R_Hello-World_Regression

Robert McCulloch

1/24/2026

Hello World Data Analysis in R

In this notebook we will do a basic data analysis in R.

Our goal is to see how the price of a used car depends on characteristics of the car (the features).

We will

- read in the data to an R data frame
- pull off price, mileage, and year

- divide price and mileage by 1,000
- do some simple summaries

- run some regressions to relate car price to car features and get the standard inference.
- learn how R handles a categorical variable

Read in the Data and Get the Variables We Want

First we will read in the data from the file susedcars.csv.

```r
cd = read.csv("http://www.rob-mcculloch.org/data/susedcars.csv")
```

cd is now a *variable* which refers to a data structure that hold the data read from the file susedcars.csv.

The cd "is' ' an R data.frame.
The data structures in R are: vector, list, and data.frame.
We will see examples below.

You can also read directly from a file on your machine but you need your *working directory* to be set to the directory (or folder) where the file is.
You can use the functions getwd and setwd to check on your working directory.
See also /Session/Set Working Directory in Rstudio.

```r
getwd()  ## note also setwd
```

```
## [1] "/home/robert/do/webpage/R"
```

```r
cat("cd is a: ",typeof(cd),"\n")
```

```
## cd is a:  list
```

```r
cat("is it a data frame:",is.data.frame(cd),"\n")
```

```
## is it a data frame: TRUE
```

```r
cat("number of rows and columns: ", dim(cd),"\n")
```

```
## number of rows and columns:  1000 7
```

```
cat("the column names are:")
```

```
## the column names are:
```

```
names(cd)
```

```
## [1] "price"        "trim"         "isOneOwner"   "mileage"      "year"
## [6] "color"        "displacement"
```

So, there are a variety of ways to see what an object is in R.
One function that gives a lot of information about an object is `str`.

```
str(cd)
```

```
## 'data.frame':    1000 obs. of  7 variables:
##  $ price       : int  43995 44995 25999 33880 34895 5995 21900 41995 39882 47995 ...
##  $ trim        : chr  "550" "550" "550" "550" ...
##  $ isOneOwner  : chr  "f" "f" "f" "f" ...
##  $ mileage     : num  36858 46883 108759 35187 48153 ...
##  $ year        : int  2008 2012 2007 2007 2007 2002 2007 2007 2008 2011 ...
##  $ color       : chr  "Silver" "Black" "White" "Black" ...
##  $ displacement: chr  "5.5" "4.6" "5.5" "5.5" ...
```

```
head(cd)
```

```
##   price trim isOneOwner mileage year  color displacement
## 1 43995  550          f   36858 2008 Silver          5.5
## 2 44995  550          f   46883 2012  Black          4.6
## 3 25999  550          f  108759 2007  White          5.5
## 4 33880  550          f   35187 2007  Black          5.5
## 5 34895  550          f   48153 2007  Black          5.5
## 6  5995  500          f  121748 2002  other        other
```

Each row is an observation and corresponds to a used car for which we have measured the variables.
We want to see how the price relates the the other variables which are features of the car.

Let's just use mileage and year to start simply.

We will pull off the price, mileage and year columns.
Note the use of the $ notation to pull off a variable from a data.frame.
Our goal will be to see how price relates to mileage and year. We will divide both price and mileage by 1,000 to make the results easier to understand.

```
cd = cd[,c("price","mileage","year")]
cd$price = cd$price/1000
cd$mileage = cd$mileage/1000
head(cd)
```

```
##    price mileage year
## 1 43.995  36.858 2008
## 2 44.995  46.883 2012
## 3 25.999 108.759 2007
## 4 33.880  35.187 2007
## 5 34.895  48.153 2007
## 6  5.995 121.748 2002
```

vectors

The function c creates a vector.
A vector is just a list of things but they all must have the same type.
For example:

```
temp = c("price","mileage","year")
print(length(temp))
```

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

```
print(temp[2])
```

```
## [1] "mileage"
```

```
print(temp[2:3])
```

```
## [1] "mileage" "year"
```

```
print(temp[c(1,3)])
```

```
## [1] "price" "year"
```

```
junk = c(1,'rob')
print(junk) # 1 is turned into a character string
```

```
## [1] "1"   "rob"
```

```
stuff = c(1,5, 22)
typeof(stuff)
```

```
## [1] "double"
```

Back to our Data

We can get a summary of our data.frame.

```
summary(cd)
```

```
##      price           mileage             year     
##  Min.   : 0.995   Min.   :  1.997   Min.   :1994  
##  1st Qu.:12.995   1st Qu.: 40.133   1st Qu.:2004  
##  Median :29.800   Median : 67.919   Median :2007  
##  Mean   :30.583   Mean   : 73.652   Mean   :2007  
##  3rd Qu.:43.992   3rd Qu.:100.138   3rd Qu.:2010  
##  Max.   :79.995   Max.   :255.419   Max.   :2013  
```

Since all our variables are numeric, let's use the pairwise correlations to get a quick feeling for the relationships.

```
cor(cd)
```

```
##              price    mileage       year
## price    1.0000000 -0.8152458  0.8805373
## mileage -0.8152458  1.0000000 -0.7447292
## year     0.8805373 -0.7447292  1.0000000
```

Remember, a correlation is between -1 and 1.

The closer the correlation is to 1, the stronger the linear relationship between the variables, with a positive slope.

The closer the correlation is to -1, the stronger the linear relationship between the variables, with a negative slope.

So it looks like the bigger the mileage is, the lower the price of the car.
The bigger the year is, the higher the price of the car.

Makes sense!!

Note that we can also pull off columns and rows using integer indices starting at 1.

```
cd1 = read.csv("https://bitbucket.org/remcc/rob-data-sets/downloads/susedcars.csv")
cd1 = cd1[1:8,c(1,4,5)] #first 8 rows, columns 1, 4, and 5.
print(names(cd1))
```

```
## [1] "price"   "mileage" "year"
```

```
cd1
```

```
##   price mileage year
## 1 43995   36858 2008
## 2 44995   46883 2012
## 3 25999  108759 2007
## 4 33880   35187 2007
## 5 34895   48153 2007
## 6  5995  121748 2002
## 7 21900  115280 2007
## 8 41995   36370 2007
```

Functions, Data Strutures, Help

R basically works with functions and data structures.
For example, to read in the data, we called the function read.csv and it returned a data.frame.

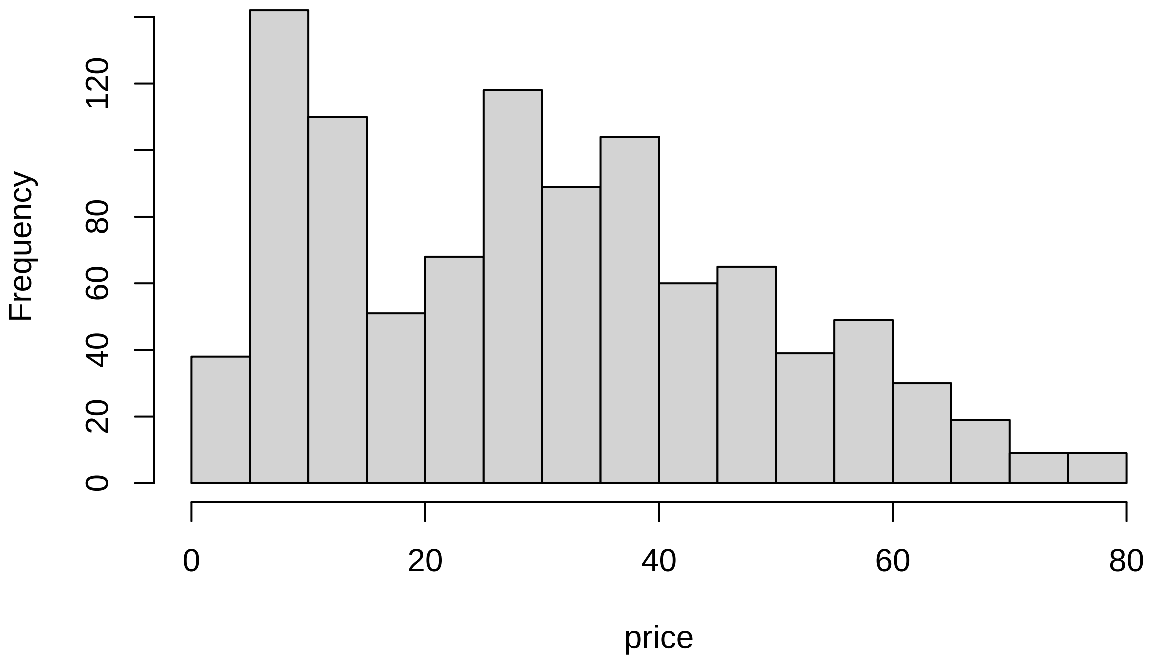
To get help on a function use ?functionName as in ?read.csv.

Functions can have a lot of arguments, many of which will be optional with defaults.

Plot y and versus each x

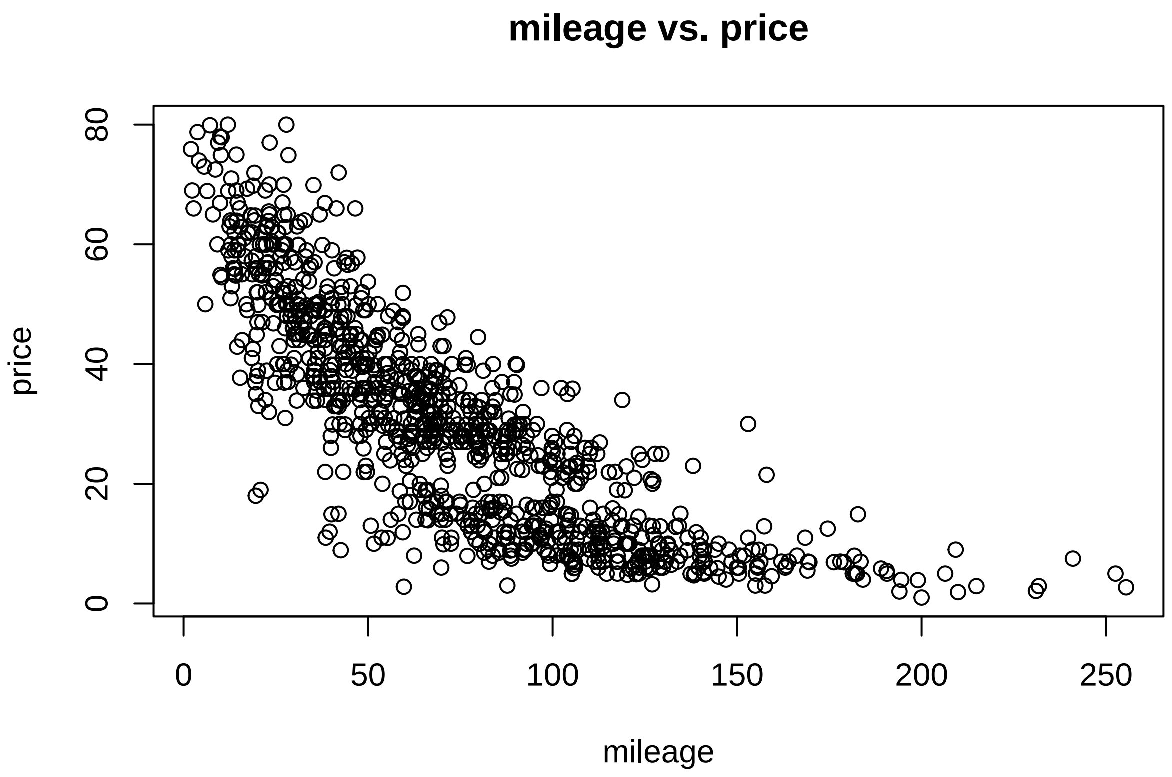
Let's get a histogram of y=price.

```
hist(cd$price,nclass=20,main='',xlab='price')  #nclass is the number of bins, it is an optional argumen
```



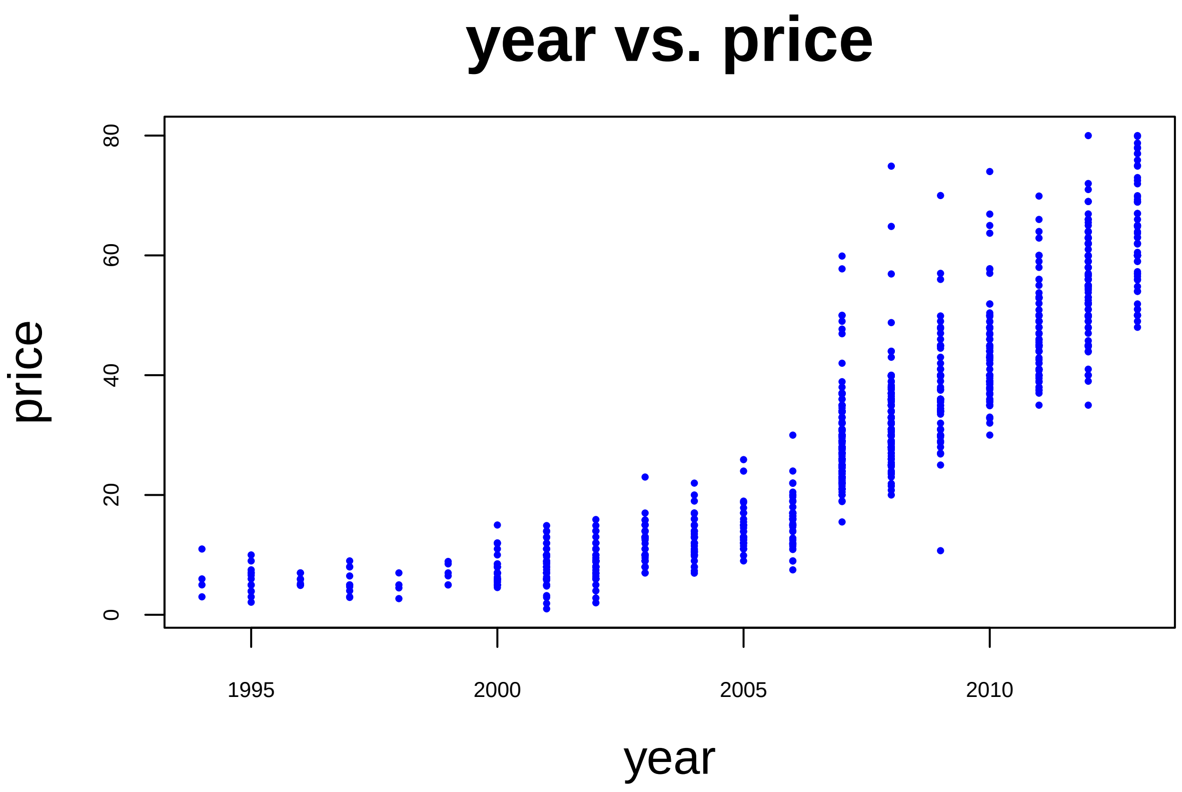
Now let's plot mileage vs. price.

```r
plot(cd$mileage,cd$price,xlab="mileage",ylab="price")
title(main="mileage vs. price")
```



And year vs. price.

```r
plot(cd$year,cd$price,xlab="year",ylab="price",col='blue',pch=16,cex=.5,cex.lab=1.5,cex.axis=.7)
title(main="year vs. price",cex.main = 2)
```



The "cex'' arguments control the size of various things.
pch controls the plot symbol.

Clearly, price is related to both year and mileage.
Clearly, the relationship is not linear !!!

What we really want to **learn** is the joint relationship betwee *price* and the pair of variables (*mileage*,*year*) !!!

Essentially, the modern statistical tools or *Machine Learning* enables us to learn the relationships from data without making strong assumptions.

In the expression:

$$price = f(mileage, year)$$

we would like to know the function f.

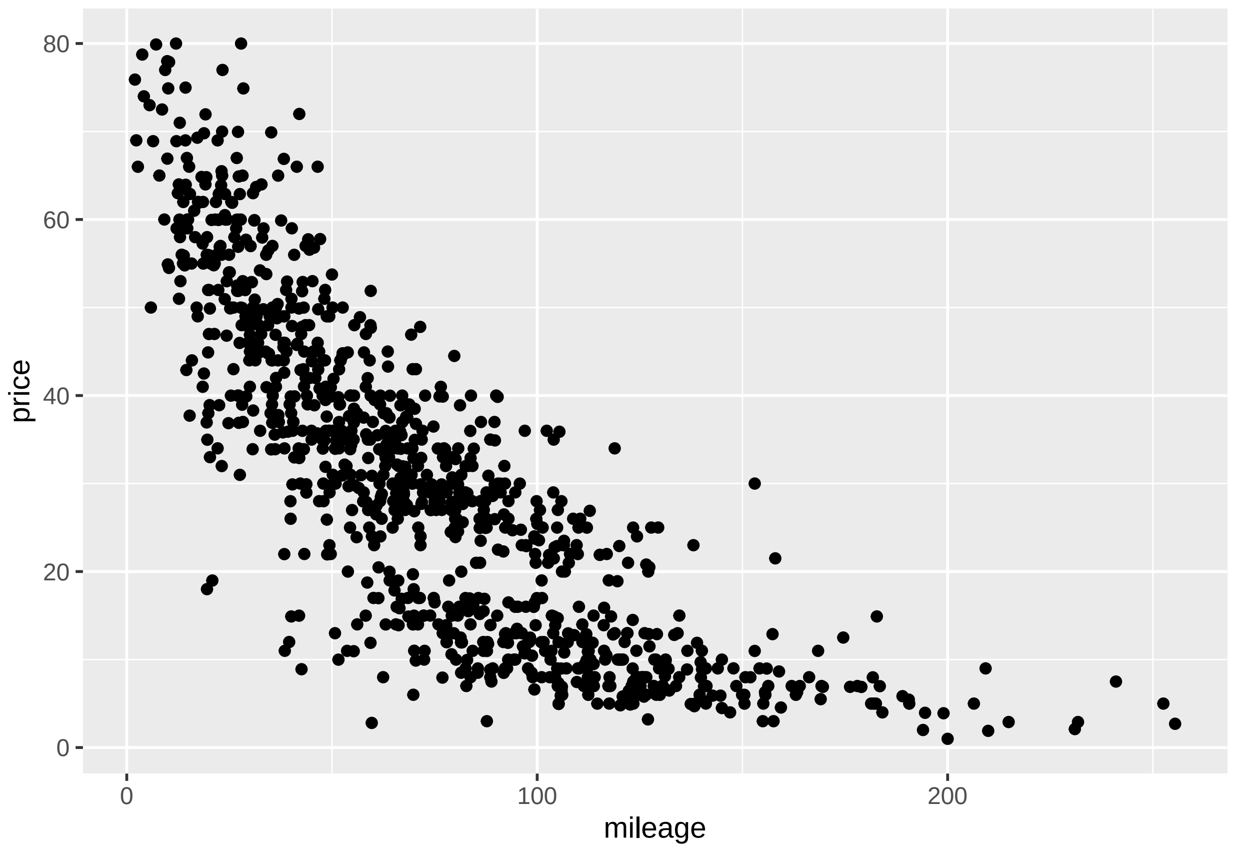
Using the ggplot R-package

Note that the vast number of R packages available in R is a major reason for using it.

Above I have used the graphics in "base R".
`ggplot` has become a very popular alternative graphics environment.

Note that to use ggplot, you would have to first install the R package and then load the library as below.

```r
# install.packages("ggplot2")  # if not already installed
library(ggplot2)
p = ggplot(data=cd,aes(x=mileage,y=price)) + geom_point()
p
```

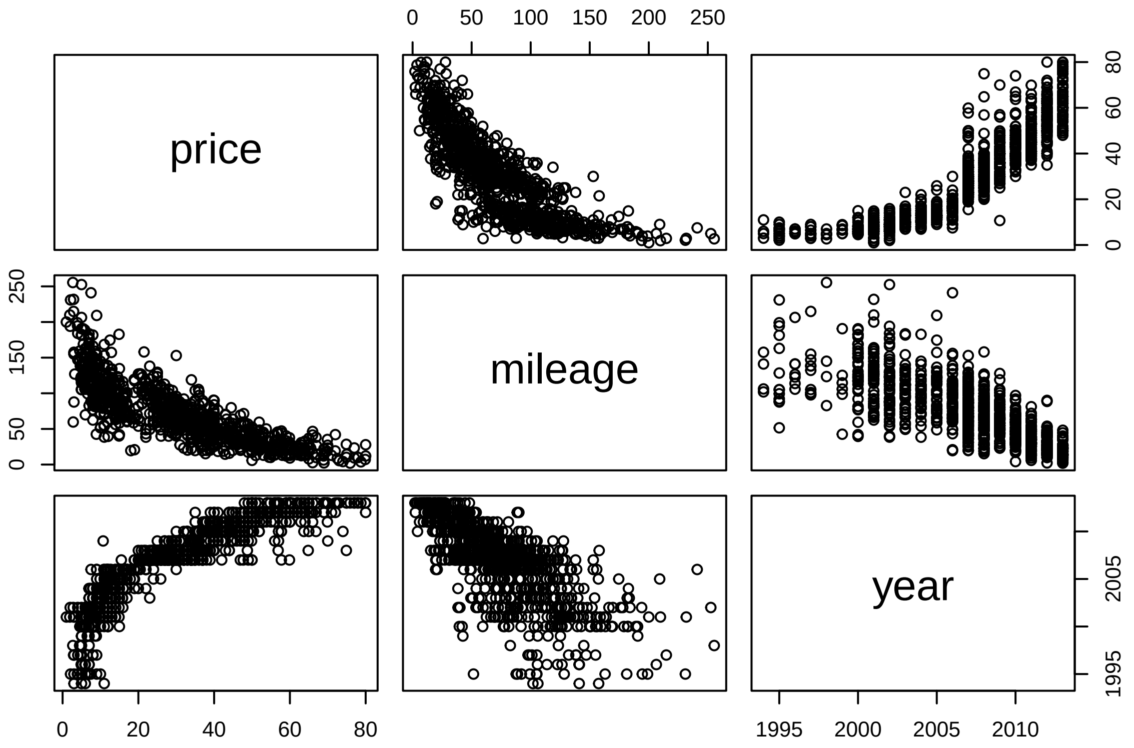


Note that in the bottom right frame in Rstudio there is a tab for managing R packages.

Note that ggplot is part of the "tidyverse" which is a set of R packages which has become quite popular. See the book "R For Data Science" by Wickham.

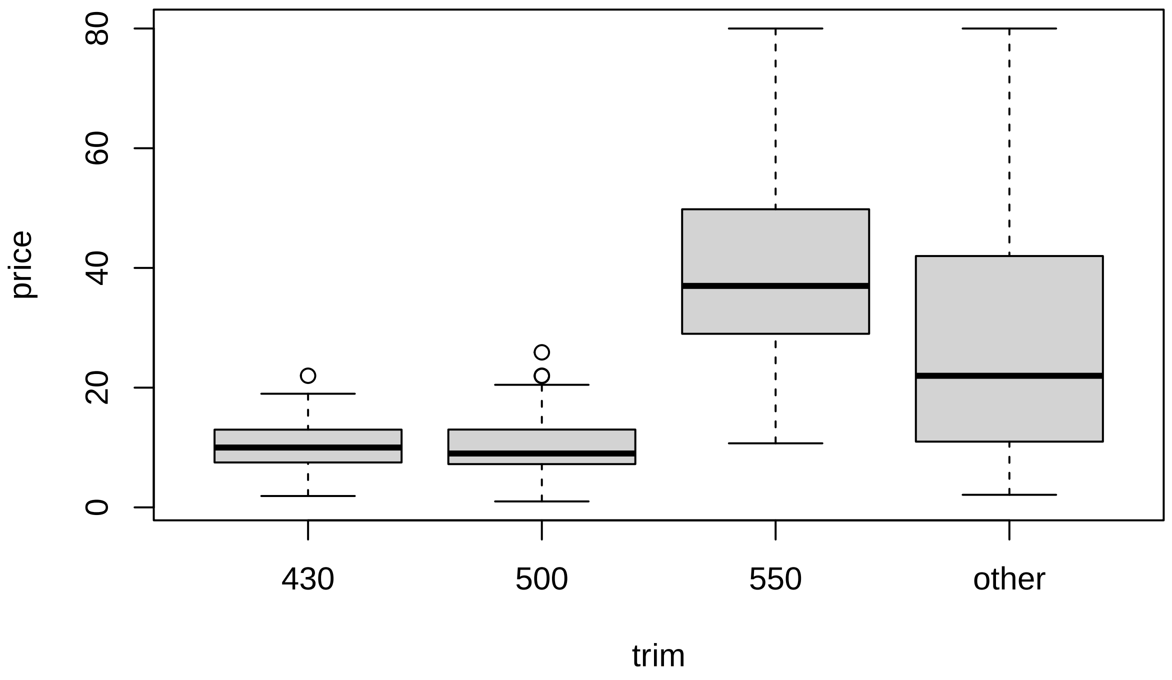
Another nice plot is

```r
pairs(cd)
```



Boxplots are super useful for plotting a numeric variable versus a categorical variable.
How does price relate to trim?

```r
cdtemp = read.csv("http://www.rob-mcculloch.org/data/susedcars.csv")
cdtemp$trim = factor(cdtemp$trim)  #see next section for factor
cdtemp$price = cdtemp$price/1000
boxplot(price~trim,cdtemp)
```



Categorical Variables in R, factors

When looking at data, a fundamental distinction is between variables which are numeric and variables which are categorical.

For example, in our cars data, price is numeric and trim is categorical.

Both in building models and plotting, you have to think about categorical variables differently from numeric ones !!

R very explicitly allows you to make the distinction between numeric and categorical variables and many important R functions adapt to the kind of variable you are using.

To see a simple example, let's add color to our data.

```
cd1 = read.csv("http://www.rob-mcculloch.org/data/susedcars.csv")
cd1 = cd1[,c(1,4,5,2)] #all rows, columns 1, 4, 5, and 2
print(names(cd1))
```

```
## [1] "price"   "mileage" "year"    "trim"
```

```
cd1$price = cd1$price/1000
cd1$mileage = cd1$mileage/1000
summary(cd1)
```

```
##      price           mileage             year          trim
##  Min.   : 0.995   Min.   :  1.997   Min.   :1994   Length:1000
##  1st Qu.:12.995   1st Qu.: 40.133   1st Qu.:2004   Class :character
##  Median :29.800   Median : 67.919   Median :2007   Mode  :character
##  Mean   :30.583   Mean   : 73.652   Mean   :2007
##  3rd Qu.:43.992   3rd Qu.:100.138   3rd Qu.:2010
##  Max.   :79.995   Max.   :255.419   Max.   :2013
```

trim is treated differently from the numerics, but not too intelligently.
We can fix this, by telling R that trim is a *factor*.

```
cd1$trim = as.factor(cd1$trim)
summary(cd1)
```

```
##      price           mileage             year         trim
##  Min.   : 0.995   Min.   :  1.997   Min.   :1994   430  :143
##  1st Qu.:12.995   1st Qu.: 40.133   1st Qu.:2004   500  :127
##  Median :29.800   Median : 67.919   Median :2007   550  :591
##  Mean   :30.583   Mean   : 73.652   Mean   :2007   other:139
##  3rd Qu.:43.992   3rd Qu.:100.138   3rd Qu.:2010
##  Max.   :79.995   Max.   :255.419   Max.   :2013
```

The variable `trim` is now treated as cateorical.
A categorical has a fixed set of possible categories or "levels".

```
levels(cd1$trim)
```

```
## [1] "430"   "500"   "550"   "other"
```

You can check "what" a variable is.

```
cat("is color a factor: ",is.factor(cd1$trim),"\n")
```

```
## is color a factor:  TRUE
```

```
cat("is price a factor: ",is.factor(cd1$price),"\n")
```

```
## is price a factor:  FALSE
```

apply

```
sapply(cd1,is.factor) # simple apply
```

```
##   price mileage    year    trim
##   FALSE   FALSE   FALSE    TRUE
```

There are a bunch of apply functions.

```
print(apply(cd,2,mean)) ## apply function mean to columns of cd
```

```
##      price    mileage       year
##   30.58332   73.65241 2006.93900
```

```
print(c(mean(cd[,1]),mean(cd[,2]),mean(cd[,3])))
```

```
## [1]   30.58332   73.65241 2006.93900
```

Regression with just mileage

Our goal is to relate y=price to x=(mileage,price). Let's start simple and just use mileage.

Our *simple linear regression model* is

$$price = \beta_0 + \beta_1\, mileage + \epsilon$$

ϵ represents the part of price we cannot learn from mileage.

Let's regress price on mileage to estimate the model:

```
#regress y=price on x=mileage, price and mileage are in the data.frame cd
lmmod1 = lm(price~mileage,cd)
cat("estimated coefficients are: ",lmmod1$coef,"\n")
```

```
## estimated coefficients are:  56.35978 -0.3499745
```

```
summary(lmmod1)
```

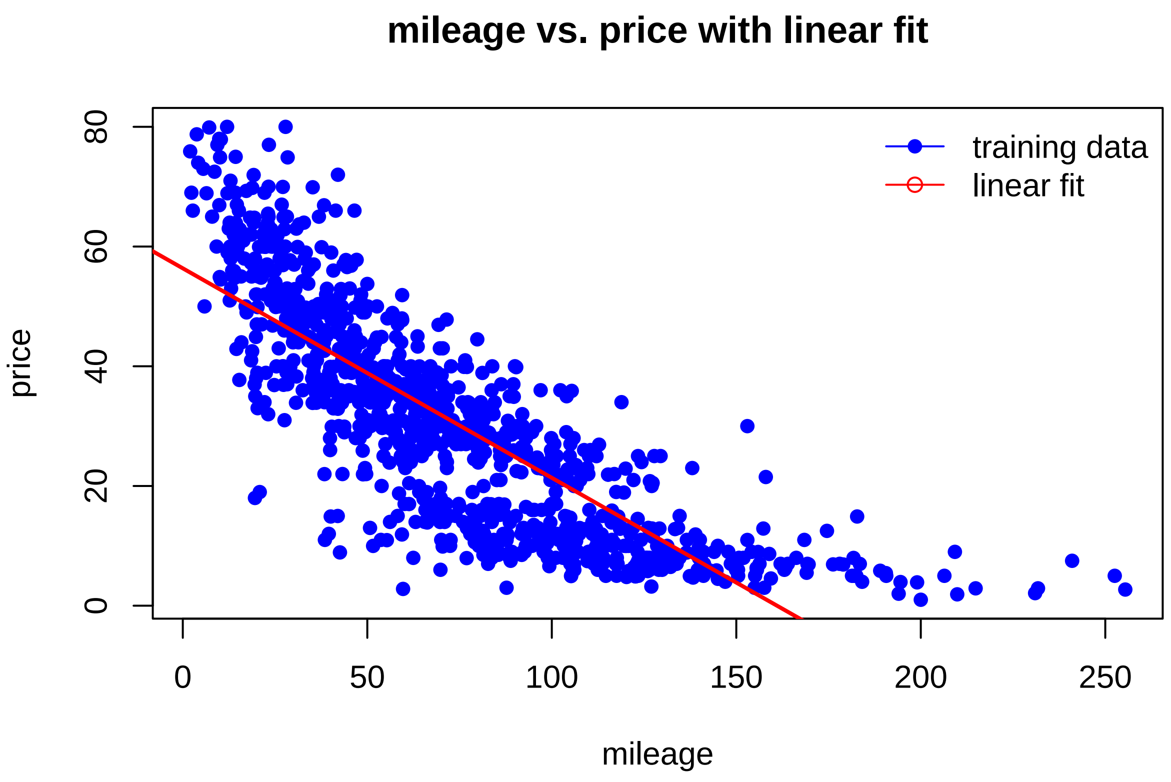
```
##
## Call:
## lm(formula = price ~ mileage, data = cd)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -32.670  -7.063   0.239   6.293  37.024
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 56.35978    0.67063   84.04   <2e-16 ***
## mileage     -0.34997    0.00787  -44.47   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.67 on 998 degrees of freedom
## Multiple R-squared:  0.6646, Adjusted R-squared:  0.6643
## F-statistic:  1978 on 1 and 998 DF,  p-value: < 2.2e-16
```

So, the fitted model is

$$price = 56.36 - .35\, mileage + \epsilon$$

Let's plot the training data with the fitted line.

```r
plot(cd$mileage,cd$price,xlab="mileage",ylab="price",col="blue",pch=16)
abline(lmmod1$coef,col="red",lwd=2)
title(main="mileage vs. price with linear fit")
legend("topright",legend=c("training data","linear fit"),col=c("blue","red"),pch=c(16,1),lwd=c(1,1),bty=
```



Pretty bad !! You better take an applied machine learning course!!

Run the Regression of y=price on X=(mileage,year)

Let's run a linear regression of *price* on *mileage* and *year*.

Our model is:

$$price = \beta_0 + \beta_1\, mileage + \beta_2\, year + \epsilon$$

This model assumes a linear relationship.

We already know this may not be a good idea !!!

Let's go ahead and *fit* the model.

Fitting the model to data will give us estimates of the parameters $(\beta_0, \beta_1, \beta_2)$.

The error term ϵ represents the part of price we cannot know from $(mileage, year)$.

```r
lmmod = lm(price~mileage+year,cd)
cat("the coefficients are:",lmmod$coefficients,"\n")
```

```
## the coefficients are: -5365.49 -0.1537219 2.69435
```

So, the fitted relationship is

$$price = -5365.49 - 0.154\, mileage + 2.7\, year$$

Note the use of the formula `price~mileage+year` to specify the model.
This is used extensively in R.

What is a list in R

What is `lmmod` ?

First we need to know what a list is in R. For example:

```r
tempL = list(a='rob',b=c(1,45,22))
cat('the list\n')
```

```
## the list
```

```r
print(tempL)
```

```
## $a
## [1] "rob"
##
## $b
## [1]  1 45 22
```

```r
cat('the components of the list\n')
```

```
## the components of the list
```

```r
print(names(tempL))
```

```
## [1] "a" "b"
```

```r
cat('just the b component\n')
```

```
## just the b component
```

```r
print(tempL$b)
```

```
## [1]  1 45 22
```

Back to lmmod

```r
cat("lmmod is a list: ",is.list(lmmod),"\n")
```

```
## lmmod is a list:  TRUE
```

```r
cat("lmmod has named components:\n ",names(lmmod),"\n")
```

```
## lmmod has named components:
##   coefficients residuals effects rank fitted.values assign qr df.residual xlevels call terms model
```

```r
## to get more information about a regression, get its summary
summary(lmmod)
```

```
##
## Call:
## lm(formula = price ~ mileage + year, data = cd)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -21.857  -4.855  -1.670   3.483  34.499
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.365e+03  1.716e+02  -31.27   <2e-16 ***
## mileage     -1.537e-01  8.339e-03  -18.43   <2e-16 ***
## year         2.694e+00  8.526e-02   31.60   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.544 on 997 degrees of freedom
## Multiple R-squared:  0.8325, Adjusted R-squared:  0.8321
## F-statistic:  2477 on 2 and 997 DF,  p-value: < 2.2e-16
```

There is useful information in the summary:

```
temp = summary(lmmod)
names(temp)
```

```
##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

Get and Plot the Fits

Let's get the fitted values.

For each observation in our data set the fits are

$$\hat{price_i} = -5365.49 - 0.154\, mileage_i + 2.7\, year_i$$

```
yhat = lmmod$fitted.values
cat("the type of yhat is:",typeof(yhat),"\n")
```

```
## the type of yhat is: double
```

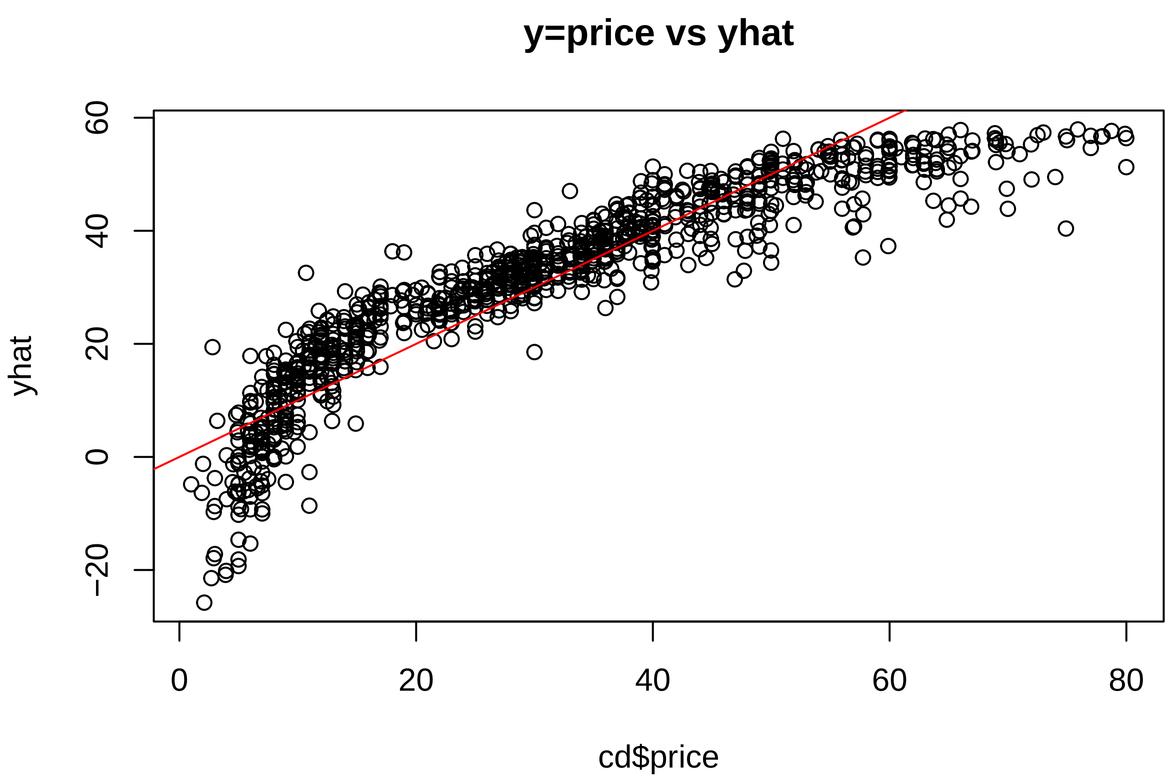
```
cat("is it a vector?:",is.vector(yhat),"\n")
```

```
## is it a vector?: TRUE
```

```
cat("length of yhat is: ",length(yhat),"\n")
```

```
## length of yhat is:  1000
```

```
plot(cd$price,yhat)
abline(0,1,col="red")
title(main='y=price vs yhat')
```



Clearly, it is really bad !!

Machine Learning will enable us to get it right fairly automatically.

Predictions

Many models in R get predictions by calling the predict function on the model data structure (object).

We have a fitted model object lmmod and we will call predict on it.

```
xpdf = data.frame(mileage=c(40,100),year=c(2010,2004))
ypred = predict(lmmod,xpdf)
print(xpdf)
```

```
##   mileage year
## 1      40 2010
## 2     100 2004
```

```
print(ypred)
```

```
##        1        2
## 44.00383 18.61442
```

Let's check it:

```
sum(lmmod$coef * as.double(c(1,xpdf[1,])))
```

```
## [1] 44.00383
```

General problems and terminology in machine learning

The data we used to "fit" our model, is called the *training data*.

When we look at predictions for observations in the training data (as we did for `yhat`) we say we are looking at *in-sample* fits or the fit on the training data.

When we predict at observations not in the training data (as we did for `ypred`), then we are predicting *out of sample*.

Out of sample prediction is the name of the game in predictive modeling or *supervised learning*.

Out of sample prediction is always a more interesting test since you have not already seen.

A *huge* part of Machine Learning is about building models to obtain good out of sample predictions!!!!

In supervised learning, we learn the parameters of our model on training data and then predict on the out of sample test data.
A good model makes accurate predictions on the out of sample test data.
We predict the target variable y with the features x.
As noted above, in our example y = price x=(mileage,year).
In statistics y would be called the dependent variable and x would be called the independent variables.

There is also unsupervised learning in which the goal is to understand struture in data without neccessarily have a target variable y that we want to predict.

For example, if I regress y=price on x=mileage, I am doing supervised learning.
If I compute the correlation between y and x that is unsupervised learning.

Another key distinction is classification versus regression.
In regression, our target variable y is numeric. In classification, y is categorical.
For example, if we were trying to predict if a used car sells in the next month, then y would be binary categorical sells/does not sell.

Many of our key modelling approaches have versions for regression and classification
where some of the basic ideas are the same but we need to change some basic details.
For example we have linear regression for a numeric y and logistic regression for a binary y.

Standard Errors from Standard Regression Output

From our linear regression fit, we got estimates for the parameters.

Or in Machine learning lingo, we learned the parameters from the training data.

Often we want to know a lot more about the model fit.

In particular, we might want to know the *standard errors* associated with the parameter estimates.

```r
summary(lmmod)
```

```
##
## Call:
## lm(formula = price ~ mileage + year, data = cd)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -21.857  -4.855  -1.670   3.483  34.499
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.365e+03  1.716e+02  -31.27   <2e-16 ***
## mileage     -1.537e-01  8.339e-03  -18.43   <2e-16 ***
## year         2.694e+00  8.526e-02   31.60   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 7.544 on 997 degrees of freedom
## Multiple R-squared:  0.8325, Adjusted R-squared:  0.8321
## F-statistic:  2477 on 2 and 997 DF,  p-value: < 2.2e-16
```

The standard error associate with the estimate of the slope for *mileage* is .0083.

The confidence interval for β_1, the *mileage* slope is:

```
-.1537 + c(-2,2)*0.0083
```

```
## [1] -0.1703 -0.1371
```

The confidence interval is supposed to give us some indication about our uncertainty about the coefficient given the information in the data:
small interval: data tells you a lot
big interval: data not that informative.

In basic statistics, there is a big emphasis on quantifying uncertainty.
In Machine Learning, it is more about prediction.

Recall that R^2 is the square of the correlation between y and $\hat{y}$:

```
yyhat = cbind(cd$price,yhat)
dim(yyhat)
```

```
## [1] 1000    2
```

```
cor(yyhat)
```

```
##                     yhat
##      1.0000000 0.9123897
## yhat 0.9123897 1.0000000
```

The squared correlation is .91239ˆ2 = 0.8324555, which is the same as in the regression output.

So, R-squared with just mileage is .66 but with year and mileage it is .83.

Let's try the variable trim

Does knowing the trim of the car help us predict?

Let's try it.
How do we add trim to a linear regression model?

We alread have cd1:

```
head(cd1)
```

```
##    price mileage year trim
## 1 43.995  36.858 2008  550
## 2 44.995  46.883 2012  550
## 3 25.999 108.759 2007  550
## 4 33.880  35.187 2007  550
## 5 34.895  48.153 2007  550
## 6  5.995 121.748 2002  500
```

The lm function "automatically" handles a factor (a categorical variable):

```
lmt = lm(price~mileage + year + trim,cd1)
print(summary(lmt))
```

```
##
## Call:
## lm(formula = price ~ mileage + year + trim, data = cd1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.428  -4.481  -1.626   3.530  30.060
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.299e+03  1.967e+02 -26.939  < 2e-16 ***
## mileage     -1.394e-01  7.975e-03 -17.477  < 2e-16 ***
## year         2.658e+00  9.792e-02  27.146  < 2e-16 ***
## trim500      2.427e+00  8.747e-01   2.775  0.00562 **
## trim550      4.775e+00  8.367e-01   5.706 1.52e-08 ***
## trimother    9.589e+00  8.680e-01  11.047  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.106 on 994 degrees of freedom
## Multiple R-squared:  0.8518, Adjusted R-squared:  0.851
## F-statistic:  1142 on 5 and 994 DF,  p-value: < 2.2e-16
```

According to R^2, trim did not do much!!

How did lm put trim in the model?

```
summary(cd1$trim)
```

```
##   430   500   550 other
##   143   127   591   139
```

We add binary indicators for three of the four color levels.

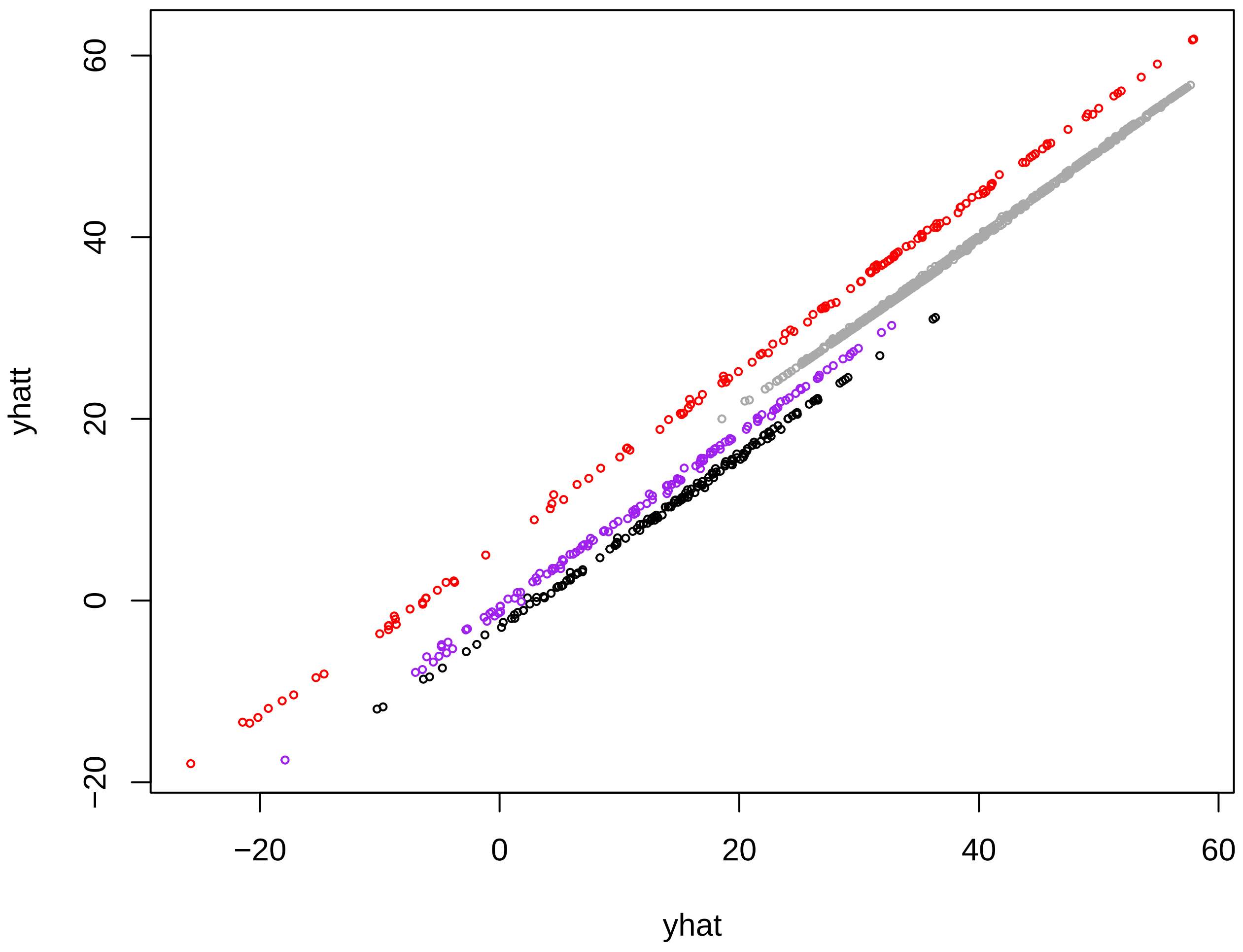
```
head(model.matrix(lmt))
```

```
##   (Intercept) mileage year trim500 trim550 trimother
## 1           1  36.858 2008       0       1         0
## 2           1  46.883 2012       0       1         0
## 3           1 108.759 2007       0       1         0
## 4           1  35.187 2007       0       1         0
## 5           1  48.153 2007       0       1         0
## 6           1 121.748 2002       1       0         0
```

```
cd1$trim[1:6]
```

```
## [1] 550 550 550 550 550 500
## Levels: 430 500 550 other
```

```
yhatt = lmt$fitted.values
cvec= rep('white',nrow(cd1))
cvec[cd1$trim=='430'] = 'black'
cvec[cd1$trim == '500'] = 'purple'
cvec[cd1$trim=='550'] = 'darkgrey'
cvec[cd1$trim == 'other'] = 'red'
plot(yhat,yhatt,col=cvec,cex=.5)
```



Even though R-squared was not much bigger when we included trim, it does seem to make a bit of a difference as a practial matter.

Regression In Matrix Notation

Let's write our multiple regression model using vector/matrix notation and use basic matrix operations to check the predicted and fitted values.

The general multiple regression model is written:

$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots \beta_p x_{ip} + \epsilon_i,\ i = 1, 2, \dots, n,$$

where i indexes observations and x_{ij} is the value of the j^{th} x in the i^{th} observation. If we let

$$x_i = \begin{bmatrix} 1 \\ x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{bmatrix},\ X = \begin{bmatrix} x_1' \\ x_2' \\ \vdots \\ x_n' \end{bmatrix},\ y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix},\ \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix},\ \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} \tag{1}$$

,

then we can write the model in matrix form:

$$y = X\beta + \epsilon.$$

Let's check this.

```r
X = cbind(rep(1,nrow(cd)),as.matrix(cd[,c(2,3)]))
head(X)
```

```
##        mileage year
## [1,] 1  36.858 2008
## [2,] 1  46.883 2012
## [3,] 1 108.759 2007
## [4,] 1  35.187 2007
## [5,] 1  48.153 2007
## [6,] 1 121.748 2002
```

```r
bhat = matrix(lmmod$coef,ncol=1)
yhatM = X %*% bhat
print(summary(yhatM - yhat))
```

```
##        V1
##  Min.   :-2.633e-09
##  1st Qu.:-8.164e-11
##  Median :-8.139e-11
##  Mean   :-8.393e-11
##  3rd Qu.:-8.112e-11
##  Max.   :-8.052e-11
```