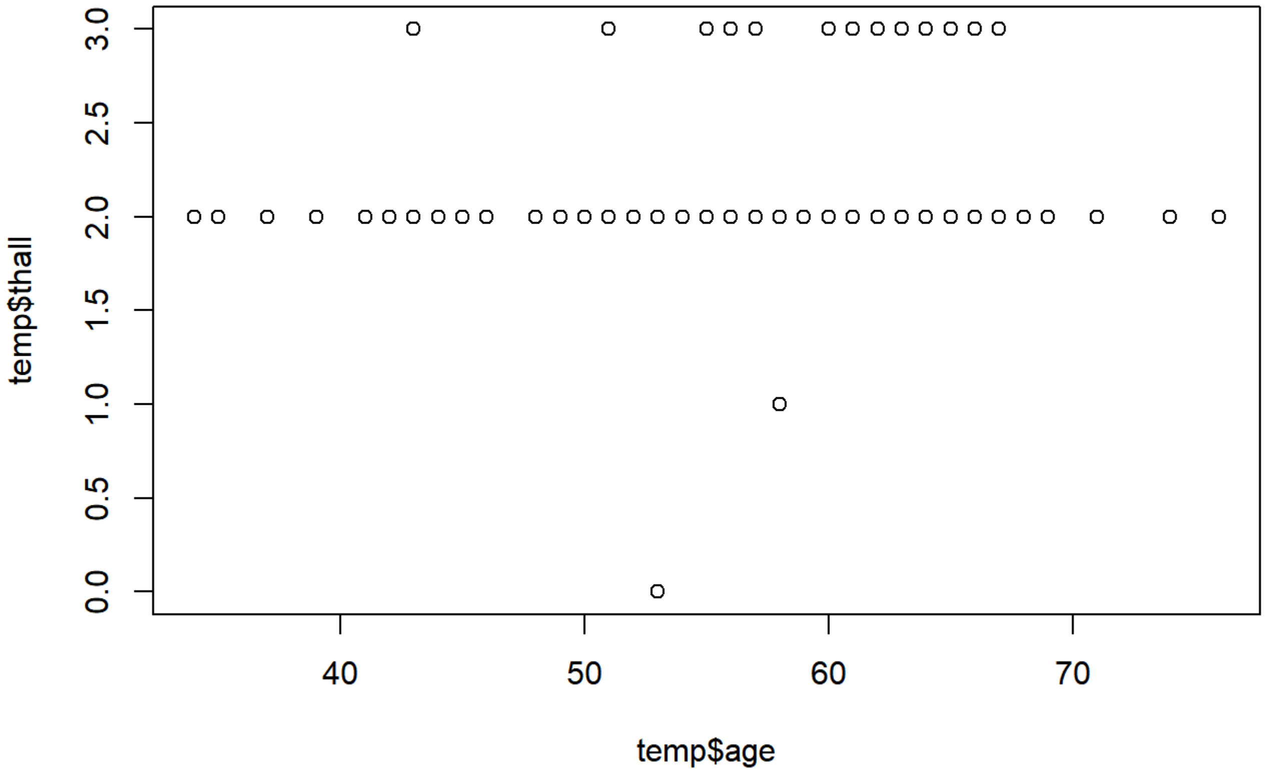
```
##       age             sex          cp            trtbps           chol      
##  Min.   :34.00   Min.   :0   Min.   :0.000   Min.   : 94.0   Min.   :141.0  
##  1st Qu.:49.75   1st Qu.:0   1st Qu.:0.000   1st Qu.:120.0   1st Qu.:214.8  
##  Median :57.00   Median :0   Median :1.000   Median :131.0   Median :253.0  
##  Mean   :55.68   Mean   :0   Mean   :1.042   Mean   :133.1   Mean   :261.3  
##  3rd Qu.:63.00   3rd Qu.:0   3rd Qu.:2.000   3rd Qu.:140.0   3rd Qu.:296.8  
##  Max.   :76.00   Max.   :0   Max.   :3.000   Max.   :200.0   Max.   :564.0  
##       fbs           restecg          thalachh          exng       
##  Min.   :0.000   Min.   :0.0000   Min.   : 96.0   Min.   :0.0000  
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:141.2   1st Qu.:0.0000  
##  Median :0.000   Median :1.0000   Median :157.0   Median :0.0000  
##  Mean   :0.125   Mean   :0.5729   Mean   :151.1   Mean   :0.2292  
##  3rd Qu.:0.000   3rd Qu.:1.0000   3rd Qu.:165.0   3rd Qu.:0.0000  
##  Max.   :1.000   Max.   :2.0000   Max.   :192.0   Max.   :1.0000  
##     oldpeak           slp             caa             thall      
##  Min.   :0.000   Min.   :0.000   Min.   :0.0000   Min.   :0.000  
##  1st Qu.:0.000   1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:2.000  
##  Median :0.600   Median :1.000   Median :0.0000   Median :2.000  
##  Mean   :0.876   Mean   :1.427   Mean   :0.5521   Mean   :2.125  
##  3rd Qu.:1.400   3rd Qu.:2.000   3rd Qu.:1.0000   3rd Qu.:2.000  
##  Max.   :6.200   Max.   :2.000   Max.   :3.0000   Max.   :3.000  
##      output    
##  Min.   :0.00  
##  1st Qu.:0.75  
##  Median :1.00  
##  Mean   :0.75  
##  3rd Qu.:1.00  
##  Max.   :1.00
```

```
summary(filter(df, age == 65))
```

```
##       age          sex            cp            trtbps           chol      
##  Min.   :65   Min.   :0.0   Min.   :0.000   Min.   :110.0   Min.   :177.0  
##  1st Qu.:65   1st Qu.:0.0   1st Qu.:0.000   1st Qu.:131.2   1st Qu.:242.2  
##  Median :65   Median :0.5   Median :1.000   Median :139.0   Median :261.5  
##  Mean   :65   Mean   :0.5   Mean   :1.125   Mean   :138.5   Mean   :279.0  
##  3rd Qu.:65   3rd Qu.:1.0   3rd Qu.:2.000   3rd Qu.:151.2   3rd Qu.:301.5  
##  Max.   :65   Max.   :1.0   Max.   :3.000   Max.   :160.0   Max.   :417.0  
##       fbs          restecg        thalachh          exng      oldpeak     
##  Min.   :0.00   Min.   :0.00   Min.   :114.0   Min.   :0   Min.   :0.400  
##  1st Qu.:0.00   1st Qu.:0.00   1st Qu.:136.8   1st Qu.:0   1st Qu.:0.750  
##  Median :0.00   Median :0.00   Median :149.5   Median :0   Median :0.800  
##  Mean   :0.25   Mean   :0.25   Mean   :146.1   Mean   :0   Mean   :1.075  
##  3rd Qu.:0.25   3rd Qu.:0.25   3rd Qu.:157.2   3rd Qu.:0   3rd Qu.:1.100  
##  Max.   :1.00   Max.   :1.00   Max.   :174.0   Max.   :0   Max.   :2.800  
##       slp             caa           thall          output   
##  Min.   :1.000   Min.   :0.00   Min.   :1.00   Min.   :0.0  
##  1st Qu.:1.000   1st Qu.:0.00   1st Qu.:2.00   1st Qu.:0.0  
##  Median :2.000   Median :1.00   Median :2.00   Median :0.5  
##  Mean   :1.625   Mean   :1.00   Mean   :2.25   Mean   :0.5  
##  3rd Qu.:2.000   3rd Qu.:1.25   3rd Qu.:3.00   3rd Qu.:1.0  
##  Max.   :2.000   Max.   :3.00   Max.   :3.00   Max.   :1.0  
```

Creating a Scatter Plot

```
plot(temp$age, temp$thall)
```



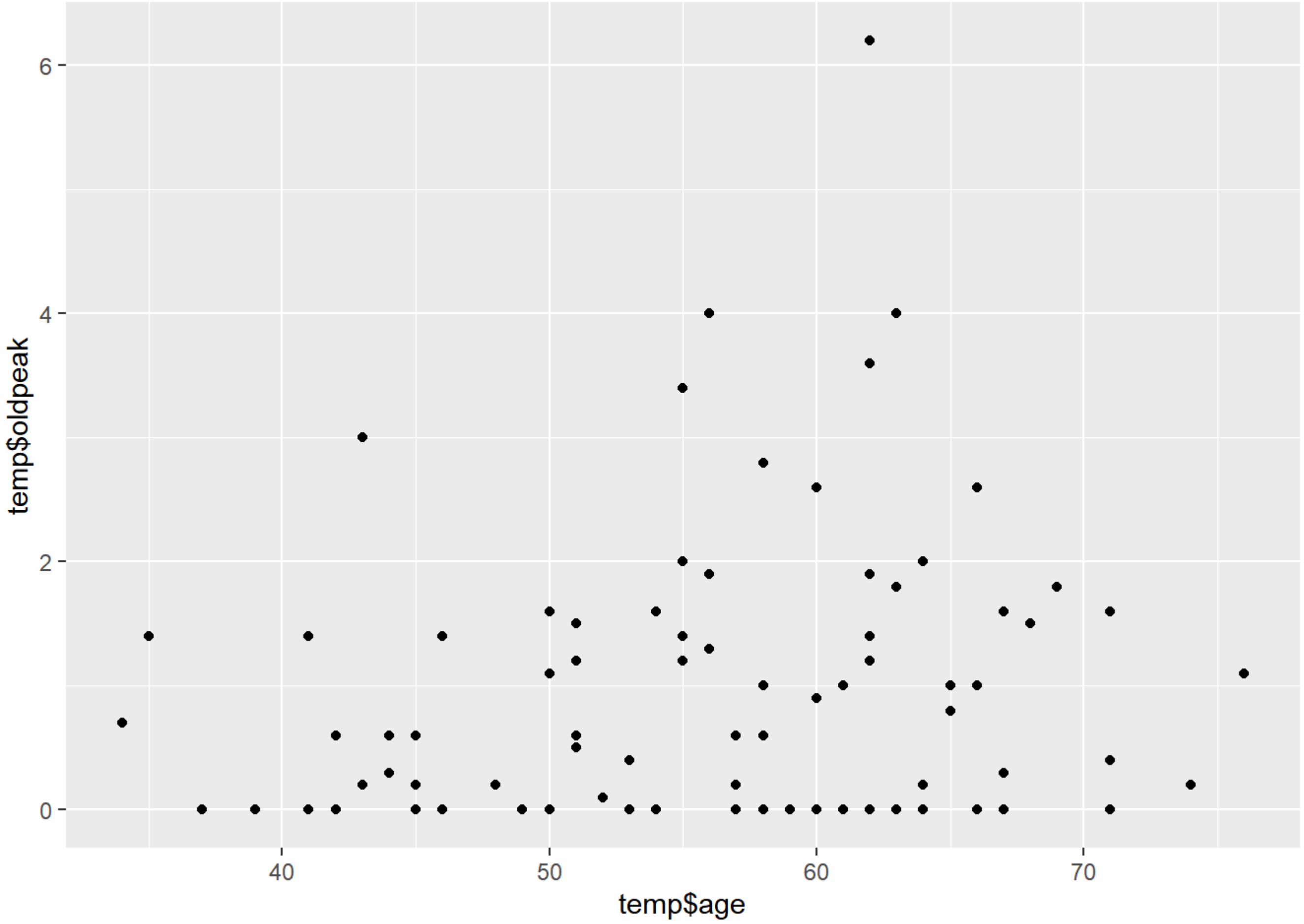
##With ggplot2, you can get a similar result using the ggplot() function

```
library(ggplot2)
```

```
ggplot(temp, aes(x = temp$age, y = temp$oldpeak)) + geom_point()
```

```
## Warning: Use of `temp$age` is discouraged. Use `age` instead.
```

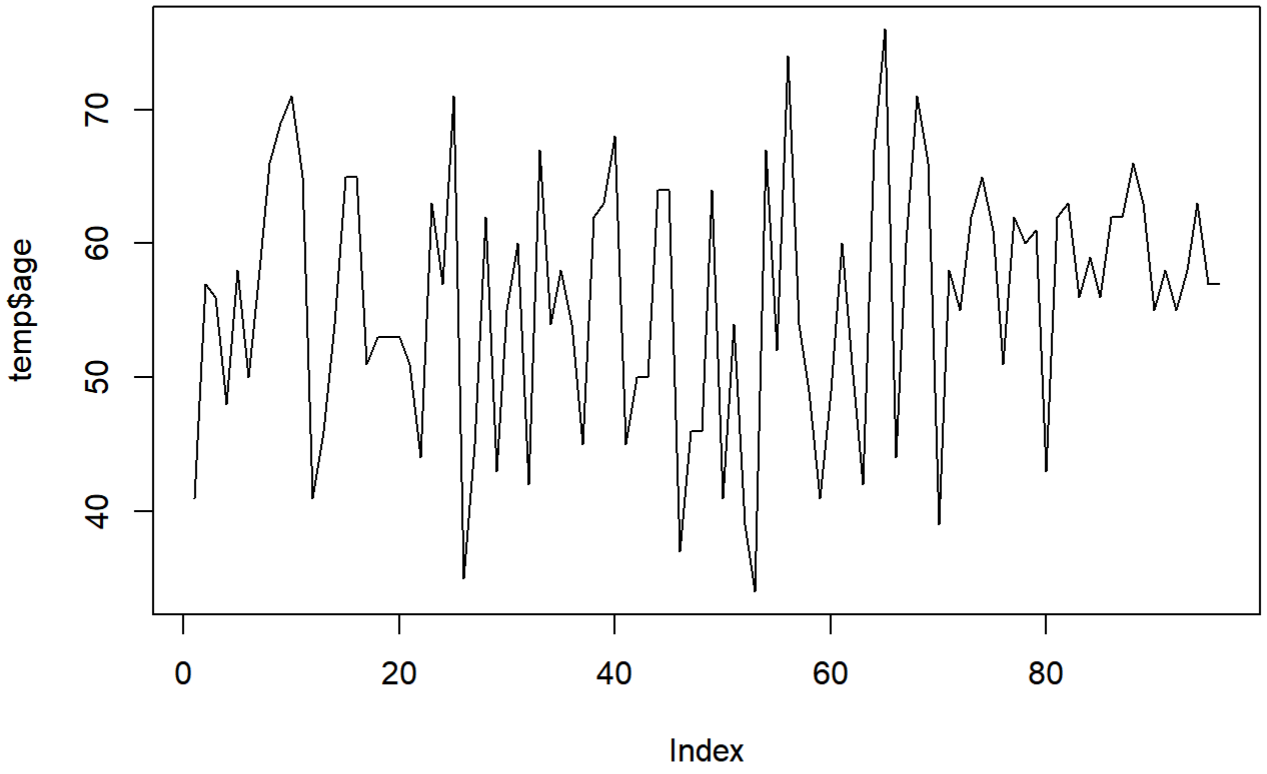
```
## Warning: Use of `temp$oldpeak` is discouraged. Use `oldpeak` instead.
```



##

Creating a Line Graph

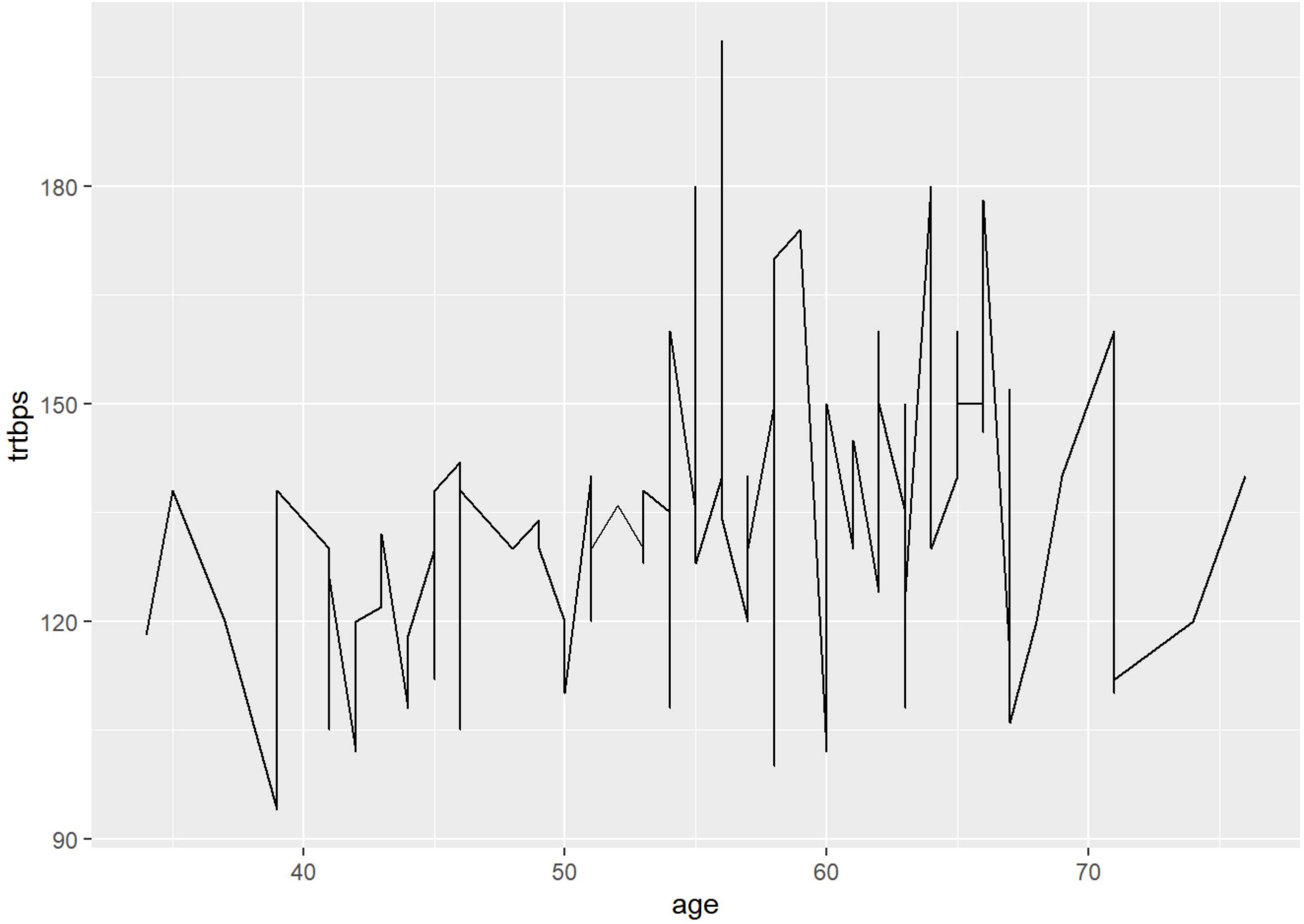
```
plot(temp$age, type = 'l')
```



##

With ggplot2

```
ggplot(temp, aes(x = age, y = trtbps)) + geom_line()
```



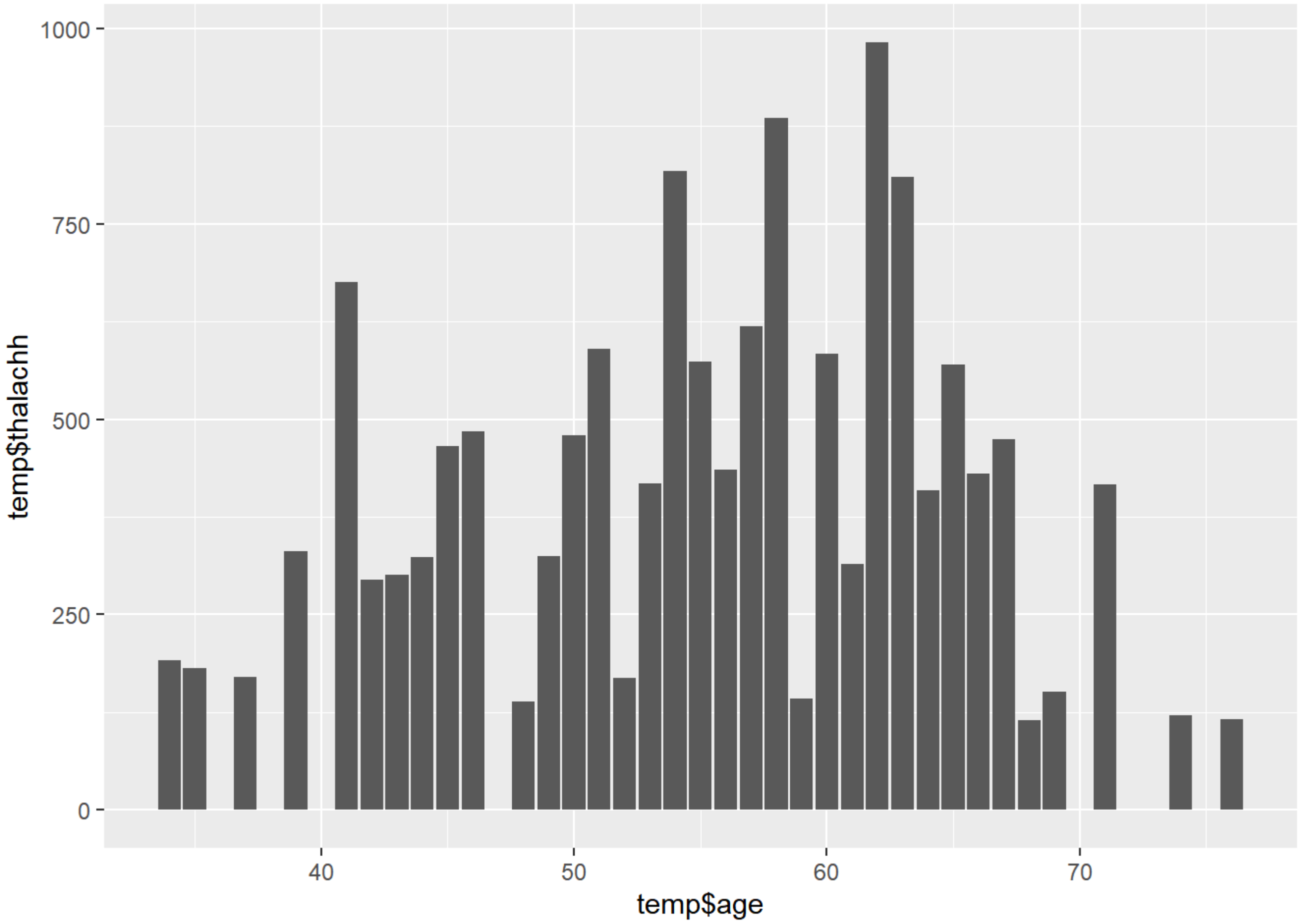
##

Creating a Bar Graph

```
ggplot(temp, aes(x = temp$age, y = temp$thalachh)) + geom_col()
```

```
## Warning: Use of `temp$age` is discouraged. Use `age` instead.
```

```
## Warning: Use of `temp$thalachh` is discouraged. Use `thalachh` instead.
```

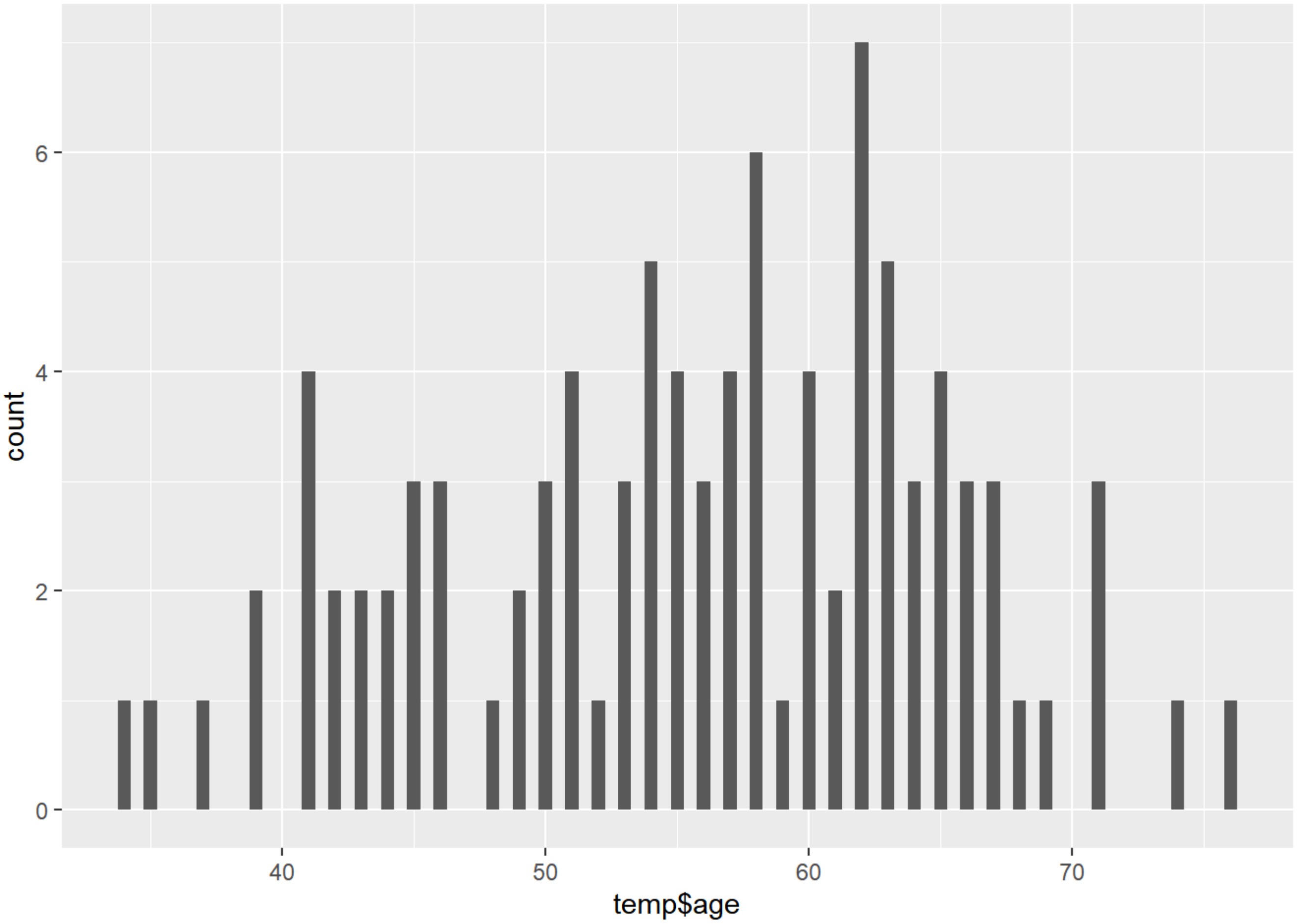


##

Creating a Histogram

```
ggplot(temp, aes(x = temp$age)) + geom_histogram(binwidth = .5)
```

```
## Warning: Use of `temp$age` is discouraged. Use `age` instead.
```



##

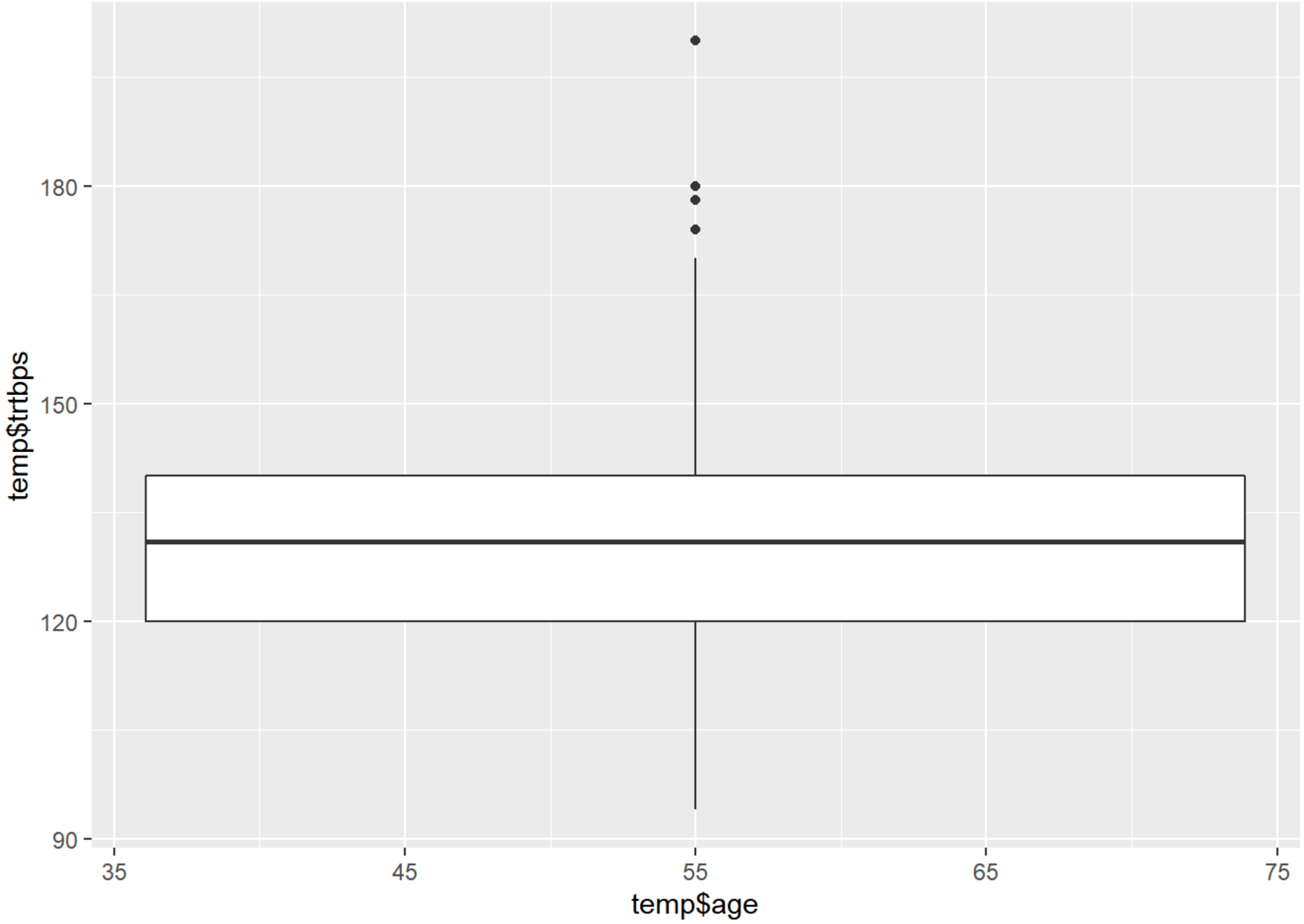
Creating a Box Plot

```
ggplot(temp, aes(x = temp$age, y = temp$trtbps)) + geom_boxplot()
```

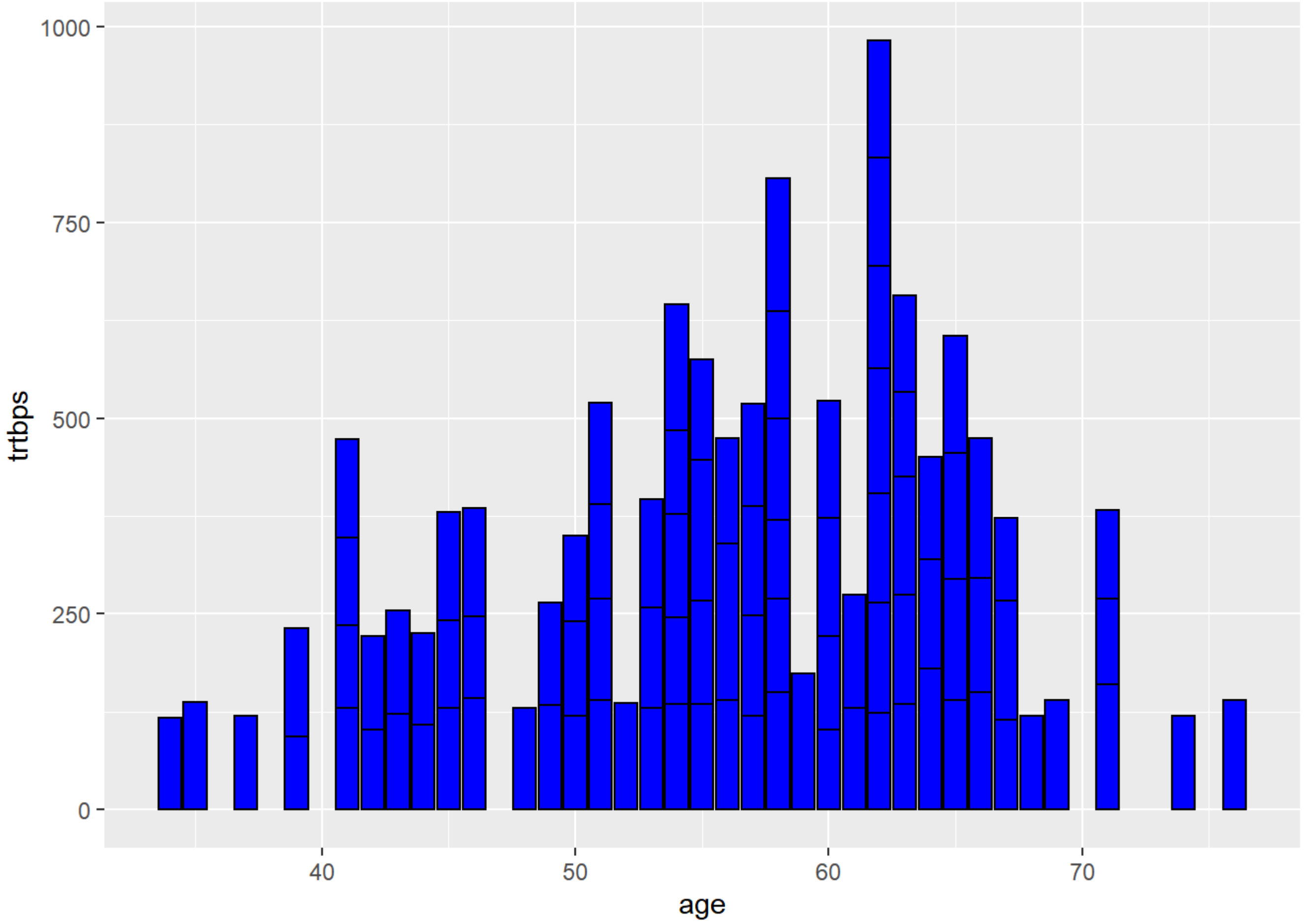
```
## Warning: Use of `temp$age` is discouraged. Use `age` instead.
```

```
## Warning: Use of `temp$trtbps` is discouraged. Use `trtbps` instead.
```

```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```



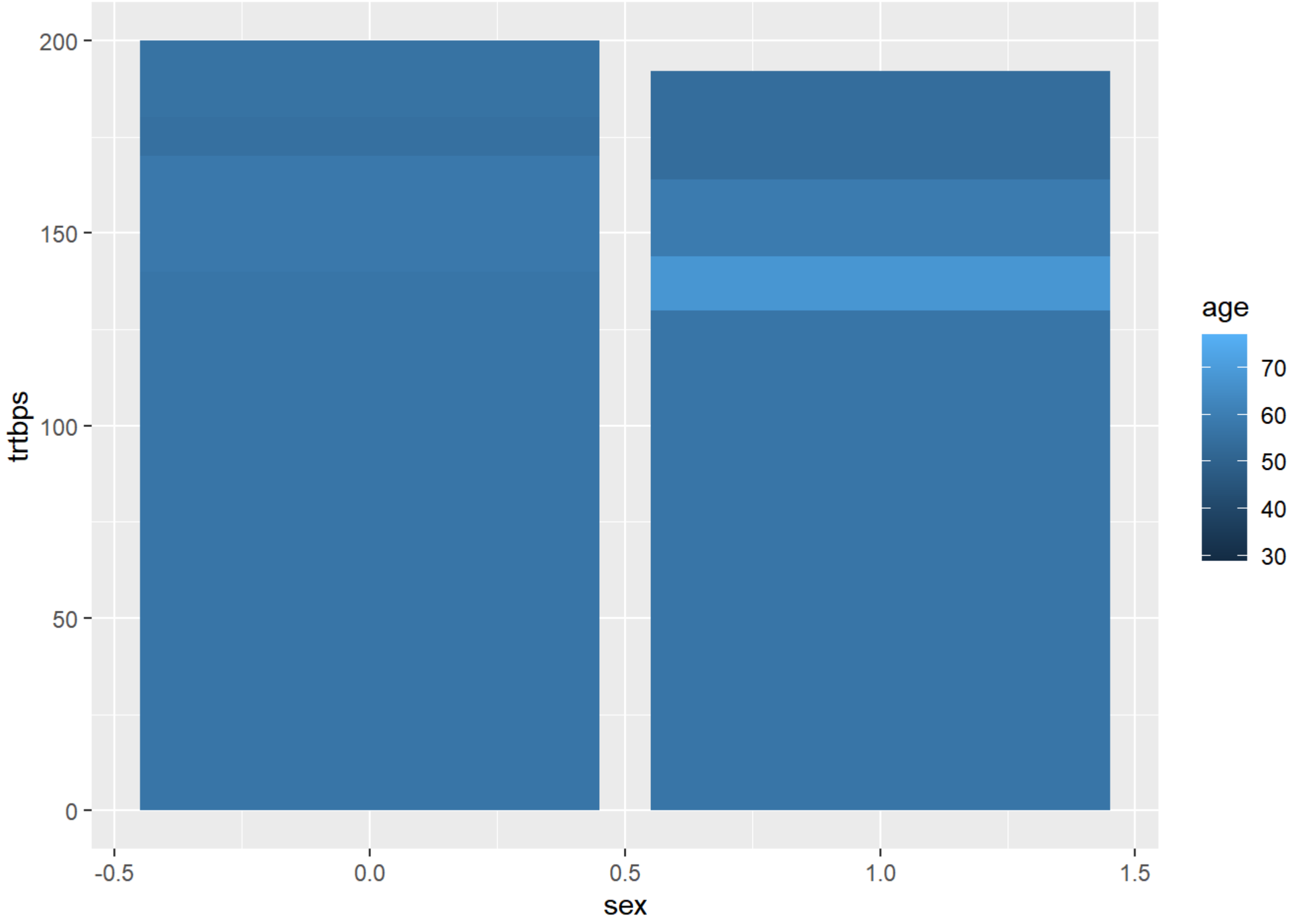
```
ggplot(temp, aes(x = age, y = trtbps)) + geom_col(fill = "blue", colour = "black")
```



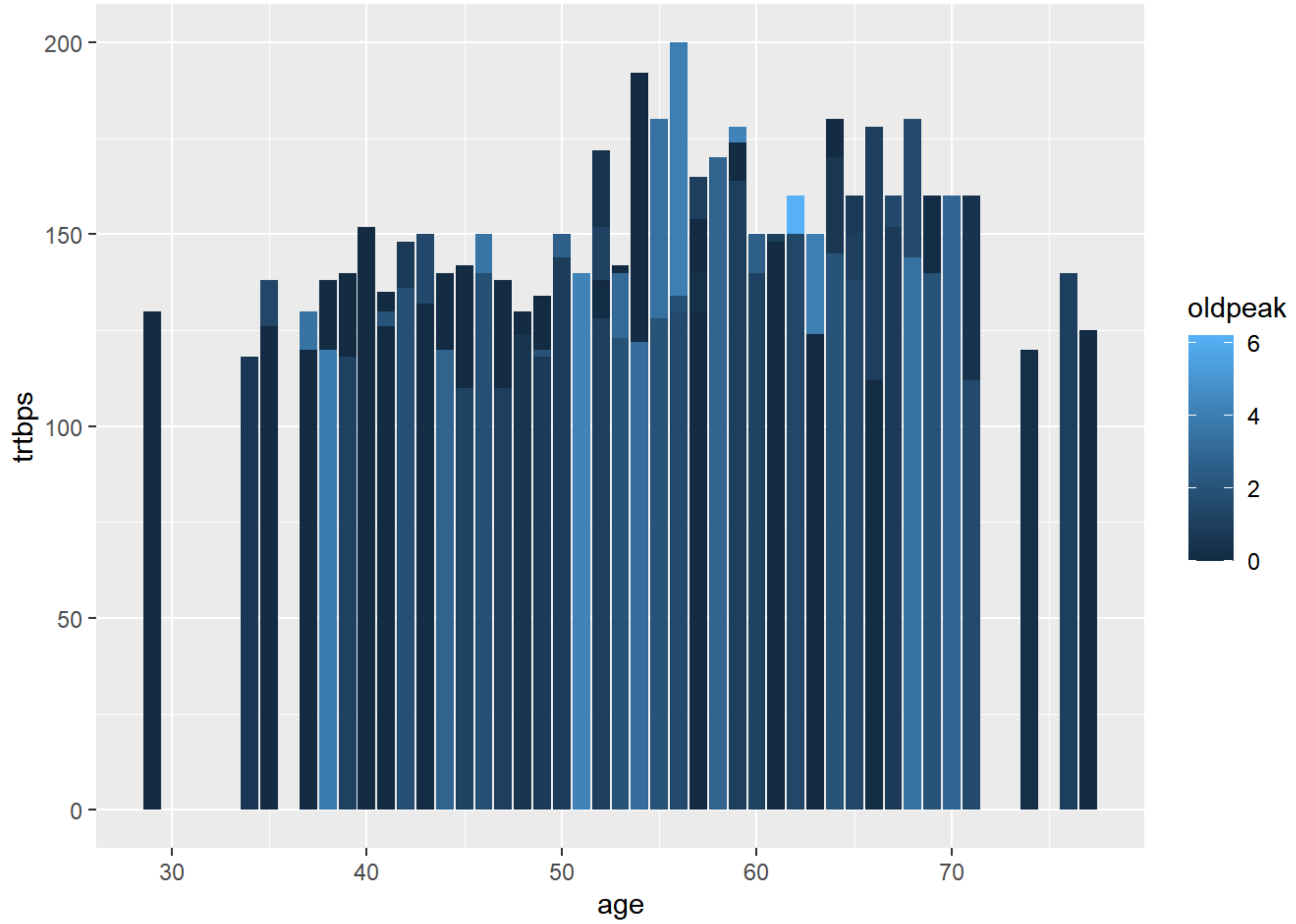
##

Grouping Bars Together

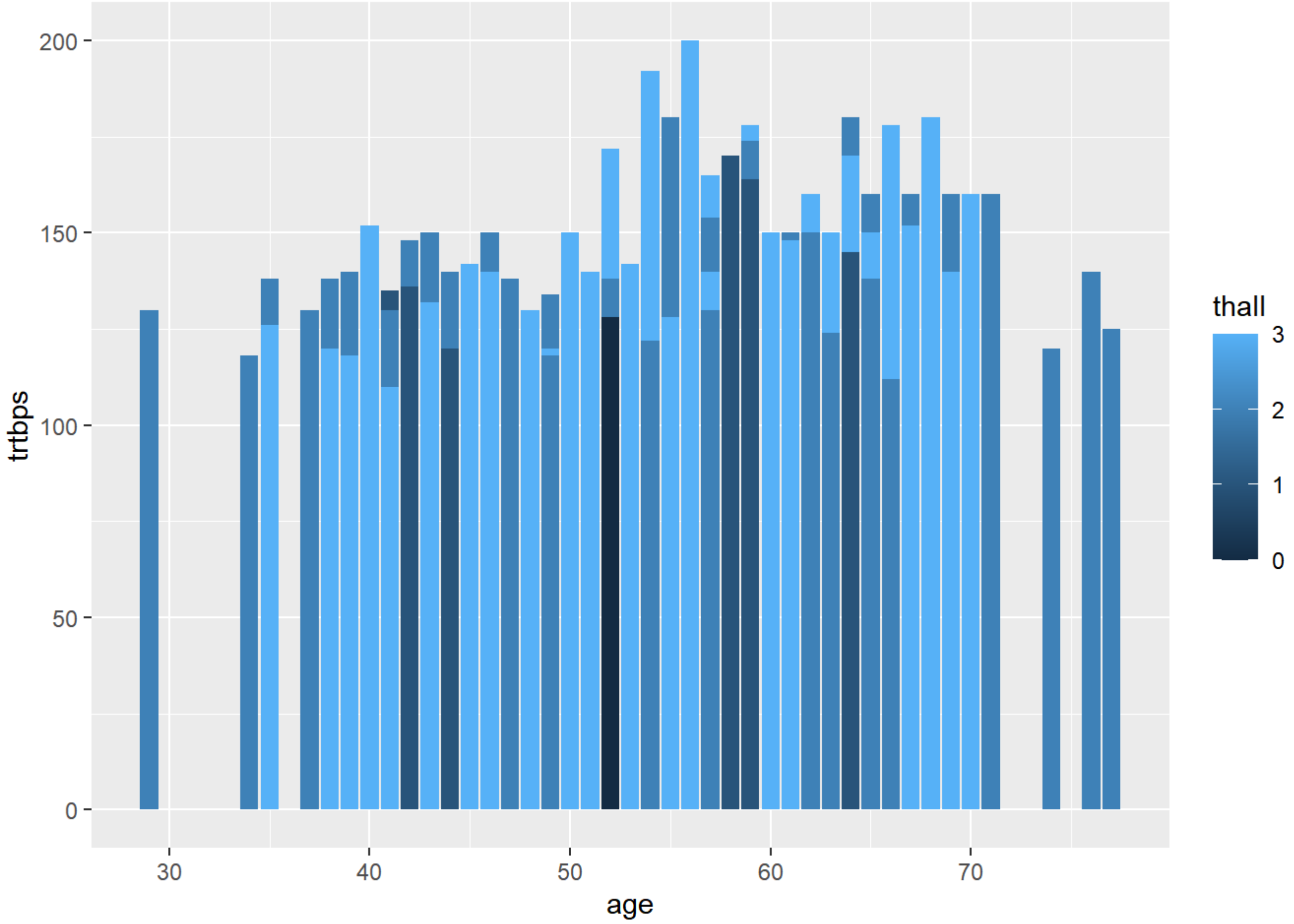
```
ggplot(df, aes(x = sex, y = trtbps, fill = age)) + geom_col(position = "dodge")
```



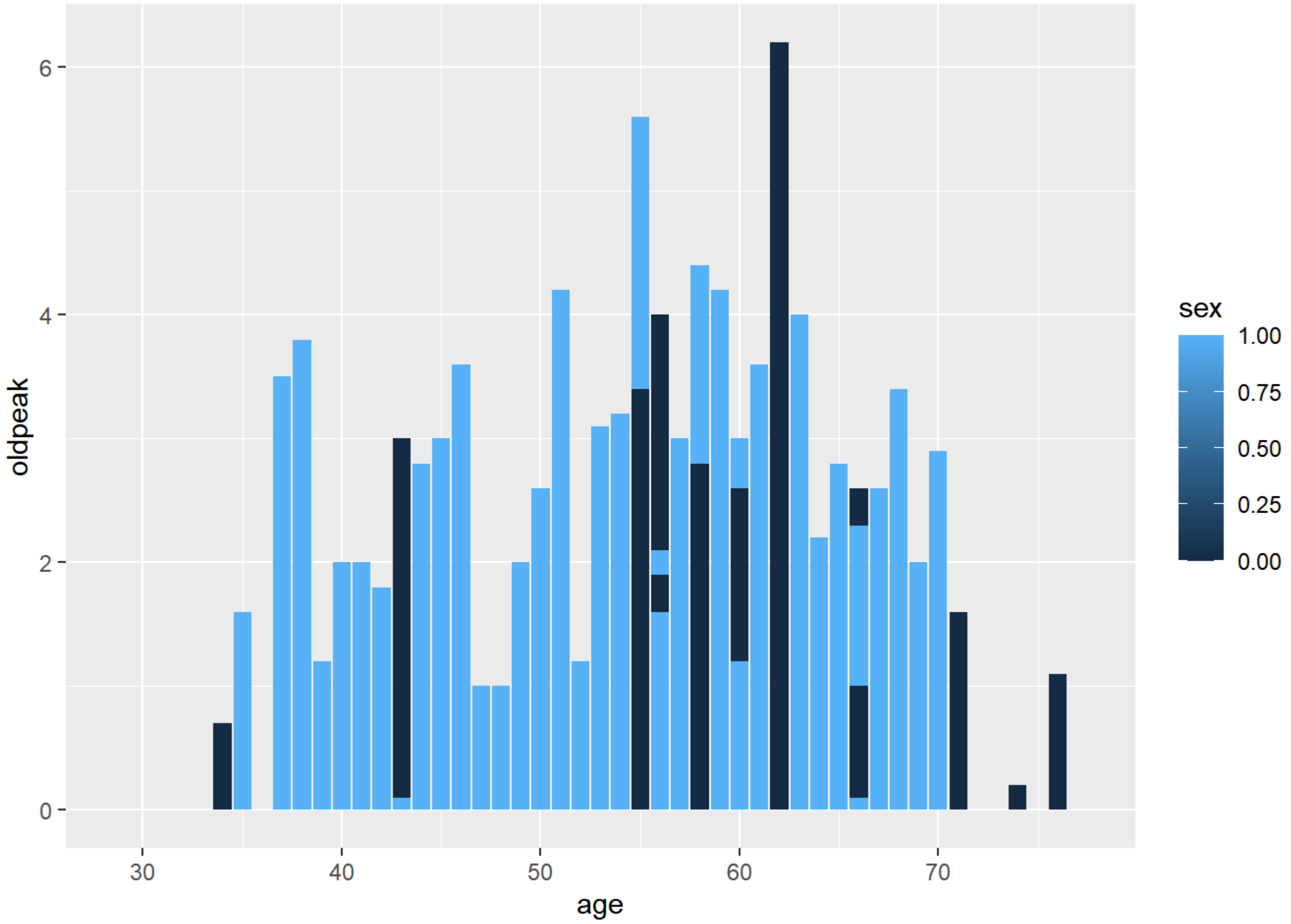
```
ggplot(df, aes(x = age, y = trtbps, fill = oldpeak)) + geom_col(position = "dodge")
```



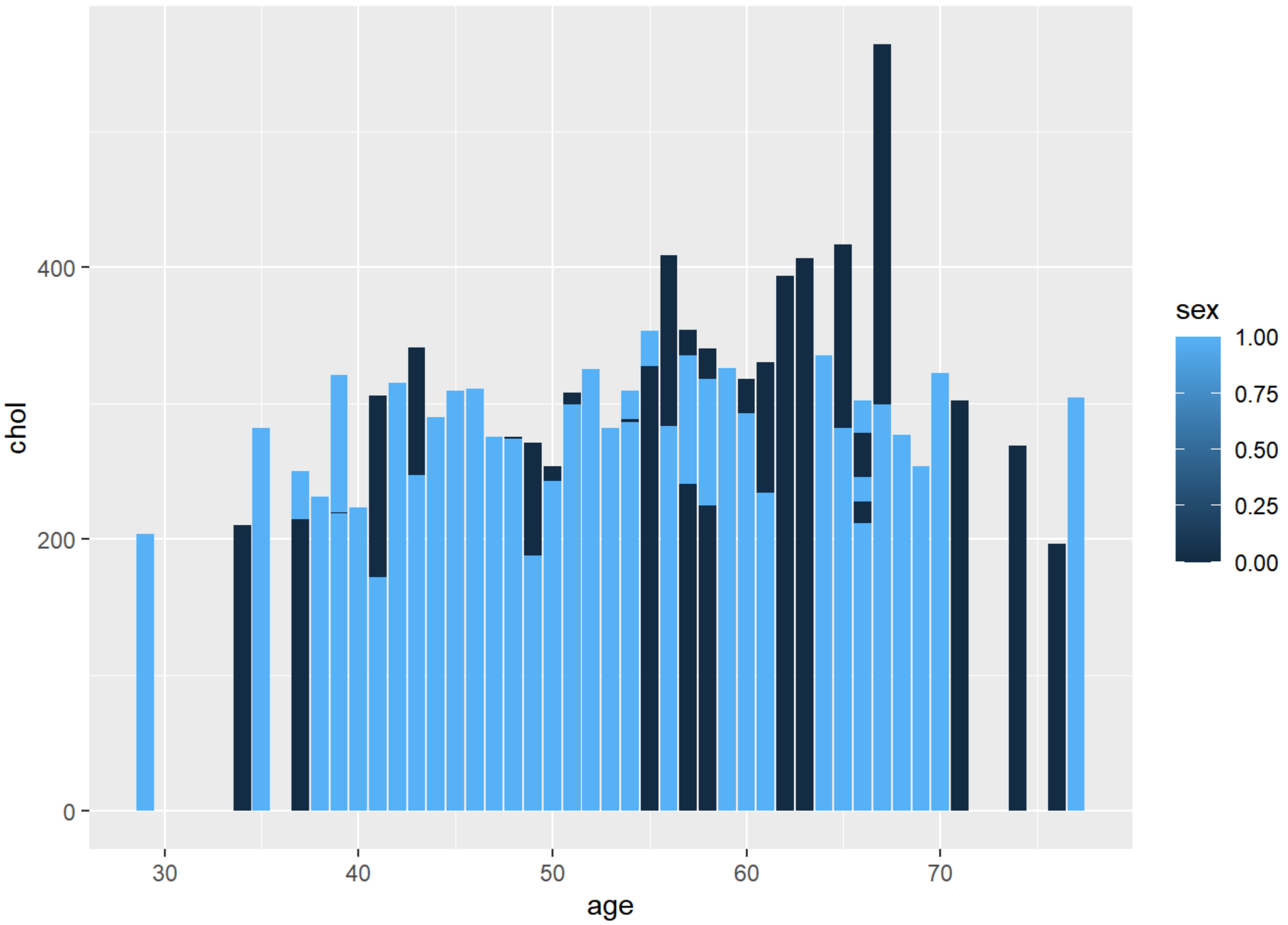
```
ggplot(df, aes(x = age, y = trtbps, fill = thall)) + geom_col(position = "dodge")
```



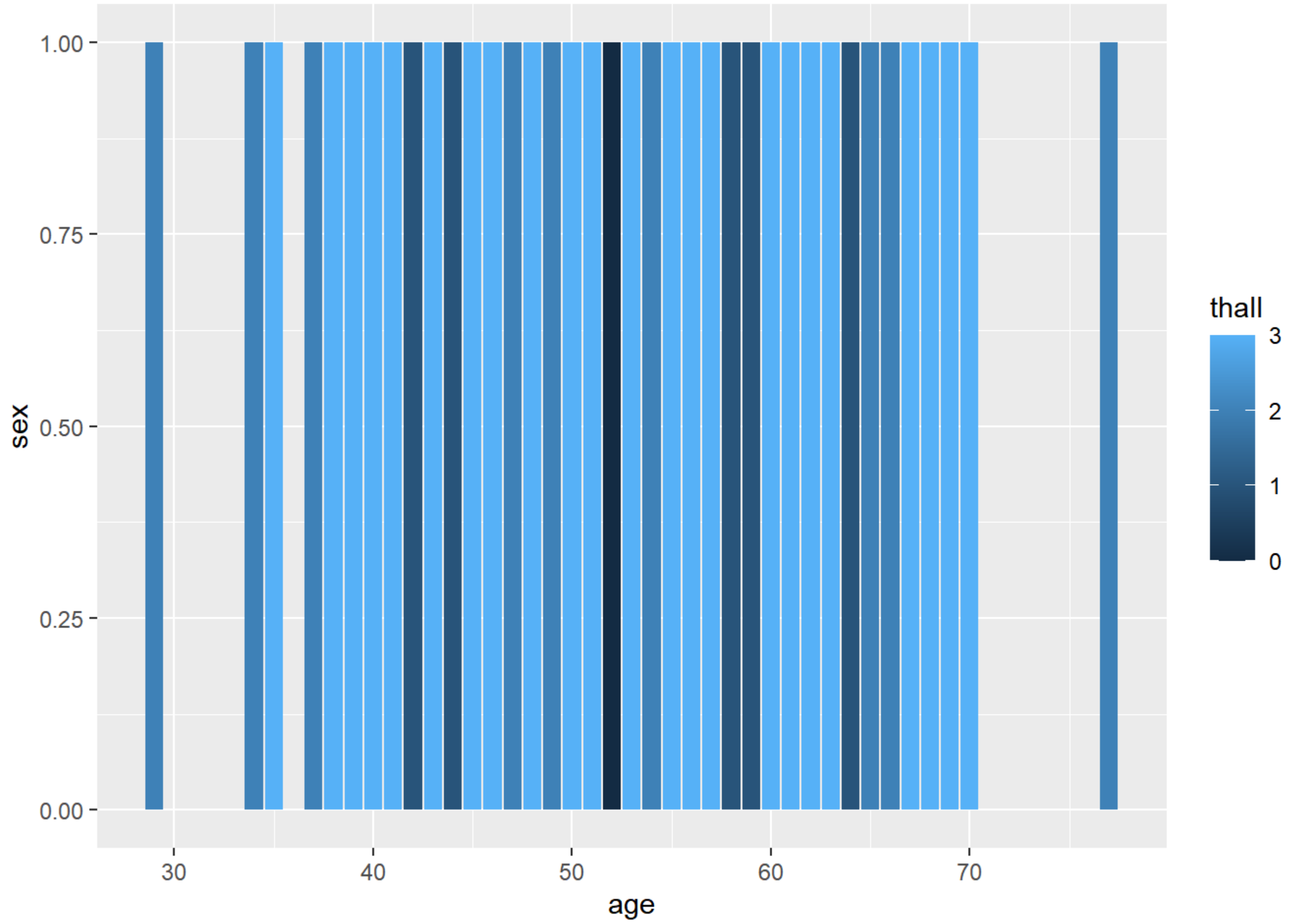
```
ggplot(df, aes(x = age, y = oldpeak, fill = sex)) + geom_col(position = "dodge")
```



```
ggplot(df, aes(x = age, y = chol, fill = sex)) + geom_col(position = "dodge")
```



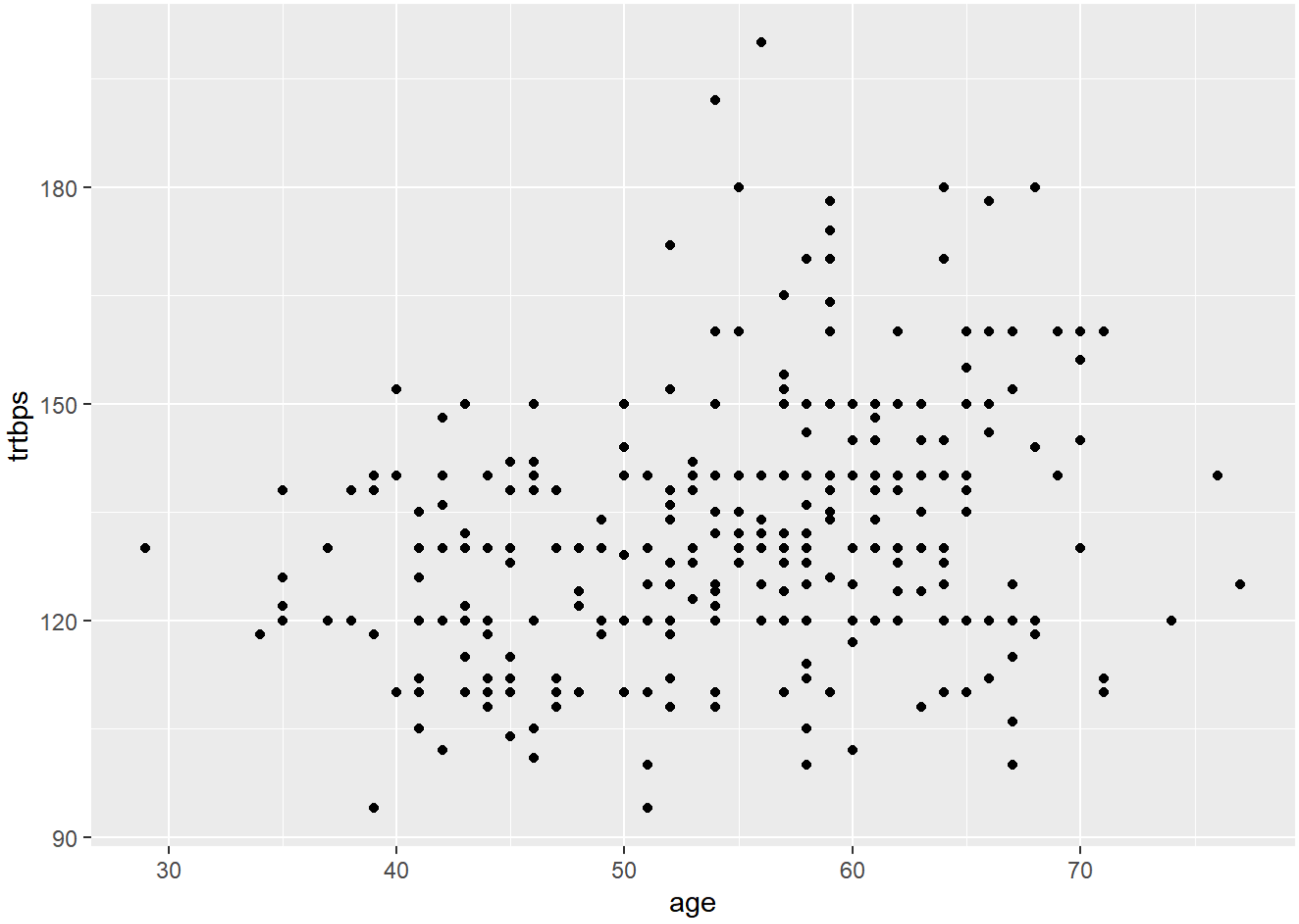
```
ggplot(df, aes(x = age, y = sex, fill = thall)) + geom_col(position = "dodge")
```



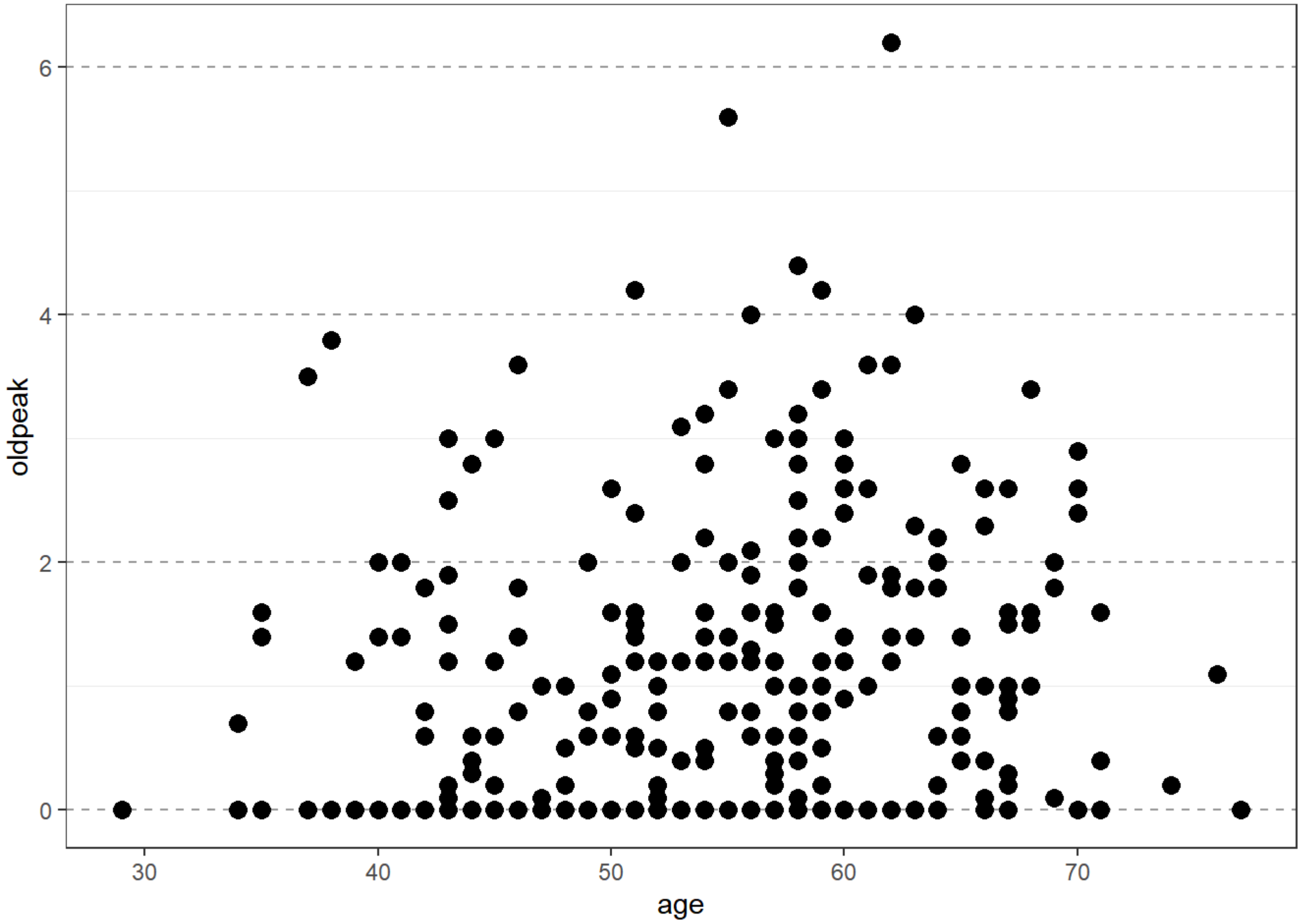
```
##
```

Making a Cleveland Dot Plot

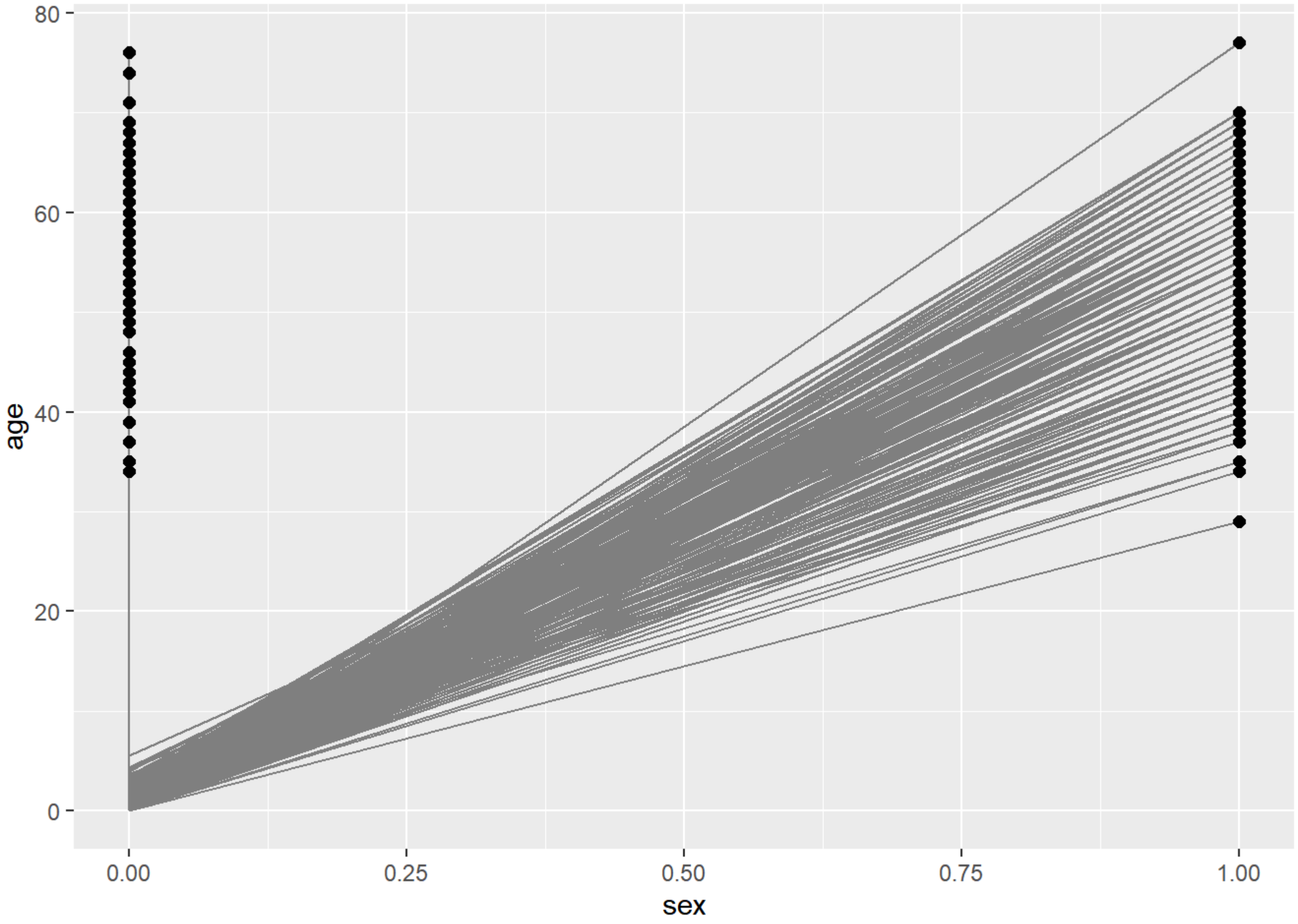
```
ggplot(df, aes(x = age, y = trtbps)) + geom_point()
```



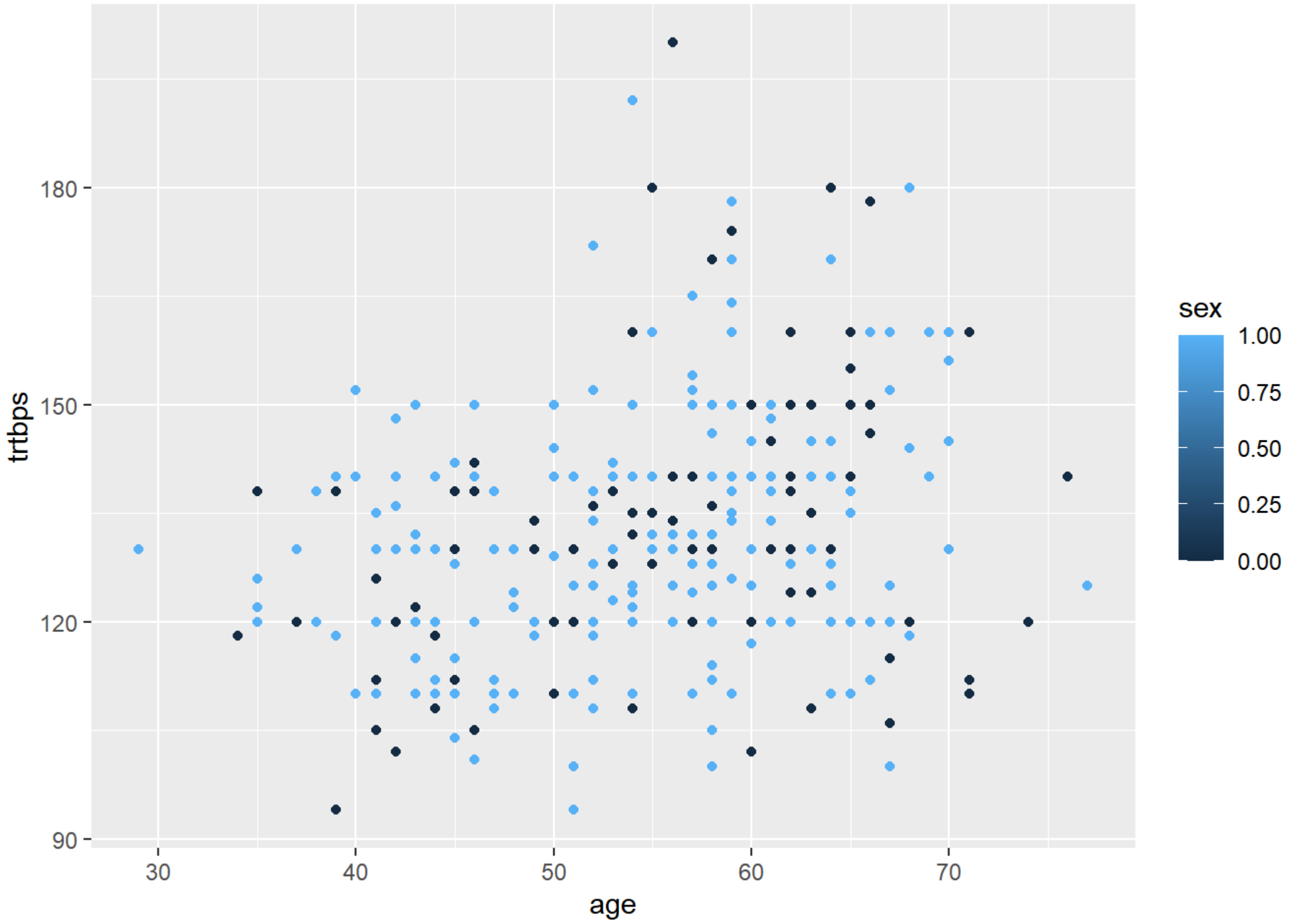
```
ggplot(df, aes(x = age, y = oldpeak)) +
 geom_point(size = 3) + # Use a larger dot
 theme_bw() +
 theme(
 panel.grid.major.x = element_blank(),
 panel.grid.minor.x = element_blank(),
 panel.grid.major.y = element_line(colour = "grey60", linetype = "dashed")
 )
```



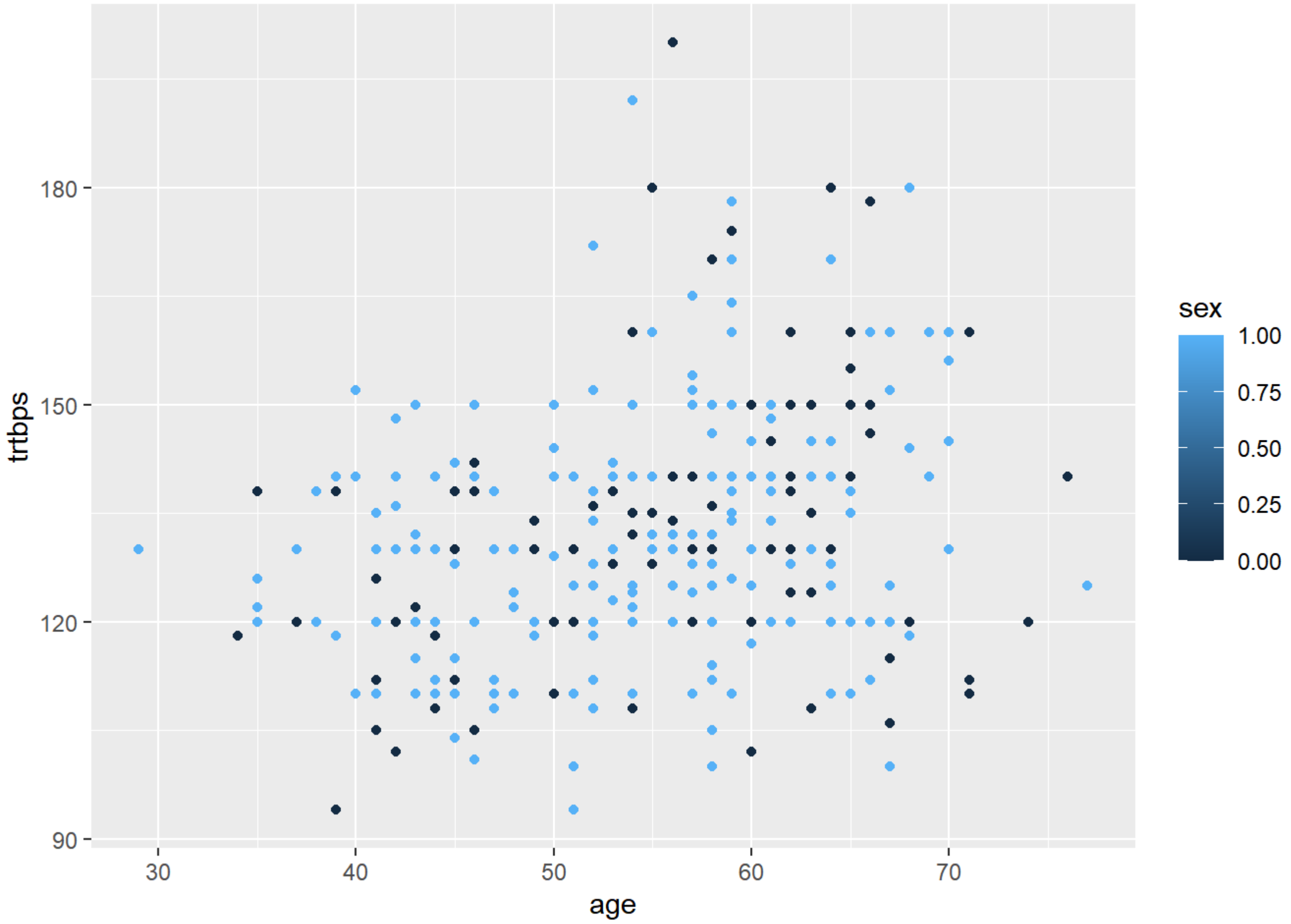
```
ggplot(df, aes(x = sex, y = age)) +
geom_segment(aes(yend = oldpeak), xend = 0, colour = "grey50") +
  geom_point(size = 2)
```



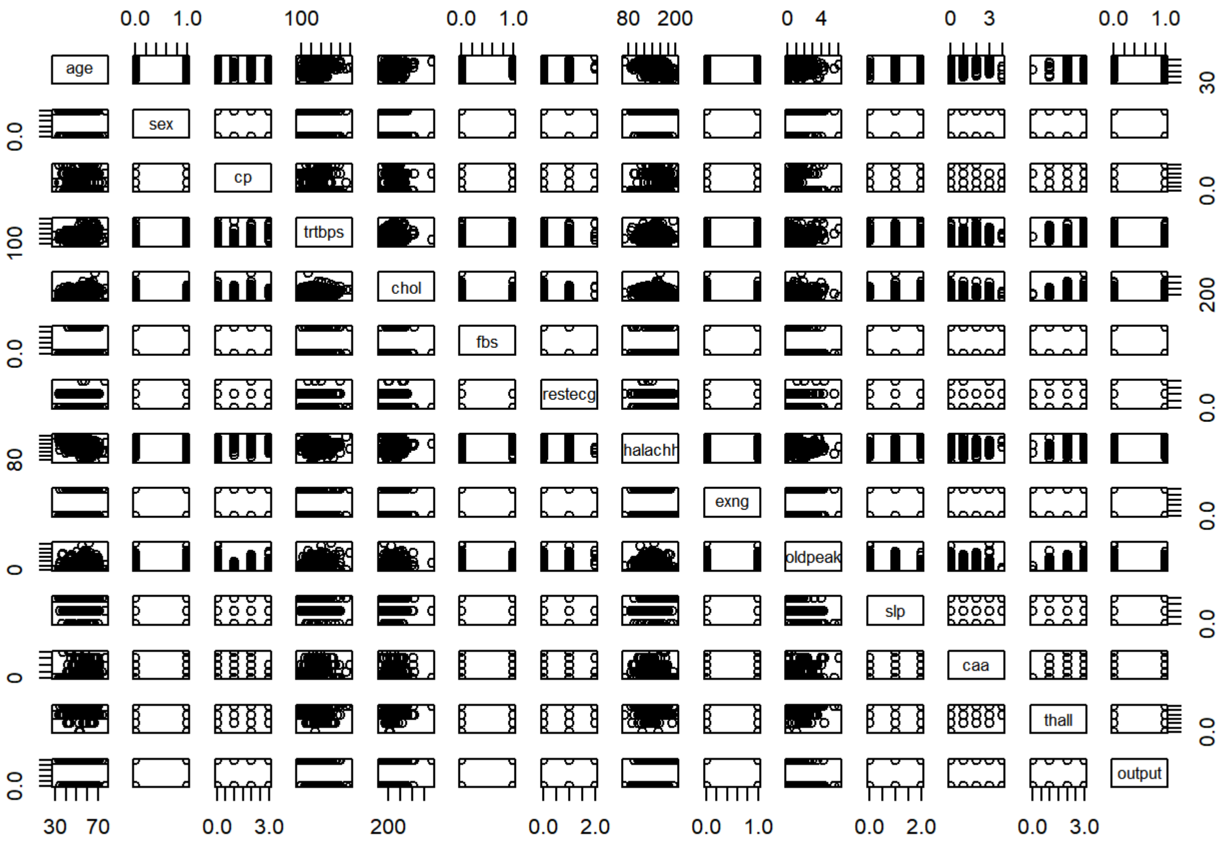
```
ggplot(df,aes(x=age, y=trtbps, colour= sex)) + geom_point()
```



```
ggplot(df,aes(x=age, y=trtbps, colour = sex)) + geom_point()
```



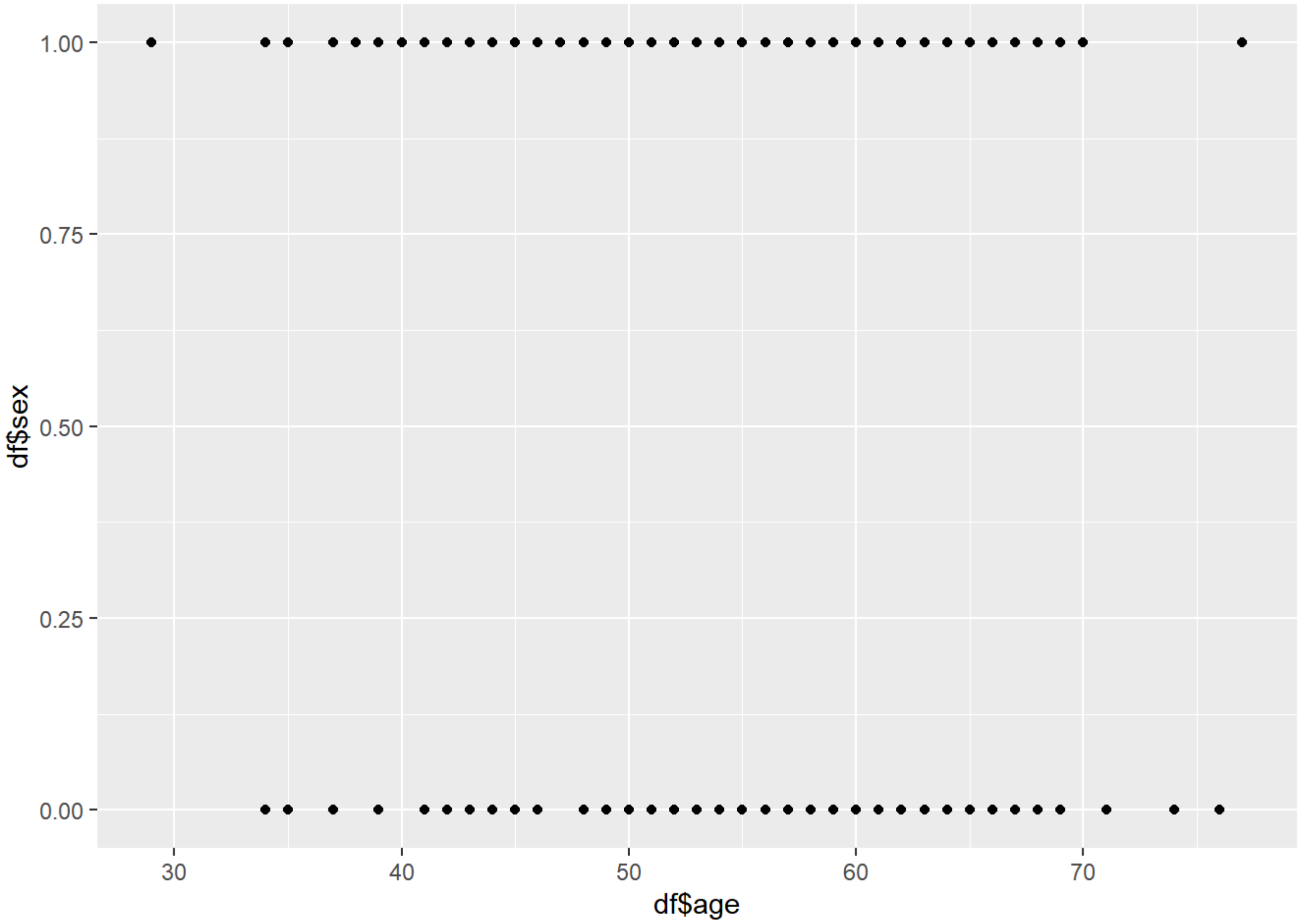
```
plot(df)
```



```
ggplot(df, aes(x = df$age, y = df$sex)) +
 geom_point(size = 1.5)
```

```
## Warning: Use of `df$age` is discouraged. Use `age` instead.
```

```
## Warning: Use of `df$sex` is discouraged. Use `sex` instead.
```



```
tt <- df %>%
 filter(df$sex == 0,) %>%
 select(age, trtbps, thall, oldpeak, cp)
head(tt)
```

```
##   age trtbps thall oldpeak cp
## 1  41    130     2     1.4  1
## 2  57    120     2     0.6  0
## 3  56    140     2     1.3  1
## 4  48    130     2     0.2  2
## 5  58    150     2     1.0  3
## 6  50    120     2     1.6  2
```

```
tt <- df %>%
 filter(df$sex == 1) %>%
 select(thalachh, chol, thall, oldpeak, cp)
head(tt)
```

```
##   thalachh chol thall oldpeak cp
## 1      150  233     1     2.3  3
## 2      187  250     2     3.5  2
## 3      178  236     2     0.8  1
## 4      148  192     1     0.4  0
## 5      173  263     3     0.0  1
## 6      162  199     3     0.5  2
```

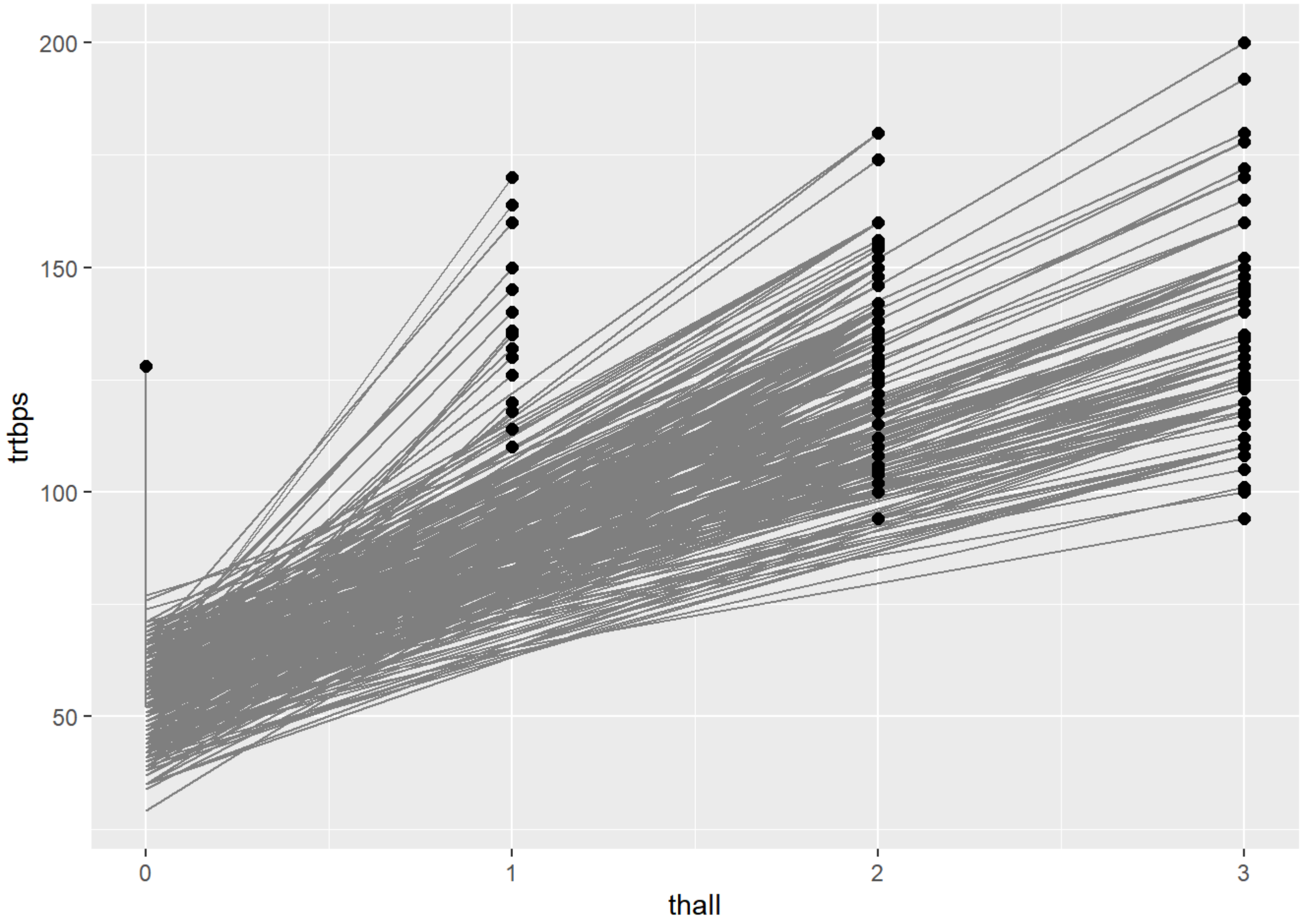
```
tt = df %>% filter(df$age == 50) %>% select(age, trtbps, thall, oldpeak, cp)
head(tt)
```

```
##   age trtbps thall oldpeak cp
## 1  50    120     2     1.6  2
## 2  50    129     2     0.0  2
## 3  50    120     2     1.1  1
## 4  50    110     2     0.0  0
## 5  50    150     3     2.6  0
## 6  50    140     3     0.6  2
```

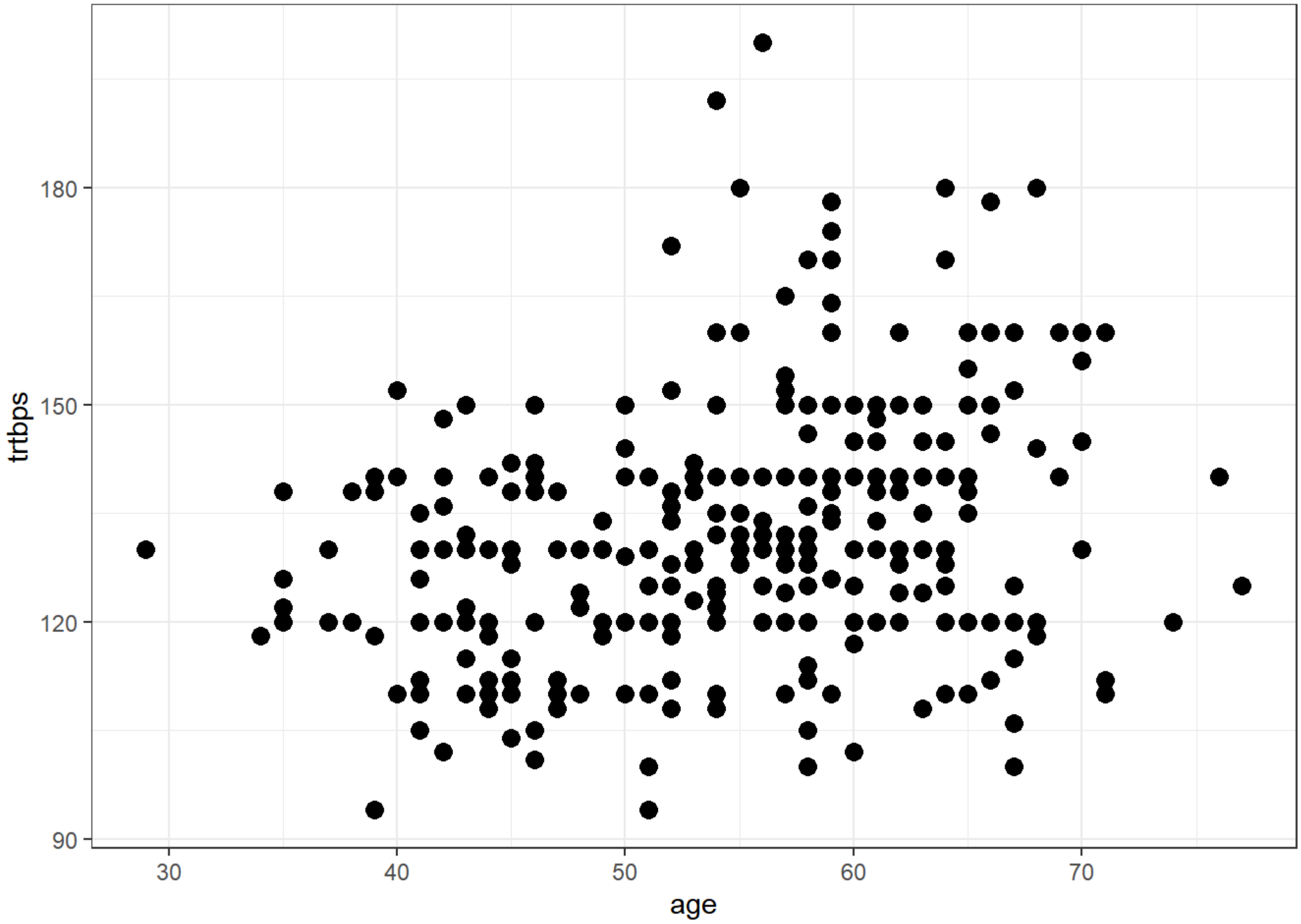
```
tt = df %>% filter(df$age == 60) %>% select(age, sex, trtbps, thall, oldpeak, cp)
head(tt)
```

```
##   age sex trtbps thall oldpeak cp
## 1  60   0    102     2     0.0  2
## 2  60   0    120     2     0.0  2
## 3  60   0    150     2     0.9  3
## 4  60   1    130     3     2.4  0
## 5  60   1    117     3     1.4  0
## 6  60   1    130     3     1.4  0
```

```
ggplot(df, aes(x = thall, y = trtbps)) +
geom_segment(aes(yend = age), xend = 0, colour = "grey50") +
  geom_point(size = 2)
```



```
ggplot(df, aes(x = age, y = trtbps)) +
 geom_point(size = 3) + # Use a larger dot
 theme_bw() +
 theme()
```



```
tt = df %>% group_by(df$sex) %>% summary()
tt
```

```
##       age             sex               cp            trtbps     
##  Min.   :29.00   Min.   :0.0000   Min.   :0.000   Min.   : 94.0  
##  1st Qu.:47.50   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:120.0  
##  Median :55.00   Median :1.0000   Median :1.000   Median :130.0  
##  Mean   :54.37   Mean   :0.6832   Mean   :0.967   Mean   :131.6  
##  3rd Qu.:61.00   3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:140.0  
##  Max.   :77.00   Max.   :1.0000   Max.   :3.000   Max.   :200.0  
##       chol            fbs            restecg          thalachh    
##  Min.   :126.0   Min.   :0.0000   Min.   :0.0000   Min.   : 71.0  
##  1st Qu.:211.0   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:133.5  
##  Median :240.0   Median :0.0000   Median :1.0000   Median :153.0  
##  Mean   :246.3   Mean   :0.1485   Mean   :0.5281   Mean   :149.6  
##  3rd Qu.:274.5   3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:166.0  
##  Max.   :564.0   Max.   :1.0000   Max.   :2.0000   Max.   :202.0  
##       exng           oldpeak          slp             caa        
##  Min.   :0.0000   Min.   :0.00   Min.   :0.000   Min.   :0.0000  
##  1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:1.000   1st Qu.:0.0000  
##  Median :0.0000   Median :0.80   Median :1.000   Median :0.0000  
##  Mean   :0.3267   Mean   :1.04   Mean   :1.399   Mean   :0.7294  
##  3rd Qu.:1.0000   3rd Qu.:1.60   3rd Qu.:2.000   3rd Qu.:1.0000  
##  Max.   :1.0000   Max.   :6.20   Max.   :2.000   Max.   :4.0000  
##      thall           output           df$sex      
##  Min.   :0.000   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:2.000   1st Qu.:0.0000   1st Qu.:0.0000  
##  Median :2.000   Median :1.0000   Median :1.0000  
##  Mean   :2.314   Mean   :0.5446   Mean   :0.6832  
##  3rd Qu.:3.000   3rd Qu.:1.0000   3rd Qu.:1.0000  
##  Max.   :3.000   Max.   :1.0000   Max.   :1.0000
```