ECN2160: Introduction

Alexandre Brilhante
Karim Itani

28 janvier 2018

0.1 Étape 1: Installer R

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

0.2 Étape 2: Installer Jupyter

http://jupyter.org/

0.3 Étape 3: Installer IRKernel

https://irkernel.github.io/

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

0.4 Alternative: installer RStudio

https://www.rstudio.com/

0.5 Documentation complète:

https://www.rdocumentation.org/

0.6 Opérations de base

```
In [1]: # Ceci est un commentaire.
```

```
In [2]: # Affectation de variable.

        x <- 345

        x
```

345

```
In [3]: # Le logarithme d'une valeur x.

        log_x <- log2(x)

        log_x
```

```
8.43045255166553
```

```
In [4]: y <- 45

        x-y
```

```
300
```

```
In [5]: # Control Flow.

        z <- 300

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

```
TRUE
```

```
In [6]: z <- x-x

        z
```

```
0
```

```
In [7]: # Fonction mult qui reproduit l'opérateur *.

        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 [8]: a*b
```

```
72
```

0.7 Série chronologique

```
In [9]: # On génère 12 valeurs aléatoires suivant une loi normale.

        input <- rnorm(12)
```

```
In [10]: input
```

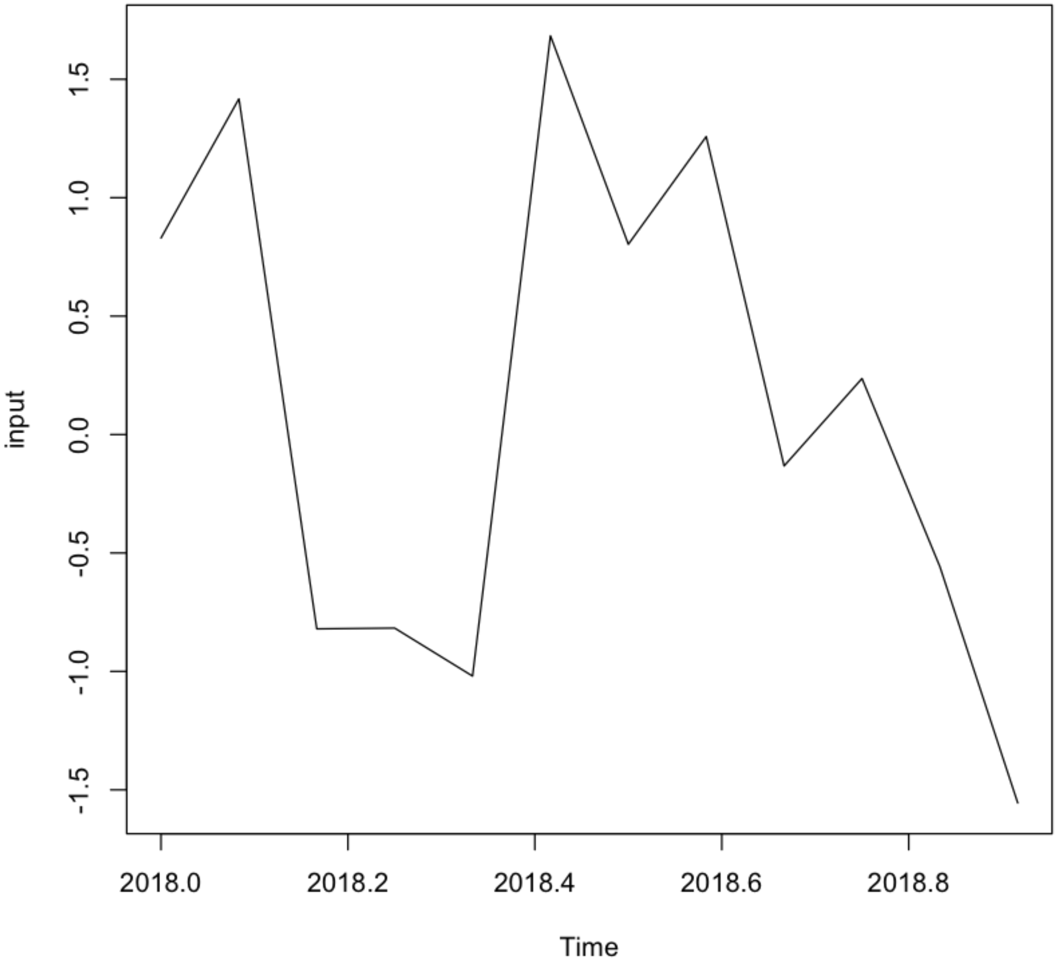
1. 0.829562920898473 2. 1.41701757322719 3. -0.820347605407959 4. -0.817118072363014
5. -1.01973932139853 6. 1.68347453674299 7. 0.803266615385764 8. 1.25785373279933
9. -0.132573407799251 10. 0.236549219777952 11. -0.557677594791627 12. -1.55571209332311

```
In [11]: # On convertit nos valeurs en série chronologique.

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

```
In [12]: input
```

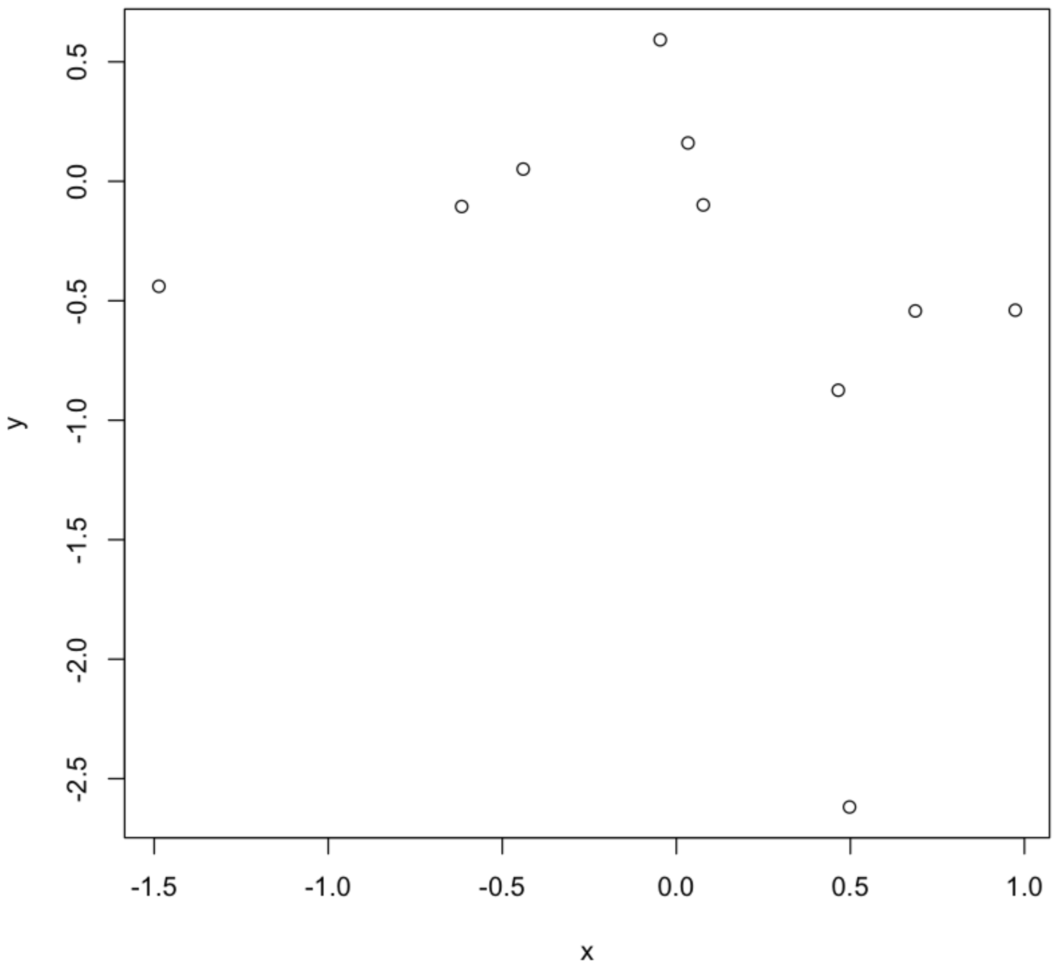
```
            Jan        Feb        Mar        Apr        May        Jun
2018  0.8295629  1.4170176 -0.8203476 -0.8171181 -1.0197393  1.6834745
            Jul        Aug        Sep        Oct        Nov        Dec
2018  0.8032666  1.2578537 -0.1325734  0.2365492 -0.5576776 -1.5557121
```

```
In [13]: plot.ts(input)
```



0.8 Nuage de point

```
In [14]: # On génère 10 points aléatoires.

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

In [15]: plot(x, y)
```



0.9 Régression linéaire

```
In [16]: # 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 [17]: cor(dat$x, dat$y)
```

-0.789499506110698

Régression linéaire simple

```
In [18]: fit <- lm(dat$y ~ dat$x)

         # Alternative: fit <- lm(formula = y ~ x, data = dat)
         # Alternative: fit <- lm(y ~ x, data = dat)

         fit
```

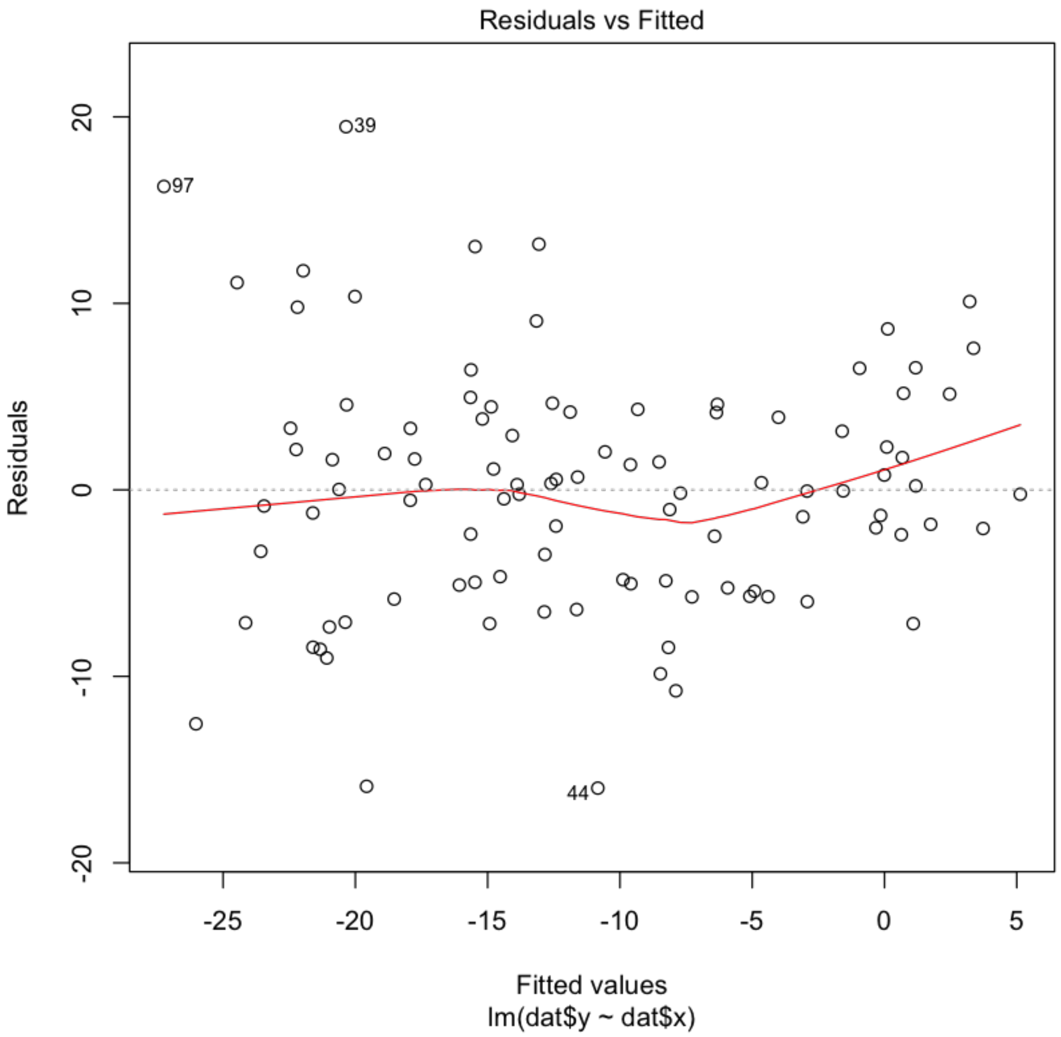
```
Call:
lm(formula = dat$y ~ dat$x)

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

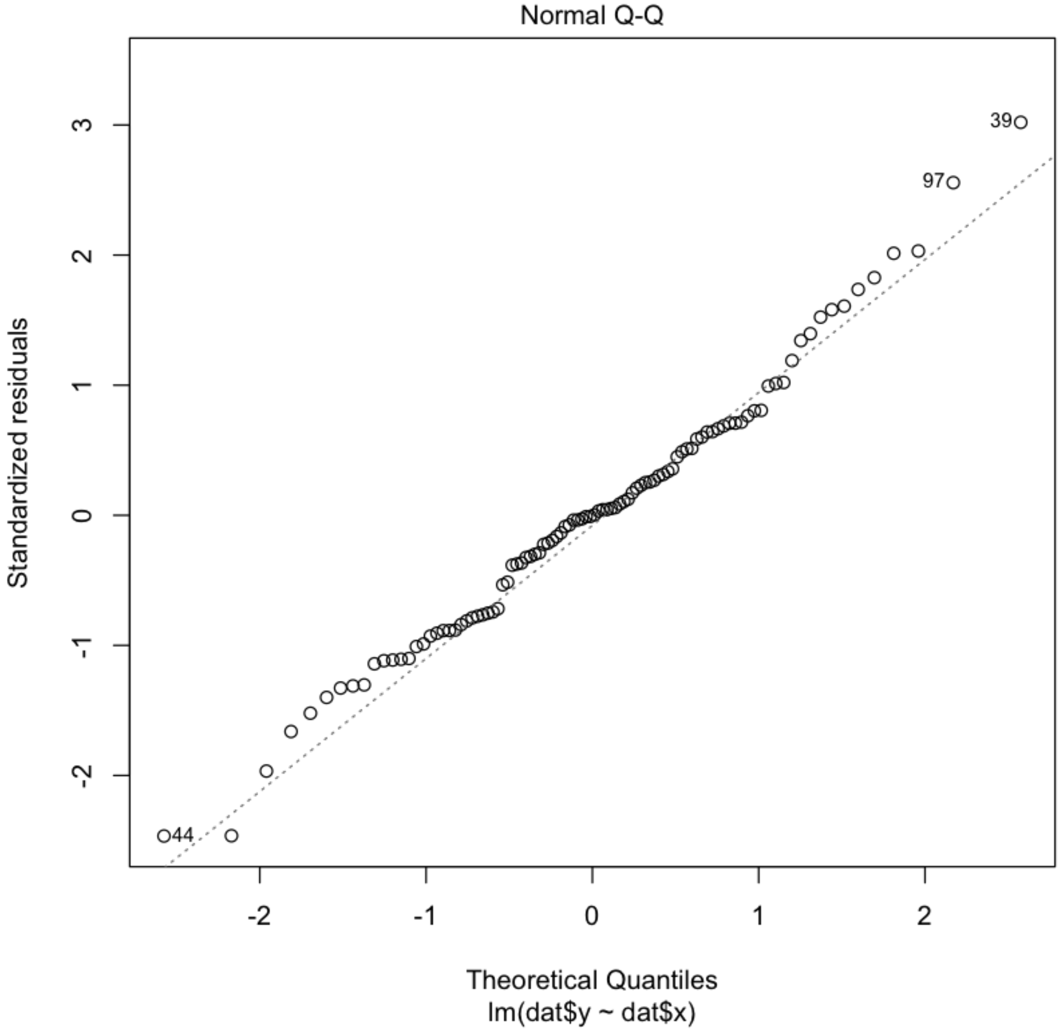
Cela veut donc dire que y = 2.365 - 1.256x.

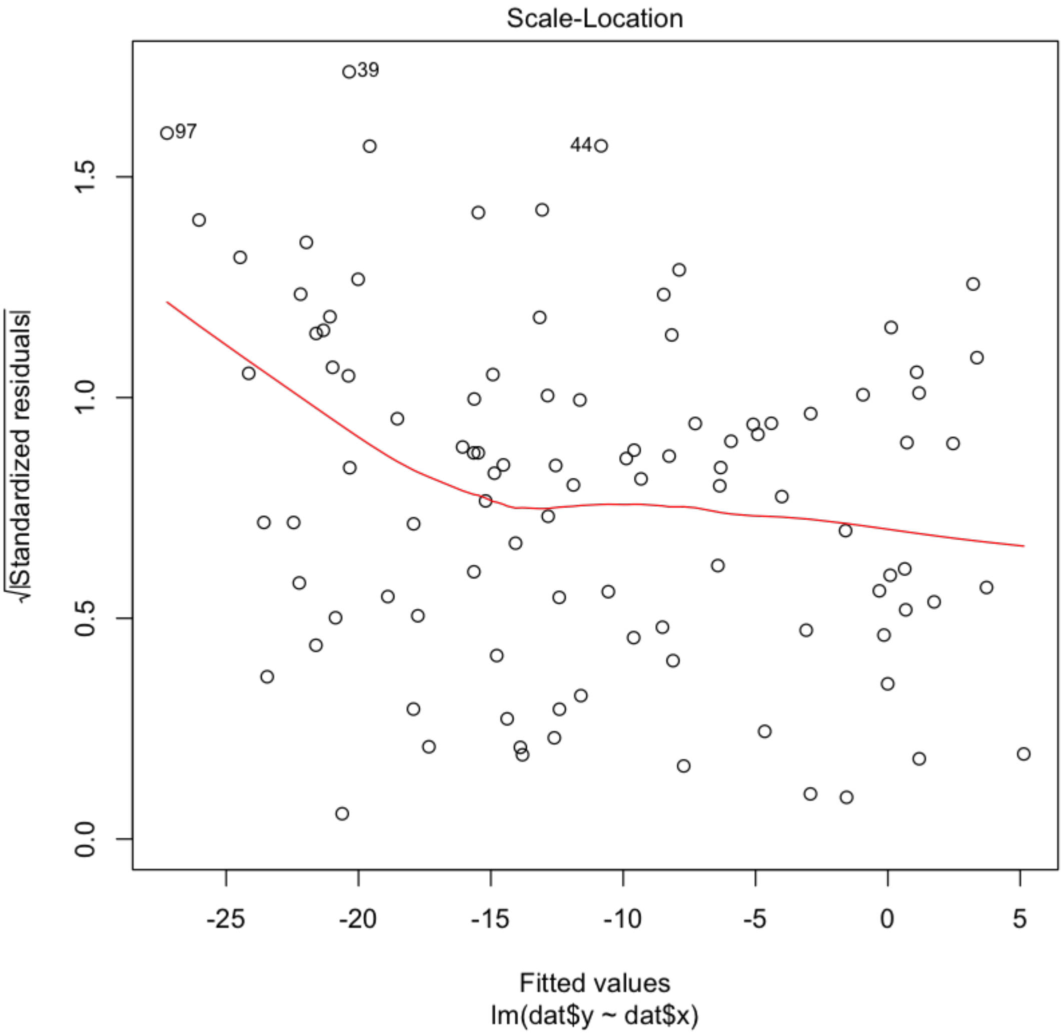
```
In [19]: summary(fit)
```

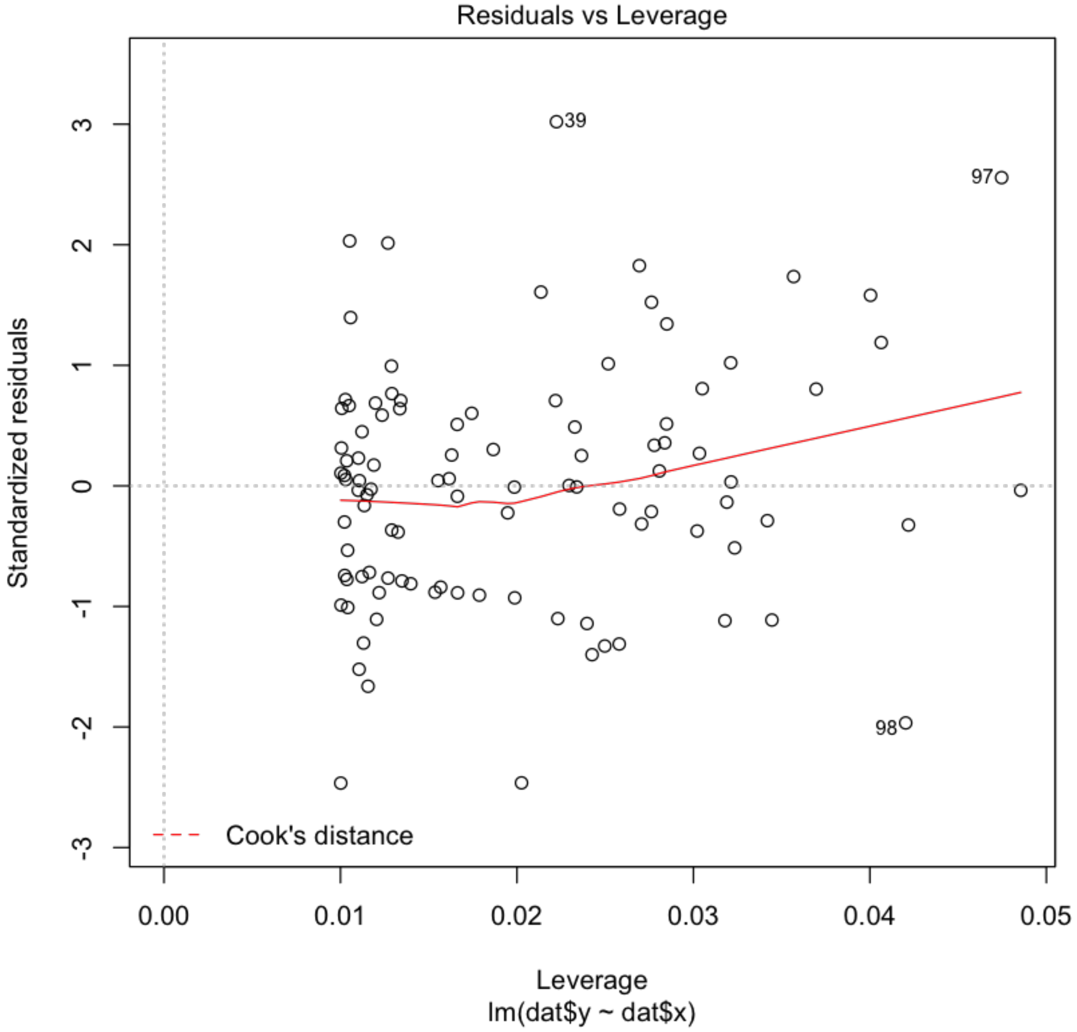
```
Call:
lm(formula = dat$y ~ dat$x)

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 [20]: plot(fit)
```


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


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


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



Régression linéaire multiple

```
In [21]: fit <- lm(dat$z ~ dat$x + dat$y)

         # Alternative: fit <- lm(formula = x ~ x + y, data = dat)
         # Alternative: fit <- lm(z ~ x + y, data = dat)

         fit
```

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

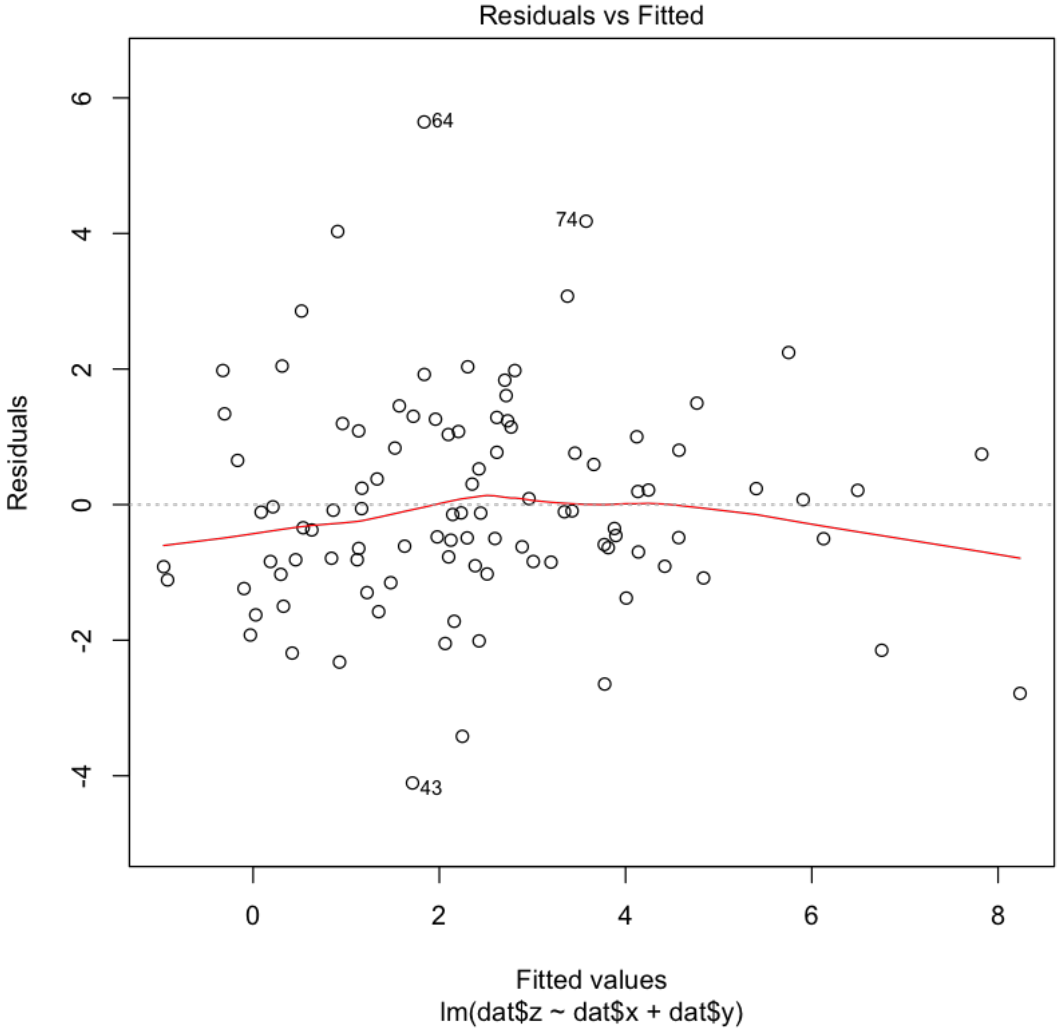
```
Coefficients:
(Intercept)       dat$x       dat$y
    -0.1735      0.4523      0.2049
```

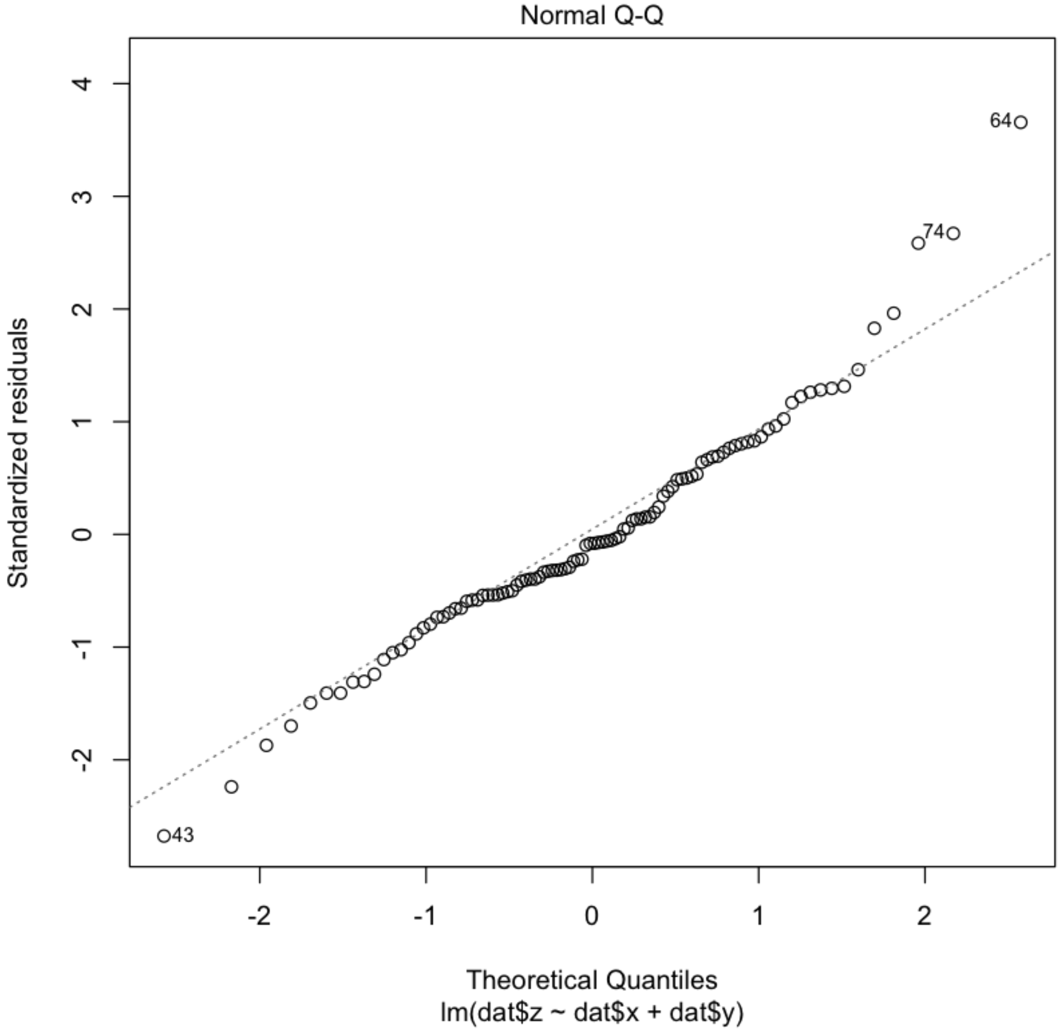
Cela veut donc dire que z = -0.1735 + 0.4523x + 0.2049y.

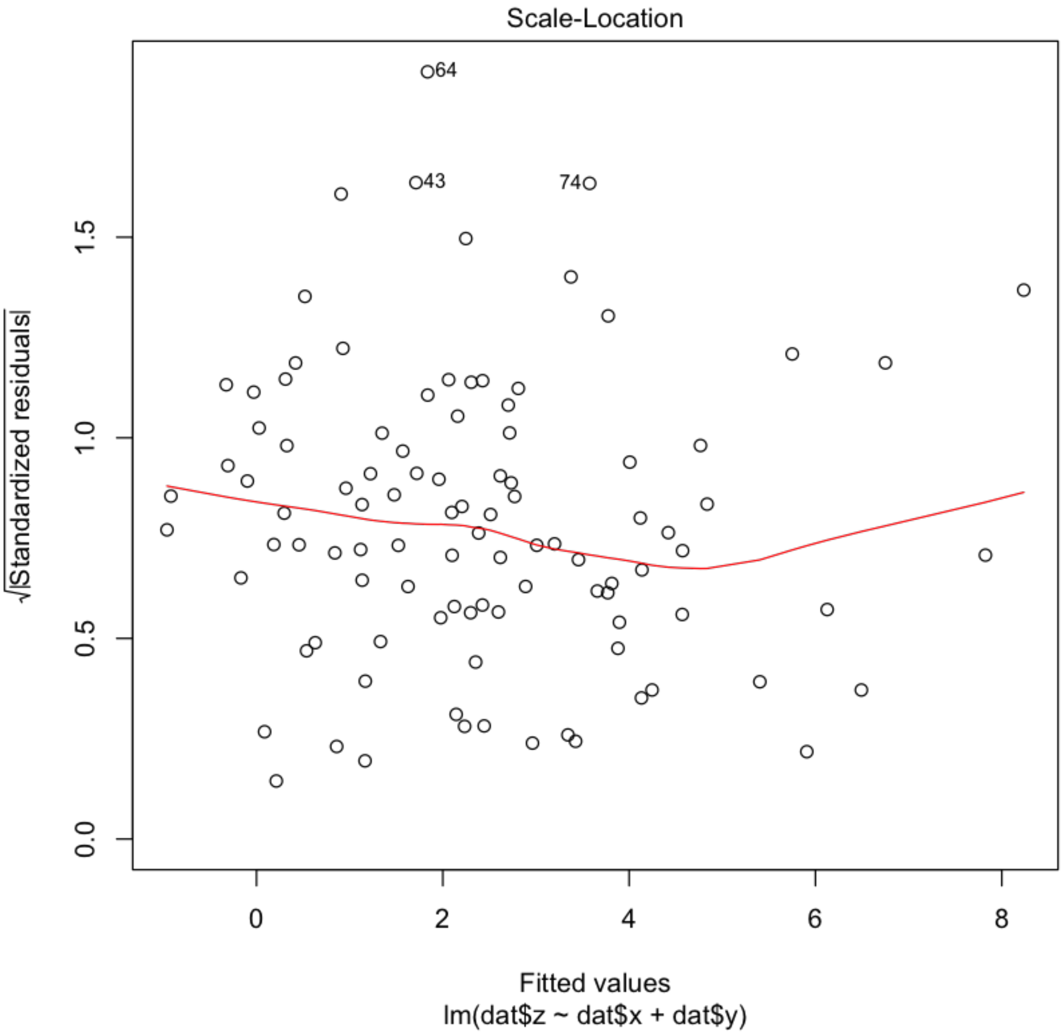
```
In [22]: summary(fit)
```

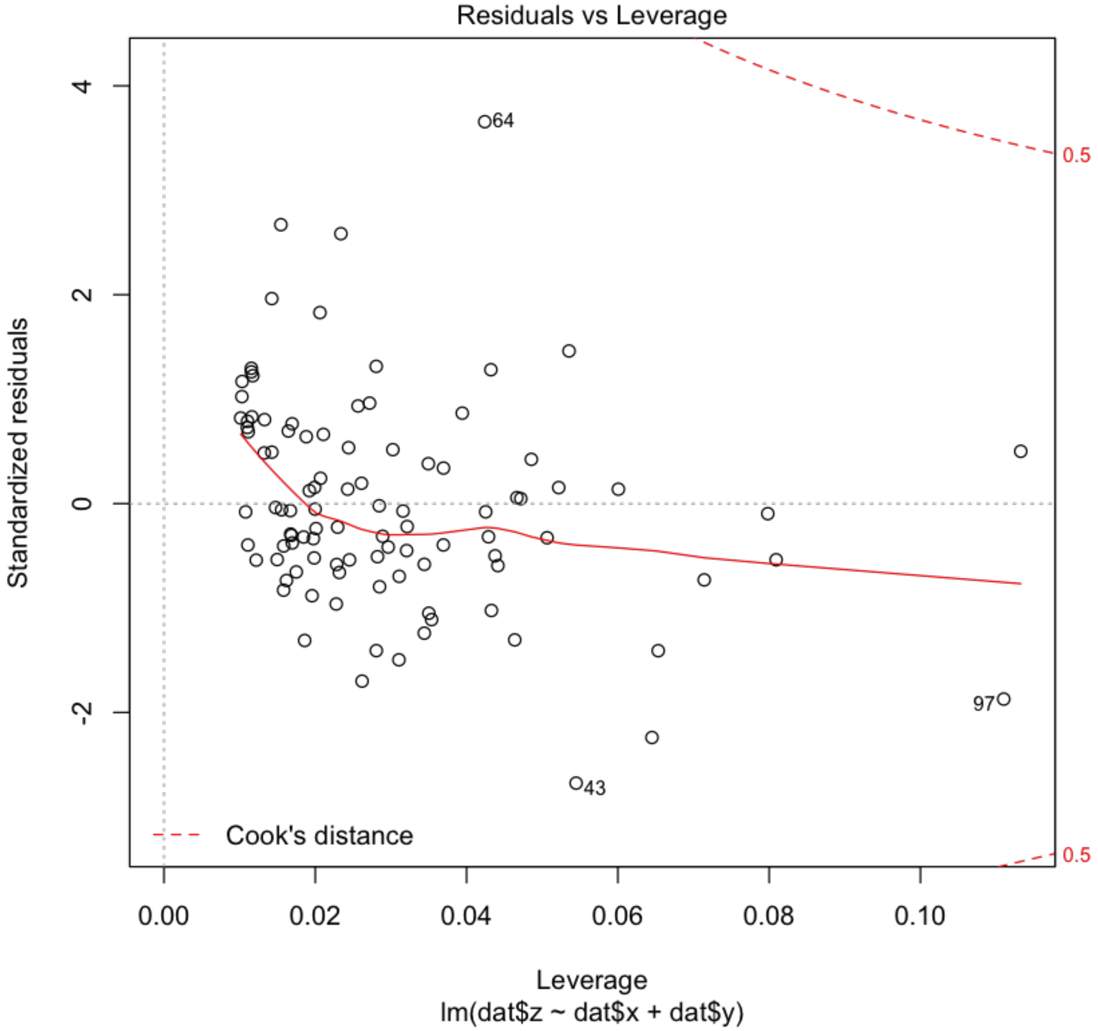
```
Call:
lm(formula = dat$z ~ dat$x + dat$y)

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 [23]: plot(fit)
```


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


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


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


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