BIO 754 - Lecture 13

18-05-2017

Contents

Correlation analysis

Before starting studying regression we will remind ourselves how correlation analysis works. Like the simplest case of linear regression, correlation analysis also studies the relationship between two continuous variables.

We can start by using a dataset by Julian Faraway, provided in the library `faraway`:

```
library(faraway)
str(stat500)
```

```
## 'data.frame':    55 obs. of  4 variables:
##  $ midterm: num  24.5 22.5 23.5 23.5 22.5 16 27.5 22.5 25 30 ...
##  $ final  : num  26 24.5 26.5 34.5 30.5 31 33.5 31 29.5 37.5 ...
##  $ hw     : num  28.5 28.2 28.3 29.2 27.3 27.5 29.7 29 27.3 27.2 ...
##  $ total  : num  79 75.2 78.3 87.2 80.3 74.5 90.7 82.5 81.8 94.7 ...
```

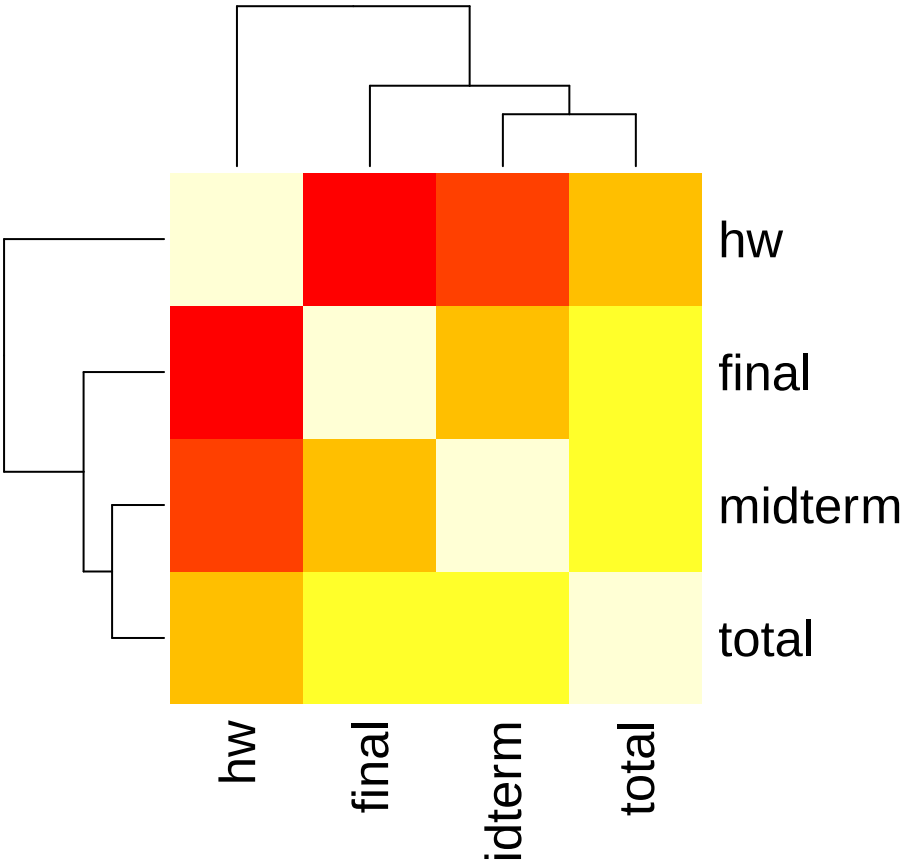
```
# midterm, final and homework grades of stat students
head(stat500)
```

```
##   midterm final   hw total
## 1    24.5  26.0 28.5  79.0
## 2    22.5  24.5 28.2  75.2
## 3    23.5  26.5 28.3  78.3
## 4    23.5  34.5 29.2  87.2
## 5    22.5  30.5 27.3  80.3
## 6    16.0  31.0 27.5  74.5
```

```
# correlation among the columns
cor(stat500)
```

```
##            midterm      final         hw     total
## midterm 1.0000000 0.54522775 0.27205756 0.8444568
## final   0.5452277 1.00000000 0.08733764 0.7788629
## hw      0.2720576 0.08733764 1.00000000 0.5644286
## total   0.8444568 0.77886293 0.56442864 1.0000000
```

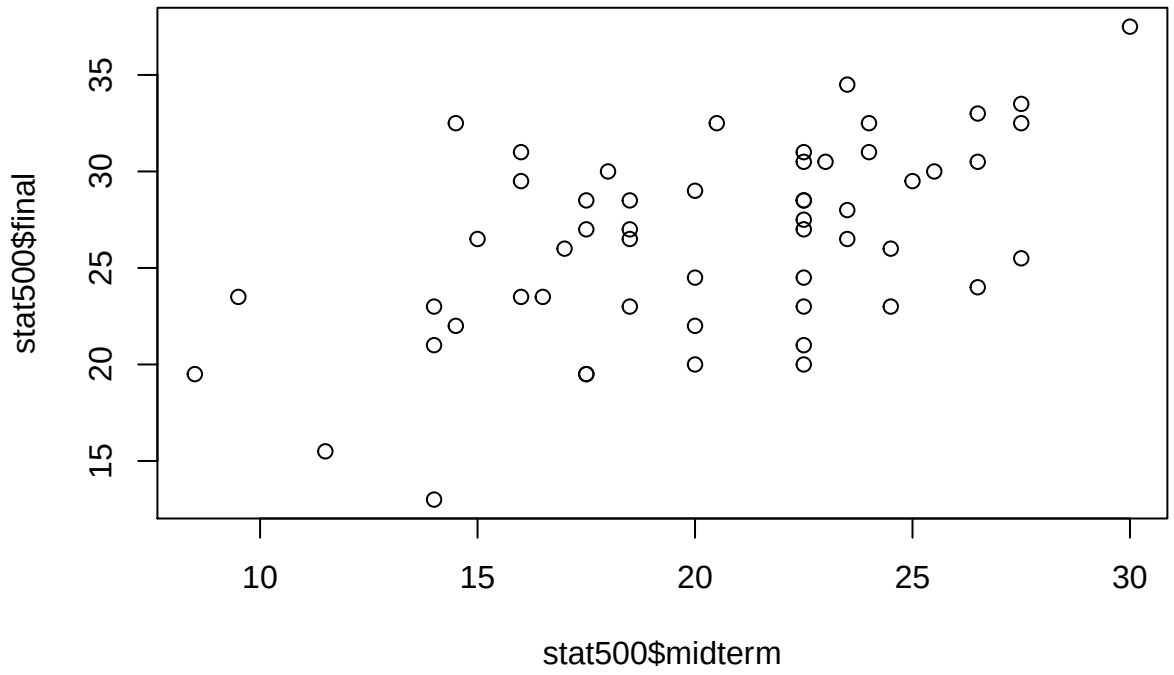
```
# summarise this in a heatmap
# tell heatmap to treat data symmetrically
heatmap(cor(stat500), symm = T)
```



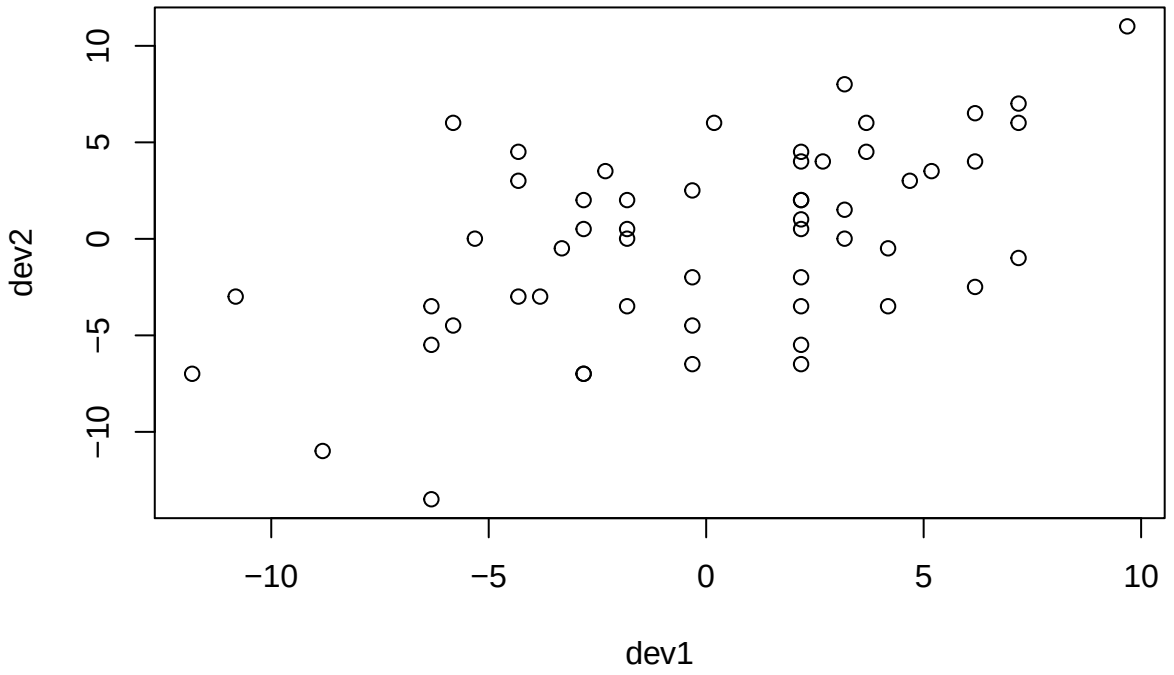
The Pearson correlation coefficient is calculated by dividing the covariance with the variance. The covariance is sum of the **product of deviations from the mean** for two variables. It thus measures if two variables are changing in the same (or in the opposite) direction.

Let's study this using the midterm and final data for the 55 students:

```
plot( stat500$midterm, stat500$final )
```




```
# deviations for midterm results
dev1 = stat500$midterm - mean(stat500$midterm)
# deviations for final results
dev2 = stat500$final - mean(stat500$final)
plot( dev1, dev2 )
```




```
# product of deviations
# will be positive if both midterm and final grades deviate in the same direction
# i.e. a student is bad in both or good in both
dev1 * dev2
```

```
##  [1]  -2.05289256  -4.34380165   0.02892562  25.48347107   8.74710744
```

```
##  [6] -19.47107438  50.33801653   9.83801653  14.08801653 106.58801653
## [11]  19.70165289  11.41983471 -11.98016529 -14.16198347  96.91983471
## [16]   2.06528926  16.60165289  32.35619835   1.42892562  24.78347107
## [21]  -3.65289256   6.34710744 -34.96198347  -0.01652893  -5.66198347
## [26]  10.75165289  34.69256198  -7.11652893 -14.59834711   4.38347107
## [31]  -0.79834711  -1.43471074  19.70165289   2.20165289 -15.39834711
## [36]  -0.92561983   1.09256198  26.12892562 -12.99380165  22.12438017
## [41]  12.91528926   4.38347107   0.63347107   1.62892562  22.05619835
## [46]  18.18347107  85.23801653  82.61983471  43.15619835  -8.13471074
## [51]   1.11074380  -0.04834711  -7.61652893  40.23801653   4.80165289
```

```
# sum of the product of deviations
# which is the covariance term
cov = sum( dev1 * dev2 )
cov
```

```
## [1] 699.4091
```

```
# the variance term
var = sqrt( sum(dev1^2) ) * sqrt( sum(dev2^2) )
var
```

```
## [1] 1282.783
```

```
cov/var
```

```
## [1] 0.5452277
```

```
cor( stat500$midterm, stat500$final )
```

```
## [1] 0.5452277
```

How do test the significance of the Pearson correlation coefficient, or r? In other words, how do we test the H0 that r is `0`.

We can use the `cor.test` function, which uses the t-test. Given r and n (sample size) it calculates the standard error for r and a t-testistic from these values. The degrees of freedom is `n-2`.

```
# check the output and make sure you understand the components
cor.test(stat500$midterm, stat500$final)
```

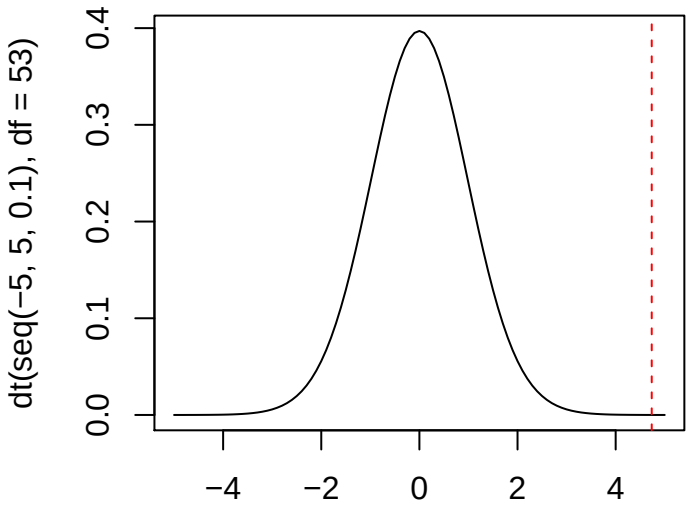
```
##
##  Pearson's product-moment correlation
##
## data:  stat500$midterm and stat500$final
## t = 4.735, df = 53, p-value = 1.675e-05
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3272692 0.7081001
## sample estimates:
##       cor
## 0.5452277
```

```
# note that this was a two-sided test
# our alternative is just r being non-0
# running this one-sided, asking if the correlation is positive
cor.test(stat500$midterm, stat500$final,
         alternative = "greater")
```

```
##
```

```
##  Pearson's product-moment correlation
##
## data:  stat500$midterm and stat500$final
## t = 4.735, df = 53, p-value = 8.374e-06
## alternative hypothesis: true correlation is greater than 0
## 95 percent confidence interval:
##  0.3657126 1.0000000
## sample estimates:
##       cor
## 0.5452277
```

```
# and let's remind ourselves how the p-value's calculated:
x = seq(-5,5,0.1)
plot(x, dt(seq(-5,5,0.1), df=53), type="l")
t = cor.test(stat500$midterm, stat500$final)$statistic
abline(v = t, col=2, lty=2)
```



The area to the right is the **one-sided p-value** when the alternative hypothesis is "greater". It is twice as much (as the distribution is symmetric) when the test is **two-sided**. What if we tested the opposite?

```
cor.test(stat500$midterm, stat500$final,
         alternative = "less")
```

```
##
##  Pearson's product-moment correlation
##
## data:  stat500$midterm and stat500$final
## t = 4.735, df = 53, p-value = 1
## alternative hypothesis: true correlation is less than 0
## 95 percent confidence interval:
##  -1.0000000  0.6856316
## sample estimates:
##       cor
## 0.5452277
```

Note that the shape of the distribution is relevant for Pearson (parametric) correlation, but not the scale. Meanwhile, the shape is irrelevant in the case of **Spearman (non-parametric) correlation**.

```
# the correlation coefficient estimate
cor.test(stat500$midterm, stat500$final)$est
```

```
##       cor
## 0.5452277
```

```
# adding a constant does not change the shape
cor.test(stat500$midterm + 10, stat500$final)$est
```

```
##       cor
## 0.5452277
```

```
# but log transformation does change the shape
cor.test(log(stat500$midterm), stat500$final)$est
```

```
##       cor
## 0.5336921
```

```
#
par(mfrow=c(2,2))
plot(stat500$midterm, stat500$final)
plot(log(stat500$midterm), stat500$final)
# compare these with the non-parametric version
cor.test(log(stat500$midterm), stat500$final,
         method = "spearman")$est
```

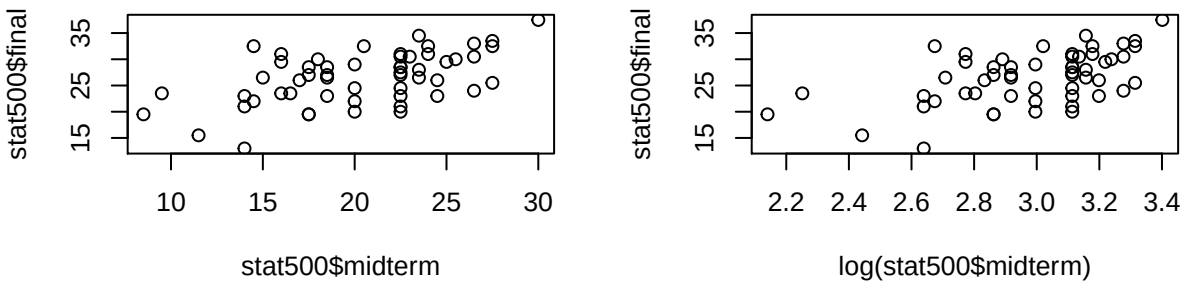
```
## Warning in cor.test.default(log(stat500$midterm), stat500$final, method =
## "spearman"): Cannot compute exact p-value with ties
```

```
##       rho
## 0.5052322
```

```
cor.test(stat500$midterm, stat500$final,
         method = "spearman")$est
```

```
## Warning in cor.test.default(stat500$midterm, stat500$final, method =
## "spearman"): Cannot compute exact p-value with ties
```

```
##       rho
## 0.5052322
```



Linear regression

Regression is useful for: * Describing the relationship between a response (dependent) variable, and predictor (independent) variables as a function. * Testing the relationship, usually with the H0 that there is no relationship. * Predicting values of the response given predictor observations.

The simplest version is just like correlation. But we now have an unequal relationship between the two variables: one is a predictor (usually plotted on the x axis), and the other the response (plotted on the y).

The `lm` function

Let's continue studying the same dataset.

```
g = lm(final ~ midterm, stat500)
g
```

```
##
## Call:
## lm(formula = final ~ midterm, data = stat500)
##
## Coefficients:
## (Intercept)      midterm
##     15.0462       0.5633
```

- The first coefficient is the intercept estimate.
- The second is the slope estimate.

We can scale the matrix columns:

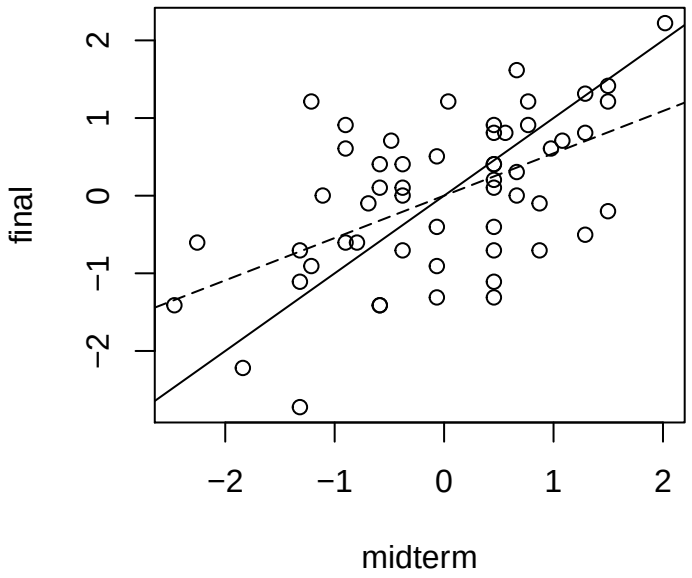
```
stat500 = data.frame( scale( stat500 ) )
plot(final ~  midterm, stat500)
# the 0,1 line, with intersept=0 and slope=1
abline(0,1)
# run the regression again
g = lm(final ~ midterm, stat500)
g
```

```
##
## Call:
## lm(formula = final ~ midterm, data = stat500)
##
## Coefficients:
## (Intercept)      midterm
##  -3.647e-16    5.452e-01
```

```
# after scaling, the slope = correlation coefficient
cor(stat500$final, stat500$midterm)
```

```
## [1] 0.5452277
```

```
# the regression line
abline(g$coef, lty=5)
```



The significance of the regression slope

We can study the significance of the terms using `summary` on an `lm` object.

```
summary(g)
```

```
##
## Call:
## lm(formula = final ~ midterm, data = stat500)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0049 -0.5363  0.1064  0.6024  1.8745
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.647e-16  1.141e-01   0.000        1
## midterm      5.452e-01  1.151e-01   4.735 1.67e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8462 on 53 degrees of freedom
## Multiple R-squared:  0.2973, Adjusted R-squared:  0.284
## F-statistic: 22.42 on 1 and 53 DF,  p-value: 1.675e-05
```

Note that the t-statistic and the p-value are the same.

We can also construct an ANOVA table, calculated using sum of squares of residuals, just like ANOVA. Residuals here are the differences between the predicted values (the line) and the observed values.

```
anova(g)
```

```
## Analysis of Variance Table
##
## Response: final
##           Df Sum Sq Mean Sq F value    Pr(>F)
## midterm    1 16.053  16.053  22.421 1.675e-05 ***
```

```
## Residuals 53 37.947   0.716
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
head( g$residuals )
```

```
##          1          2          3          4          5          6
## -0.5745774 -0.6499615 -0.3599453  1.2549289  0.5611942  1.4011895
```

```
head( stat500$final - g$fit )
```

```
##          1          2          3          4          5          6
## -0.5745774 -0.6499615 -0.3599453  1.2549289  0.5611942  1.4011895
```

```
# sum of sq residulas
sum(g$residuals^2)
```

```
## [1] 37.94724
```

```
# total sum of sq
sum((stat500$final - mean(stat500$final))^2)
```

```
## [1] 54
```

```
# sum of sq explained by the model
sum((stat500$final - mean(stat500$final))^2) - sum(g$residuals^2)
```

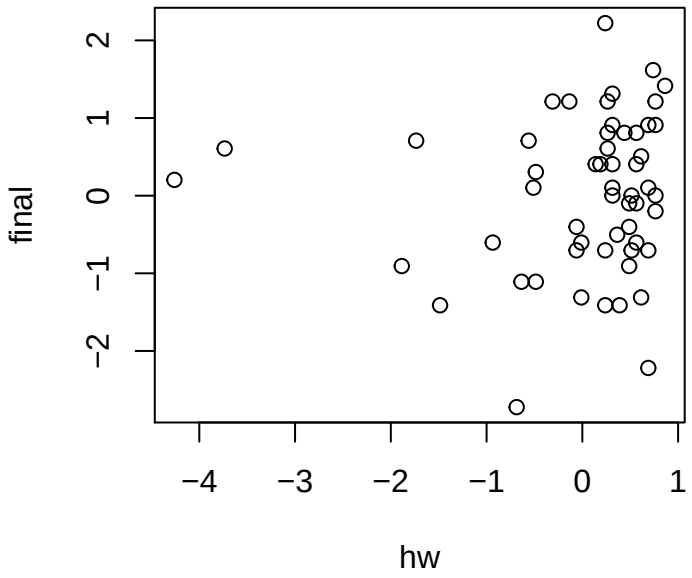
```
## [1] 16.05276
```

```
# compare these with
anova(g)
```

```
## Analysis of Variance Table
##
## Response: final
##           Df Sum Sq Mean Sq F value    Pr(>F)
## midterm    1 16.053  16.053  22.421 1.675e-05 ***
## Residuals 53 37.947   0.716
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Would `hw` also show a similar relationship correlation?

```
plot(final ~ hw, data = stat500)
```




```
# no relationship using Pearson correlation
cor.test(stat500$final, stat500$hw)
```

```
##
##  Pearson's product-moment correlation
##
## data:  stat500$final and stat500$hw
## t = 0.63827, df = 53, p-value = 0.526
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.1821807  0.3446492
## sample estimates:
##        cor
## 0.08733764
```

```
# or non-parametric correlation
cor.test(stat500$final, stat500$hw,
         method="spearman")
```

```
## Warning in cor.test.default(stat500$final, stat500$hw, method =
## "spearman"): Cannot compute exact p-value with ties
```

```
##
##  Spearman's rank correlation rho
##
## data:  stat500$final and stat500$hw
## S = 22926, p-value = 0.2067
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.1729462
```

```
g = lm(final ~ hw, stat500)
summary(g)
```

```
##
## Call:
## lm(formula = final ~ hw, data = stat500)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.66337 -0.71245  0.04259  0.77857  2.20141
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.305e-16  1.356e-01   0.000    1.000
## hw           8.734e-02  1.368e-01   0.638    0.526
##
## Residual standard error: 1.006 on 53 degrees of freedom
## Multiple R-squared:  0.007628,   Adjusted R-squared:  -0.0111
## F-statistic: 0.4074 on 1 and 53 DF,  p-value: 0.526
```

As you notice the slope is very small. And is not significant. Thus, homework grades do not seem to be related to (or influence) the final grade.

Predicting using `predict`

```
g = lm(final ~ midterm, stat500)
g
```

```
##
## Call:
## lm(formula = final ~ midterm, data = stat500)
##
## Coefficients:
## (Intercept)      midterm
##  -3.647e-16    5.452e-01
```

```
# predict the final values for all individuals given the midterm results
predict(g)
```

```
##           1           2           3           4           5           6
##  0.47548280  0.24807799  0.36178039  0.36178039  0.24807799 -0.49098768
##           7           8           9          10          11          12
##  0.81659003  0.24807799  0.53233401  1.10084606 -0.32043406 -0.43413647
##          13          14          15          16          17          18
##  0.24807799  0.24807799 -1.00264852 -0.03617804  0.41863160 -1.23005334
##          19          20          21          22          23          24
## -0.03617804  0.70288762 -0.20673165 -0.20673165 -0.66154129 -0.20673165
##          25          26          27          28          29          30
## -0.32043406  0.30492919 -0.71839250  0.81659003  0.47548280  0.24807799
##          31          32          33          34          35          36
## -0.03617804 -0.32043406 -0.32043406  0.24807799  0.70288762 -0.20673165
##          37          38          39          40          41          42
##  0.02067317 -0.66154129 -0.49098768  0.41863160 -0.49098768  0.24807799
##          43          44          45          46          47          48
## -0.03617804 -0.37728527 -0.71839250  0.58918521 -0.71839250 -1.34375575
##          49          50          51          52          53          54
##  0.81659003 -0.26358286  0.24807799 -0.60469009  0.24807799  0.70288762
##          55
##  0.36178039
```

```
# the regression coefficients
g$coefficients
```

```
##   (Intercept)       midterm
## -3.646651e-16  5.452277e-01
```

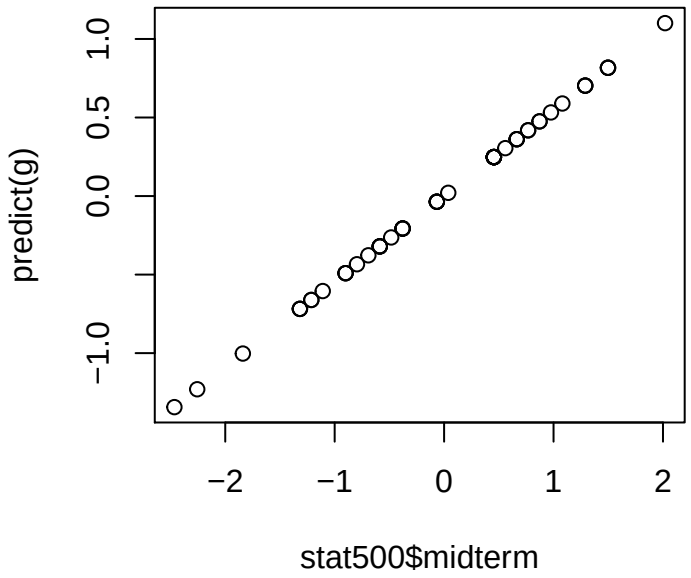
```
# the fitted y values values (predicted for each x)
g$fit
```

```
##           1           2           3           4           5           6
##  0.47548280  0.24807799  0.36178039  0.36178039  0.24807799 -0.49098768
##           7           8           9          10          11          12
##  0.81659003  0.24807799  0.53233401  1.10084606 -0.32043406 -0.43413647
##          13          14          15          16          17          18
##  0.24807799  0.24807799 -1.00264852 -0.03617804  0.41863160 -1.23005334
##          19          20          21          22          23          24
## -0.03617804  0.70288762 -0.20673165 -0.20673165 -0.66154129 -0.20673165
##          25          26          27          28          29          30
## -0.32043406  0.30492919 -0.71839250  0.81659003  0.47548280  0.24807799
##          31          32          33          34          35          36
## -0.03617804 -0.32043406 -0.32043406  0.24807799  0.70288762 -0.20673165
##          37          38          39          40          41          42
##  0.02067317 -0.66154129 -0.49098768  0.41863160 -0.49098768  0.24807799
##          43          44          45          46          47          48
## -0.03617804 -0.37728527 -0.71839250  0.58918521 -0.71839250 -1.34375575
##          49          50          51          52          53          54
##  0.81659003 -0.26358286  0.24807799 -0.60469009  0.24807799  0.70288762
##          55
##  0.36178039
```

```
# the difference between the fitted and observed y values.
g$residuals
```

```
##           1           2           3           4           5           6
## -0.57457736 -0.64996146 -0.35994531  1.25492892  0.56119422  1.40118952
##           7           8           9          10          11          12
##  0.59826000  0.66212386  0.07507891  1.12144110 -1.09074581 -0.16960628
##          13          14          15          16          17          18
## -1.35646894 -1.55832822 -1.21596846 -1.27407219  0.49157024  0.62631059
##          19          20          21          22          23          24
## -0.87035363  0.10638458  0.61228530 -0.49794074  1.87453205  0.20856674
##          25          26          27          28          29          30
##  0.72598771  0.50434301 -0.38999845 -1.01661423 -1.18015520  0.15747566
##          31          32          33          34          35          36
##  0.54266132  0.42319879 -1.09074581 -0.04438362 -1.20570074  0.30949638
##          37          38          39          40          41          42
##  1.19231759 -0.24499038  1.09840060  0.79435916 -0.11275507  0.15747566
##          43          44          45          46          47          48
## -0.36570543  0.27819071  0.01372011  0.11915735 -2.00487269 -0.06742412
##          49          50          51          52          53          54
##  0.39640073  0.97192542 -0.14531326  0.60652517 -0.95275038  0.61103277
##          55
## -0.05715639
```

```
# note that regression tries to minimize the residuals
plot( stat500$midterm, predict(g) )
```



What is the estimated final outcome for a student with midterm grade = 1 (i.e. one s.d. above the mean)?

```
# you need to define a new data frame
predict( g, data.frame(midterm=1) )
```

```
##         1
## 0.5452277
```

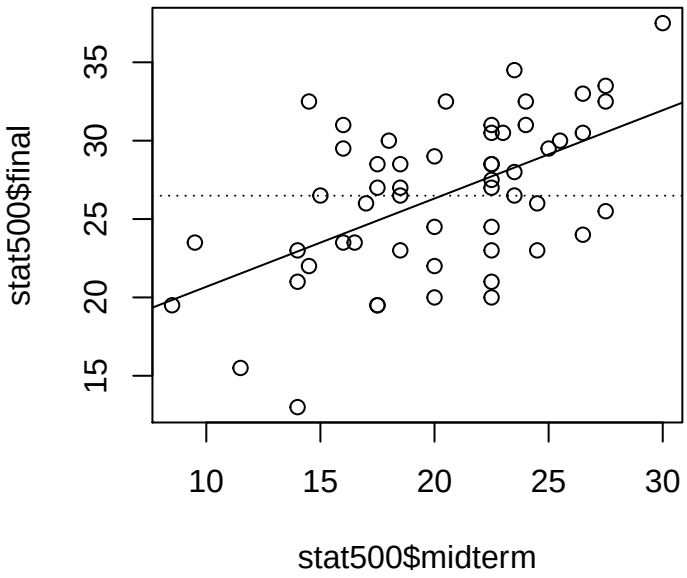
We can also make predictions using the original data:

```
# reload the original dataset
data(stat500)

plot(stat500$midterm, stat500$final)
g = lm(final ~ midterm, stat500)
g
```

```
##
## Call:
## lm(formula = final ~ midterm, data = stat500)
##
## Coefficients:
## (Intercept)      midterm
##     15.0462       0.5633
```

```
abline(g)
abline(h=mean(stat500$final), lty=3)
```




```
# now make predictions, for midterm grades 25 and 15
predict(g, data.frame(midterm=c(25,15)))
```

```
##        1        2
## 29.12806 23.49531
```

```
# we can also calculate these ourselves
g
```

```
##
## Call:
## lm(formula = final ~ midterm, data = stat500)
##
## Coefficients:
## (Intercept)      midterm
##     15.0462       0.5633
```

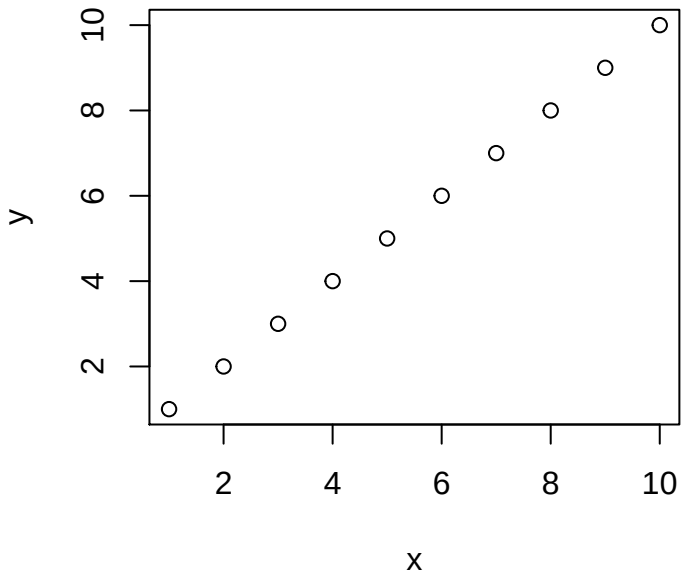
```
g$coefficients[1] + g$coefficients[2]*15
```

```
## (Intercept)
##    23.49531
```

How noise affects the slope

Adding noise to either variable affects the regression but in different ways. Let's study this situation using a small experiment with toy variables.

```
x = 1:10
y = x
plot(x, y)
```




```
lm(y ~ x )
```

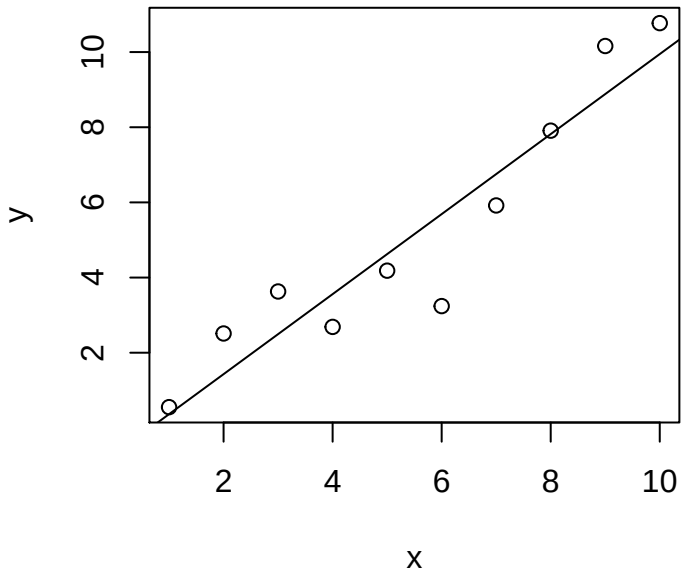
```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##   1.123e-15    1.000e+00
```

The slope is 1.

Noise in the response

Now we add random noise to the data from the standard normal distribution (mean=0, sd=1).

```
y = x + rnorm(10)
plot(x, y)
g = lm(y ~ x )
abline(g)
```

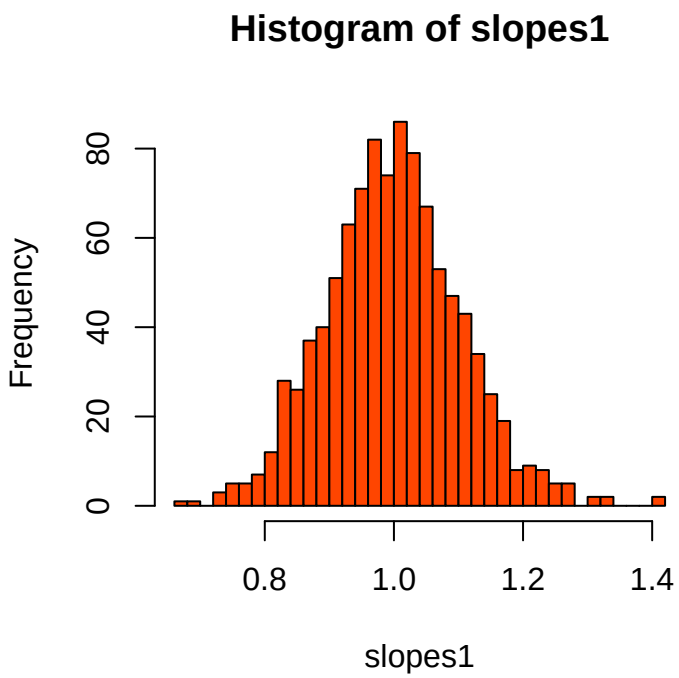



```
g
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##     -0.6987       1.0646
```

Now the slope is different from 1. How would the deviation appear if we ran this 1000 times and recorded the slope each time?

```
slopes1 = sapply(1:1000, function(i) {
  y = x + rnorm(10)
  g = lm(y ~ x )
  g$coefficients[2]
})
hist(slopes1, br=50, col='orangered1')
```




```
summary(slopes1)
```

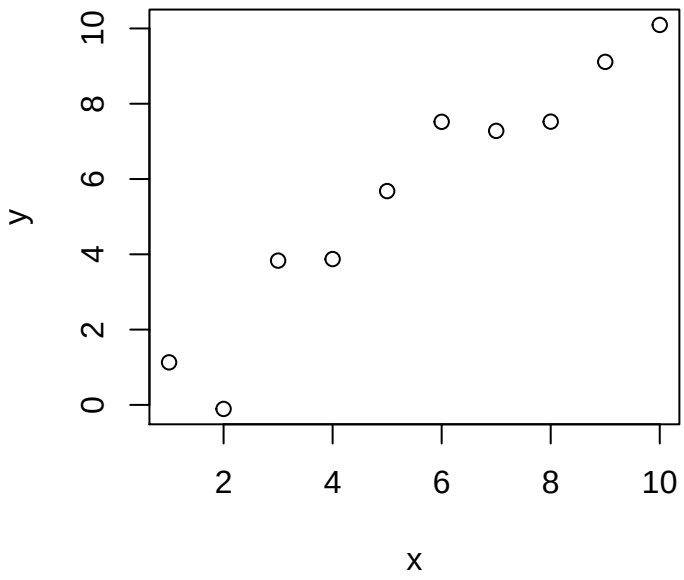
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.6654  0.9309  0.9983  0.9995  1.0630  1.4150
```

Apparently the slope estimates are still distributed around 1, the original value.

How about if we added more noise?

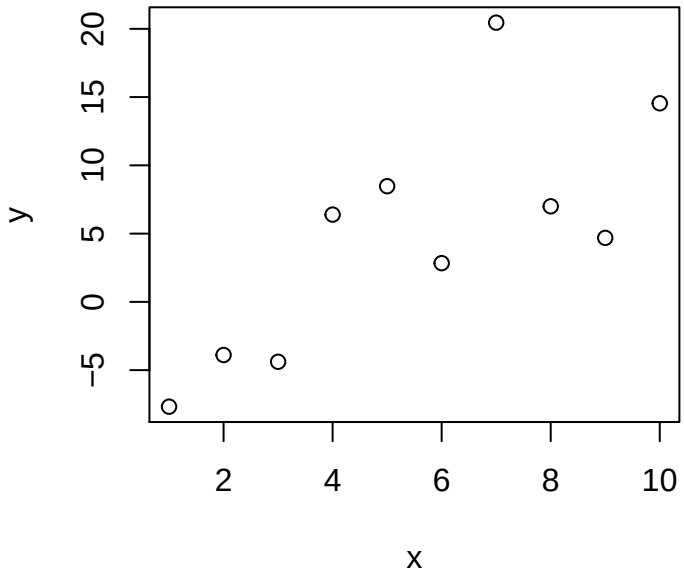
For this we could increase the sd:

```
# the original case
y = x + rnorm(10, sd=1)
plot(x, y)
```

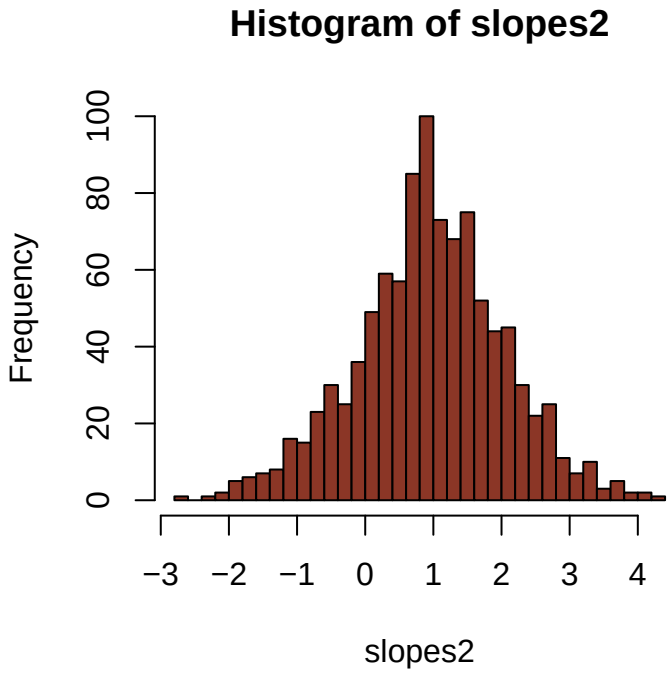



```
# more noise
y = x + rnorm(10, sd=5)
```

```r
plot(x, y)
```




```r
# now run 1000 times
slopes2 = sapply(1:1000, function(i) {
  y = x + rnorm(10, sd=10)
  g = lm(y ~ x )
  g$coefficients[2]
})
hist(slopes2, br=50, col='tomato4')
```




```r
summary(slopes2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.7230  0.3009  0.9562  0.9495  1.6270  4.3580
```

The distribution of estimated slopes is now much wider. But it is still distributed around the original value. Thus, noise in the response lowers precision of the estimate, but does not create a systematic bias.

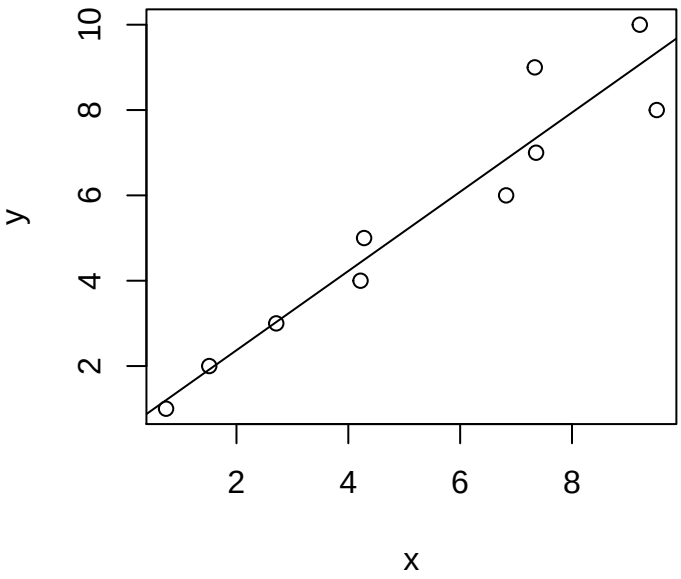
Noise in the predictor

Would the same apply if we added noise to the predictor?

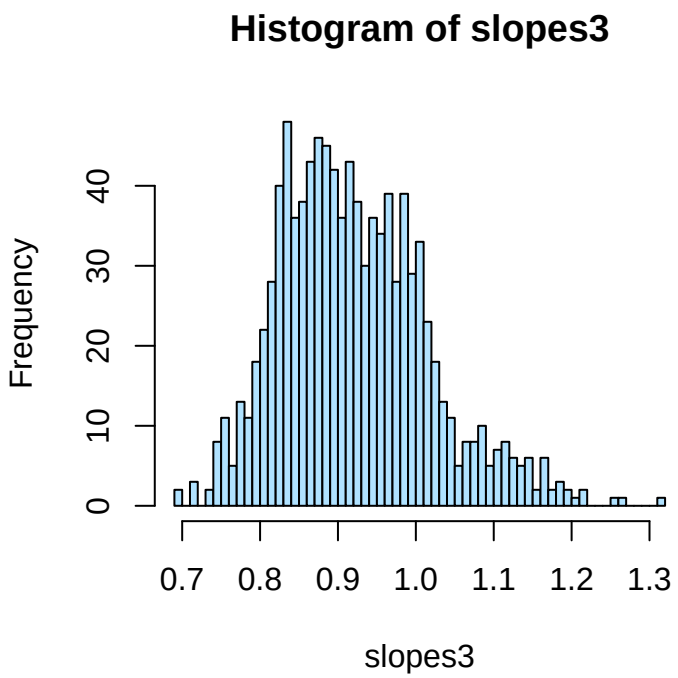
```
y = 1:10
x = 1:10 + rnorm(10, sd=1)
plot(x, y)
g = lm(y ~ x)
g
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##      0.5124       0.9287
```

```
abline(g)
```



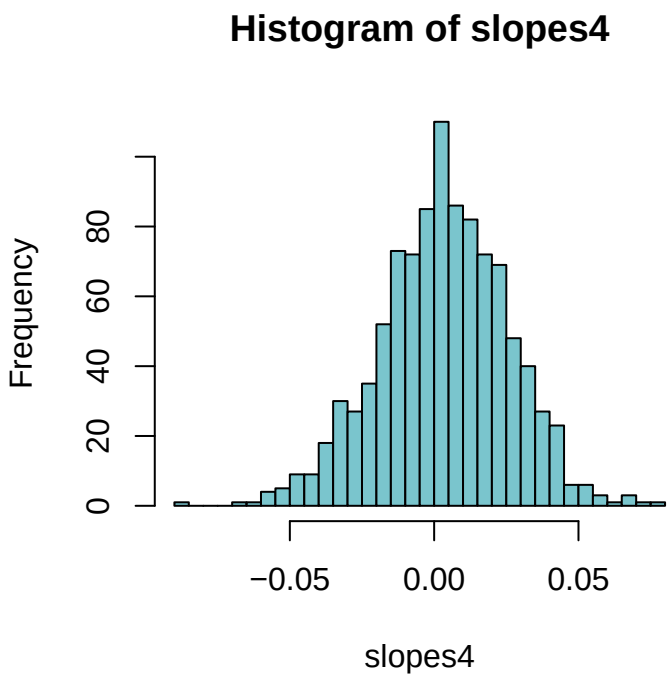

```
# now run this 1000 times
slopes3 = sapply(1:1000, function(i) {
  y = 1:10
  x = 1:10 + rnorm(10, sd=1)
  g = lm(y ~ x )
  g$coefficients[2]
})
hist(slopes3, br=50, col='lightskyblue1')
```




```
summary(slopes3)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.6954  0.8503  0.9107  0.9208  0.9808  1.3170
```

```
# and adding more noise
slopes4 = sapply(1:1000, function(i) {
  y = 1:10
  x = 1:10 + rnorm(10, sd=50)
  g = lm(y ~ x )
  g$coefficients[2]
})
hist(slopes4, br=50, col='cadetblue3')
```

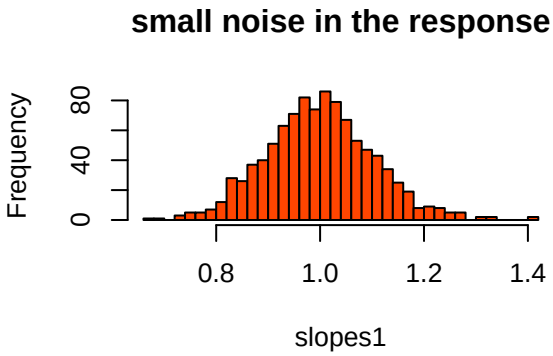


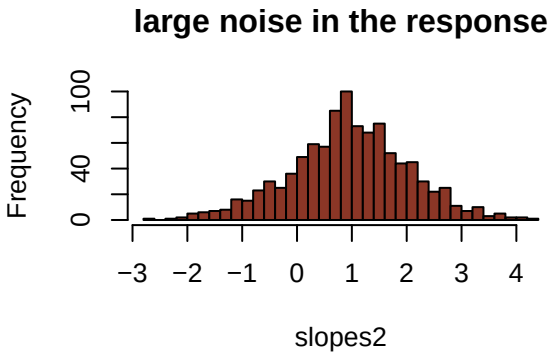

```
summary(slopes4)
```

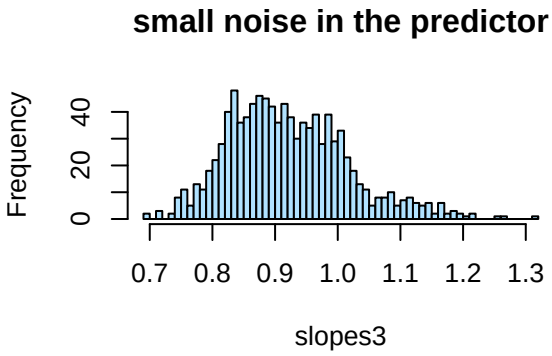
```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.085470 -0.011140  0.003376  0.003315  0.018270  0.076790
```

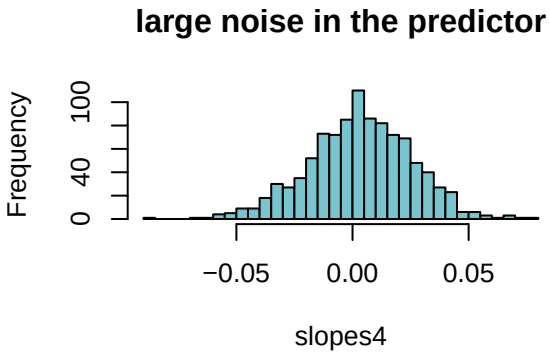
It seems the average estimate shifts to `0` as noise increases. This is because the slope is calculated to minimise the residuals on the Y-axis (it assumes noise on the y-axis, but not the x-axis).

```
par(mfrow=c(2,2))
hist(slopes1, br=50, main = "small noise in the response", col='orangered1')
hist(slopes2, br=50, main = "large noise in the response", col='tomato4')
hist(slopes3, br=50, main = "small noise in the predictor", col='lightskyblue1')
hist(slopes4, br=50, main = "large noise in the predictor", col='cadetblue3')
```







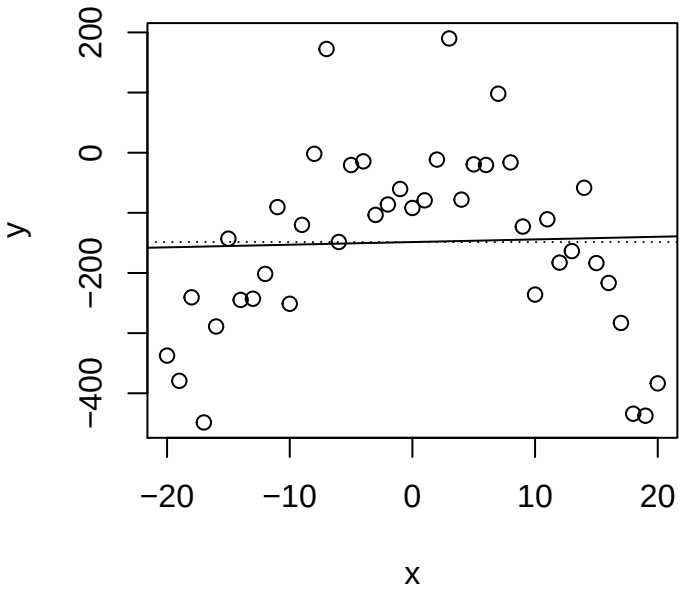


Nonlinear relationships

We can also study non-linear relationships by adding polynomial factors in the model.

Let's again study a toy case, simulating data which are related in non-linear fashion:

```
set.seed(1)
x = -20:20
y = -(x^2 + rnorm(length(x), sd=100))
plot(x, y)
# the linear model
g = lm(y ~ x)
# the mean of y, which represents the H0
abline(h=mean(y), lty=3)
abline(g)
```



Values predicted from this model are also pretty useless:

```
predict(g, data.frame(x=20))
```

```
##         1
## -139.7995
```

```
predict(g, data.frame(x=-20))
```

```
##         1
## -157.3543
```

```
summary(g)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -297.08  -90.02   21.83   88.63  337.20
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -148.5769    24.1872  -6.143 3.28e-07 ***
## x              0.4389     2.0442   0.215    0.831
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 154.9 on 39 degrees of freedom
## Multiple R-squared:  0.00118,    Adjusted R-squared:  -0.02443
## F-statistic: 0.04609 on 1 and 39 DF,  p-value: 0.8311
```

```
# is the model significant?
anova(g)
```

```
## Analysis of Variance Table
##
```

```
## Response: y
##           Df Sum Sq Mean Sq F value Pr(>F)
## x          1   1106  1105.6  0.0461 0.8311
## Residuals 39 935449 23985.9
```

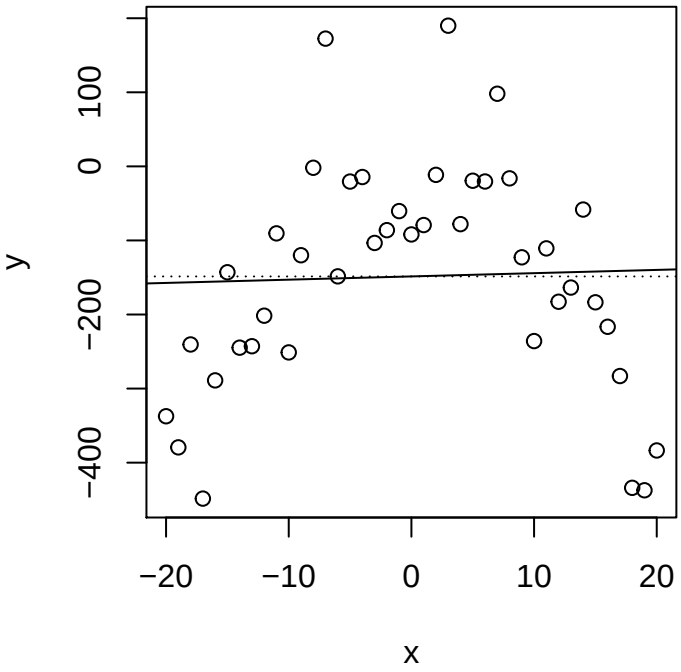
In the ANOVA table we see that the total sum of squares (SS) is composed of residuals, with nearly no SS explained by the regression. Unsurprisingly, linear regression is not significant. The regression line does not explain more variance than would be expected from random. Thus we don't need a linear model.

Spline regression using `smooth.spline`

To study possible non-linear relationships we could instead use spline regression.

For an introduction check: http://people.stat.sfu.ca/~cschwarz/Consulting/Trinity/Phase2/TrinityWorkshop/Workshop-handouts/TW-04-Intro-splines.pdf

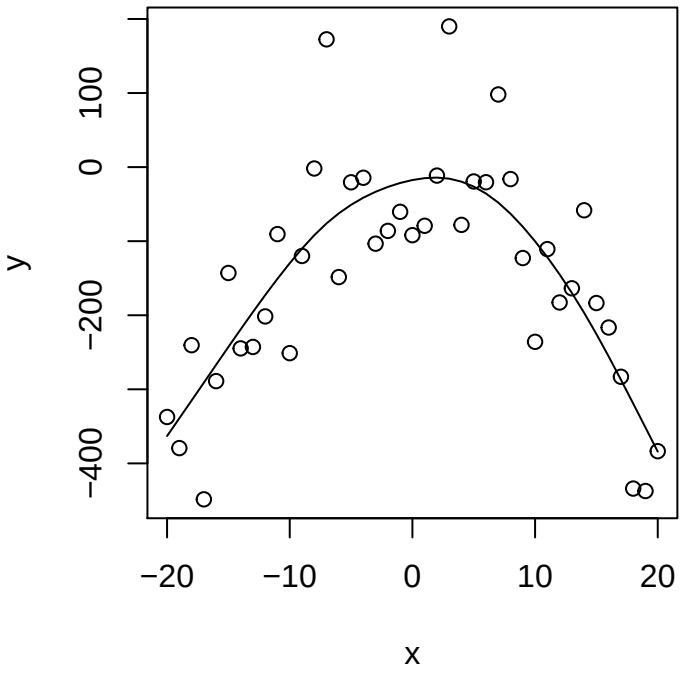
```
plot(x, y)
g = lm(y ~ x)
# mean
abline(h=mean(y), lty=3)
# linear
abline(g)
```




```
# smooth spline regression
smooth.spline(y ~ x )
```

```
## Call:
## smooth.spline(x = y ~ x)
##
## Smoothing Parameter  spar= 0.7952583  lambda= 0.006225094 (11 iterations)
## Equivalent Degrees of Freedom (Df): 4.240411
## Penalized Criterion: 298090.5
## GCV: 9044.627
```

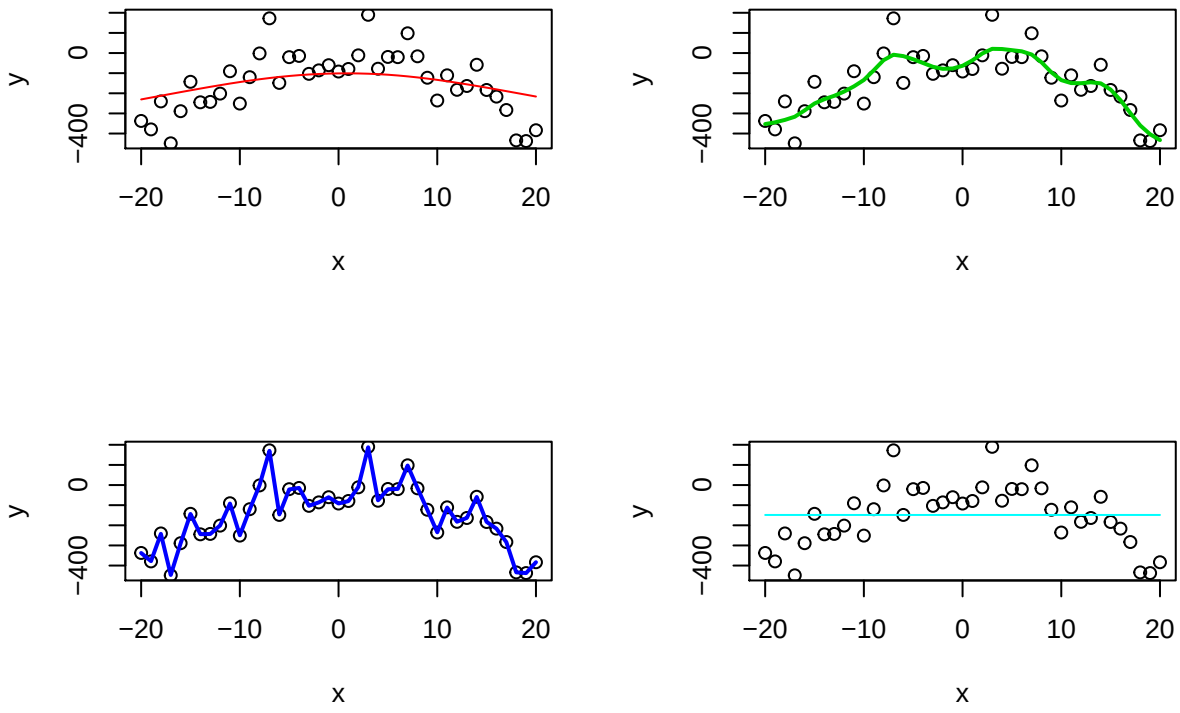
```r
# predict y-values for each x
plot(x, y)
lines(predict(smooth.spline(y ~ x )))
```



You can also change the **smoothing** (how wide the window will be) using the `spar` parameter. This is related to the degrees of freedomg - the less the smoothing, the more rigid the model will be, and will be using more parameters. The less the smoothing, the fewer parameters (and in the most extreme case - no parameters, which means just estimating the mean).

```r
par(mfrow=c(2,2))
plot(x, y); lines(predict(smooth.spline(y ~ x, spar = 1)), col=2)
plot(x, y); lines(predict(smooth.spline(y ~ x, spar = 0.5)), col=3, lwd=2)
# no smoothing - the line is the points, uses 41 degrees of freedom
plot(x, y); lines(predict(smooth.spline(y ~ x, spar = 0)), col=4, lwd=2)
# degrees of freedom = 1, only estimates the mean
plot(x, y); lines(predict(smooth.spline(y ~ x, spar = 5 )), col=5)
```

```
## Warning in smooth.spline(y ~ x, spar = 5): smoothing parameter value too large
## setting df = 1  __use with care!__
```



Polynomial regression

Splines are useful for exploring relationships. We could alternatively check possible polynomial terms, such as quadratic or cubic:

```
# this is wrong in R syntax
g2 = lm(y ~ x + x^2)
g2
```

```
##
## Call:
## lm(formula = y ~ x + x^2)
##
## Coefficients:
## (Intercept)            x
##   -148.5769       0.4389
```

```
summary(g2)
```

```
##
## Call:
## lm(formula = y ~ x + x^2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -297.08  -90.02   21.83   88.63  337.20
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -148.5769    24.1872  -6.143 3.28e-07 ***
## x              0.4389     2.0442   0.215    0.831
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 154.9 on 39 degrees of freedom
## Multiple R-squared:  0.00118,    Adjusted R-squared:  -0.02443
## F-statistic: 0.04609 on 1 and 39 DF,  p-value: 0.8311
```

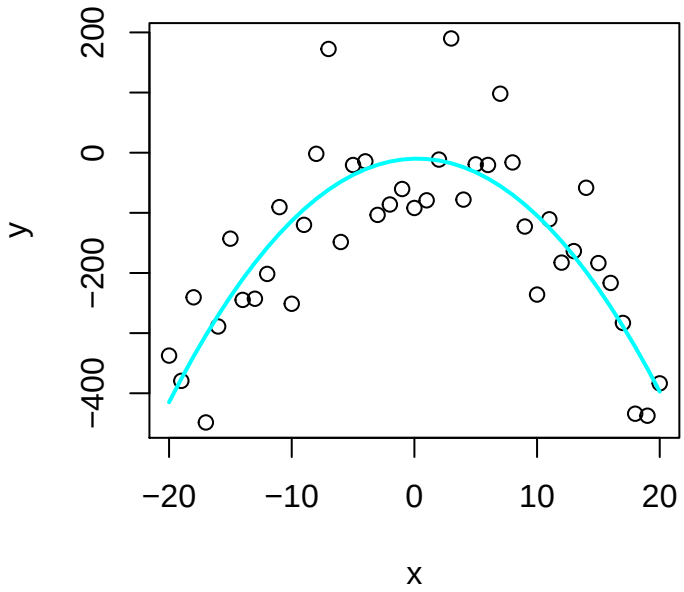
```
# this is correct
# I(...) calculates the values and includes them in the model
g2 = lm(y ~ x + I(x^2))
g2
```

```
##
## Call:
## lm(formula = y ~ x + I(x^2))
##
## Coefficients:
## (Intercept)            x       I(x^2)
##     -9.9326       0.4389      -0.9903
```

```
summary(g2)
```

```
##
## Call:
## lm(formula = y ~ x + I(x^2))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -144.93  -59.83    1.56   42.67  234.00
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -9.9326    21.0351  -0.472    0.639
## x             0.4389     1.1846   0.370    0.713
## I(x^2)       -0.9903     0.1120  -8.839 9.37e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 89.75 on 38 degrees of freedom
## Multiple R-squared:  0.6732, Adjusted R-squared:  0.656
## F-statistic: 39.14 on 2 and 38 DF,  p-value: 5.914e-10
```

```
plot(x, y)
lines(x, predict(g2), col=5, lwd=2)
```




```
# similar to the spline curve, but simpler and explicit
```

So the quadratic term is significant. How about adding a cubic term?

```
g3 = lm(y ~ x + I(x^2) + I(x^3))
g3
```

```
##
## Call:
## lm(formula = y ~ x + I(x^2) + I(x^3))
##
## Coefficients:
## (Intercept)            x       I(x^2)       I(x^3)
##    -9.93262      3.55594     -0.99032     -0.01238
```

```
summary(g3)
```

```
##
## Call:
## lm(formula = y ~ x + I(x^2) + I(x^3))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -152.762  -55.625   -5.354   40.693  251.574
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -9.93262   20.94616  -0.474    0.638
## x            3.55594    2.95516   1.203    0.237
## I(x^2)      -0.99032    0.11156  -8.877 1.06e-10 ***
## I(x^3)      -0.01238    0.01076  -1.150    0.257
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 89.37 on 37 degrees of freedom
## Multiple R-squared:  0.6845, Adjusted R-squared:  0.6589
## F-statistic: 26.75 on 3 and 37 DF,  p-value: 2.238e-09
```

As expected, the cubic term not significant.

We could also compare different models to decide if one is superior over the other - that is, whether it explains a non-significant portion of the variance.

```
# a model with only the quadratic term
g1 = lm(y ~ I(x^2))
g1
```

```
##
## Call:
## lm(formula = y ~ I(x^2))
##
## Coefficients:
## (Intercept)       I(x^2)
##     -9.9326      -0.9903
```

```
summary(g1)
```

```
##
## Call:
## lm(formula = y ~ I(x^2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -152.394  -65.536    2.437   46.883  230.928
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -9.9326    20.8011  -0.478    0.636
## I(x^2)       -0.9903     0.1108  -8.939 5.55e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 88.75 on 39 degrees of freedom
## Multiple R-squared:  0.672,  Adjusted R-squared:  0.6636
## F-statistic:  79.9 on 1 and 39 DF,  p-value: 5.553e-11
```

```
# is the only-quadratic model better than linear+quadratic?
anova(g1, g2)
```

```
## Analysis of Variance Table
##
## Model 1: y ~ I(x^2)
## Model 2: y ~ x + I(x^2)
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     39 307192
## 2     38 306086  1    1105.6 0.1373 0.7131
```

```
# no. how about the linear model g vs. linear+quadratic?
anova(g, g2)
```

```
## Analysis of Variance Table
##
## Model 1: y ~ x
## Model 2: y ~ x + I(x^2)
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1     39 935449
```

```
## 2     38 306086  1    629363 78.134 9.37e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# as we checked before, this one was significant.
```

The Galapagos dataset

Plant species numbers, endemic species numbers, and area, elevation, etc of Galapagos islands.

https://upload.wikimedia.org/wikipedia/commons/e/e7/Galapagos%2Bmap.jpg

```
# str gives a useful summary
str(gala)
```

```
## 'data.frame':    30 obs. of  7 variables:
##  $ Species  : num  58 31 3 25 2 18 24 10 8 2 ...
##  $ Endemics : num  23 21 3 9 1 11 0 7 4 2 ...
##  $ Area     : num  25.09 1.24 0.21 0.1 0.05 ...
##  $ Elevation: num  346 109 114 46 77 119 93 168 71 112 ...
##  $ Nearest  : num  0.6 0.6 2.8 1.9 1.9 8 6 34.1 0.4 2.6 ...
##  $ Scruz    : num  0.6 26.3 58.7 47.4 1.9 ...
##  $ Adjacent : num  1.84 572.33 0.78 0.18 903.82 ...
```

```
head(gala)
```

```
##              Species Endemics  Area Elevation Nearest Scruz Adjacent
## Baltra            58       23 25.09       346     0.6   0.6     1.84
## Bartolome         31       21  1.24       109     0.6  26.3   572.33
## Caldwell           3        3  0.21       114     2.8  58.7     0.78
## Champion          25        9  0.10        46     1.9  47.4     0.18
## Coamano            2        1  0.05        77     1.9   1.9   903.82
## Daphne.Major      18       11  0.34       119     8.0   8.0     1.84
```

```
summary(gala)
```

```
##     Species          Endemics          Area            Elevation      
##  Min.   :  2.00   Min.   : 0.00   Min.   :   0.010   Min.   :  25.00  
##  1st Qu.: 13.00   1st Qu.: 7.25   1st Qu.:   0.258   1st Qu.:  97.75  
##  Median : 42.00   Median :18.00   Median :   2.590   Median : 192.00  
##  Mean   : 85.23   Mean   :26.10   Mean   : 261.709   Mean   : 368.03  
##  3rd Qu.: 96.00   3rd Qu.:32.25   3rd Qu.:  59.237   3rd Qu.: 435.25  
##  Max.   :444.00   Max.   :95.00   Max.   :4669.320   Max.   :1707.00  
##     Nearest           Scruz           Adjacent      
##  Min.   : 0.20   Min.   :  0.00   Min.   :   0.03  
##  1st Qu.: 0.80   1st Qu.: 11.03   1st Qu.:   0.52  
##  Median : 3.05   Median : 46.65   Median :   2.59  
##  Mean   :10.06   Mean   : 56.98   Mean   : 261.10  
##  3rd Qu.:10.03   3rd Qu.: 81.08   3rd Qu.:  59.24  
##  Max.   :47.40   Max.   :290.20   Max.   :4669.32
```

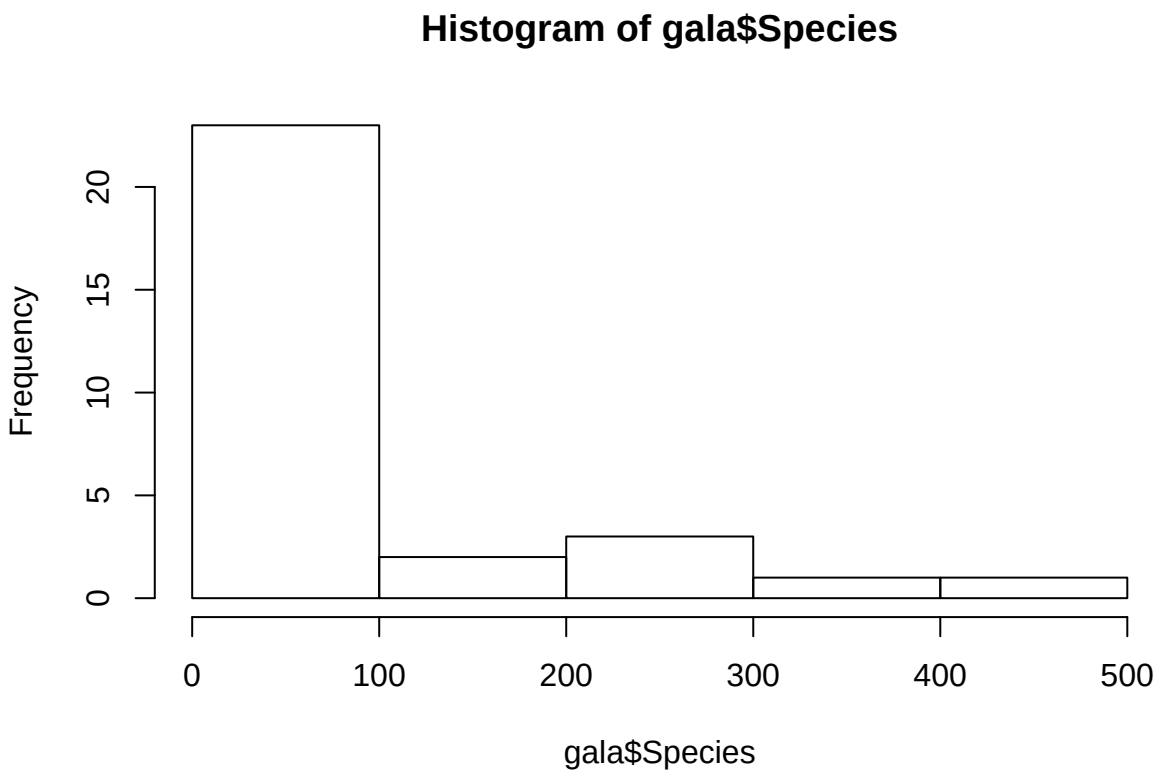
```
dim(gala)
```

```
## [1] 30  7
```

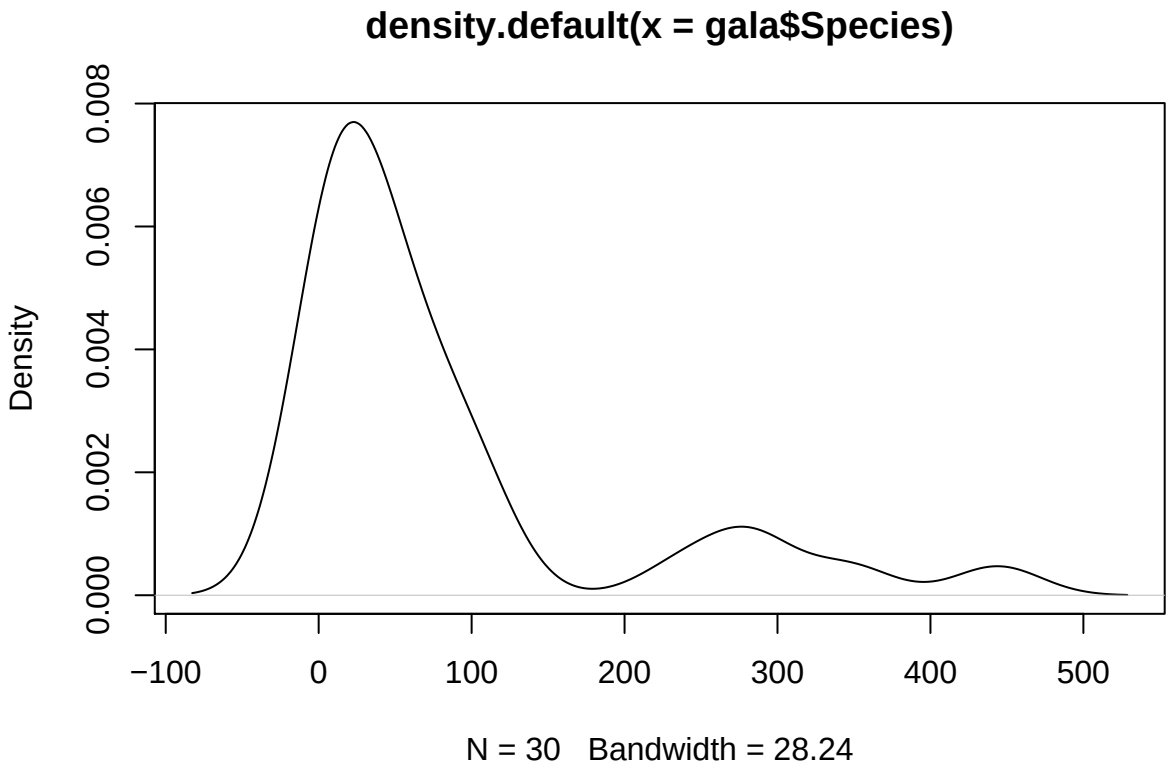
The `density` function

`Species` is the variable of highest interest and we hope to explain in species numbers across islands. How is it distributed? Instead of a histrogram we can use a density plot.

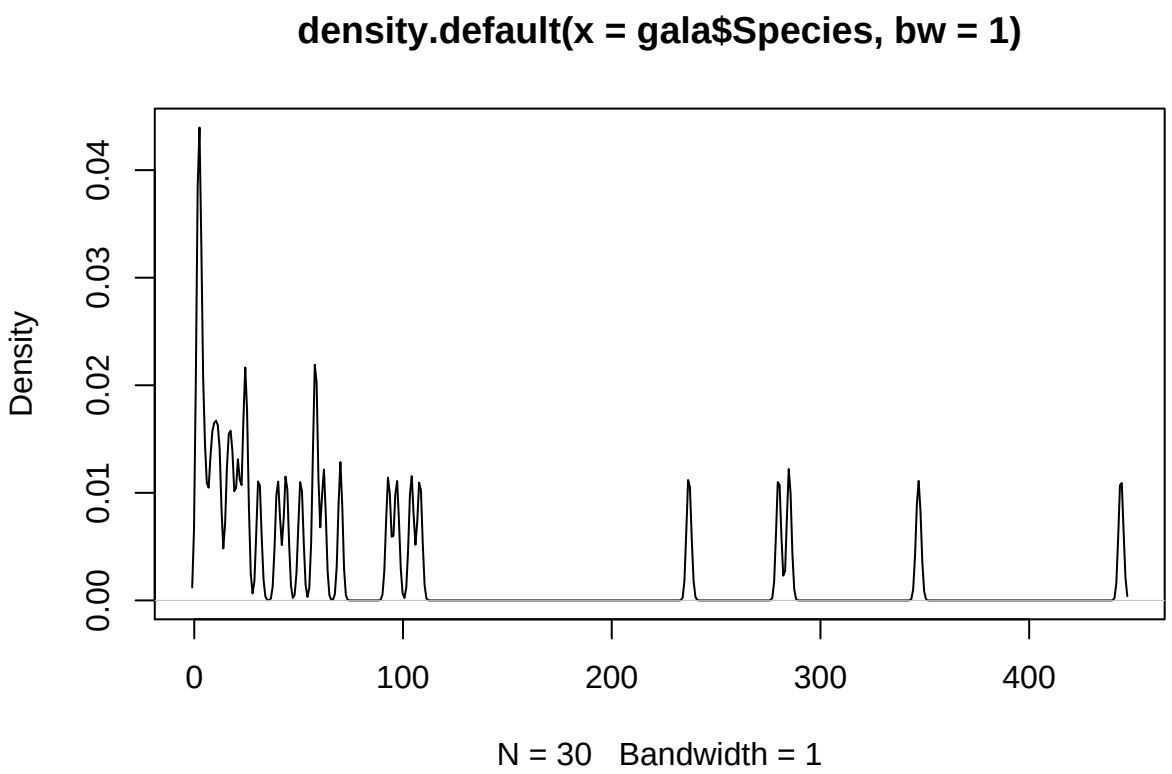
```
hist(gala$Species)
```



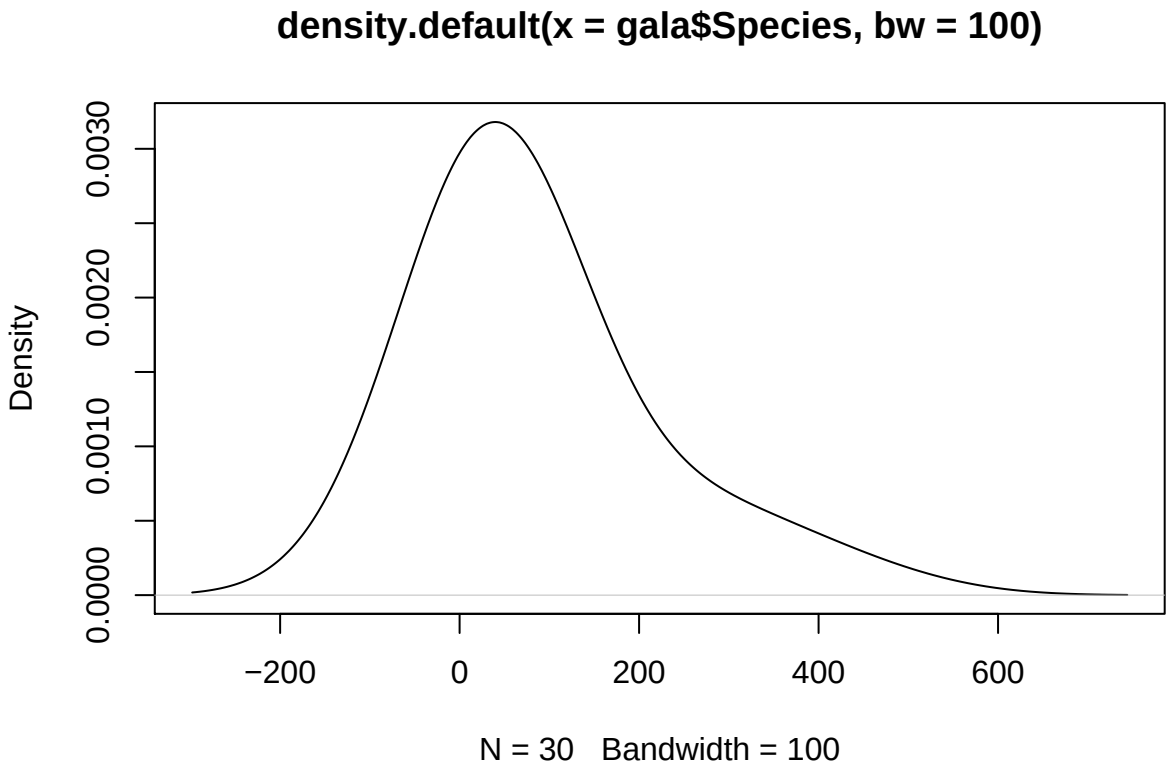

```
plot(density(gala$Species))
```




```
plot(density(gala$Species, bw = 1))
```



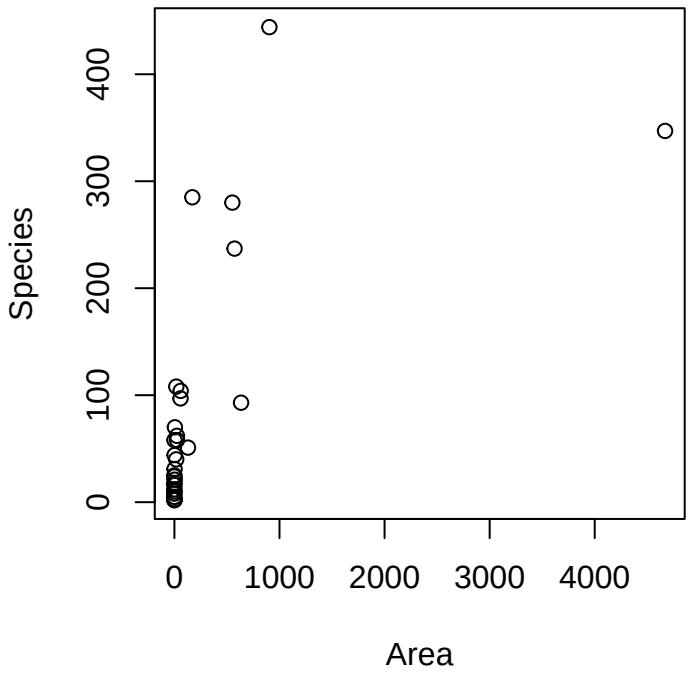

```
plot(density(gala$Species, bw = 100))
```



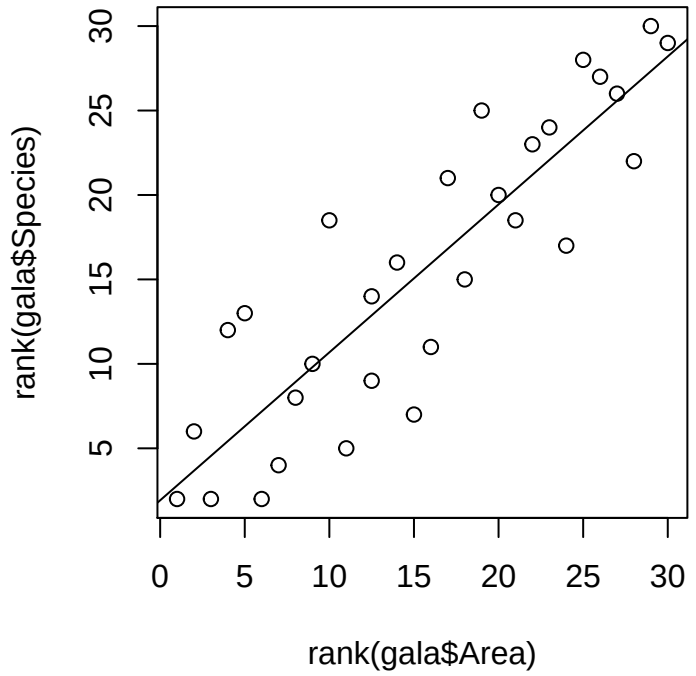
biplot and pairs functions

The variables interrelated? We can study one-by-one or in bulk:

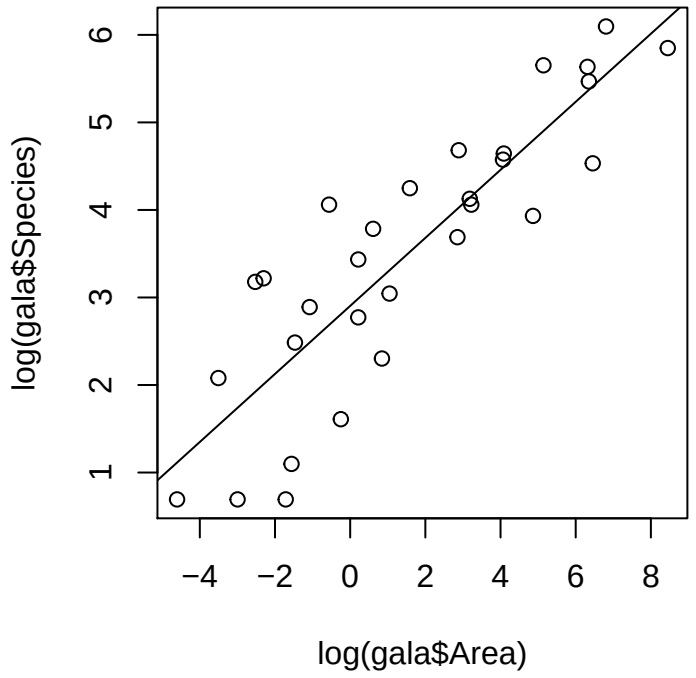
```
plot(Species ~ Area, gala)
```



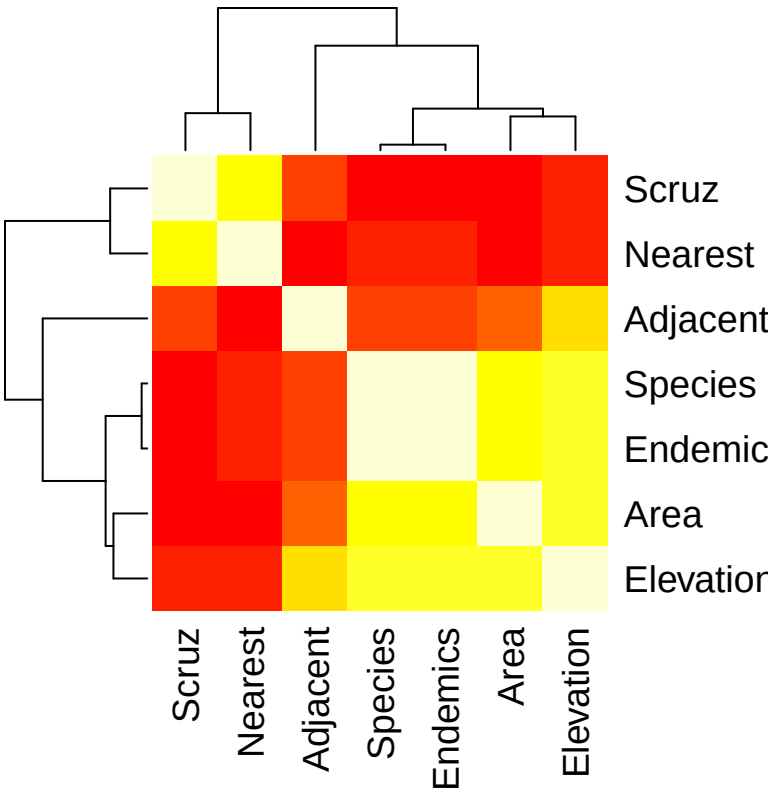
```
plot(rank(gala$Area), rank(gala$Species) )
abline( lm(rank(gala$Species) ~ rank(gala$Area)) )
```




```
plot(log(gala$Area), log(gala$Species) )
abline( lm(log(gala$Species) ~ log(gala$Area)) )
```

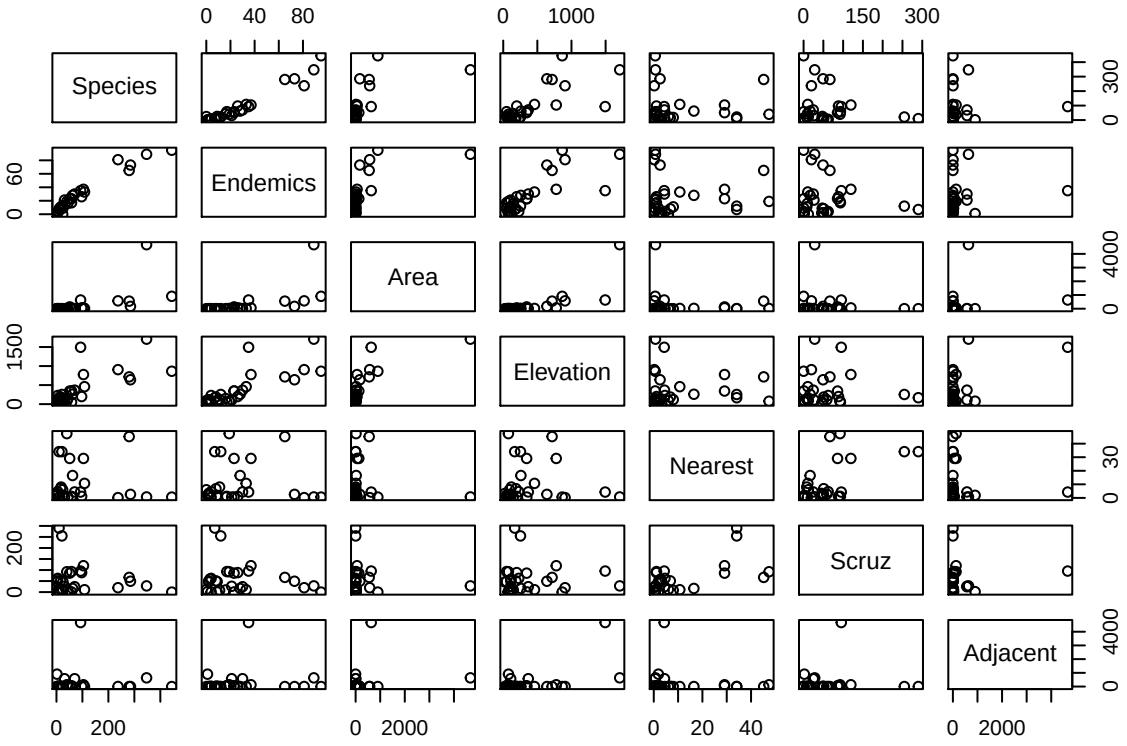



```
heatmap(cor(gala), symm = T)
```




```
# correlation between pairs of variables
pairs(gala)

# this runs the same
plot(gala)
```



```r
# biplot of PCA
pc = prcomp(gala, scale = T)
biplot(pc)
```