Data Exploration with the Boston Housing Data

Karen Mazidi

Load the package, installing if necessary

```
if (!require("MASS")){
  install.packages("MASS")
}
```

```
## Loading required package: MASS
```

```
#library(MASS)  # not needed with above code
```

Load Boston

The Boston housing data set is a collection of data from Boston neighborhoods in the late 1970s. There are 506 rows representing different neighborhoods, and 14 variables:

- crim - per capita crime by town
- zn - propostion of residential land zoned for lots over 25K sq ft
- indus - proportion of non-retail business acres per town
- chas - =1 if tract bounds the Charles river; 0 overwise
- nox - nitrous oxide concentration in parts per 10 million
- rm - average number of rooms per dwelling
- age - proportion of owner-occupied units built prior to 1940
- rad - index of accessibility to radial highways
- tax - full-value property tax rate per $10K
- ptratio - pupil-teacher ratio by town
- black - proportion of blacks by town; seriously?
- lstat - lower status of the populatio as a percent
- medv - median value of owner-occupied homes in 1000s of dollars

The str() function tells you about the structure of the data set.

```
data(Boston)
str(Boston)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

Built-in R Functions

First we attach the data. The advantage of attaching the data is that we can type **mean(lstat)** instead of having to specify the data frame: **mean(Boston$lstat) * *.Thedataframe$column format is how columns are accessed. However, if we attach the data, R can find the columns without having to specify the data frame. The disadvantage to attaching data is that you may have column names from different data frames that are the same. This can be confusing.

```r
attach(Boston)
range(medv)
```

```
## [1]  5 50
```

```r
median(tax)
```

```
## [1] 330
```

```r
mean(lstat)
```

```
## [1] 12.65306
```

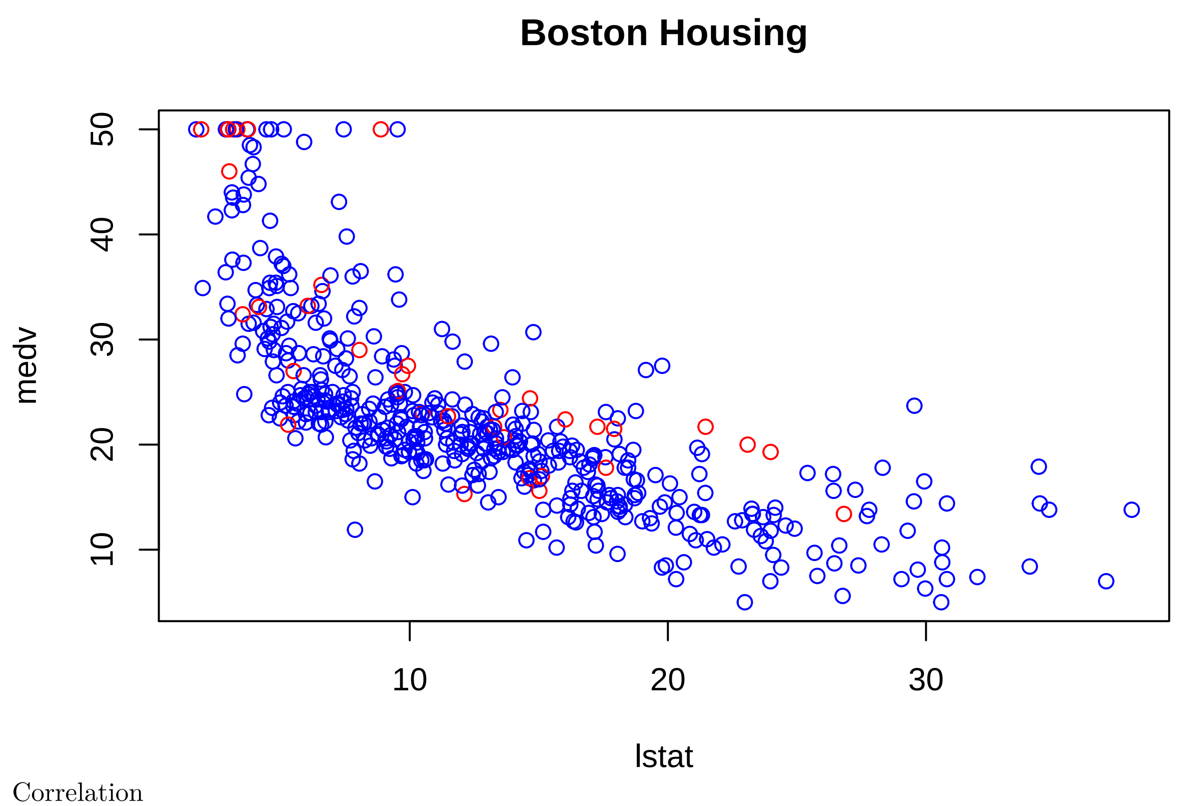
```r
summary(age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.90   45.02   77.50   68.57   94.08  100.00
```

Plot

The plot function is powerful and versatile. Read more by typing ?plot at the console. Here we plot median home value as a function of the percent of lower economic status persons in the neighborhood. The color of the points will indicate whether or not the neighborhood is close to the Charles River. The unclass() functions converts the factors to integers so they can index the color choices blue or red. We also added a main heading. We did not specify x or y axis headings so R just used the variable names.

```r
plot(medv~lstat, col=c("blue","red")[unclass(chas)+1], main="Boston Housing")
```



###

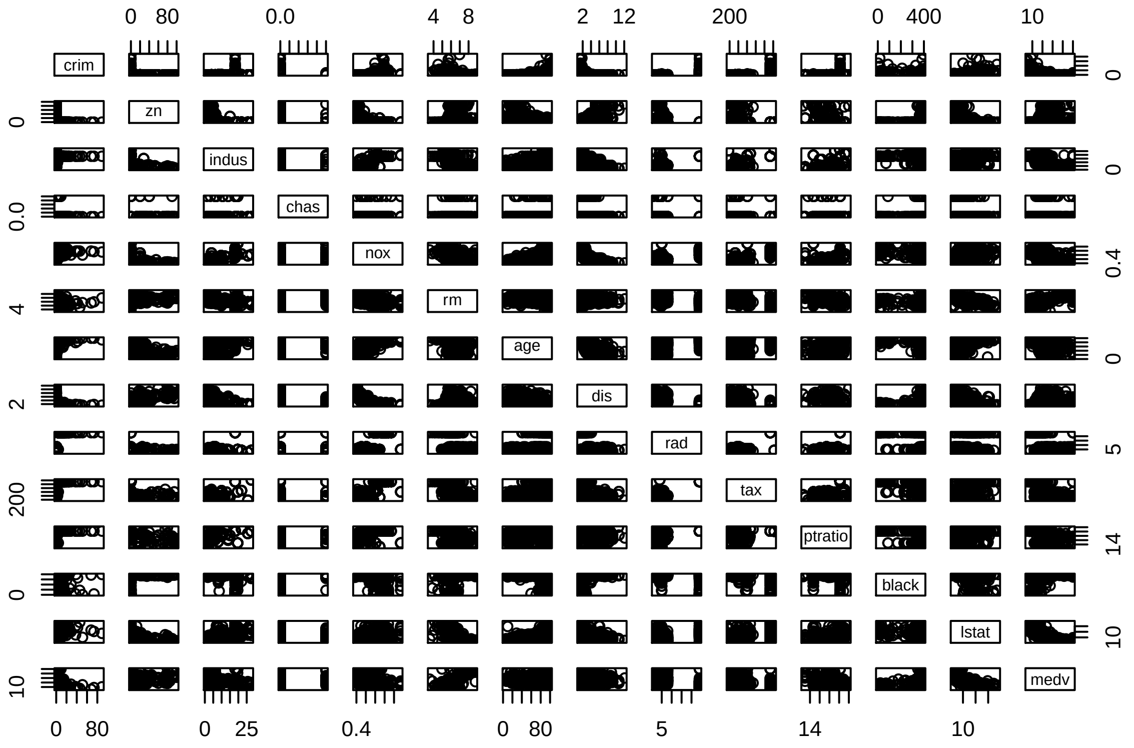
Correlation

We can check for correlation by creating a table with the cor() function or visually look for correlations by plotting pairs().

```r
cor(Boston)
```

```
##                crim          zn       indus         chas         nox
## crim     1.00000000 -0.20046922  0.40658341 -0.055891582  0.42097171
## zn      -0.20046922  1.00000000 -0.53382819 -0.042696719 -0.51660371
## indus    0.40658341 -0.53382819  1.00000000  0.062938027  0.76365145
## chas    -0.05589158 -0.04269672  0.06293803  1.000000000  0.09120281
## nox      0.42097171 -0.51660371  0.76365145  0.091202807  1.00000000
## rm      -0.21924670  0.31199059 -0.39167585  0.091251225 -0.30218819
## age      0.35273425 -0.56953734  0.64477851  0.086517774  0.73147010
## dis     -0.37967009  0.66440822 -0.70802699 -0.099175780 -0.76923011
## rad      0.62550515 -0.31194783  0.59512927 -0.007368241  0.61144056
## tax      0.58276431 -0.31456332  0.72076018 -0.035586518  0.66802320
## ptratio  0.28994558 -0.39167855  0.38324756 -0.121515174  0.18893268
## black   -0.38506394  0.17552032 -0.35697654  0.048788485 -0.38005064
## lstat    0.45562148 -0.41299457  0.60379972 -0.053929298  0.59087892
## medv    -0.38830461  0.36044534 -0.48372516  0.175260177 -0.42732077
##                  rm         age         dis          rad         tax    ptratio
## crim    -0.21924670  0.35273425 -0.37967009  0.625505145  0.58276431  0.2899456
## zn       0.31199059 -0.56953734  0.66440822 -0.311947826 -0.31456332 -0.3916785
## indus   -0.39167585  0.64477851 -0.70802699  0.595129275  0.72076018  0.3832476
## chas     0.09125123  0.08651777 -0.09917578 -0.007368241 -0.03558652 -0.1215152
## nox     -0.30218819  0.73147010 -0.76923011  0.611440563  0.66802320  0.1889327
## rm       1.00000000 -0.24026493  0.20524621 -0.209846668 -0.29204783 -0.3555015
## age     -0.24026493  1.00000000 -0.74788054  0.456022452  0.50645559  0.2615150
## dis      0.20524621 -0.74788054  1.00000000 -0.494587930 -0.53443158 -0.2324705
## rad     -0.20984667  0.45602245 -0.49458793  1.000000000  0.91022819  0.4647412
## tax     -0.29204783  0.50645559 -0.53443158  0.910228189  1.00000000  0.4608530
## ptratio -0.35550149  0.26151501 -0.23247054  0.464741179  0.46085304  1.0000000
## black    0.12806864 -0.27353398  0.29151167 -0.444412816 -0.44180801 -0.1773833
## lstat   -0.61380827  0.60233853 -0.49699583  0.488676335  0.54399341  0.3740443
## medv     0.69535995 -0.37695457  0.24992873 -0.381626231 -0.46853593 -0.5077867
##              black      lstat       medv
## crim    -0.38506394  0.4556215 -0.3883046
## zn       0.17552032 -0.4129946  0.3604453
## indus   -0.35697654  0.6037997 -0.4837252
## chas     0.04878848 -0.0539293  0.1752602
## nox     -0.38005064  0.5908789 -0.4273208
## rm       0.12806864 -0.6138083  0.6953599
## age     -0.27353398  0.6023385 -0.3769546
## dis      0.29151167 -0.4969958  0.2499287
## rad     -0.44441282  0.4886763 -0.3816262
## tax     -0.44180801  0.5439934 -0.4685359
## ptratio -0.17738330  0.3740443 -0.5077867
## black    1.00000000 -0.3660869  0.3334608
## lstat   -0.36608690  1.0000000 -0.7376627
## medv     0.33346082 -0.7376627  1.0000000
```

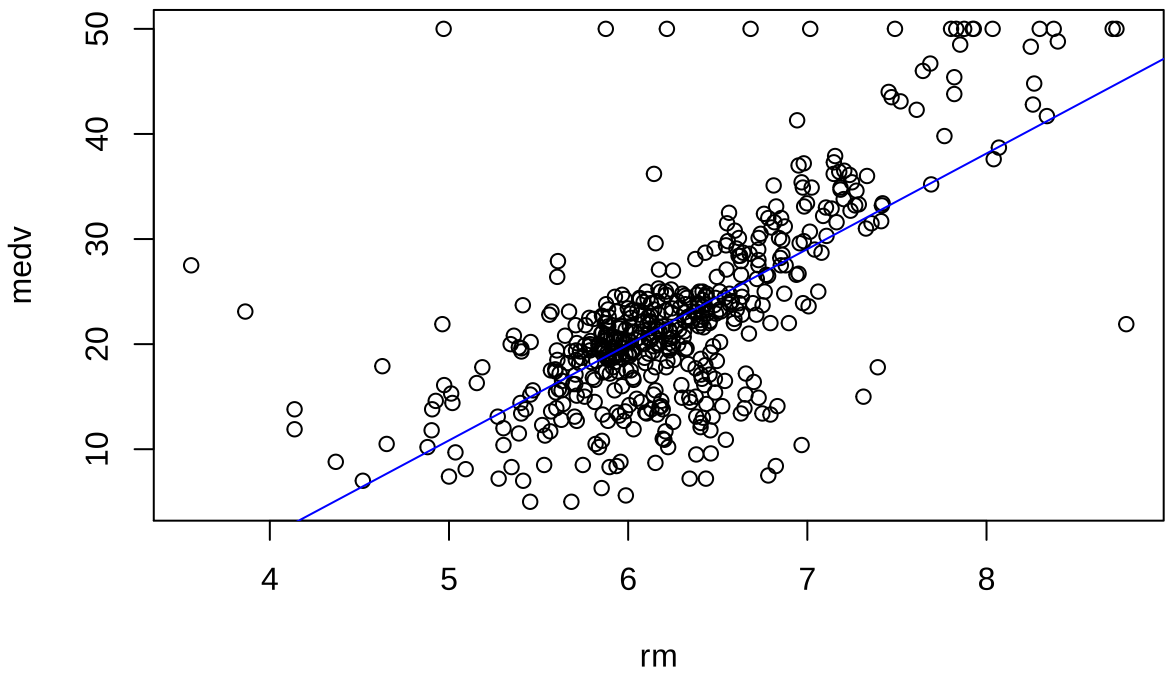
```r
pairs(Boston)
```



Plotting a regression line

Next we plot number of rooms on the x axis and median home value on the y axis. Then we use function abline() to plot a blue regression line on top of the points. The lm() function creates a linear regression model predicting median value as a function of rooms. We will learn more later, for now just realize that the regression line tries to plot through the middle of the trend. The trend is up. Not surprisingly, houses with more rooms tend to be more expensive.

```r
plot(rm, medv)
abline(lm(medv~rm), col="blue")
```



That's all for now. In future notebooks we will revisit the Boston housing data.