Bivariate Data

2020

In this section we review the relationship between 2 variables.

Bivariate categorical data

We use the same functions as in the case of one categorical variable.

Let's say we want to evaluate if the students who smoke study less?

And we have the following data for the `smoke` and `amount` variables.
Where amount is presented as follows:
1 - less than 5 hours of studying;
2 - 5 - 10 hours of studying;
3 - more than 10 hours of studying.

```
> smokes <- c("Yes", "No", "No", "Yes", "No", "Yes",
"Yes", "Yes", "No", "Yes")
> amount <- c(1, 2, 2, 3, 3, 1, 2, 1, 3, 2)
> table(smokes, amount)
      amount
smokes 1 2 3
   No  0 2 2
   Yes 3 2 1
```

Let's review the proportions.

To see it clearer we set the numbers to be rounded to the 3 decimal place

```
> options(digits = 3)
```

We can calculate proportions using the `prop.table` function or the conditional proportions using:
`1` means that the proportions in the rows will sum to 1
`2` means that the proportions in the columns will sum to 1

```
> prop.table(table(smokes, amount))
```

```
      amount
smokes   1   2   3
   No  0.0 0.2 0.2
   Yes 0.3 0.2 0.1
```

```
> prop.table(table(smokes, amount), 1)
```

```
      amount
smokes     1     2     3
   No  0.000 0.500 0.500
   Yes 0.500 0.333 0.167
```

```
> prop.table(table(smokes, amount), 2)
```

```
      amount
smokes     1     2     3
   No  0.000 0.500 0.667
   Yes 1.000 0.500 0.333
```

Another example

```
> hair <- c("blond", "blond", "black", "blond", "brown",
"brown",
+           "brown", "brown", "black", "brown", "black",
"brown",
+           "black", "black", "black", "brown", "brown",
"brown",
+           "brown", "brown", "black", "brown", "black",
"brown",
+           "blond", "blond", "black", "blond", "brown",
"brown",
+           "brown", "brown", "black", "brown", "black",
"brown",
+           "brown", "brown", "black", "brown", "black",
"brown",
+           "blond", "blond", "black", "blond", "brown",
"brown")
> eyes <- c("blue", "green", "brown",  "blue", "green",
"brown",
+           "brown", "black", "black", "green", "brown",
"brown",
```

```
+           "green", "black", "black", "brown", "brown",
"black",
+           "green", "black", "black", "brown", "brown",
"black",
+           "brown",  "blue", "green", "brown", "brown",
"black",
+           "black", "green", "brown",  "blue", "green",
"brown",
+           "brown", "black", "brown",  "blue", "green",
"brown",
+           "blue", "green", "brown",  "blue", "green",
"brown")
```

```
> table(hair, eyes)
```

```
       eyes
hair     black blue brown green
  black      4    0     6     4
  blond      0    5     2     2
  brown      7    2    11     5
```

```
> prop.table(table(hair, eyes))
```

```
       eyes
hair      black   blue  brown  green
  black 0.0833 0.0000 0.1250 0.0833
  blond 0.0000 0.1042 0.0417 0.0417
  brown 0.1458 0.0417 0.2292 0.1042
```

```
> prop.table(table(hair, eyes), 1)
```

```
       eyes
hair     black  blue brown green
  black 0.286 0.000 0.429 0.286
  blond 0.000 0.556 0.222 0.222
  brown 0.280 0.080 0.440 0.200
```

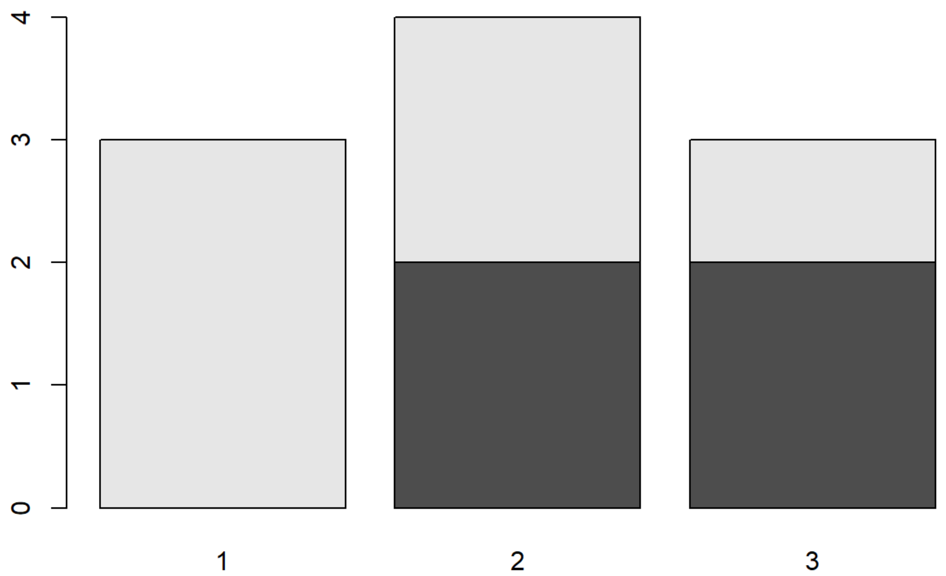
```
> prop.table(table(hair, eyes), 2)
```

```
       eyes
hair     black  blue brown green
  black 0.364 0.000 0.316 0.364
```

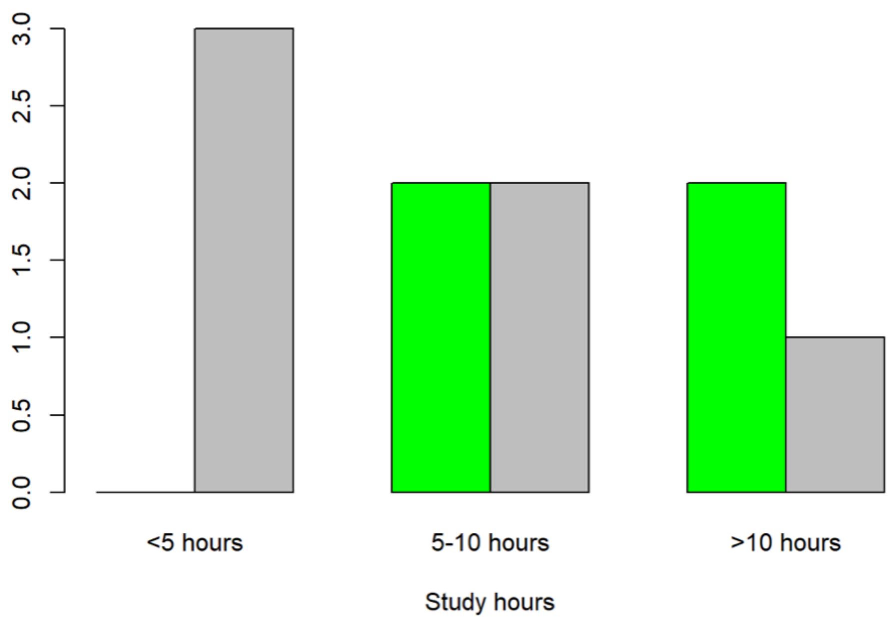
```
  blond 0.000 0.714 0.105 0.182
  brown 0.636 0.286 0.579 0.455
```

We can plot the data using `barplot` and `pie` functions.

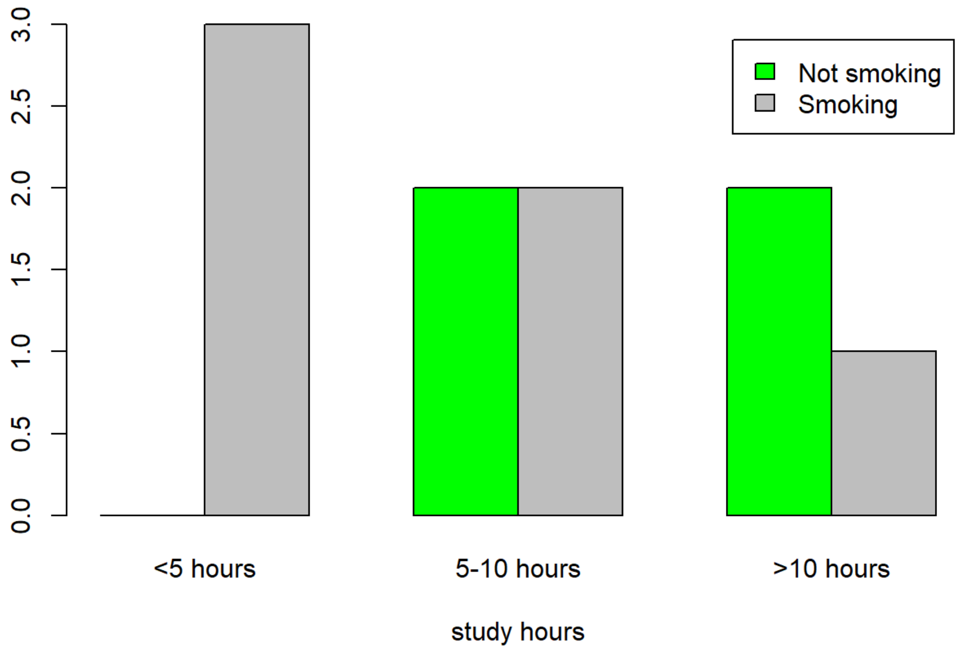
```
> barplot(table(smokes, amount))
```



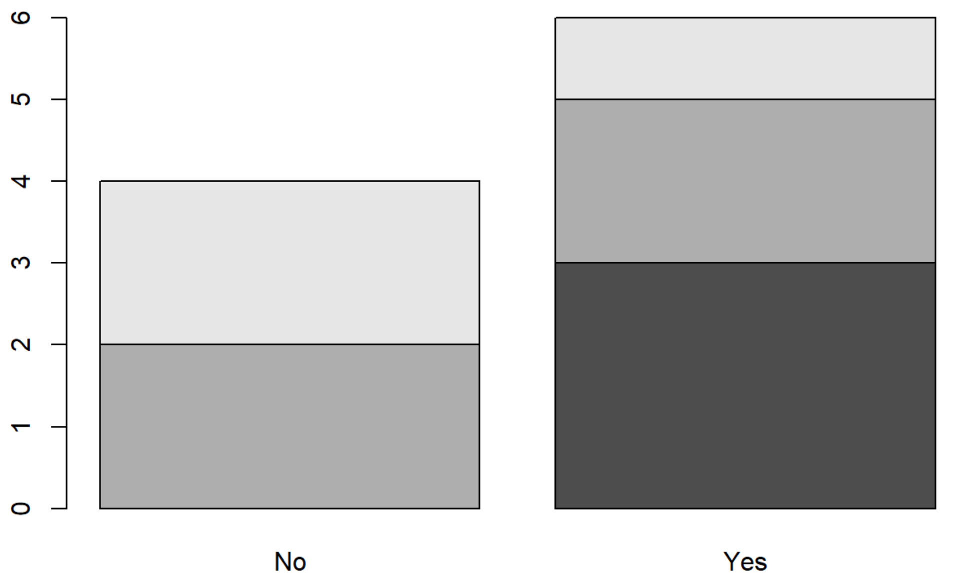

```
> barplot(table(smokes, amount),
+         names.arg = c("<5 hours", "5-10 hours", ">10 
hours"),
+         beside = TRUE,
+         col = c("Green", "Grey"),
+         xlab = "Study hours")
```



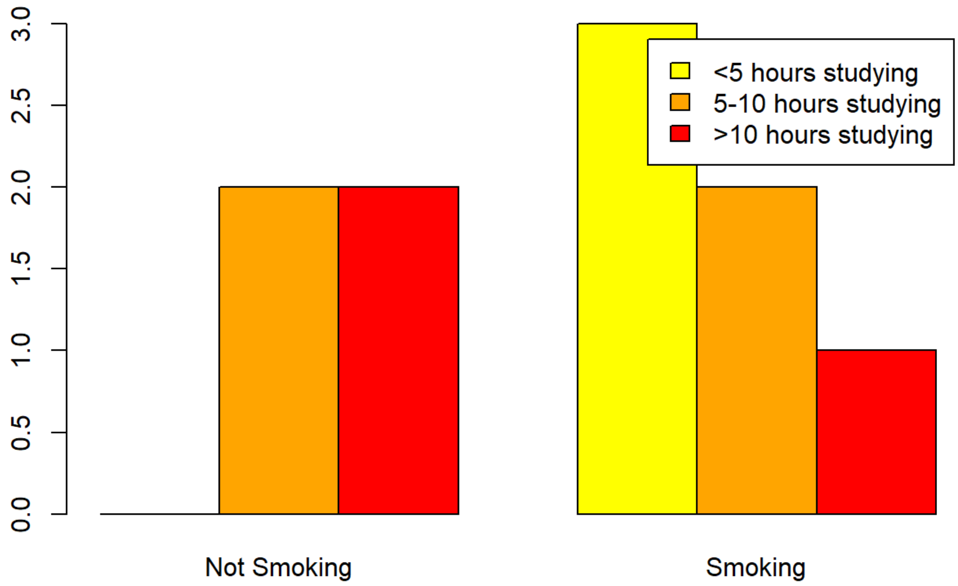
```
> barplot(table(smokes,amount),
+           names.arg = c("<5 hours", "5-10 hours", ">10
hours"),
+           legend.text = c("Not smoking", "Smoking"),
+           beside = TRUE,
+           col = c("Green", "Grey"),
+           xlab = "study hours")
```




```
> barplot(table(amount, smokes))
```



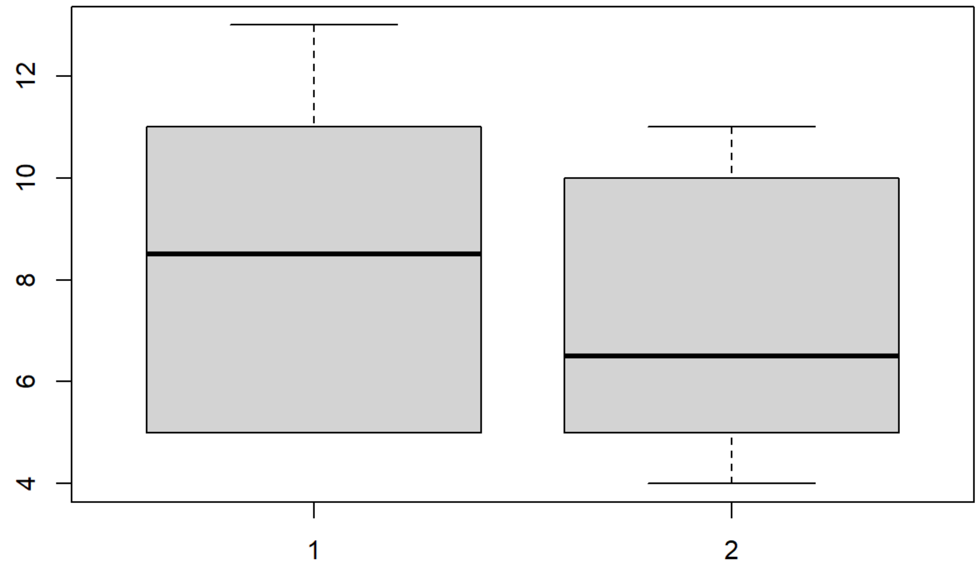
```
> barplot(table(amount,smokes),
+         names.arg = c("Not Smoking", "Smoking"),
+         legend.text = c("<5 hours studying","5-10 hours
studying",">10 hours studying"),
+         beside = TRUE,
+         col = c("Yellow", "Orange", "Red"))
```



Categorical vs. numerical data

Simple example might be a drug test, where you have data for an experimental group and for a control group

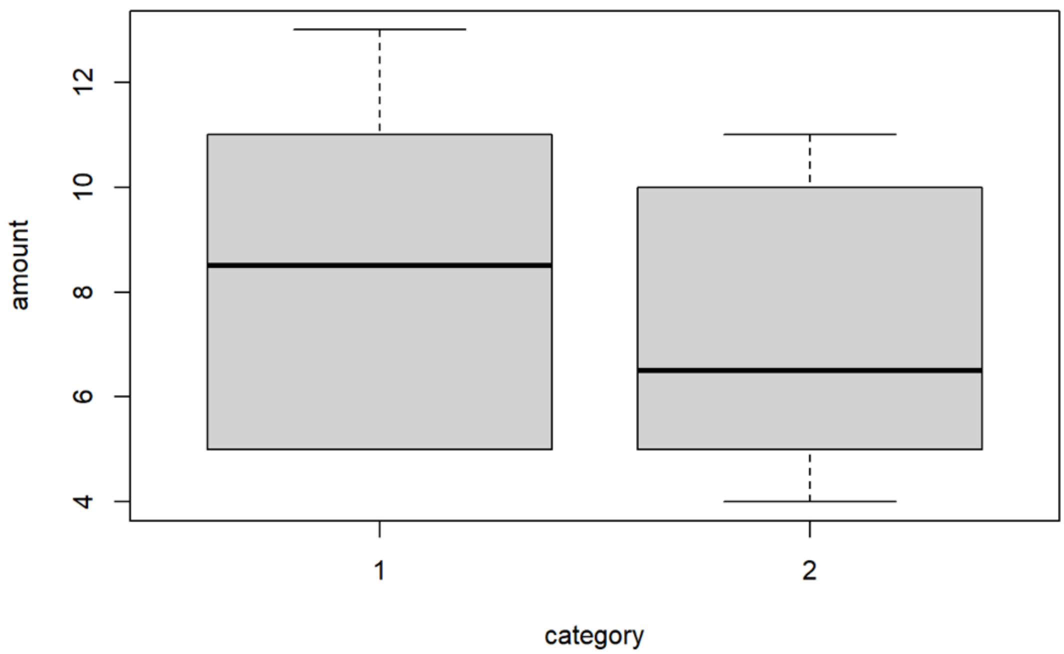
```
> group.experimental <- c(5, 5, 5, 13, 7, 11, 11, 9, 8,
9)
> group.control <- c(11, 8, 4, 5, 9, 5, 10, 5, 4, 10)
> boxplot(group.experimental, group.control)
```



From the plot we see that the control group has smaller observations than the experimental group.

Another example will be if we have the same data, but structured differently. Here we can use the `response ~ predictor`, which will break up the values in response, by the categories in category

```
> amount = c(5, 5, 5, 13, 7, 11, 11, 9, 8, 9, 11, 8, 4,
5, 9, 5, 10, 5, 4, 10)
> category = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2)
> boxplot(amount ~ category)
```



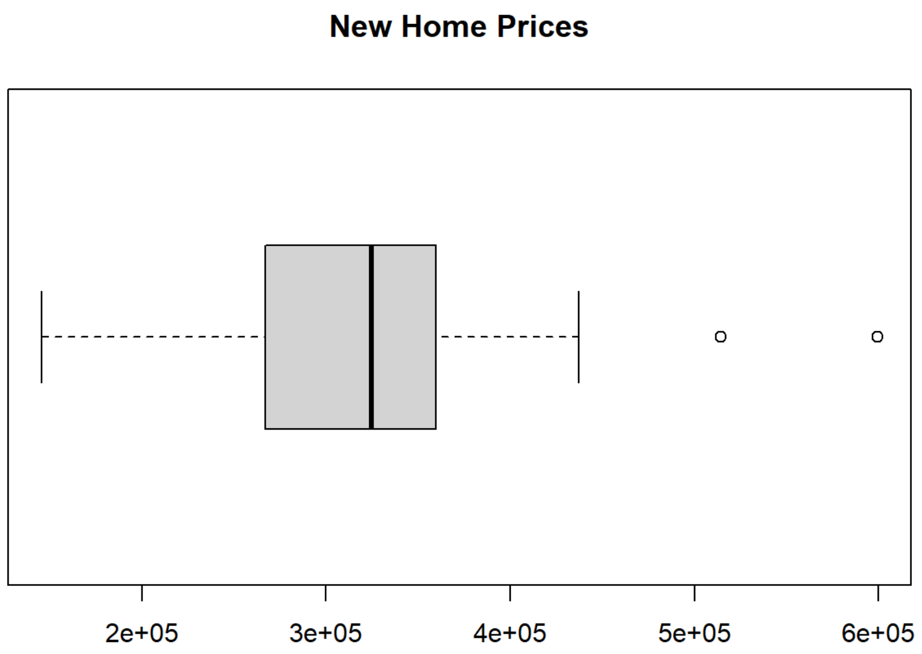
Numerical vs. numerical data

Comparing two numerical variables can be done in different ways. If the two variables are thought to be independent samples you might like to compare their distributions in some manner. However, if you expect a relationship between the variables, you might like to look for that by plotting pairs of points.

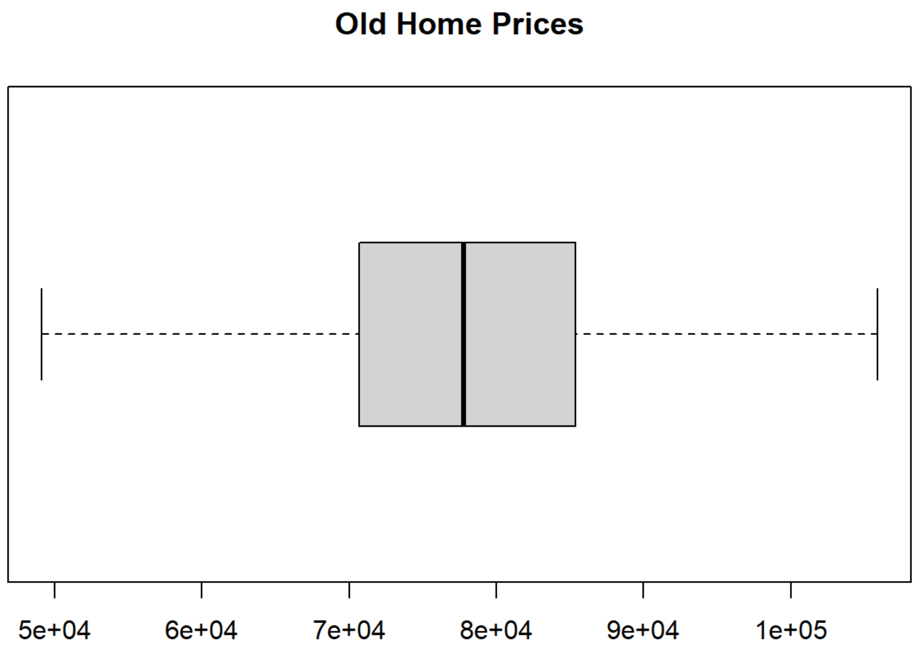
Comparing two distributions with plot

Let's compare the distributions of the old and the new prices of homes from the `home` data frame:

```
> library(UsingR)
Warning: package 'UsingR' was built under R version 4.0.3
Loading required package: MASS
Loading required package: HistData
Loading required package: Hmisc
Loading required package: lattice
Loading required package: survival
Loading required package: Formula
Loading required package: ggplot2

Attaching package: 'Hmisc'
The following objects are masked from 'package:base':

    format.pval, units

Attaching package: 'UsingR'
The following object is masked from 'package:survival':

    cancer
> head(home)
    old    new
1 64200 257500
2 72100 276800
3 87600 364600
4 59000 160400
5 83200 333500
6 49100 145600
> boxplot(home$new, horizontal = TRUE, main = "New Home
Prices")
```

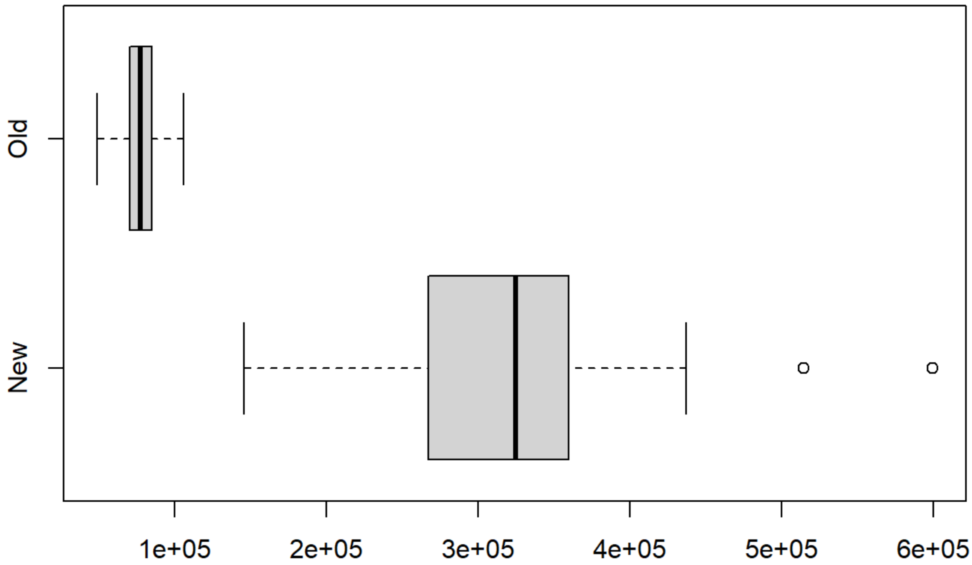



```
> boxplot(home$old, horizontal = TRUE, main = "Old Home Prices")
```




```
> boxplot(home$new, home$old, horizontal = TRUE, names = c("New", "Old"))
```

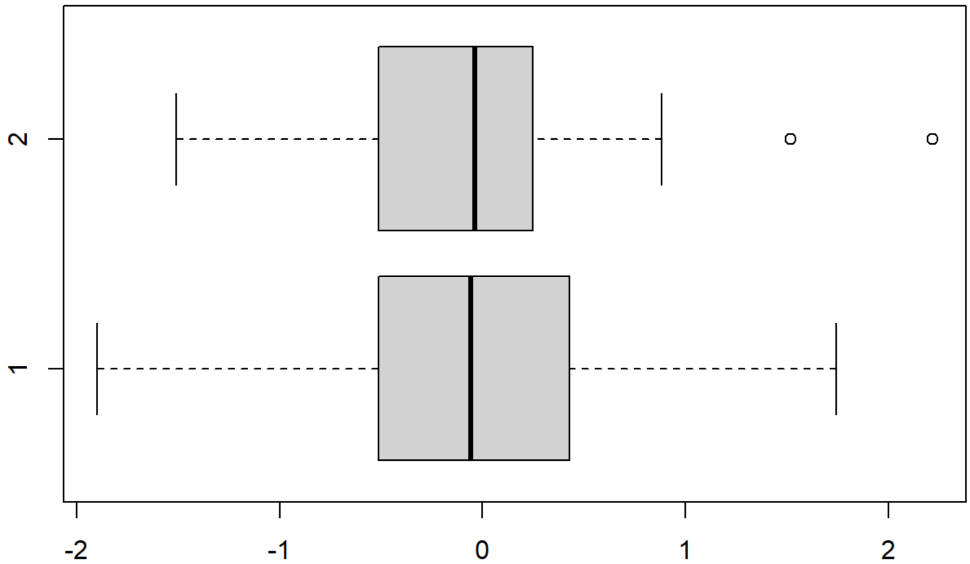
`scale` puts the two data sets on the same scale so they can sensibly be compared



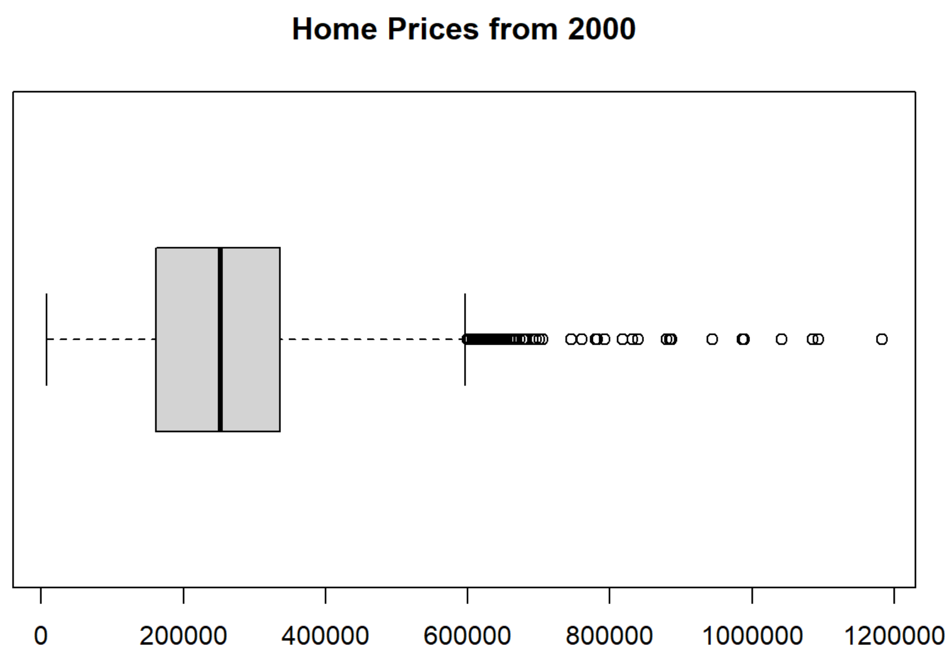

```
> head(home$old)
[1] 64200 72100 87600 59000 83200 49100
> mean(home$old)
[1] 78707
> sd(home$old)
[1] 15590
> head((home$old - mean(home$old)) / sd(home$old))
[1] -0.931 -0.424  0.570 -1.264  0.288 -1.899
> head(scale(home$old))
       [,1]
[1,] -0.931
[2,] -0.424
[3,]  0.570
[4,] -1.264
[5,]  0.288
[6,] -1.899
> head(home$new)
[1] 257500 276800 364600 160400 333500 145600
> mean(home$new)
[1] 329320
> sd(home$new)
[1] 121774
> head((home$new - mean(home$new)) / sd(home$new))
[1] -0.5898 -0.4313  0.2897 -1.3872  0.0343 -1.5087
> head(scale(home$new))
        [,1]
```

```
[1,] -0.5898
[2,] -0.4313
[3,]  0.2897
[4,] -1.3872
[5,]  0.0343
[6,] -1.5087
> boxplot(c(scale(home$old)), c(scale(home$new)),
horizontal = TRUE)
```

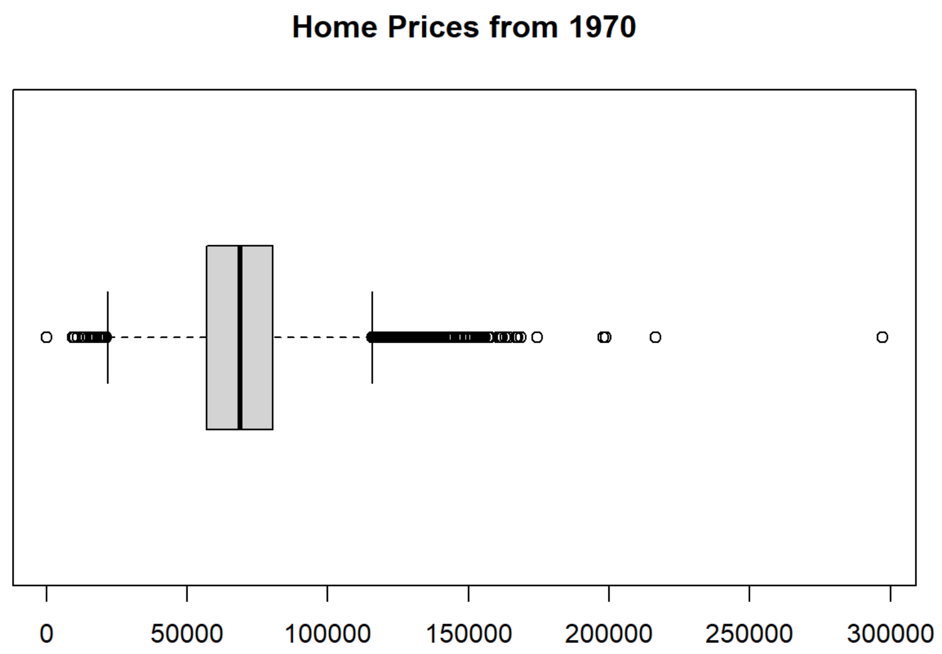
Another example with different data frame on the same house prices




```
> library(MASS)
> head(homedata)
   y1970  y2000
1  89700 359100
2 118400 504500
3 116400 477300
4 122000 500400
5  91500 433900
6 102800 464800
> boxplot(homedata$y2000, horizontal = TRUE, main = "Home
Prices from 2000")
```

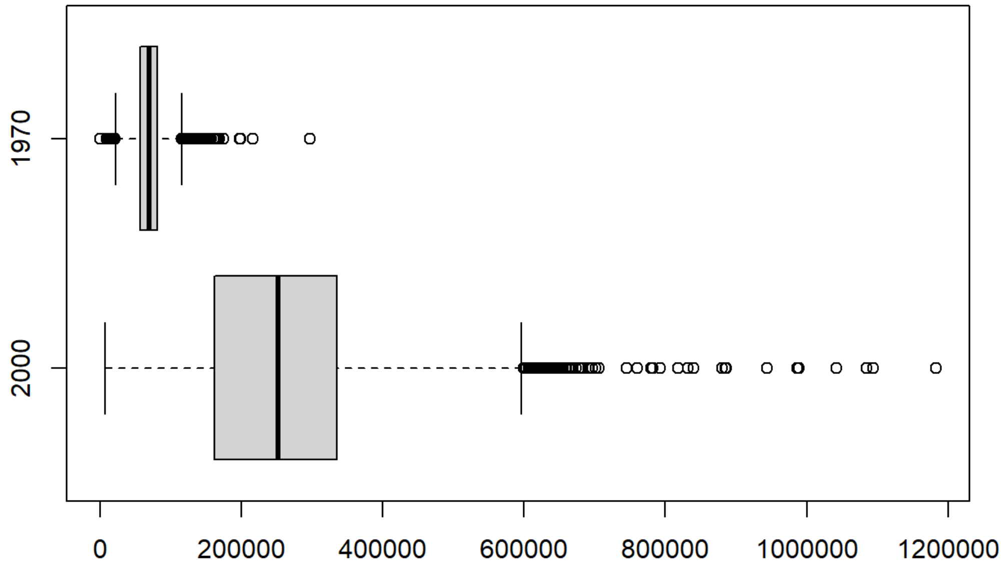



```
> boxplot(homedata$y1970, horizontal = TRUE, main = "Home
Prices from 1970")
```



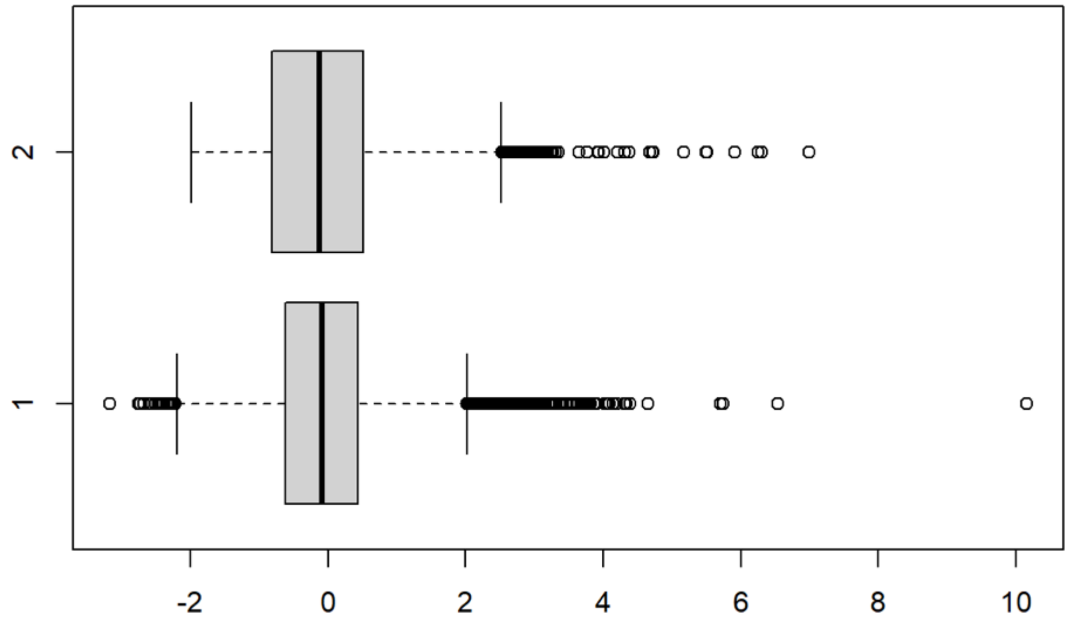

```
> boxplot(homedata$y2000, homedata$y1970, horizontal =
TRUE, names = c("2000", "1970"))
```

```
> head(homedata$y1970)
[1]  89700 118400 116400 122000  91500 102800
> mean(homedata$y1970)
```



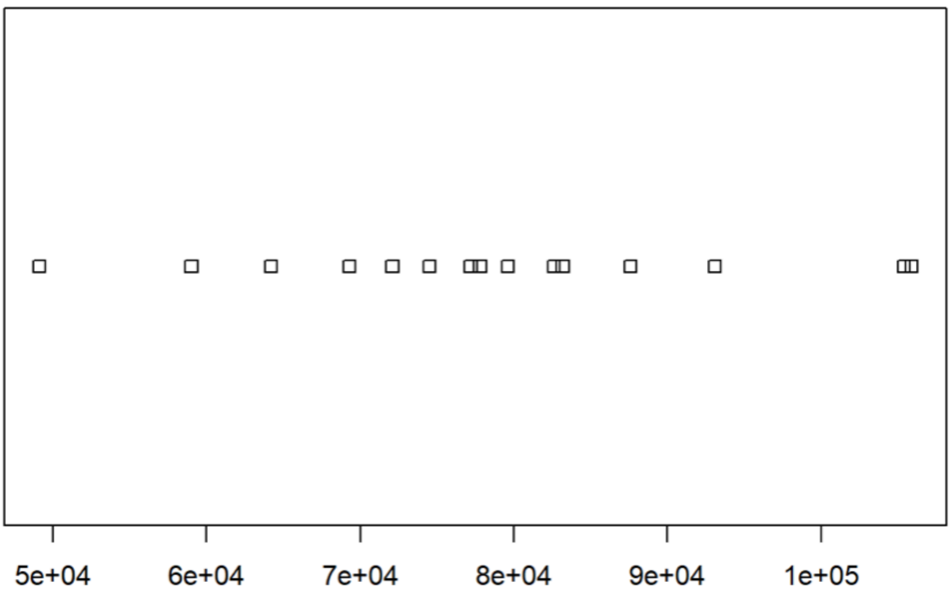

```
[1] 70821
> sd(homedata$y1970)
[1] 22284
> head((homedata$y1970 - mean(homedata$y1970)) /
sd(homedata$y1970))
[1] 0.847 2.135 2.045 2.297 0.928 1.435
> head(scale(homedata$y1970))
      [,1]
[1,] 0.847
[2,] 2.135
[3,] 2.045
[4,] 2.297
[5,] 0.928
[6,] 1.435
> head(homedata$y2000)
[1] 359100 504500 477300 500400 433900 464800
> mean(homedata$y2000)
[1] 268370
> sd(homedata$y2000)
[1] 130734
> head((homedata$y2000 - mean(homedata$y2000)) /
sd(homedata$y2000))
[1] 0.694 1.806 1.598 1.775 1.266 1.503
> head(scale(homedata$y2000))
      [,1]
```

```
[1,] 0.694
[2,] 1.806
[3,] 1.598
[4,] 1.775
[5,] 1.266
[6,] 1.503
> boxplot(c(scale(homedata$y1970)),
c(scale(homedata$y2000)), horizontal = TRUE)
```

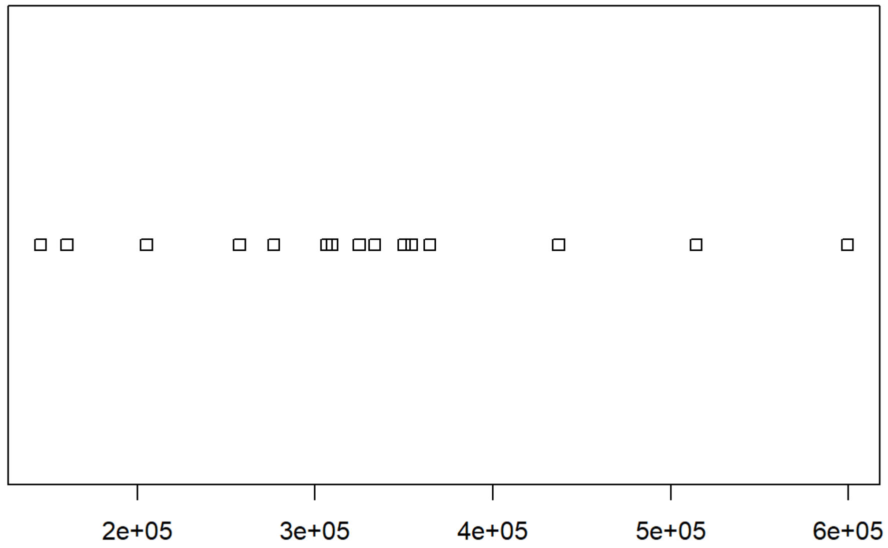


Also we can present the two distributions using stripcharts or dotplots

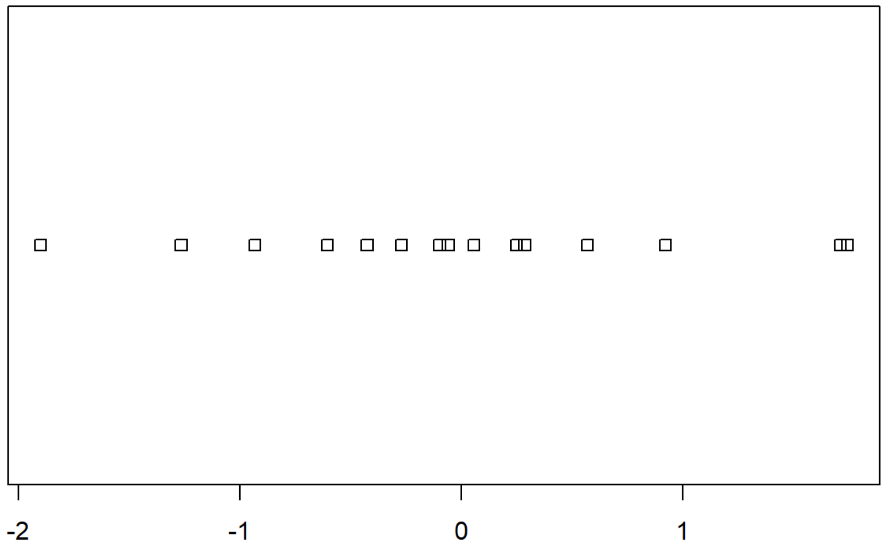
```
> stripchart(home$old)
```



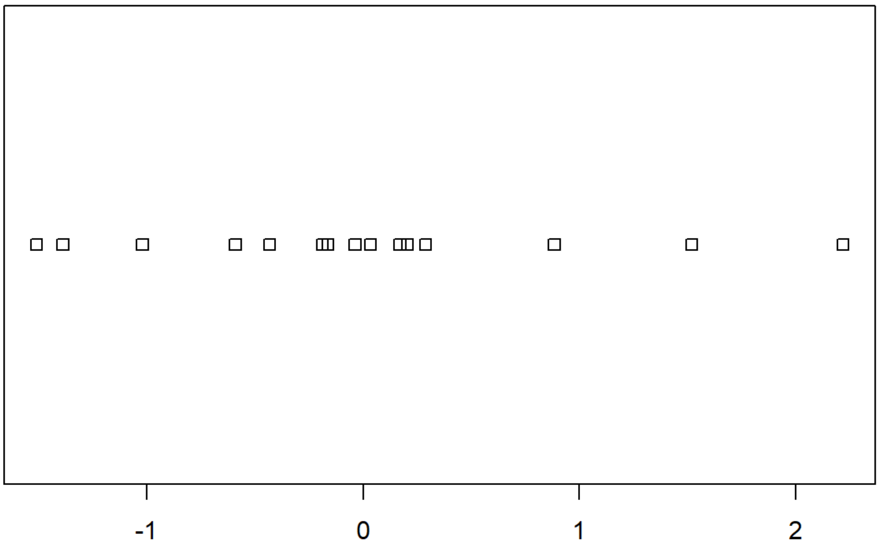
```
> stripchart(home$new)
```



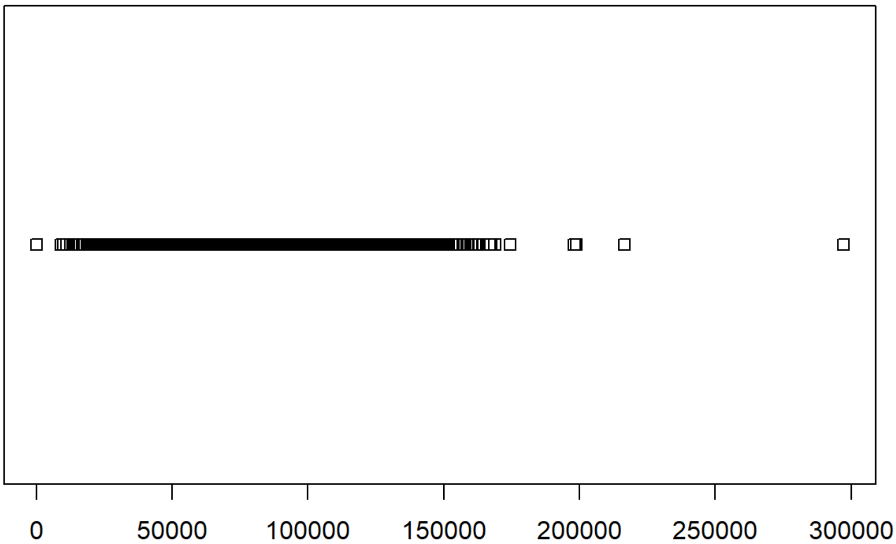

```
> stripchart(scale(home$old))
```



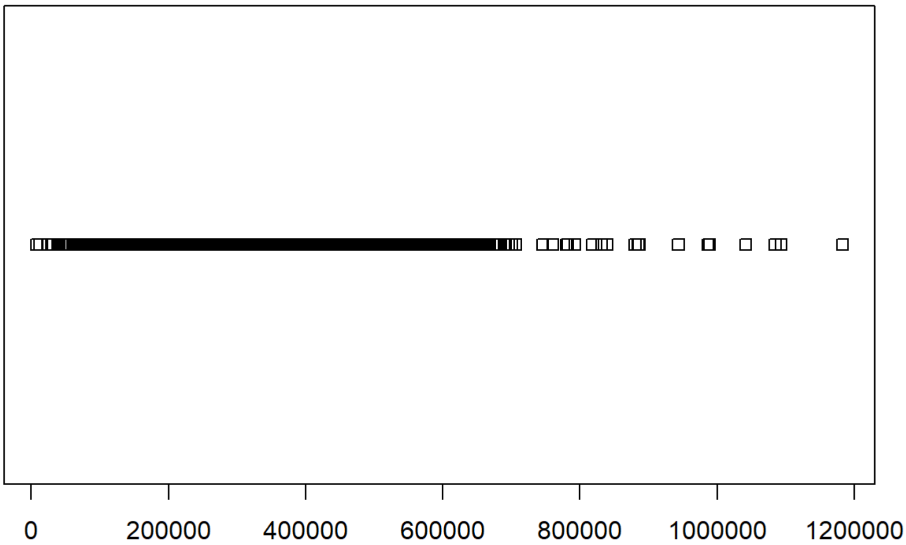

```
> stripchart(scale(home$new))
```



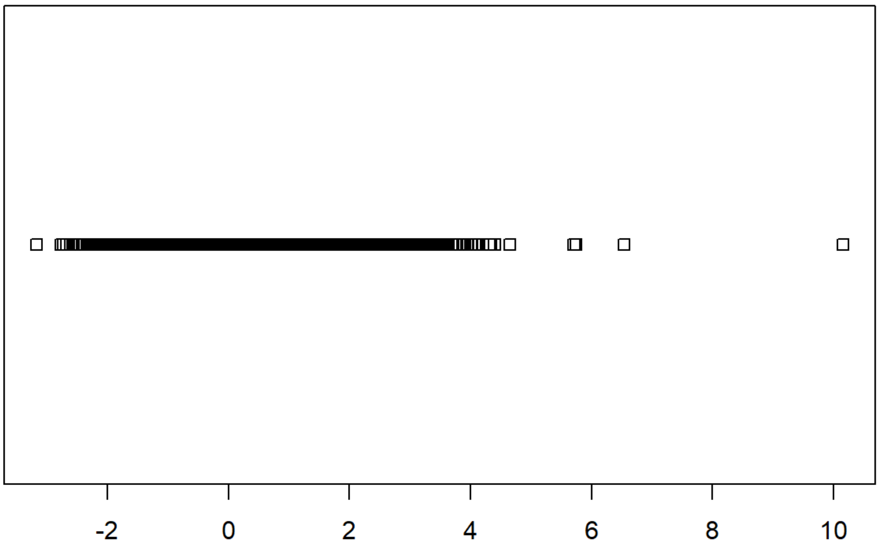

```
> stripchart(homedata$y1970)
```



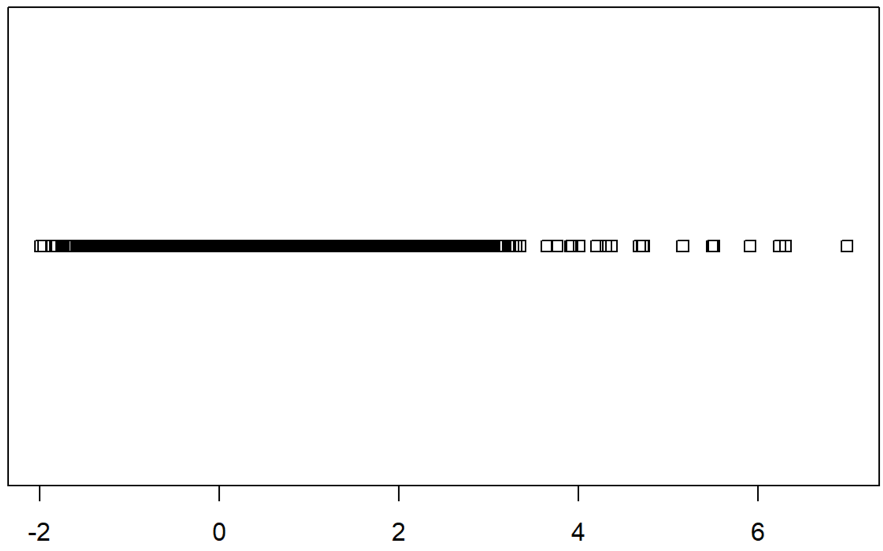

```
> stripchart(homedata$y2000)
```




```
> stripchart(scale(homedata$y1970))
```

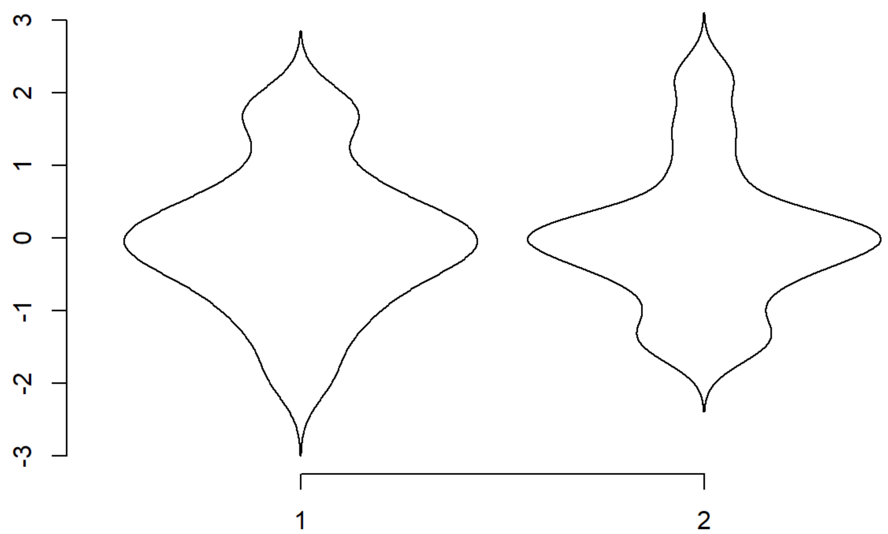


```
> stripchart(scale(homedata$y2000))
```

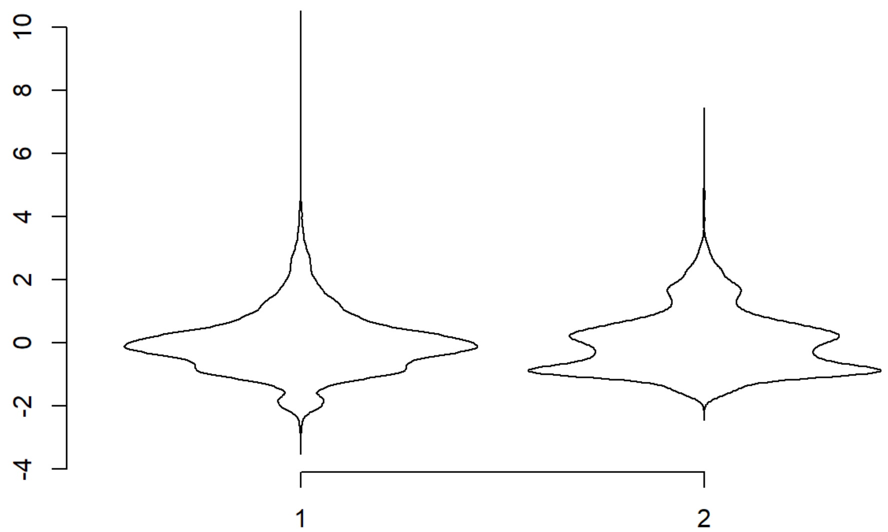


We can also compare shapes of distributions using the violin plots

```
> violinplot(scale(home$old), scale(home$new))
> simple.violinplot(scale(home$old),scale(home$new))
```



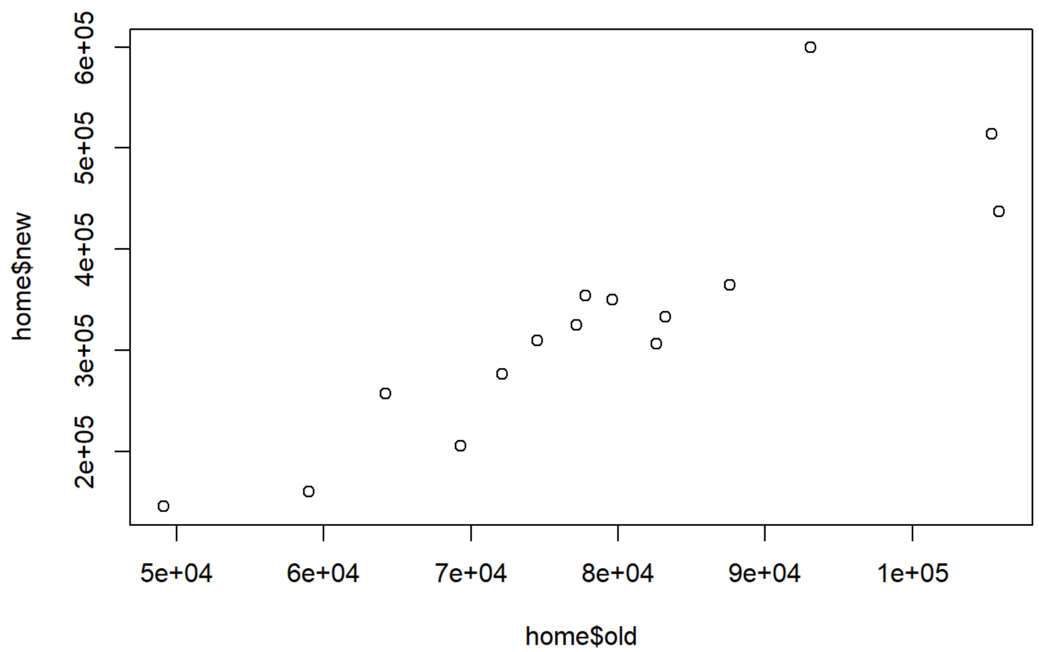
```
> violinplot(scale(homedata$y1970), scale(homedata$y2000))
> simple.violinplot(scale(homedata$y1970),scale(homedata$y2000))
```



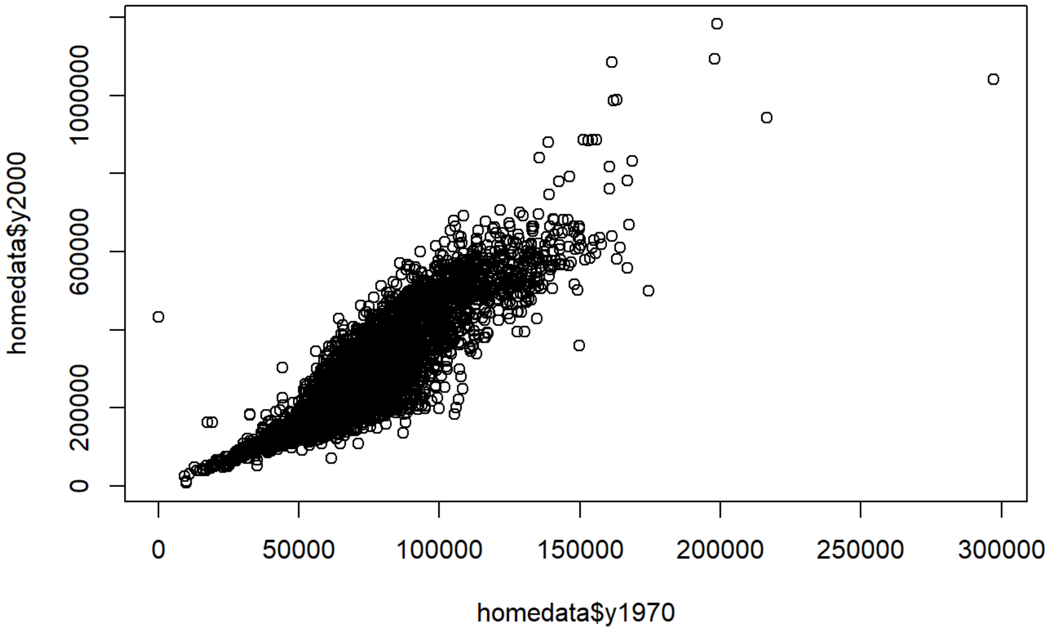
Comparing relationships

To investigate one numerical variable against another, we can use the scatter plot

```
> plot(home$old, home$new)
```



```
> plot(homedata$y1970, homedata$y2000)
```



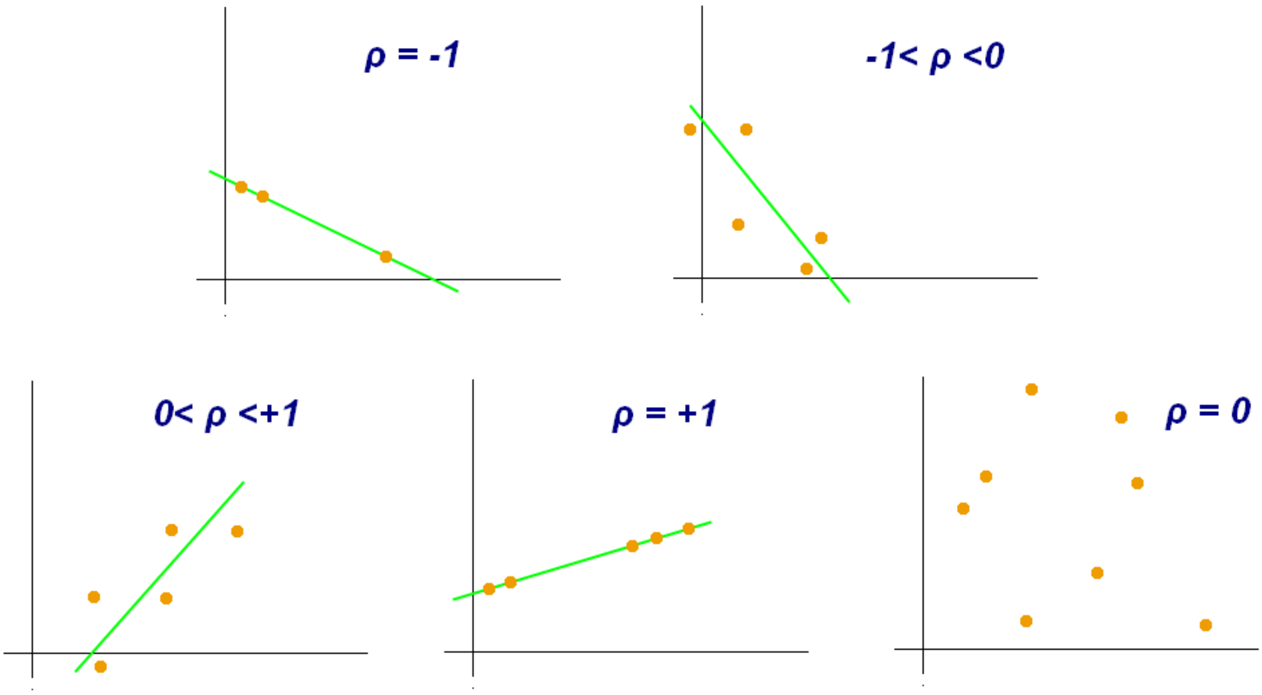
It looks like we have a strong linear trend. The modeling of such relationships is a common statistical practice. It allows us to make predictions of the y variable based on the value of the x variable.

Pearson correlation coefficient R

The Pearson correlation coefficient indicates the strength of a linear relationship between two variables.

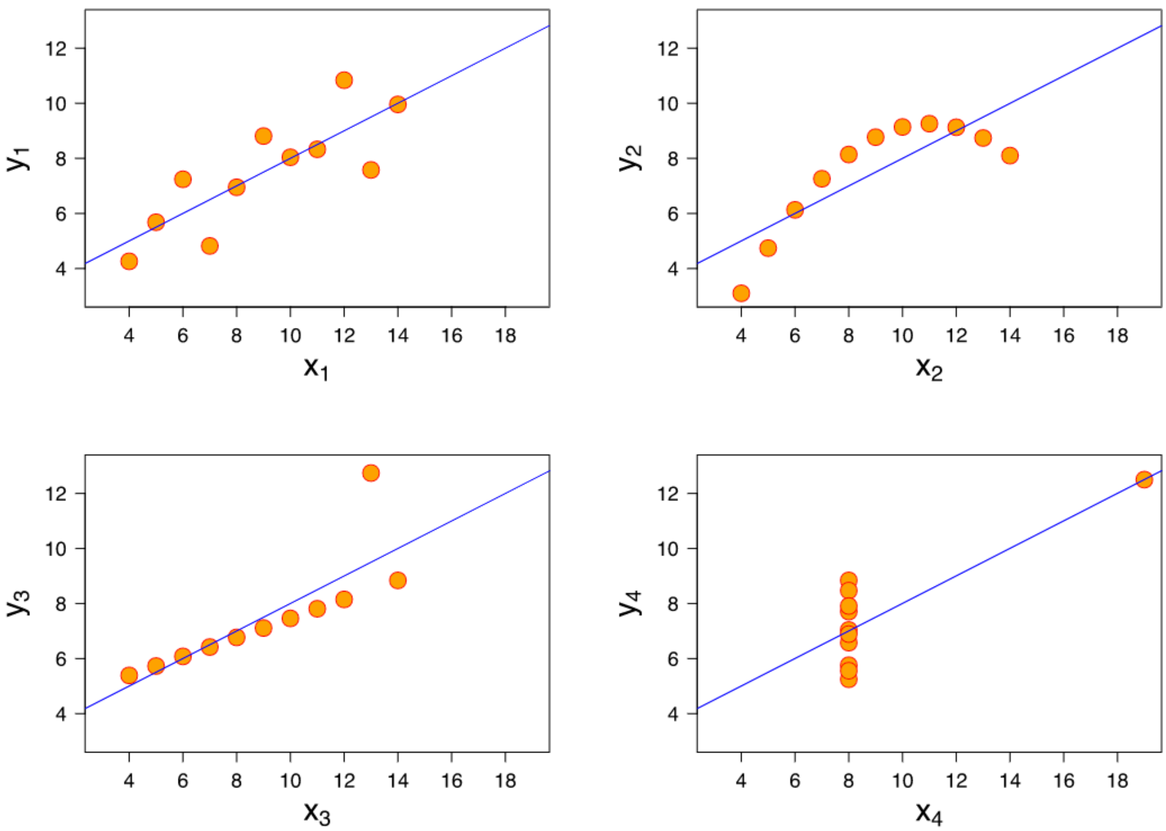
$$R = \frac{\sum (x_i - \overline{X})(y_i - \overline{Y})}{\sqrt{\sum (x_i - \overline{X})^2 \sum (y_i - \overline{Y})^2}} \in [-1,1]$$

This is a scaled covariance between X and Y. It measures how one variable varies as the other does. A value of
-1 indicates strong negative linear relationship
0 indicates no linear relationship
+1 indicates strong positive linear relationship



Pearson correlation value generally does not completely characterize their relationship.

The following image displays four data sets called Anscombe's quartet. In each case, the mean and standard deviation of each variable is the same, and the correlation between the two random variables is 0.816. However, the data look very different when graphed.



Time for some practice. Let's play some funny online games to predict correlations Game 1, Game 2

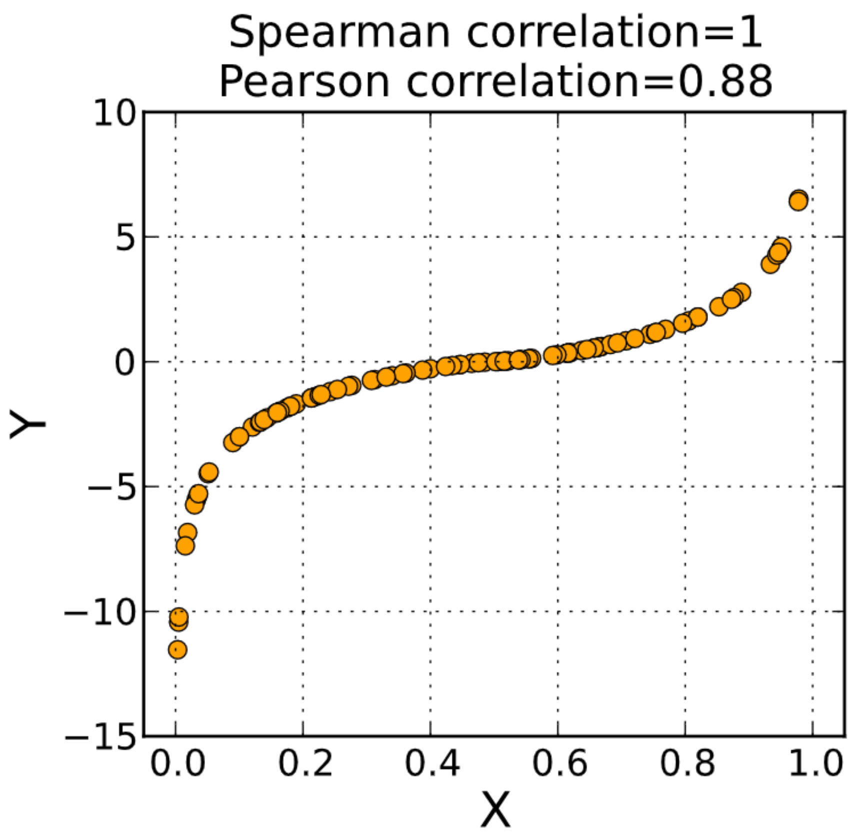
```
> cov(home$old, home$new)
[1] 1.67e+09
> cov(homedata$y1970, homedata$y2000)
[1] 2.61e+09
> cor(home$old, home$new)
[1] 0.881
> cor(homedata$y1970, homedata$y2000)
[1] 0.896
> cor(home$old, home$new)^2
[1] 0.776
> cor(homedata$y1970, homedata$y2000)^2
[1] 0.803
```

Spearman rank correlation

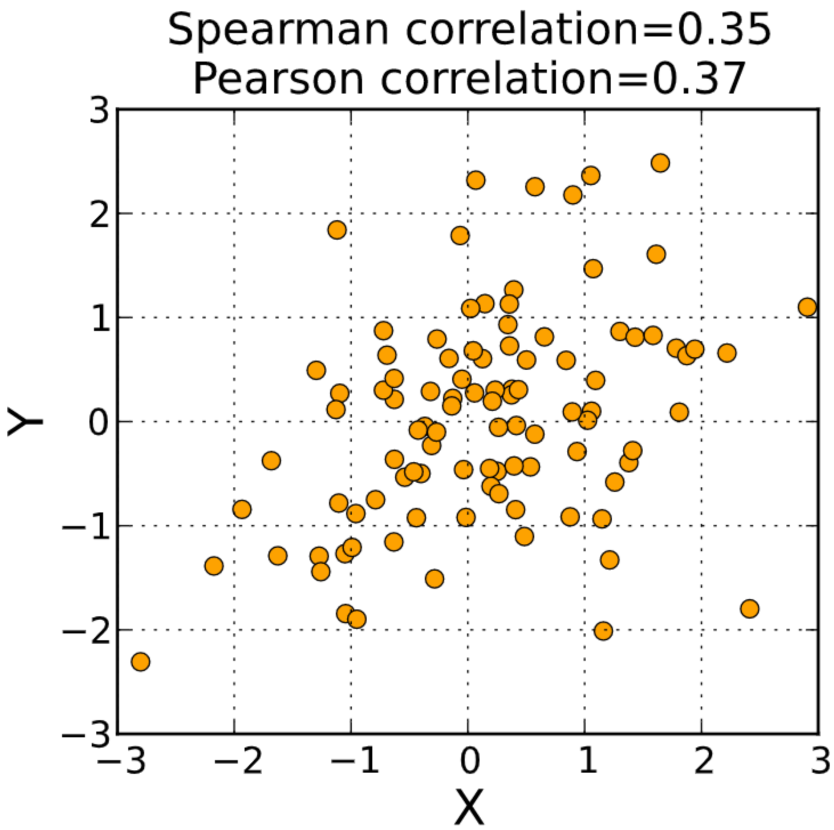
Same thing only applied to the ranks of the data.

```
> cor(rank(home$old), rank(home$new))
[1] 0.925
> cor(rank(homedata$y1970), rank(homedata$y2000))
[1] 0.888
```

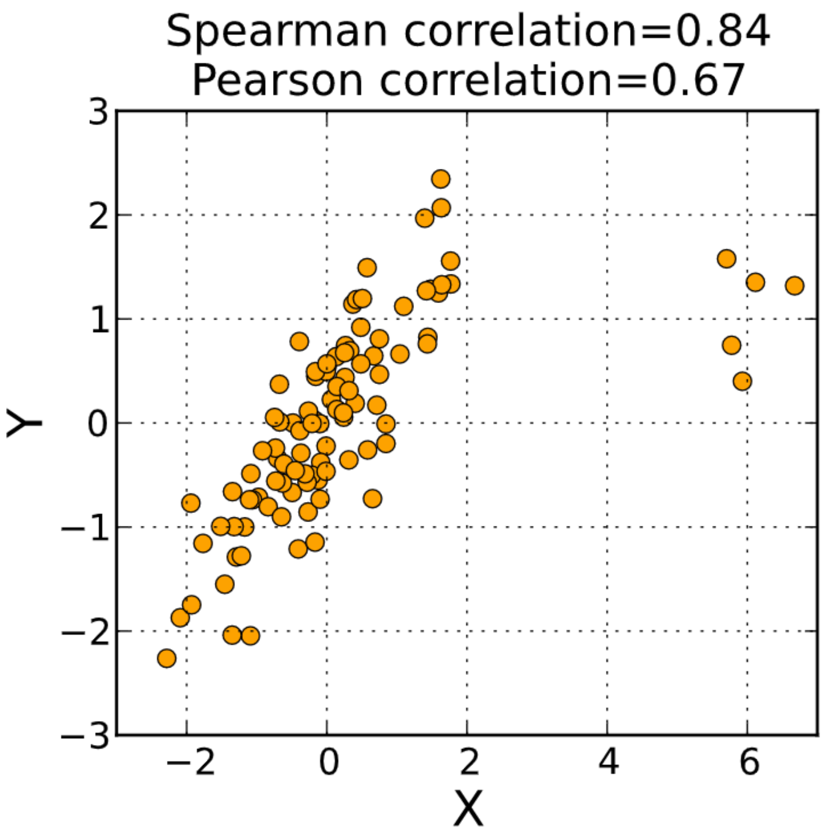
Spearman rank correlation is close to 1 (or -1) if there is a strong increasing (decreasing) trend in the data. (The trend need not be linear.)



When the data are roughly elliptically distributed and there are no prominent outliers, the Spearman correlation and Pearson correlation give similar values.



The Spearman correlation is less sensitive than the Pearsin correlation to strong outliers that are in the tails of both samples.



Simple linear regression

Linear regression is the name of a procedure that fits a straight line to the data. The idea is that the x value is something the experimenter controls, the y value one the experimenter measures. The line is used to predict the value of y for a known value of x. The variable x is the **predictor** variable and y the **response** variable.

Suppose we write the equation of the line as

$$\hat{y} = b_0 + b_1 x$$

Then, for each x_i the predicted value would be

$$\hat{y} = b_0 + b_1 x_i$$

But the measured value is y_i, the difference is called the **residual** and is simply

$$\varepsilon_i = y_i - \hat{y}_i$$

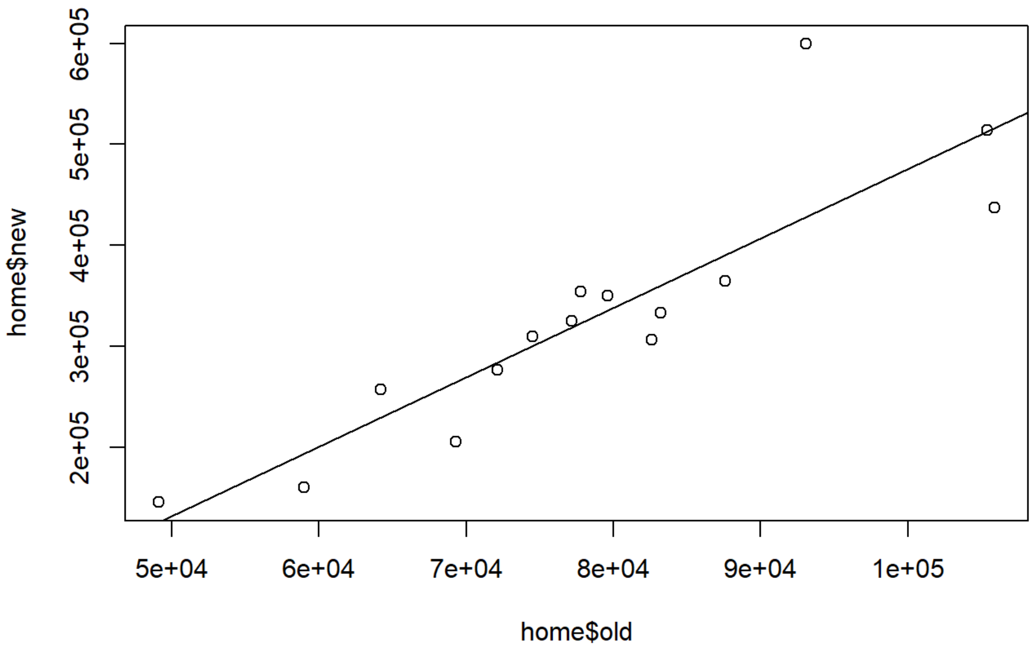
The method of least square is used to choose the values of b_0 and b_1 that minimize the sum or the squares of the residual errors $\sum (y_i - \hat{y}_i)^2$

Solving, gives

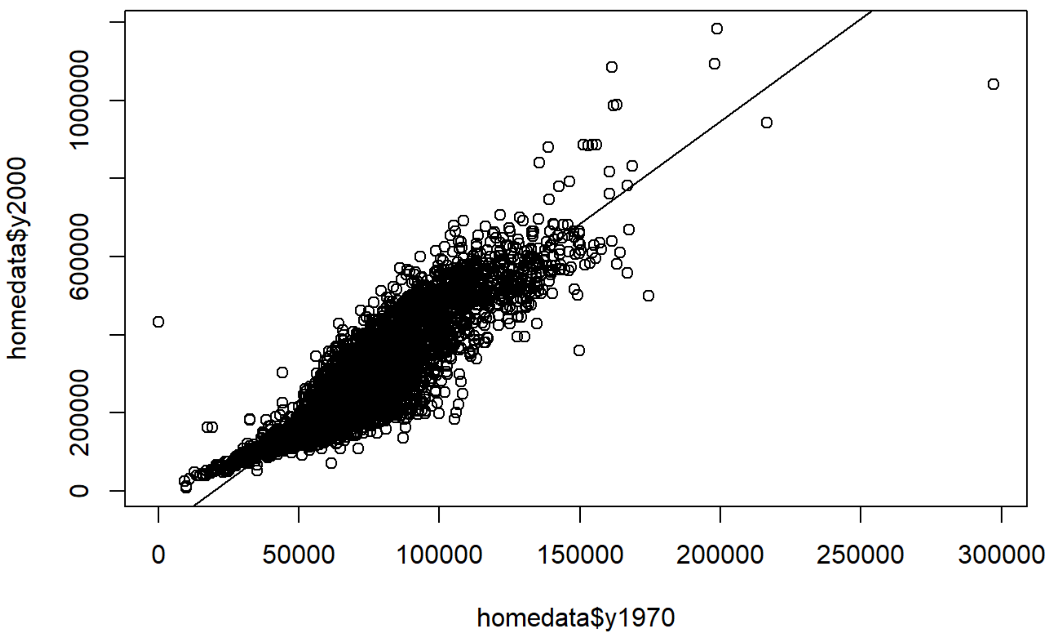
$$b_0 = \overline{Y} - b_1 \overline{X}, \;\; b_1 = \frac{s_{xy}}{s_x^2} = \frac{\sum (x_i - \overline{X})(y_j - \overline{Y})}{\sum (x_i - \overline{X})^2}$$

This is, a line with slope given by b_1.

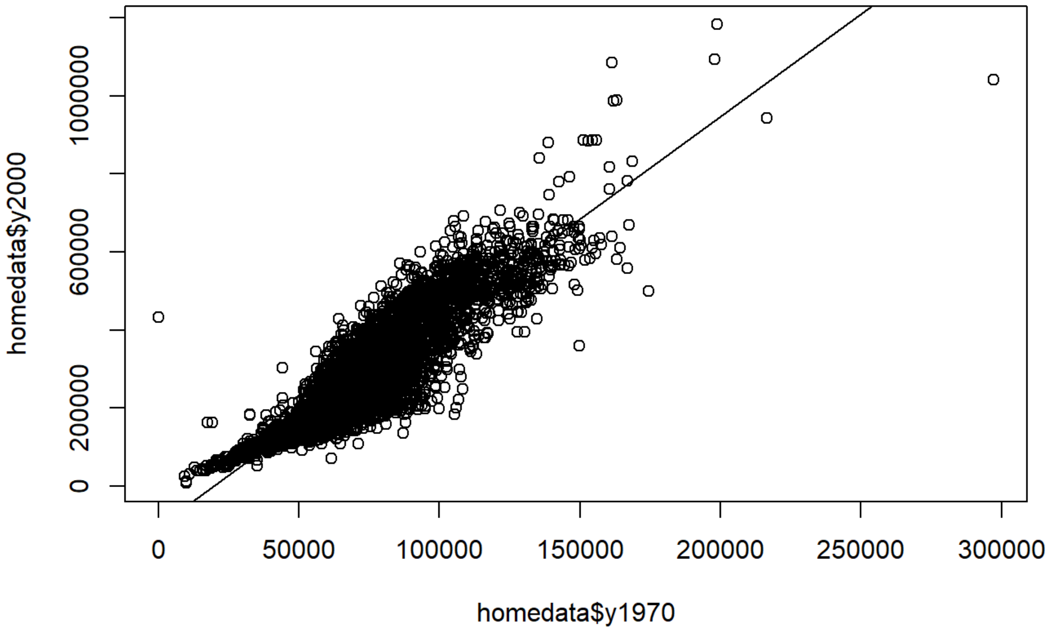
```
> plot(home$old, home$new)
> abline(lm(home$new ~ home$old))
```




```
> plot(homedata$y1970, homedata$y2000)
> abline(lm(homedata$y2000 ~ homedata$y1970))
```

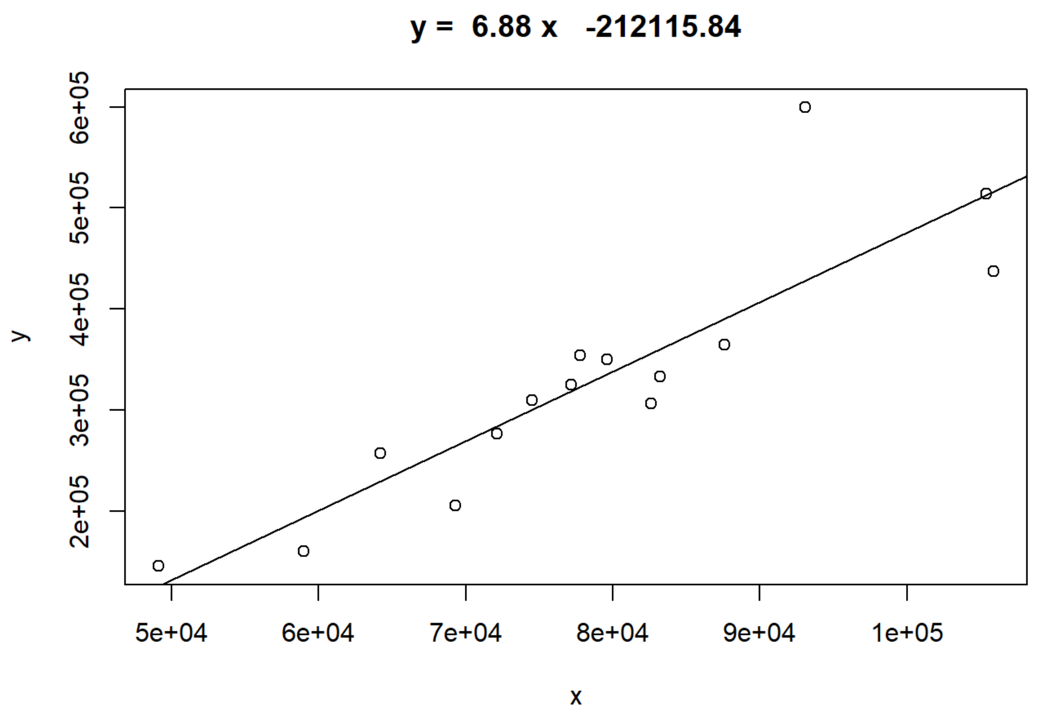



```
> plot(homedata$y2000 ~ homedata$y1970)
> abline(lm(homedata$y2000 ~ homedata$y1970))
```



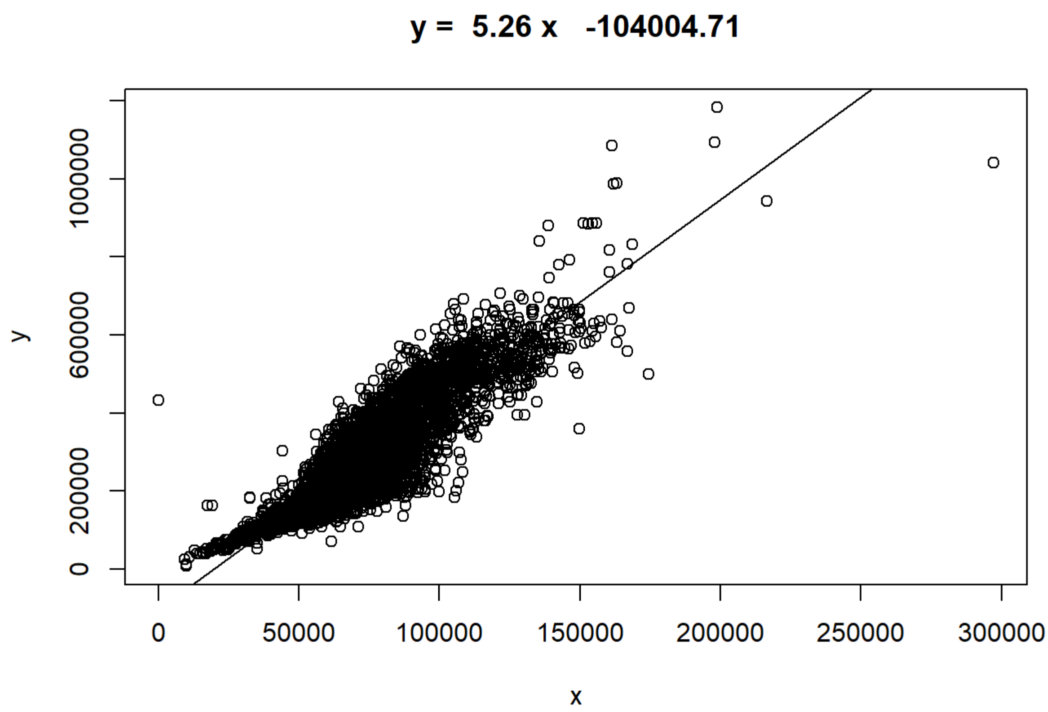
`abline` function points lines on the current graph window
`lm` function for the linear model
The syntax `y ~ x` tells R to model the y variable as a linear function of x `simple.lm` function will make the same plot and return the regression coefficients

```
> simple.lm(home$old, home$new)
```



```
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
  -2.12e+05     6.88e+00
```

```
> pl <- simple.lm(homedata$y1970, homedata$y2000)
```



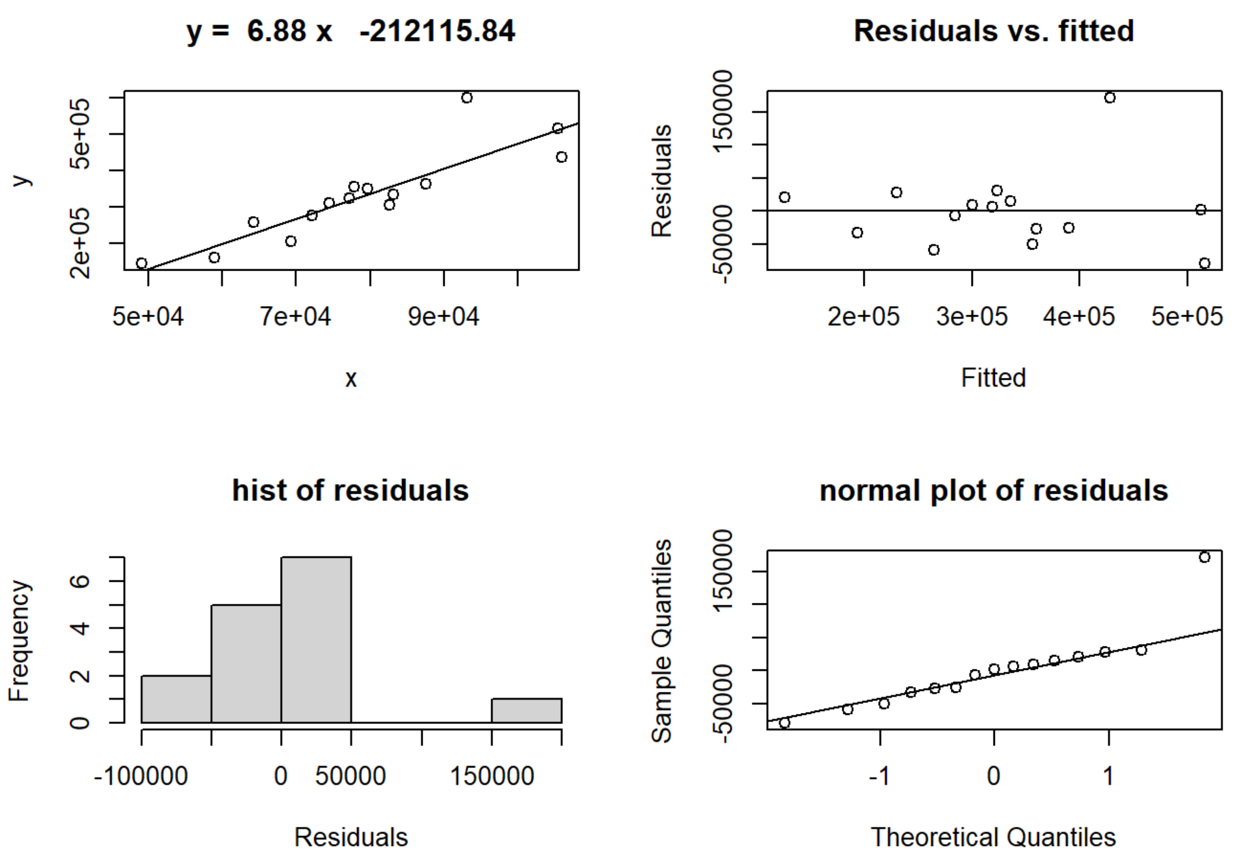

```
> attributes(pl)
$names
 [1] "coefficients"  "residuals"     "effects"
"rank"
 [5] "fitted.values" "assign"        "qr"
"df.residual"
 [9] "xlevels"       "call"          "terms"
"model"

$class
[1] "lm"
```

So we can directly access the `coefficients`, `residuals` and other parameters

```
> pl$coefficients
(Intercept)           x
  -1.04e+05    5.26e+00
> head(pl$residuals)
     1       2       3       4       5       6
536605   80331   58825   64125   77867   85328
```

Residual plot

```
> simple.lm(home$old, home$new, show.residuals = TRUE)
```



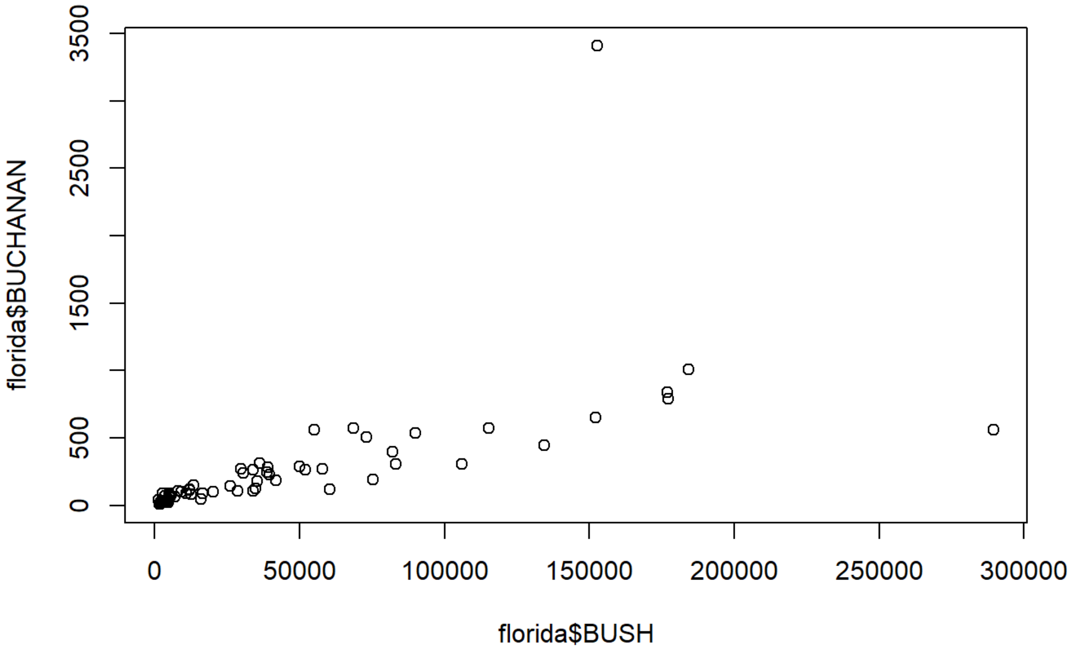

```
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
  -2.12e+05     6.88e+00
```

The lower left is a histogram of the residuals. If the standard model is applicable, then this should appear “bell” shaped.

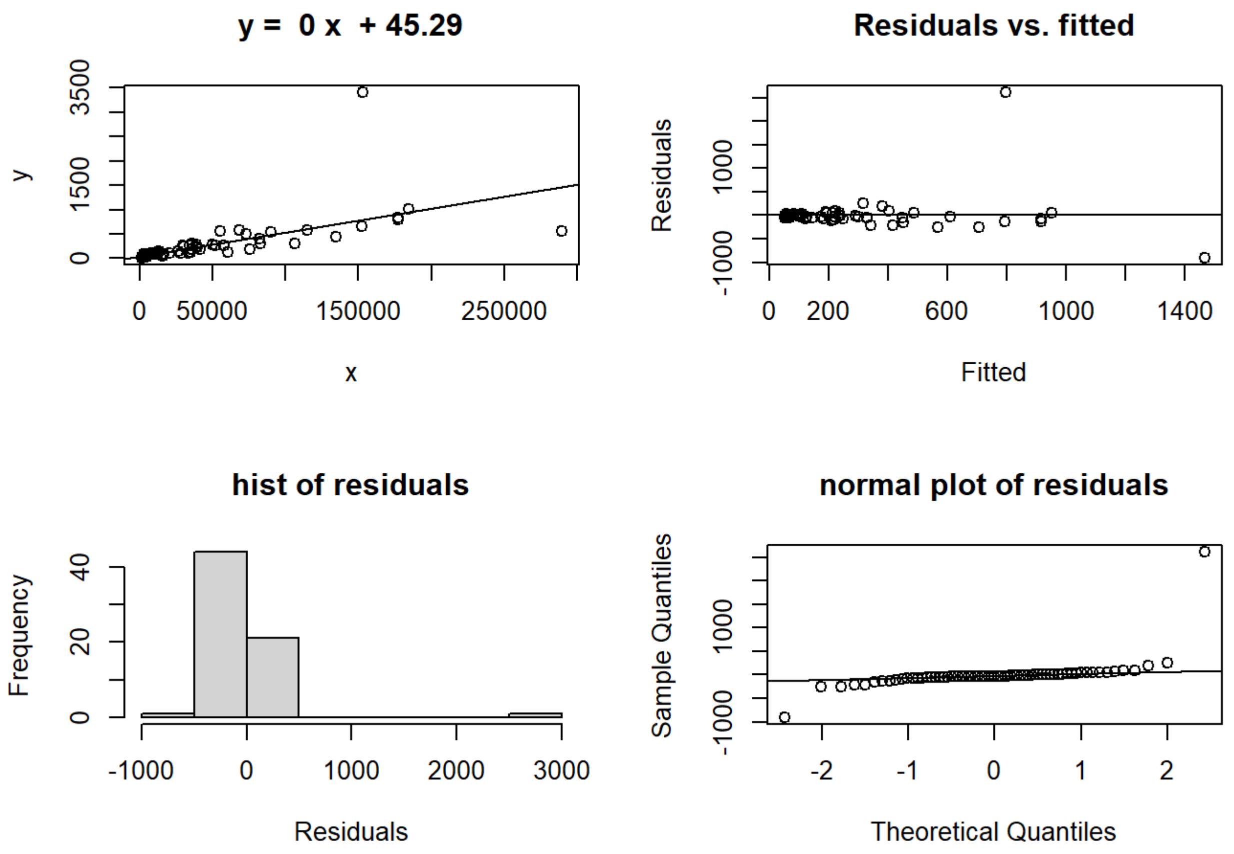
Locating points

Consider the `florida` data frame containing results of year 2000 US presidential election in Florida. Let's review the `BUSH` and `Buchanan` votes.

```
> head(florida)
    County   GORE   BUSH BUCHANAN NADER BROWN HAGELIN
HARRIS MCREYNOLDS
1  ALACHUA  47300  34062      262  3215   658      42
4          658
2    BAKER   2392   5610       73    53    17       3
0            0
3      BAY  18850  38637      248   828   171      18
5            3
4 BRADFORD   3072   5413       65    84    28       2
0            0
5  BREVARD  97318 115185      570  4470   643      39
11          11
6  BROWARD 386518 177279      789  7099  1212     128
49          35
  MOOREHEAD PHILLIPS  Total
1        21       20  86242
2         3        3   8154
3        37       18  58815
4         3        2   8669
5        76       72 218395
6       123       74 573306
> cor(florida$BUSH, florida$BUCHANAN)
[1] 0.624
> cor(rank(florida$BUSH), rank(florida$BUCHANAN))
[1] 0.944
> plot(florida$BUSH, florida$BUCHANAN)
```




```
> simple.lm(florida$BUSH, florida$BUCHANAN)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
   45.28986      0.00492
> simple.lm(florida$BUSH, florida$BUCHANAN,
show.residuals = TRUE)
```

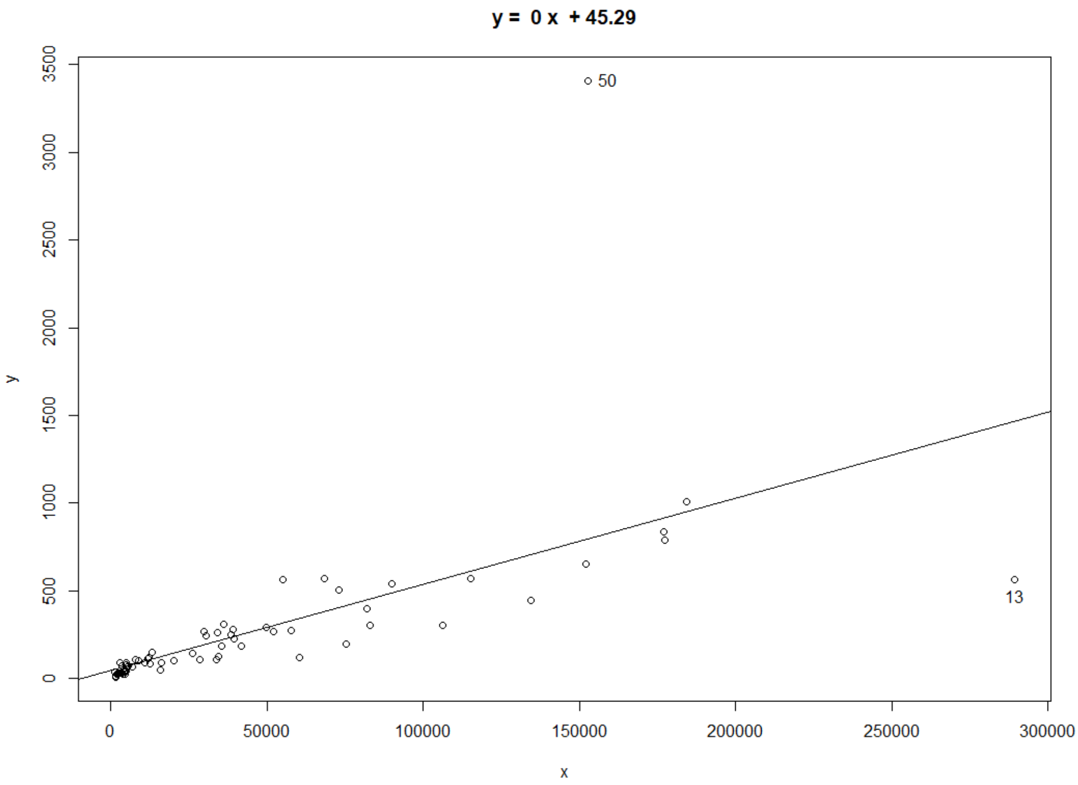



```
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
   45.28986      0.00492
```

We see a strong linear relationship, except for two "outliers". How can we identify these points? `identify` find the index of the closest (x, y) coordinates to the mouse click.

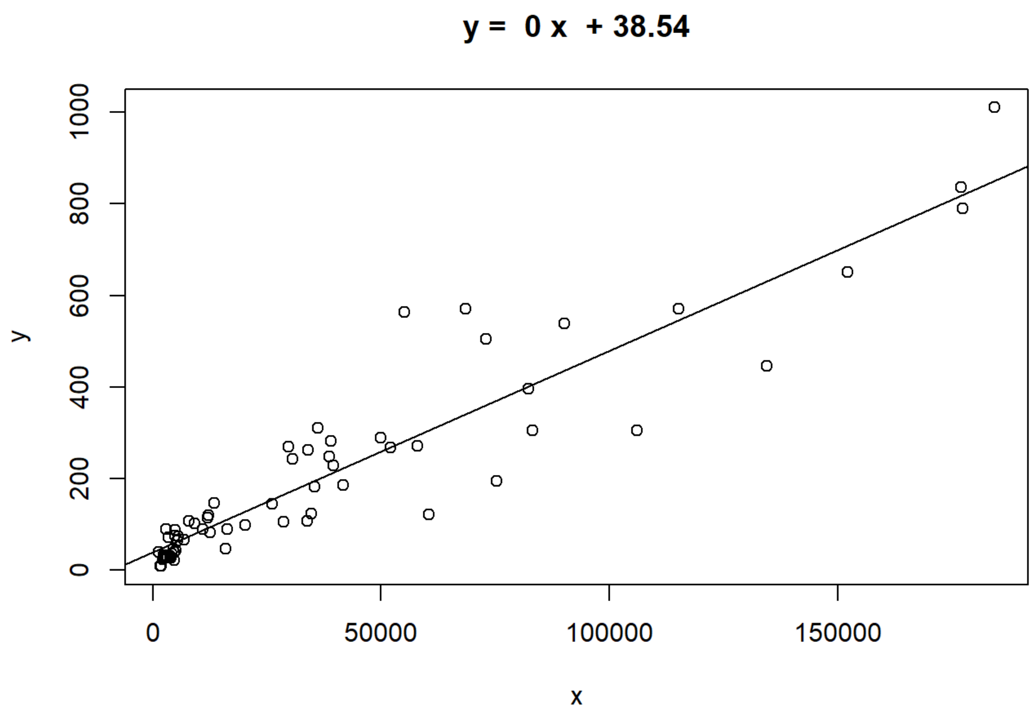
```
> simple.lm(florida$BUSH, florida$BUCHANAN)
> identify(florida$BUSH, florida$BUCHANAN)
```

Click on the two outliers and then press 'Esc' to finish and print the identified coordinates.

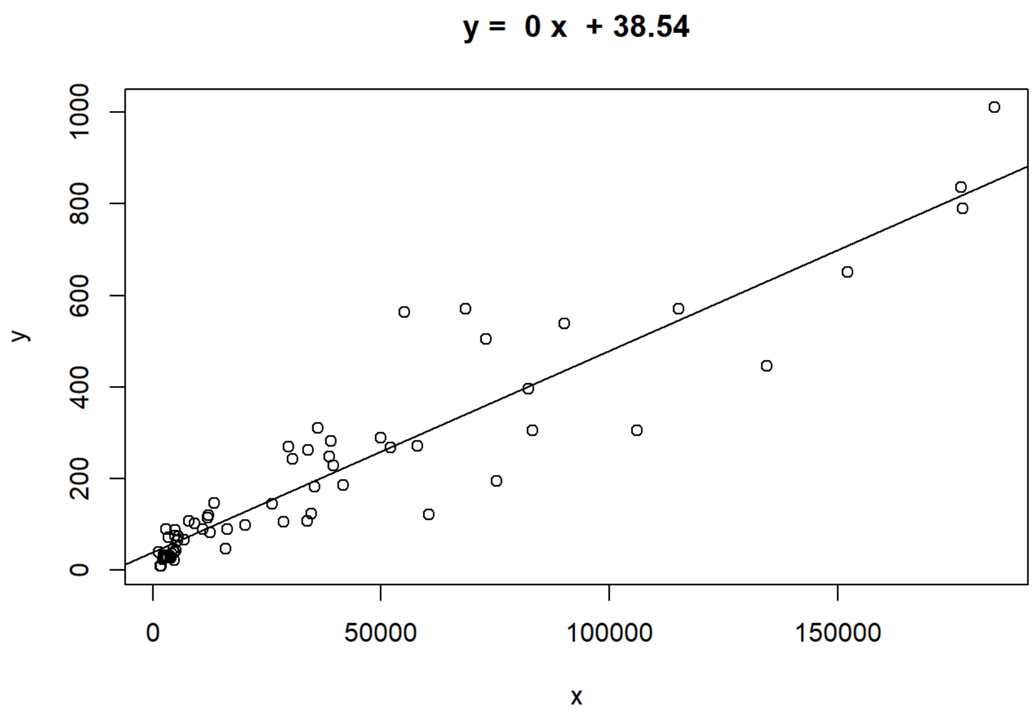


Let's see what will happen when we remove this observations.

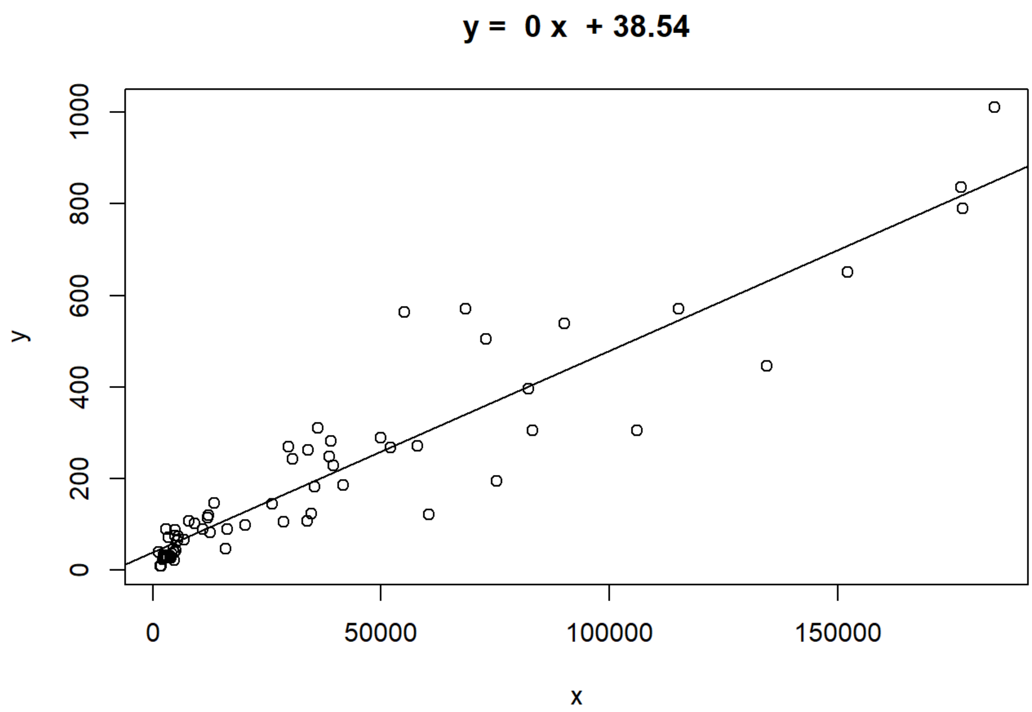
```
> florida[c(13,50),]
         County   GORE   BUSH BUCHANAN NADER BROWN HAGELIN
HARRIS MCREYNOLDS
13         DADE 328702 289456      561  5355   759     119
88           36
50 PALM BEACH 268945 152846     3407  5564   743     143
45          302
   MOOREHEAD PHILLIPS  Total
13       124       69 625269
50       103      188 432286
> florida.cleaned <- florida[-c(13, 50), ]
> linearmodel <- simple.lm(florida.cleaned$BUSH,
florida.cleaned$BUCHANAN)
```




```
> linearmodel$coefficients[1]
(Intercept)
       38.5
> linearmodel$coefficients[2]
     x
0.0044
> bush.palm.beach <- 152846
> buchanan.palm.beach <- linearmodel$coefficients[1] +
linearmodel$coefficients[2] * bush.palm.beach
> buchanan.palm.beach
(Intercept)
        712
> simple.lm(florida.cleaned$BUSH,
florida.cleaned$BUCHANAN, pred = florida$BUSH[50])
```



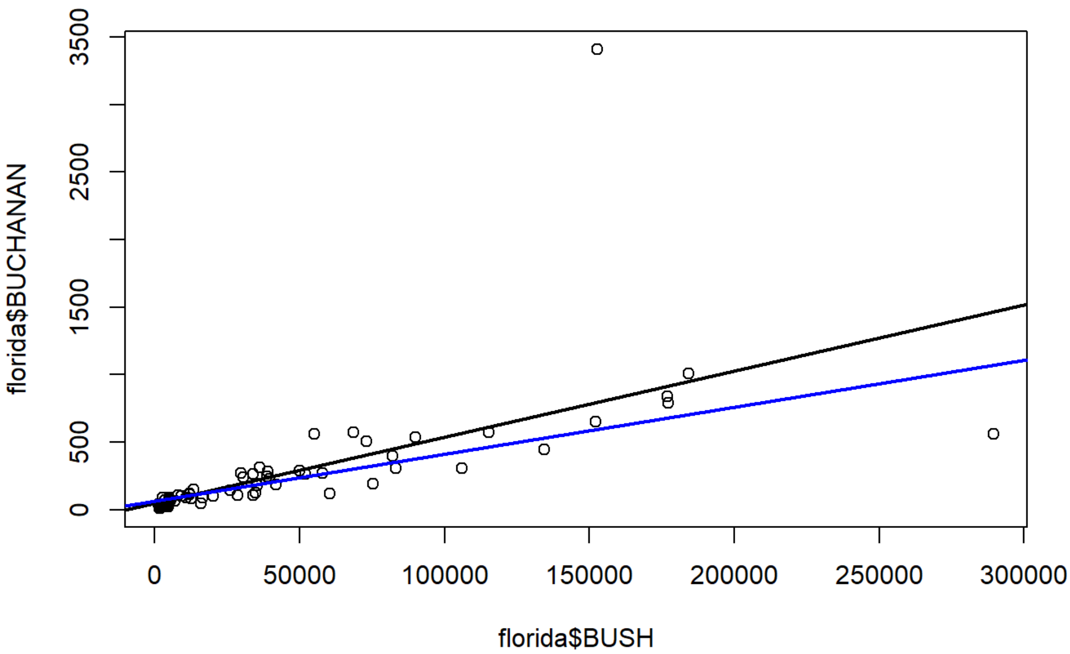

```
  1
712

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
    38.5363       0.0044

> bush.dade <- 289456
> buchanan.dade <- linearmodel$coefficients[1] +
linearmodel$coefficients[2] * bush.dade
> buchanan.dade
(Intercept)
       1313
> simple.lm(florida.cleaned$BUSH,
florida.cleaned$BUCHANAN, pred = florida$BUSH[13])
```




```
   1
1313

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
    38.5363       0.0044
```

We expect Buchanan to have received 712 votes in Palm Beach and 1313 in Dade, not the actual received.

```
> plot(florida$BUSH, florida$BUCHANAN)
> abline(lm(florida$BUCHANAN ~ florida$BUSH), lwd = 2)
> abline(65.6, 0.00348, col = "Blue", lwd = 2)
```


florida$BUCHANAN
0
500
1500
2500
3500
0
50000
100000
150000
200000
250000
300000
florida$BUSH