CS6220: Data mining techniques
Linear regression

Olga Vitek

September 22, 2015

Outline

Example dataset

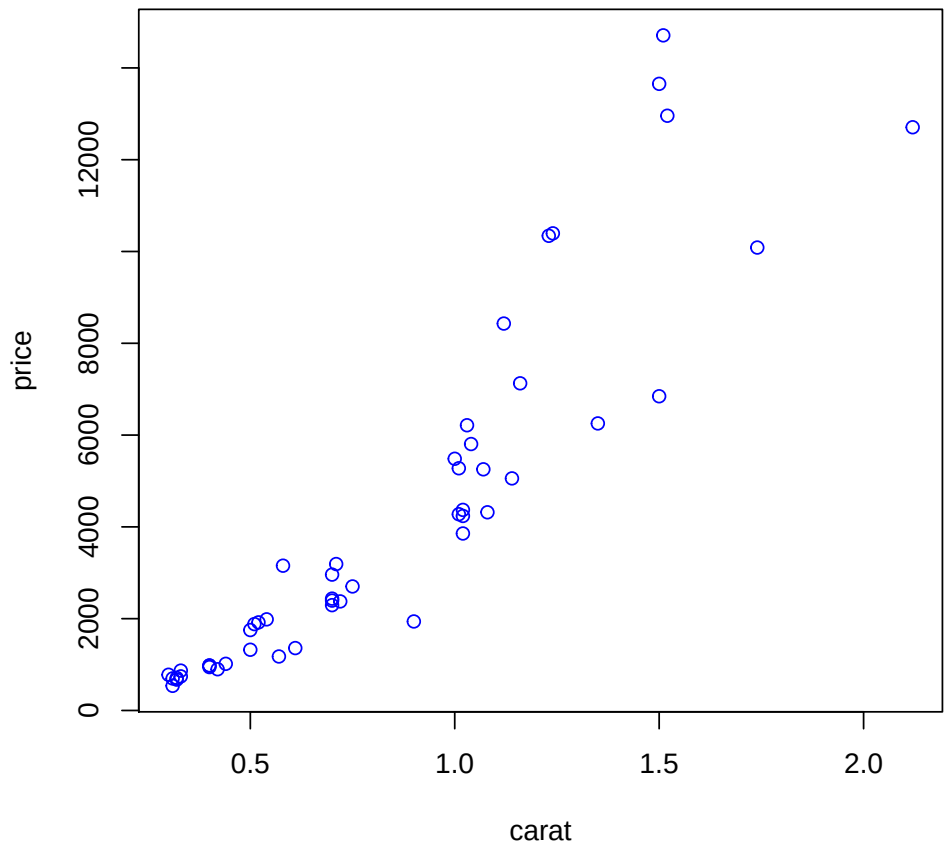
Example dataset:

- Let's take a random sample of 50 diamonds

```
> library(ggplot2)
> set.seed(123)
> index <- sample(1:nrow(diamonds), 50) # try a subset first
> diamonds2 <- diamonds[index,]
```

Example dataset

```
> library(ggplot2)
> plot(price ~ carat, data=diamonds2, col="blue")
```



Pearson correlation

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

```
> cor(diamonds2$price, diamonds2$carat) # random 50

[1] 0.9087753

> cor(diamonds$price, diamonds$carat) # all the diamonds

[1] 0.9215913
```

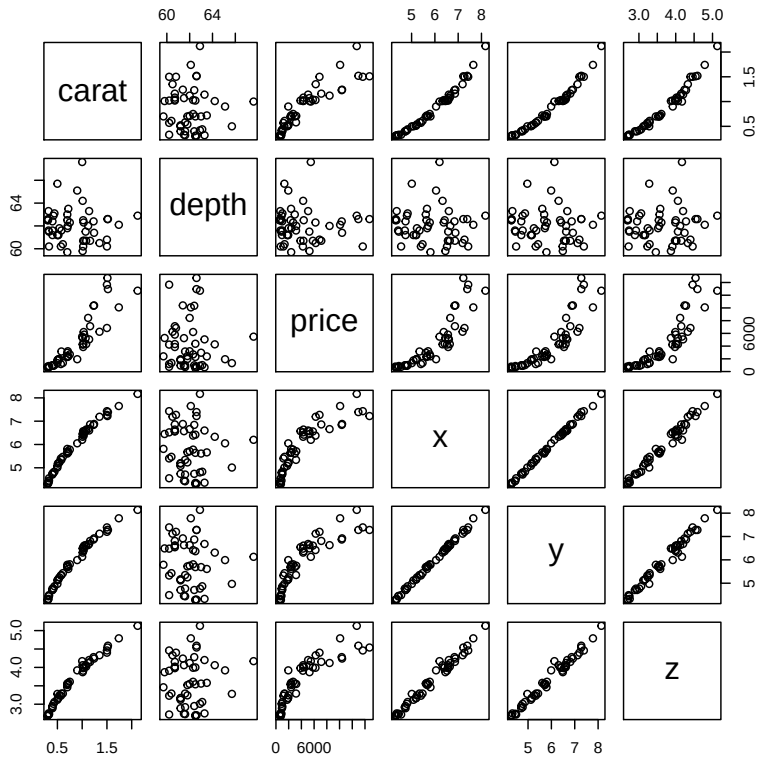
Pearson correlation of all pairs of quantitative variables

```
> cor(diamonds2[,c('carat','depth','price','x','y','z')])

              carat       depth      price           x
carat   1.00000000 -0.02969662  0.9087753  0.98219007
depth  -0.02969662  1.00000000 -0.0388721 -0.09155637
price   0.90877528 -0.03887210  1.0000000  0.87962993
x       0.98219007 -0.09155637  0.8796299  1.00000000
y       0.98108235 -0.09332294  0.8804835  0.99878864
z       0.98100572  0.04940613  0.8778840  0.98957213
                y           z
carat   0.98108235 0.98100572
depth  -0.09332294 0.04940613
price   0.88048354 0.87788405
x       0.99878864 0.98957213
y       1.00000000 0.98939476
z       0.98939476 1.00000000
```

Visualize all pairs

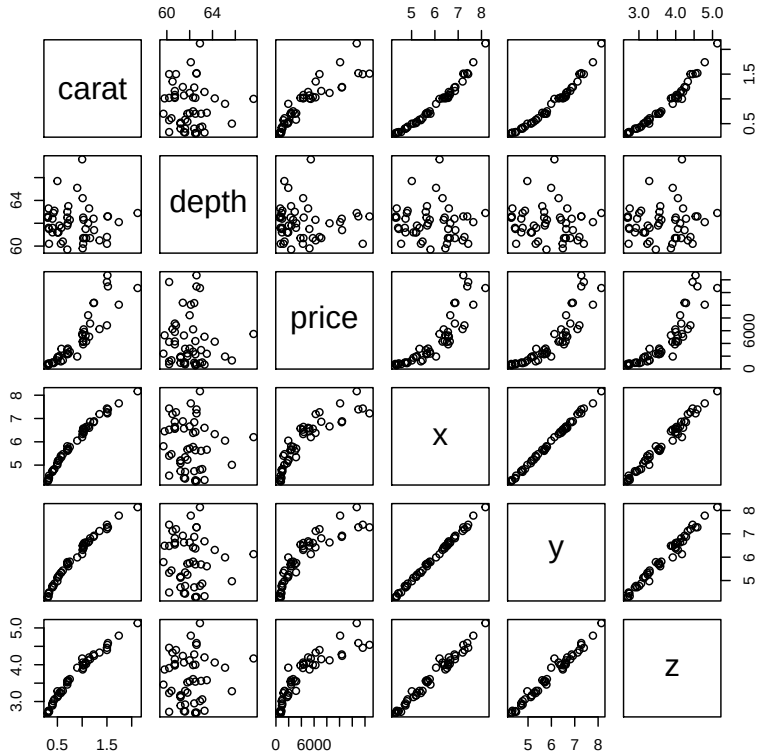
```
> pairs(diamonds2[,c('carat','depth','price','x','y','z')])
```



Visualize all pairs

Equivalent:

```
> library(dplyr)
> diamonds2 %>% select(carat,depth,price,x,y,z) %>% pairs()
```

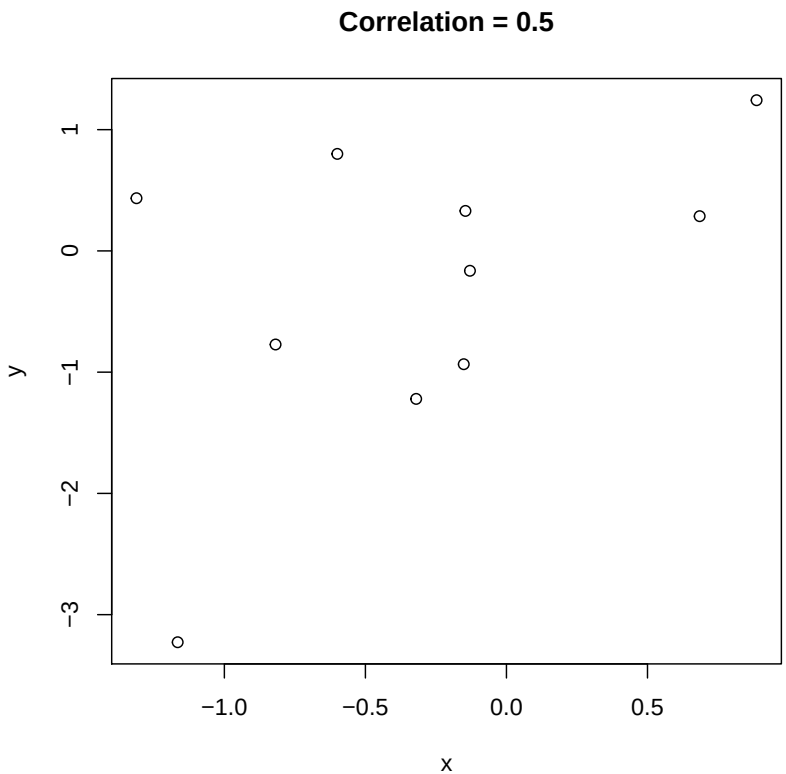


Problem: Usefull, but does not tell all the story (non-linearity? spurious associations? confounding and multiple associations?)

Correlation close to 0 indicates no association

Let's generate random and unrelated numbers from Normal distribution

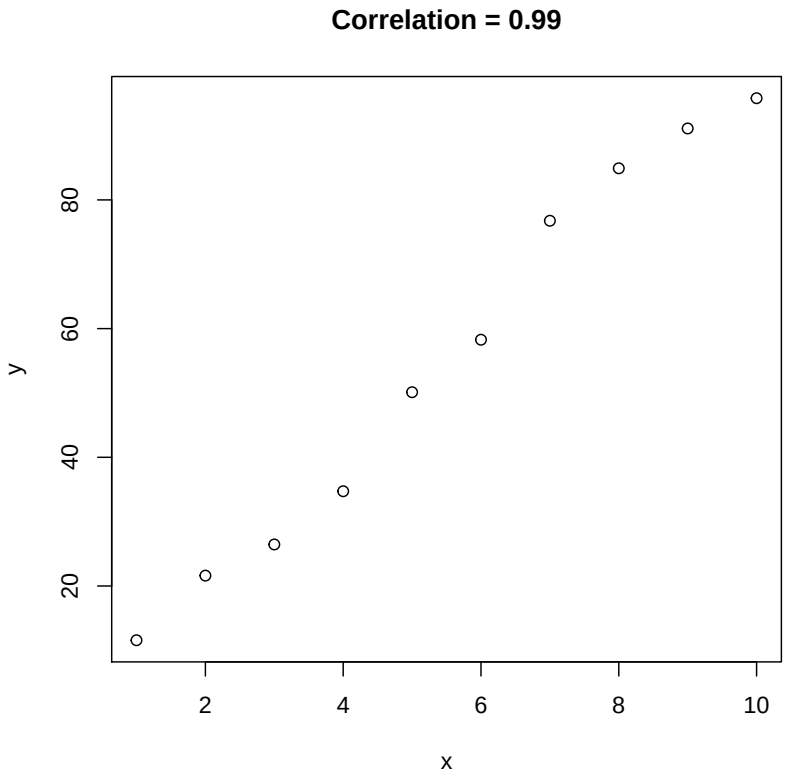
```
> x <- rnorm(10)
> y <- rnorm(10)
> plot(x, y, main=paste('Correlation =', round(cor(x,y),
+                                              digits=2)))
```



Correlation quantifies linear associations

A linear relationship has a high correlation:

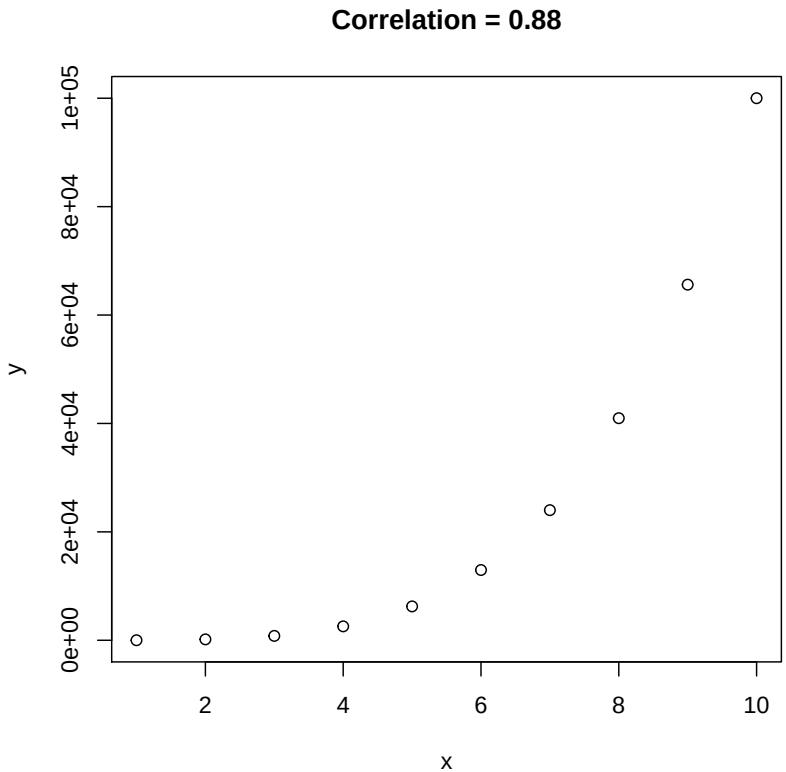
```
> x <- 1:10
> y <- 10*x + rnorm(10, sd=4)
> plot(x, y, main=paste('Correlation =', round(cor(x,y),
+                                              digits=2)))
```



Correlation quantifies linear associations

A non-linear relationship also has a high correlation:

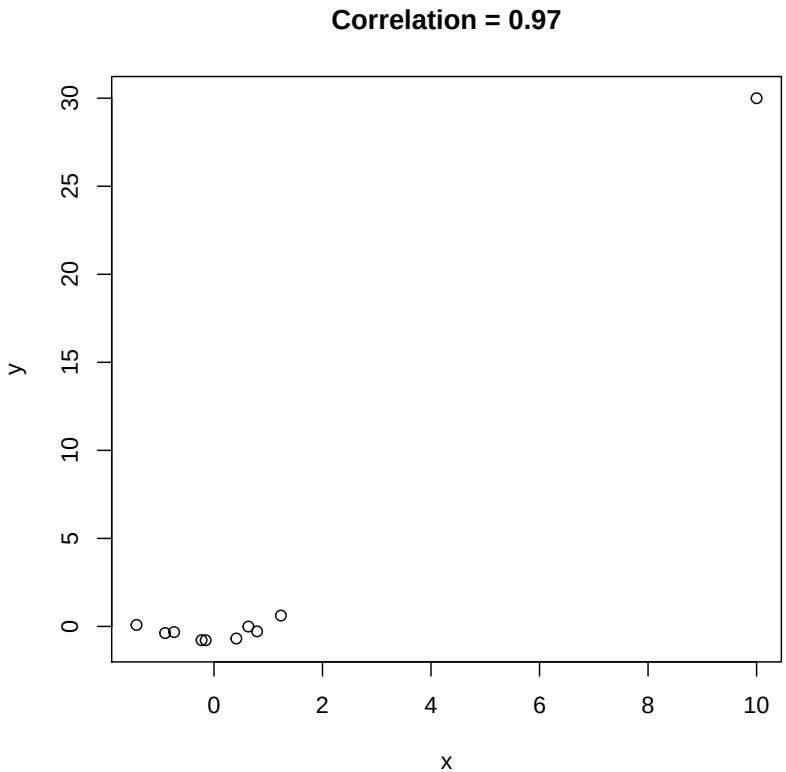
```
> x <- 1:10
> y <- 10*x^4 + rnorm(10, sd=4)
> plot(x, y, main=paste('Correlation =', round(cor(x,y),
+                                              digits=2)))
```



Correlation quantifies linear associations

A non-linear relationship with an outlier also has a high correlation:

```
> x <- c(rnorm(9), 10)
> y <- c(rnorm(9), 30)
> plot(x, y, main=paste('Correlation =', round(cor(x,y),
+                                              digits=2)))
```

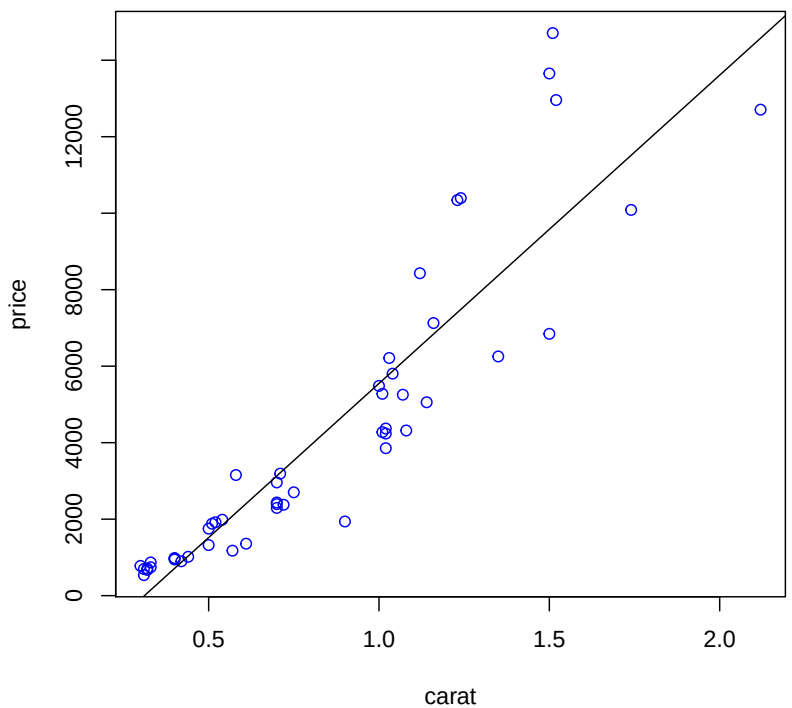


Linear regression

Linear regression extends correlation beyond pairwise associations

To quantify the association, fits a straight line. But important distinction: considers the probability distribution of y (here, price) as function of x (here, carat).

```
> plot(price ~ carat, data=diamonds2, col="blue")
> abline(lm(price ~ carat, data=diamonds2))
```



Extends correlation beyond pairwise associations

```
> summary(lm(price ~ carat, data=diamonds2))

Call:
lm(formula = price ~ carat, data = diamonds2)

Residuals:
    Min      1Q  Median      3Q     Max
-2803.9  -913.7   -20.2   583.3  5049.3

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -2511.5      502.6  -4.997 8.16e-06 ***
carat         8060.3      534.2  15.088  < 2e-16 ***
---
Signif. codes:
0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ

Residual standard error: 1613 on 48 degrees of freedom
Multiple R-squared:  0.8259,        Adjusted R-squared:  0.8222
F-statistic: 227.7 on 1 and 48 DF,  p-value: < 2.2e-16
```

Extracting parameters of model fit

```
> fit <- lm(price ~ carat, data=diamonds2)
> coef(fit)

(Intercept)       carat
  -2511.453    8060.345

> names(summary(fit))

 [1] "call"          "terms"         "residuals"
 [4] "coefficients"  "aliased"       "sigma"
 [7] "df"            "r.squared"     "adj.r.squared"
[10] "fstatistic"    "cov.unscaled"

> summary(fit)$r.squared

[1] 0.8258725
```

Prediction

```
> head(predict(fit))

    15512     42521     22060     47628     50726      2458
5790.702 1518.719 11513.547 1599.323 3130.788 3211.392
```

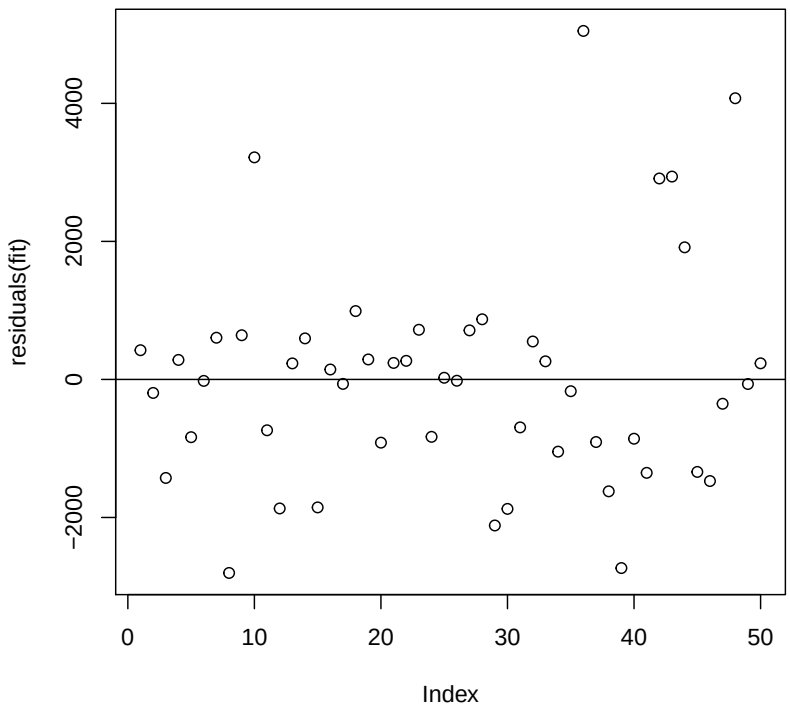
Residuals

```
> head(resid(fit), n=4)

    15512       42521       22060       47628
423.2978  -195.7192 -1427.5473   282.6774
```

Residual plot

```
> plot(residuals(fit))
> abline(h=0)
```



Statistical inference

Sampling distribution of the slope

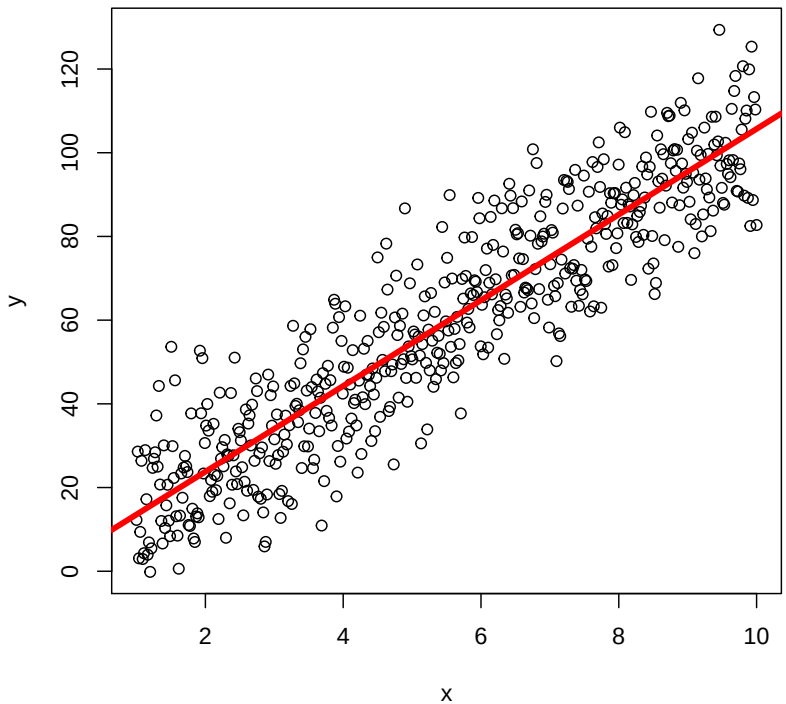
Simulation: sampling distribution depends on the variability of the data around the straight line, and on the sample size

```
> n <- 100
> param <- rep(NA, n)
> for (i in 1:n) {
+         x <- seq(from=1, to=10, length=500)
+         y <- 5 + 10*x + rnorm(length(x), sd=12)
+         param[i] <- coef(lm(y~x))[2]
+ }
```

Sampling distribution of the slope

Simulation: sampling distribution depends on the variability of the data around the straight line, and on the sample size

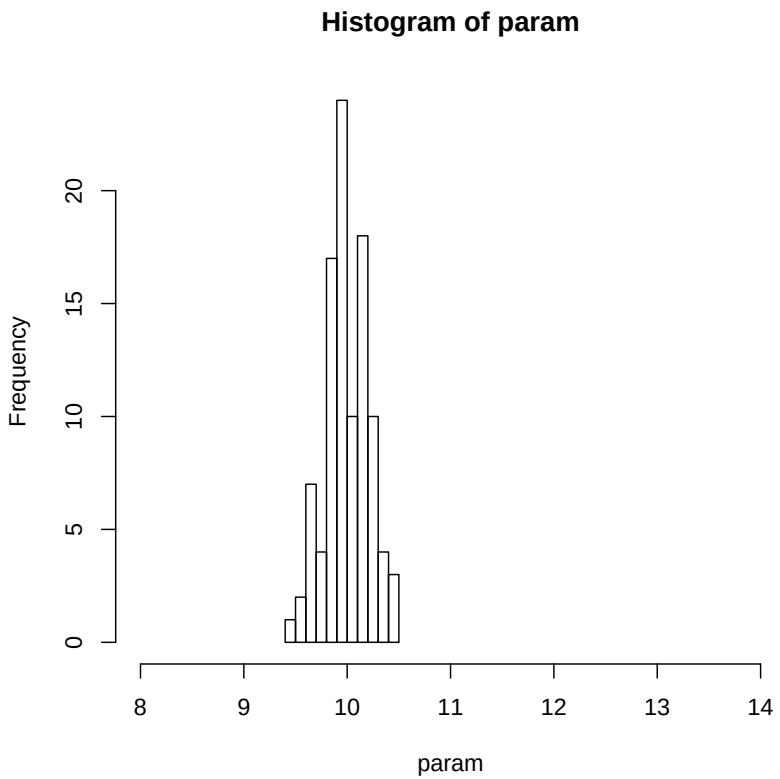
```
> plot(x,y)
> abline(lm(y~x), col='red', lwd=4)
```



Sampling distribution of the slope

Simulation: sampling distribution depends on the variability of the data around the straight line, and on the sample size

```
> hist(param, xlim=c(8,14))
```



Confidence intervals for parameters

Use ?predict.lm to see more options

```
> confint(fit)
```

```
                  2.5 %    97.5 %
(Intercept) -3522.082 -1500.825
carat        6986.249  9134.442
```

Confidence intervals for the line

Use ?predict.lm to see more options

```
> head(predict(fit, interval='confidence'))
```

```
             fit         lwr       upr
15512  5790.702  5287.9285  6293.476
42521  1518.719   933.4478  2103.990
22060 11513.547 10441.9885 12585.106
47628  1599.323  1020.6606  2177.985
50726  3130.788  2648.5774  3612.999
2458   3211.392  2732.3831  3690.400
```

Prediction intervals

Use ?predict.lm to see more options

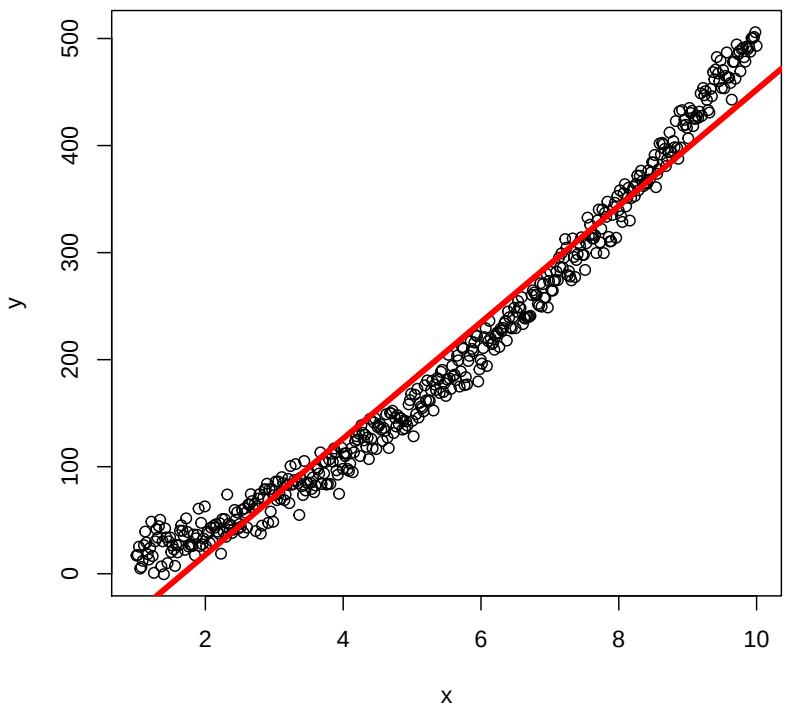
```
> head(predict(fit, interval='prediction'))
```

```
             fit          lwr       upr
15512  5790.702  2508.29006  9073.114
42521  1518.719 -1777.33763  4814.776
22060 11513.547  8097.45507 14929.640
47628  1599.323 -1695.56701  4894.212
50726  3130.788  -148.53719  6410.114
2458   3211.392   -67.46439  6490.248
```

Inference only holds if assumptions are correct

Example: true relationship is non-linear

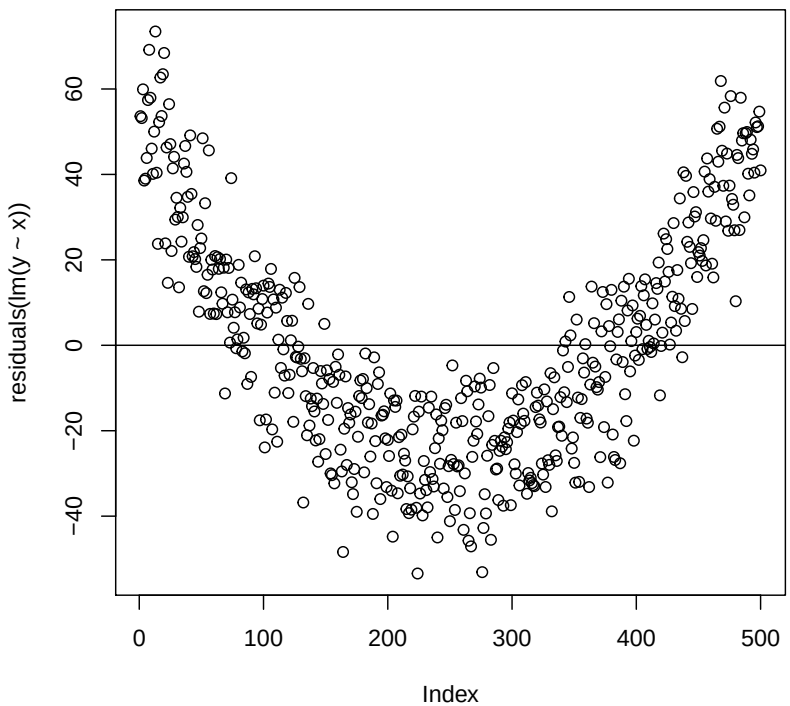
```
> x <- seq(from=1, to=10, length=500)
> y <- 5 + 10*x + 4*x^2 + rnorm(length(x), sd=12)
> plot(x, y)
> abline(lm(y~x), col='red', lwd=4)
```



Inference only holds if assumptions are correct

Example: true relationship is non-linear

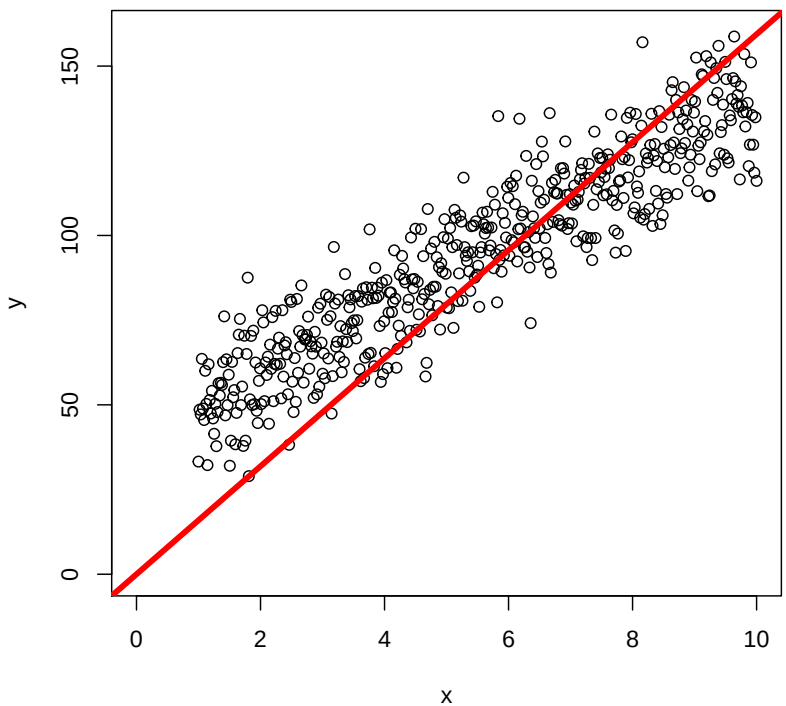
```
> plot(residuals(lm(y~x)))
> abline(h=0)
```



Inference only holds if assumptions are correct

Example: assuming no intercept may be wrong

```
> x <- seq(from=1, to=10, length=500)
> y <- 40 + 10*x + rnorm(length(x), sd=12)
> plot(x, y, xlim=c(0,10), ylim=c(0,160))
> abline(lm(y~x+0), col='red', lwd=4)
```



Inference only holds if assumptions are correct

Example: assuming no intercept may be wrong

```
> x <- seq(from=1, to=10, length=500)
> y <- 40 + 10*x + rnorm(length(x), sd=12)
> plot(residuals(lm(y~x+0)))
> abline(h=0)
```