ECN2160: Introduction

Alexandre Brilhante
Karim Itani

28 janvier 2018

0.1 Installer R

https://www.r-project.org/

0.2 Installer Jupyter

http://jupyter.org/

0.3 Installer IRKernel

https://irkernel.github.io/

Sur un terminal: jupyter notebook Un notebook comme celui-ci s'ouvrira sur votre fureteur.

0.4 Alternative: installer RStudio

https://www.rstudio.com/

0.5 Opérations de base

```
In [1]: x <- 345

        x
```

345

```
In [2]: log_x <- log2(x)

        log_x
```

8.43045255166553

```
In [3]: y <- 45

        x-y
```

300

In [4]:
```
z <- 300

      if (x-y == z) {
          return(TRUE)
      } else {
          return(FALSE)
      }
```

TRUE

In [5]:
```
z <- x-x

      z
```

0

In [6]:
```
a <- 24
      b <- 3

      mult <- function(a, b) {
          result <- 0
          for (i in 1:b) {
              result <- result+a
          }
          return(result)
      }

      mult(a, b)
```

72

In [7]:
```
a*b
```

72

0.6 Série chronologique

In [8]:
```
# On génère 12 valeurs aléatoires.

      input <- rnorm(12)
```

In [9]:
```
input
```

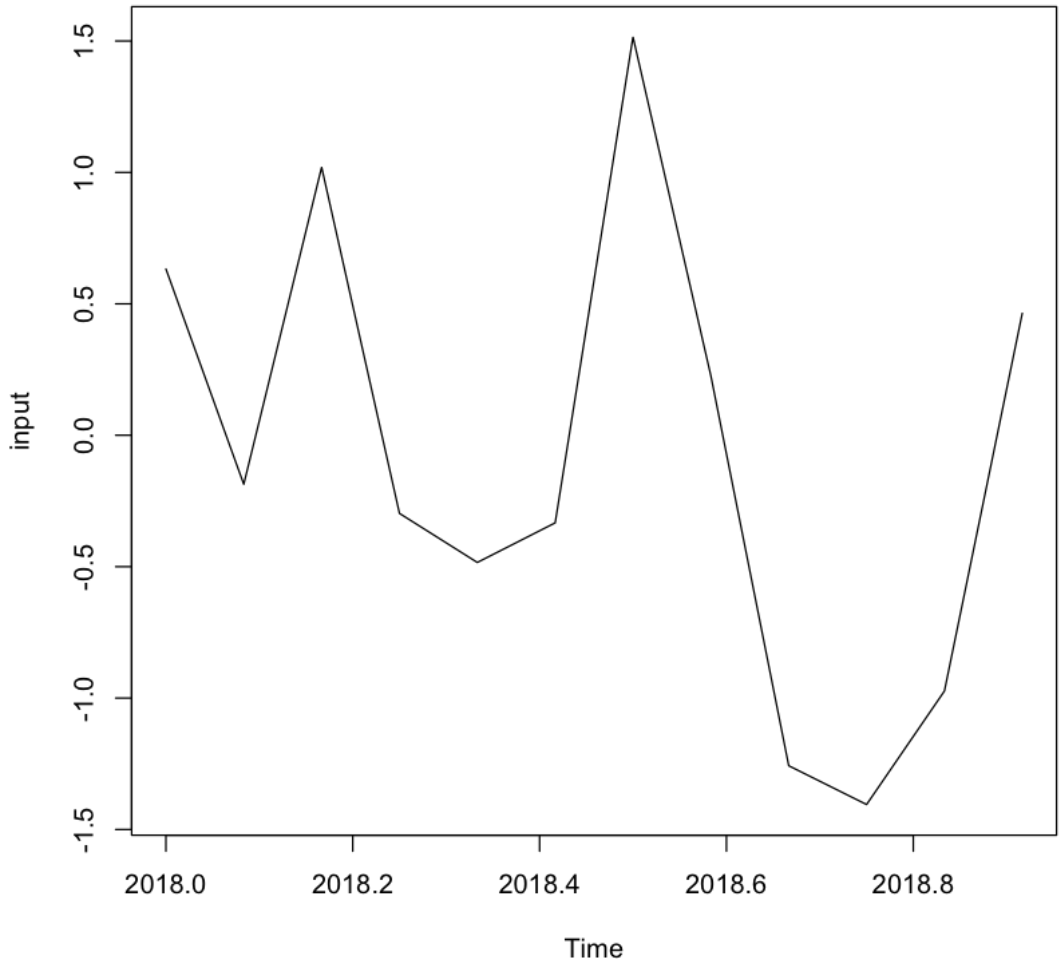
1. 0.63268462382664 2. -0.186421772308832 3. 1.01887968323517 4. -0.297304114335721 5. -0.483965435795269 6. -0.33327251542413 7. 1.51388337749175 8. 0.225710957771721 9. -1.25736090221971 10. -1.40500551148294 11. -0.97308907337095 12. 0.464128996757302

In [10]:
```
# On convertit nos valeurs en série chronologique.

       input <- ts(input, start = c(2018, 1), frequency = 12)
```

In [11]: input

```
            Jan        Feb        Mar        Apr        May        Jun
2018  0.6326846 -0.1864218  1.0188797 -0.2973041 -0.4839654 -0.3332725
            Jul        Aug        Sep        Oct        Nov        Dec
2018  1.5138834  0.2257110 -1.2573609 -1.4050055 -0.9730891  0.4641290
```

In [12]: plot.ts(input)

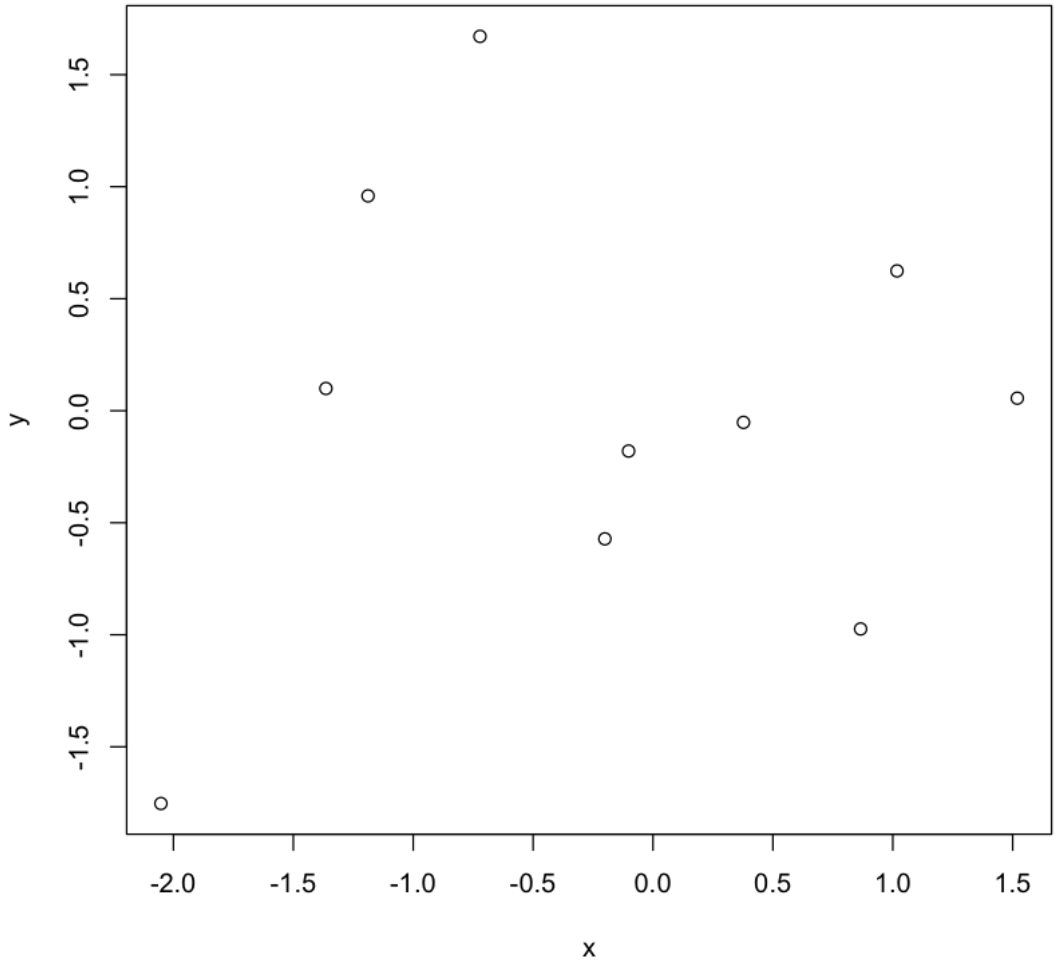


0.7 Nuage de point

In [25]: *# On génère 10 points aléatoires.*

```
x <- rnorm(10)
y <- rnorm(10)
```

In [26]:
```
plot(x, y)
```



0.8 Régression linéaire

In [27]:
```
# Création de données.
set.seed(123)

x <- 1:20 + rnorm(100, sd = 3)
z <- 1:20/4 + rnorm(100, sd = 2)
y <- -2*x + x*z/5 + 3 + rnorm(100, sd = 4)
```

```
# On créé un dataframe avec nos données.
dat <- data.frame(x = x, y = y, z = z)
```

```
head(dat)
tail(dat)
```

x	y	z
-0.6814269	13.317660	-1.1708131
1.3094675	5.896216	1.0137674
7.6761249	-13.018866	0.2566162
4.2115252	-2.993443	0.3049148
5.3878632	-10.136997	-0.6532371
11.1451950	-18.052556	1.4099446

	x	y	z
95	19.08196	-22.847579	1.128397
96	14.19922	-2.428712	7.994427
97	23.56200	-10.967331	5.451418
98	22.59783	-38.571634	1.997457
99	18.29290	-20.595608	3.527668
100	16.92074	-16.944758	2.629040

Corrélation

In [16]:
```
cor(dat$x, dat$y)
```

-0.789499506110698

Régression linéaire simple

In [17]:
```
fit <- lm(dat$y ~ dat$x)
```

```
fit
```

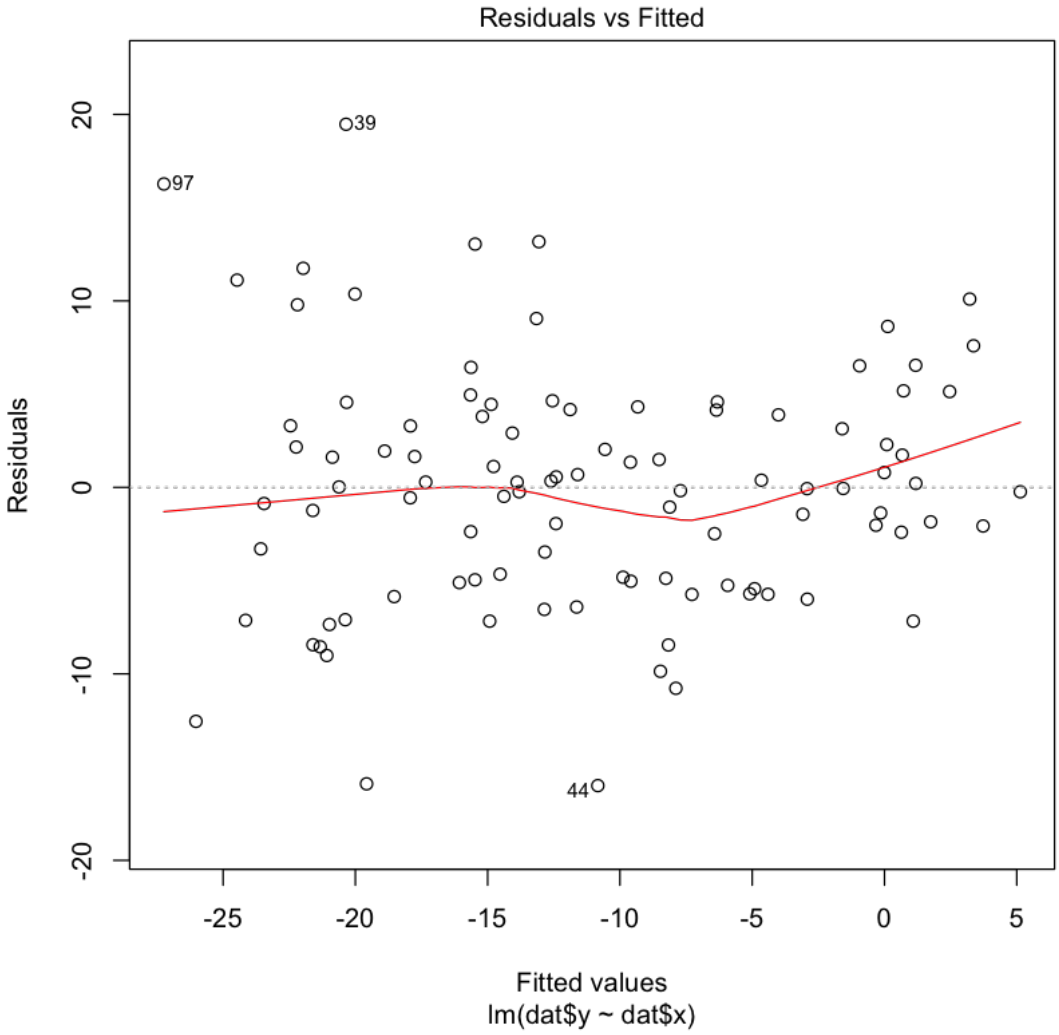
```
Call:
lm(formula = dat$y ~ dat$x)

Coefficients:
(Intercept)        dat$x
      2.365       -1.256
```

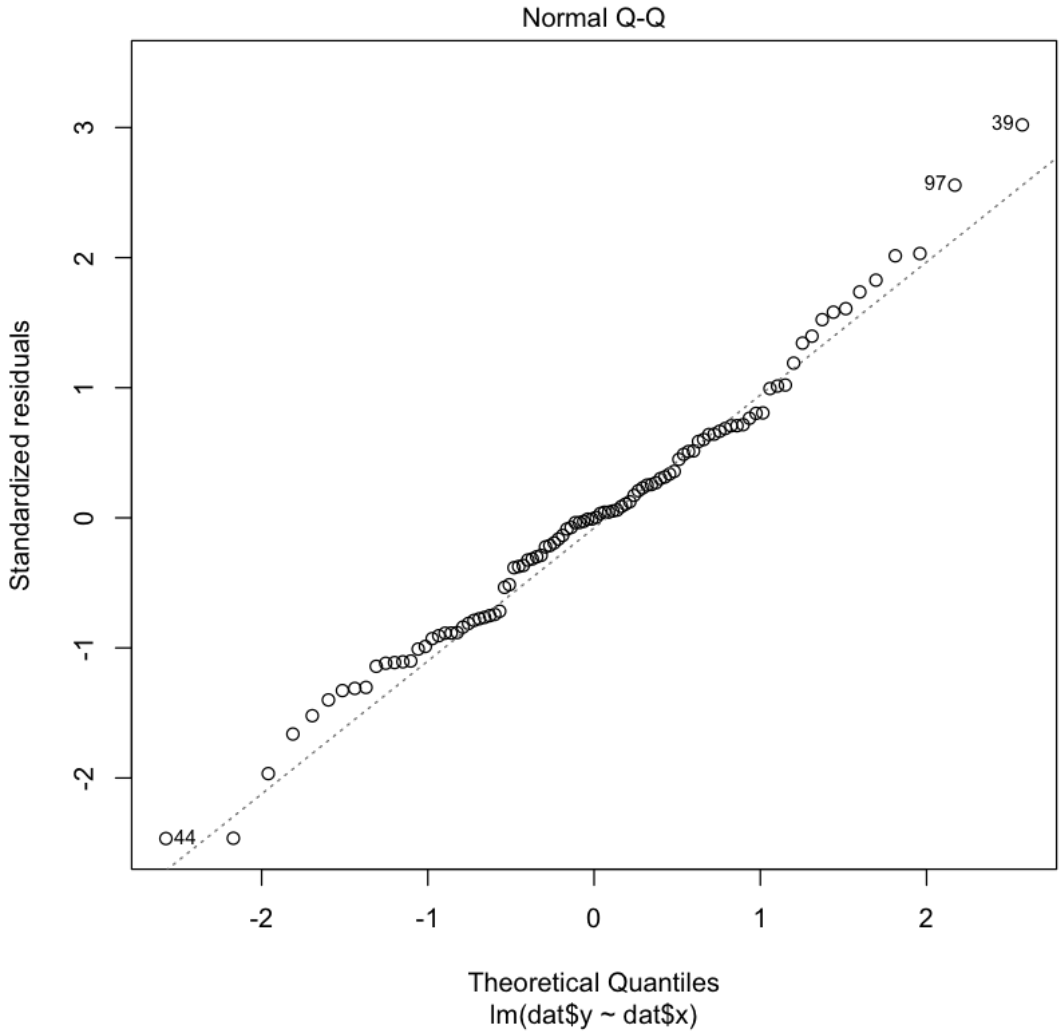
Cela veut donc dire que y = 2.365 - 1.256x.

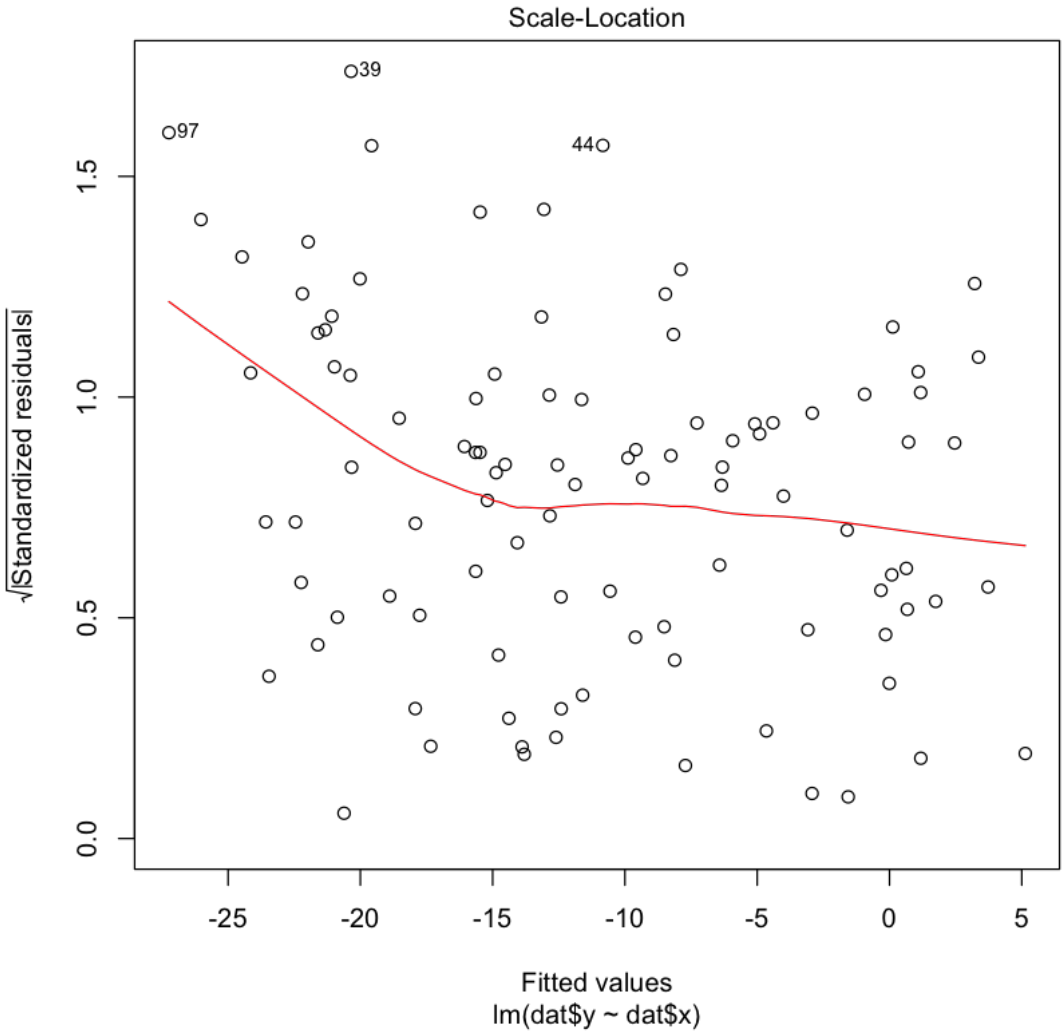
In [18]:
```
summary(fit)
```

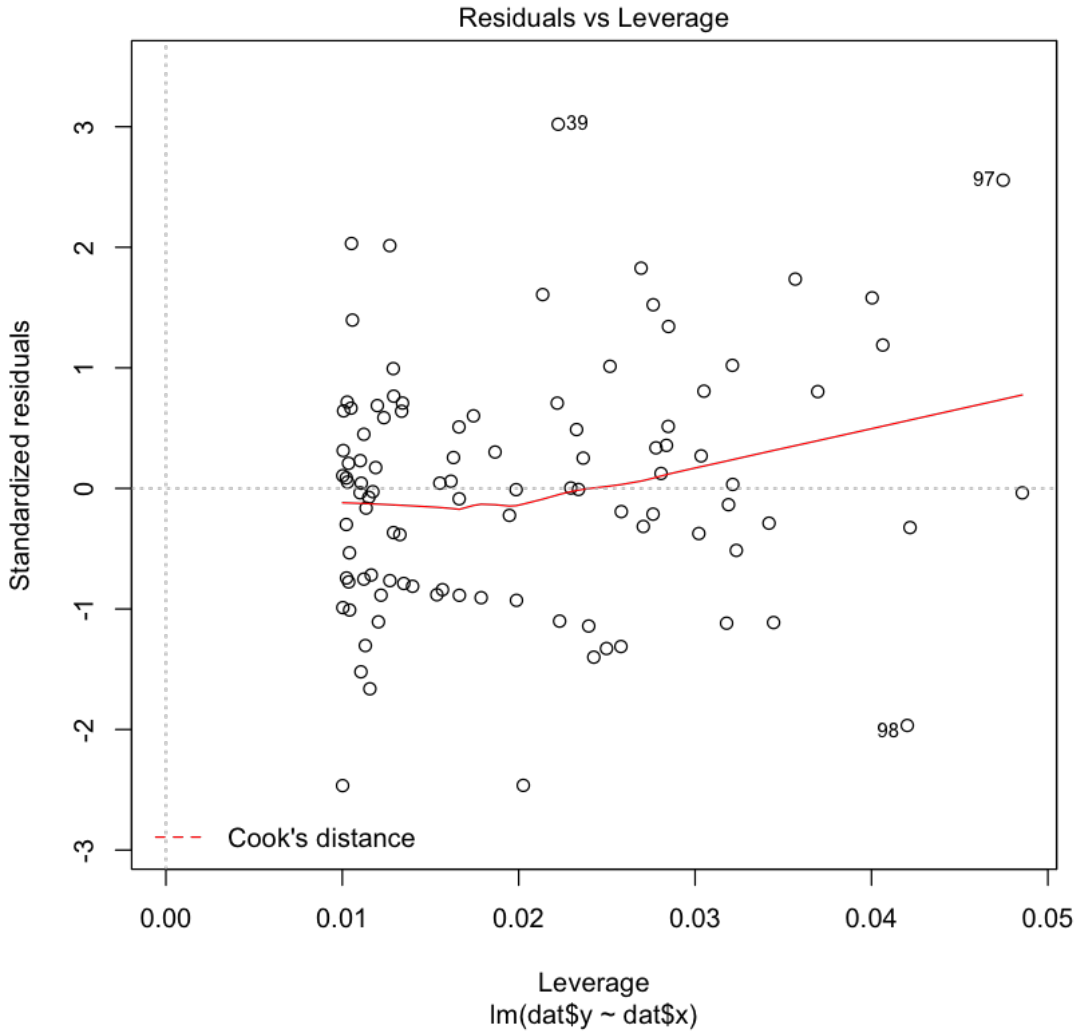
```
Call:
lm(formula = dat$y ~ dat$x)

Residuals:
     Min       1Q   Median       3Q      Max
-15.9923  -4.9756  -0.0181   3.9530  19.4712

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.36489    1.24670   1.897   0.0608 .
dat$x       -1.25632    0.09866 -12.734   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.52 on 98 degrees of freedom
Multiple R-squared:  0.6233,Adjusted R-squared:  0.6195
F-statistic: 162.2 on 1 and 98 DF,  p-value: < 2.2e-16
```

In [19]:
```
plot(fit)
```


Residuals vs Fitted
Residuals
Fitted values
lm(dat$y ~ dat$x)
39
97
44
20
10
0
-10
-20
-25
-20
-15
-10
-5
0
5


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(dat$y ~ dat$x)
39
97
44
-2
-1
0
1
2
3


Scale-Location
39
97
44
√|Standardized residuals|
0.0
0.5
1.0
1.5
-25
-20
-15
-10
-5
0
5
Fitted values
lm(dat$y ~ dat$x)



Régression linéaire multiple

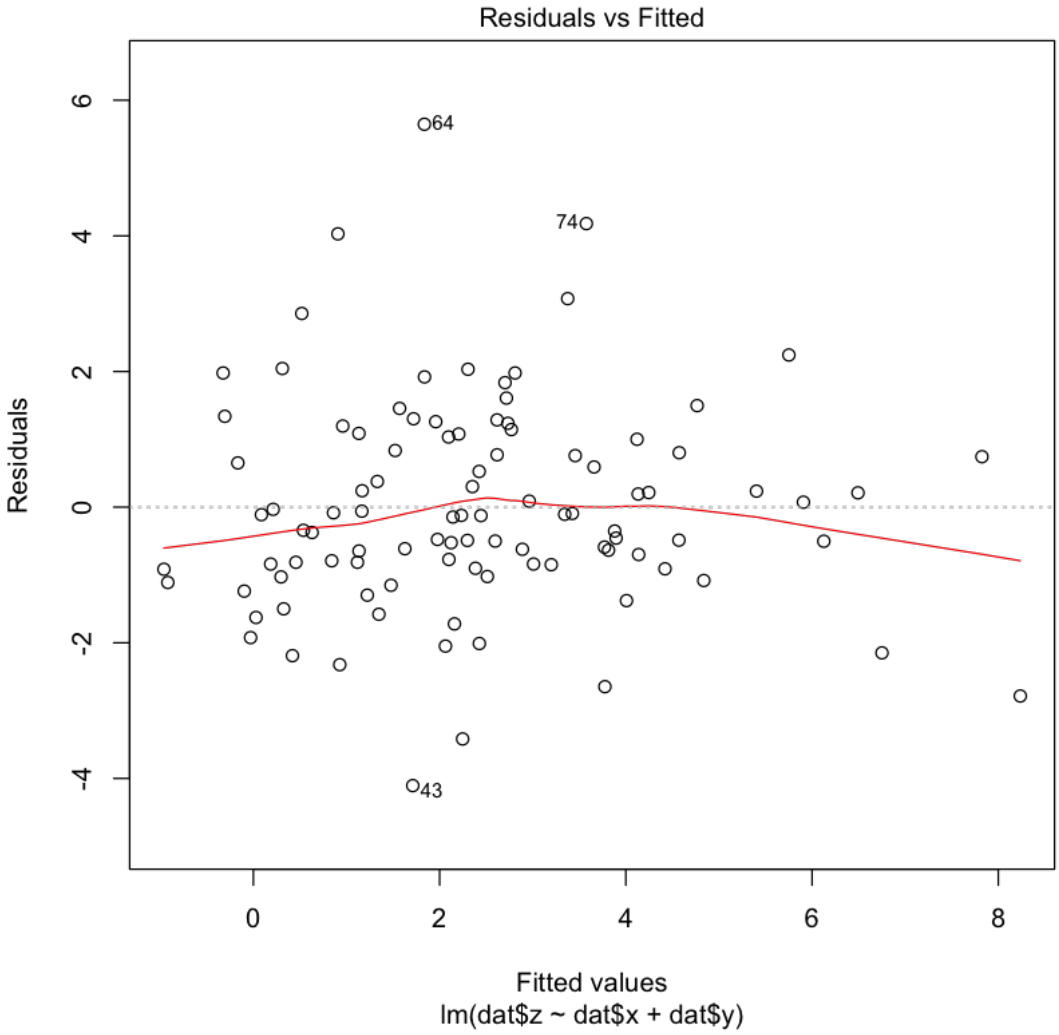
```
In [20]: fit <- lm(dat$z ~ dat$x + dat$y)

         fit


Call:
lm(formula = dat$z ~ dat$x + dat$y)

Coefficients:
(Intercept)        dat$x        dat$y
    -0.1735       0.4523       0.2049
```

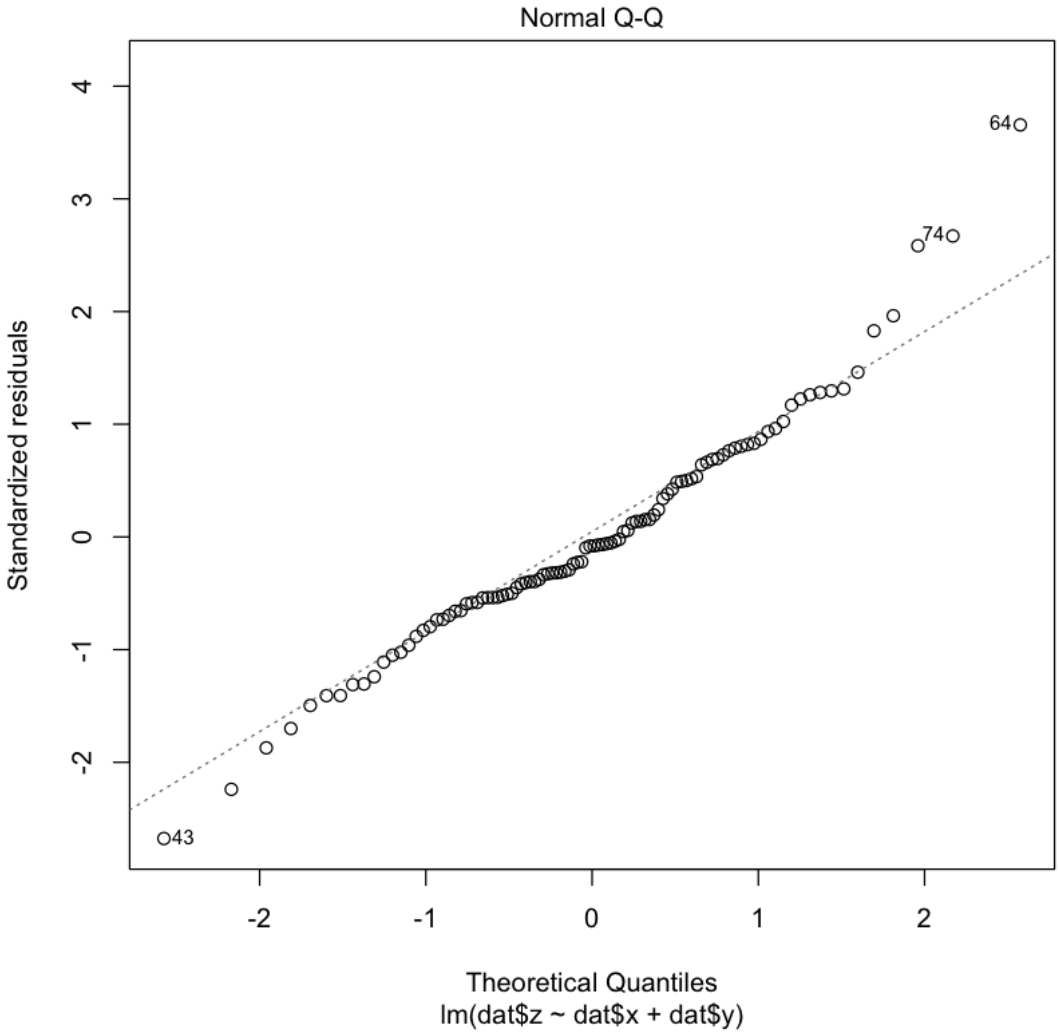
Cela veut donc dire que z = -0.1735 + 0.4523x + 0.2049y.

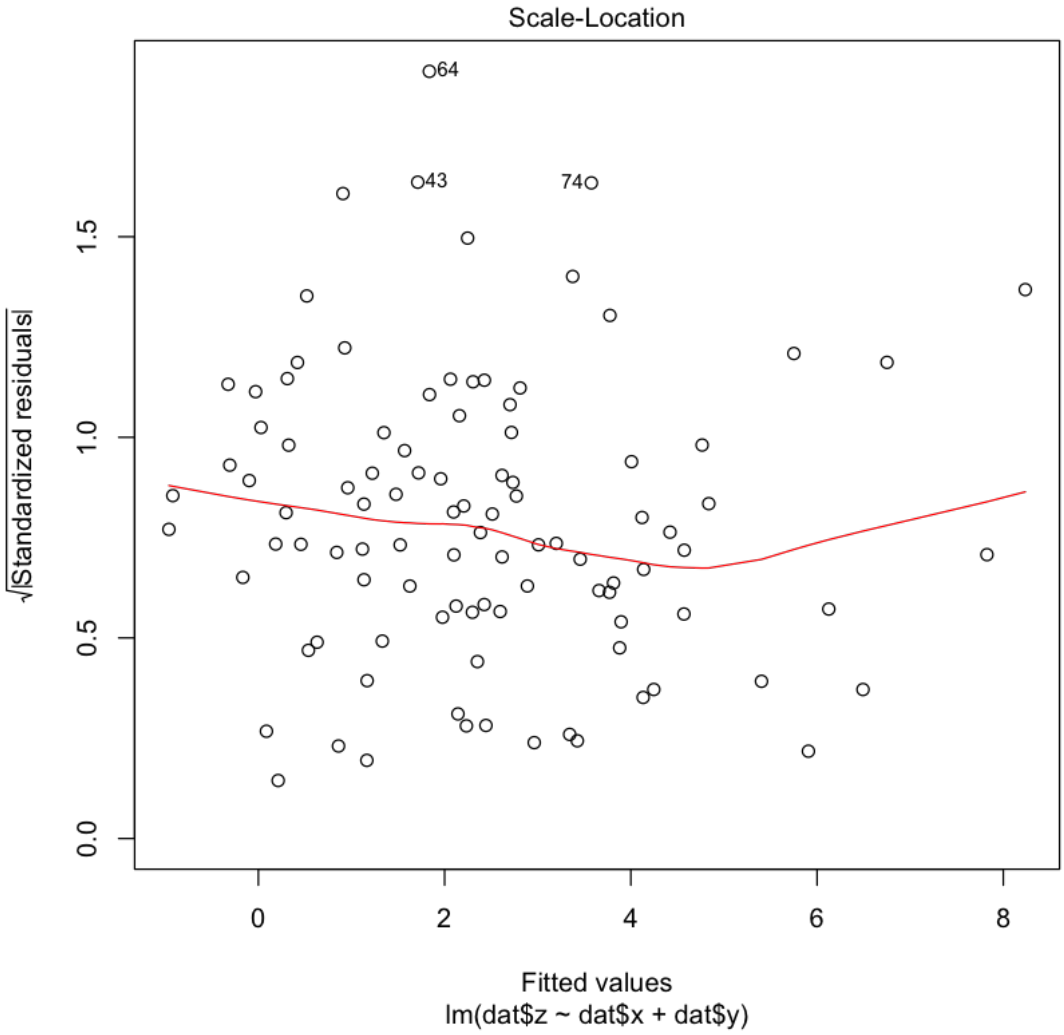
In [21]: `summary(fit)`

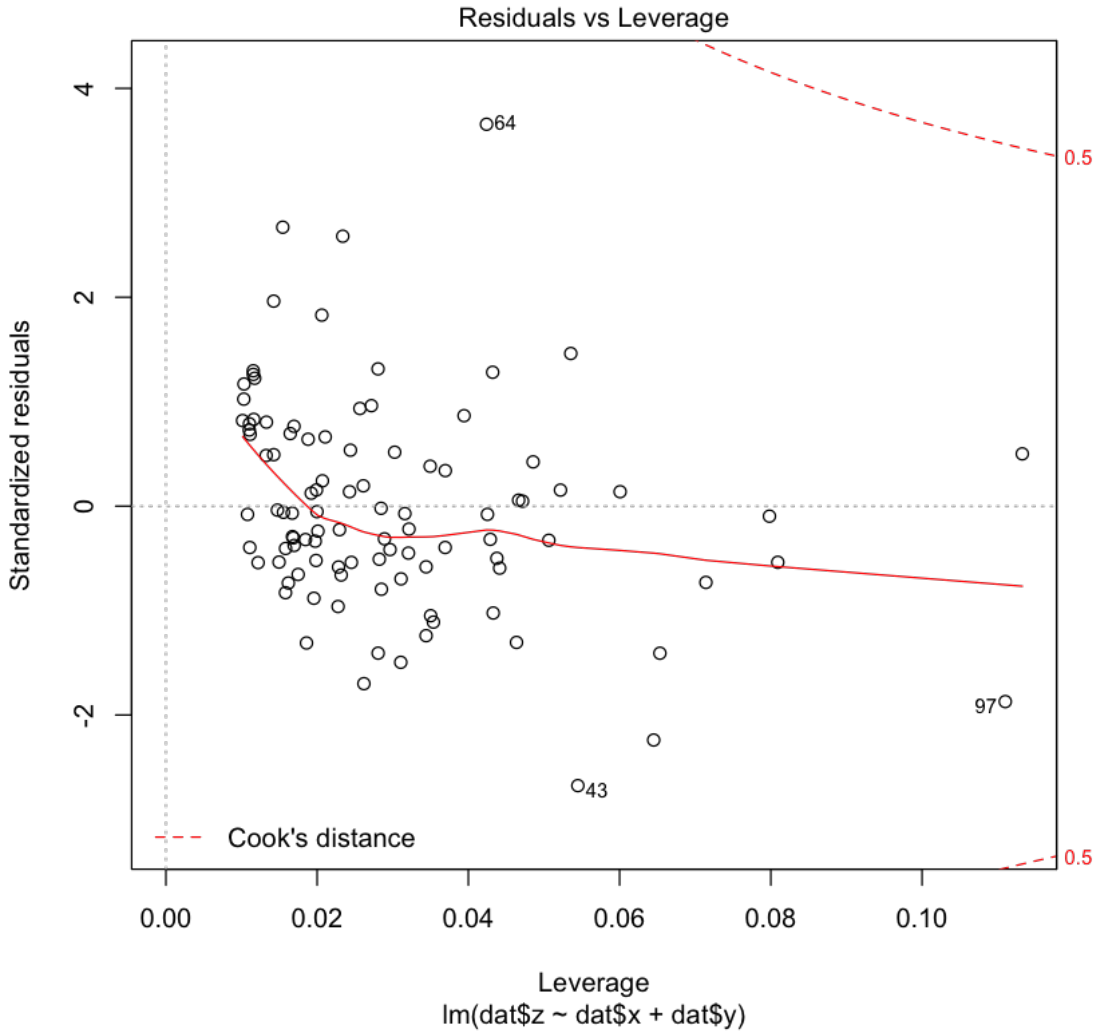
```
Call:
lm(formula = dat$z ~ dat$x + dat$y)

Residuals:
    Min      1Q  Median      3Q     Max
-4.1060 -0.8619 -0.1232  1.0091  5.6455

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.17355    0.30721  -0.565    0.573
dat$x        0.45229    0.03890  11.626  < 2e-16 ***
dat$y        0.20491    0.02445   8.382 4.12e-13 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.578 on 97 degrees of freedom
Multiple R-squared:  0.5852,Adjusted R-squared:  0.5767
F-statistic: 68.43 on 2 and 97 DF,  p-value: < 2.2e-16
```

In [22]: `plot(fit)`


Residuals vs Fitted
Residuals
Fitted values
lm(dat$z ~ dat$x + dat$y)
64
74
43
6
4
2
0
-2
-4
0
2
4
6
8


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(dat$z ~ dat$x + dat$y)
64
74
43
-2
-1
0
1
2
3
4


Scale-Location
64
43
74
√|Standardized residuals|
0.0
0.5
1.0
1.5
0
2
4
6
8
Fitted values
lm(dat$z ~ dat$x + dat$y)


Residuals vs Leverage
Standardized residuals
Leverage
lm(dat$z ~ dat$x + dat$y)
Cook's distance
0.5
0.5
64
97
43
4
2
0
-2
0.00
0.02
0.04
0.06
0.08
0.10