R pour MTH2302D

April 13, 2018

Luc Adjengue

Contents

- Le logiciel R et son interface conviviale RStudio sont installés dans les laboratoires informatiques. On peut également utiliser R dans Anaconda (jupyter) et produire des note book tels que le présent document. Tous ces logiciels peuvent également être installés sur votre ordinateur personnel fonctionnant sous Windows, IOS ou Linux (voir sur le site Moodle du cours)
- Vous pouvez exécuter une après l'autre les commandes (les lignes débutant avec In[]) du présent fichier directement sur la console de R (ou de RStudio) ou sous forme d'un script dans RStudio.
- Nous présentons ici quelques exemples d'utilisation du logiciel R, dans le cadre du cours MTH2302D :
-   – Calcul de probabilité avec les lois discrètes et les lois continues usuelles
-   – Graphiques et calcul de statistiques descriptives
-   – Tests d'hypothèses et intervalles de confiance
-   – Régression linéaire

1 Calcul de probabilités

Le calcul des probabilités se fait généralement par la fonction de répartition en utilisant `ploi`(x, paramètres) où '`loi`' désigne le nom de la loi utilisée et x, le point pour lequel on désire obtenir la probabilité cumulative. Quelques exemples de noms de lois :

```
- binom => loi binomiale
- geom => loi géométrique (Attention ! dans R, il s'agit du nombre d'échecs avant le succès)
- pois => loi de Poisson
- norm => loi normale
- chisq => loi du chi2
- t => loi de Student
- f => loi de Fisher
- exp => loi exponentielle
- etc. (pour une liste complète, taper ?distributions() dans la console de R)
```

Par exemple `pnorm(1.96, mean = 0, sd = 1)` donne la valeur de la fonction de répartition d'une loi normale de moyenne 0 et d'écart type 1, soit 0,975. On obtient le même résultat avec `pnorm(1.96,0,1)`. Dans le cas d'une normale $N(0,1)$, on obtient le résultat simplement avec `pnorm(1.96)`.
Taper `?pnorm()`, ou `?ppois()`, de manière générale `?ploi()` pour obtenir des précisions sur le calcul avec la 'loi' voulue.
De manière similaire,

- On remplace `p` par `d` pour calculer la fonction de masse ou de densité au point `x`. Par exemple, `dpois(x,c)` donne la valeur de $P(X = x)$ pour une loi de Poisson de moyenne c.
- On remplace `p` par `q` pour calculer les quantile de la loi. Exemple `qnorm(.95)` donne 1.645.

- On remplace `p` par `r` pour calculer générer des observations selon la loi de probabilités. Par exemple, `rnorm(100)` produit 100 observations selon une loi N(0, 1).

La commande `ls()` donne la liste des variables en mémoire à ce moment là.
La commande `rm(list=ls())` efface toutes les données en mémoire à ce moment là.

```
In [1]: rm(list=ls())
```

1.1 Exemple : exercice 6.1 page 183

Les commandes suivantes produisent successivement les réponses des questions a) à h) de l'exercice 6.1 page 183.

```
In [2]: pnorm(2) - pnorm(0)          # question a)
        pnorm(1) - pnorm(-1)         # question b)
        pnorm(1.65)                  # question c)
        1-pnorm(-1.96)               # question d)
        2*(1-pnorm(1.5))             # question e)
        pnorm(2) - pnorm(-1.9)       # question f)
        pnorm(1.37)                  # question g)
        pnorm(2.57) - pnorm(-2.57)   # question h)
```

0.477249868051821
0.682689492137086
0.950528531966352
0.97500210485178
0.133614402537716
0.948533308235819
0.914656549178033
0.989830148502018

1.2 Exemple : exercice 6.3 page 183

Rappel : Les quantiles sont de la forme `qloi(1-alpha, paramètres)`, c'est - à - dire `1-alpha` désigne l'aire à gauche. On obtient le même résultat en écrivant `qloi(alpha, paramètres, lower.tail=FALSE)` ou simplement `qloi(alpha, paramètres, lower.tail=F)`.

Par exemple : $z_{0{,}05}$ s'obtient avec la comande `qnorm(.95, mu=0, sd=1)`, où mu est la moyenne et sd, l'écart type.

Pour une normale quelconque, on précise la moyenne et l'écart type.

Dans le cas d'une $N(0,1)$ on peut simplement écrire `qnorm(.95)` qui donne $z_{0{,}05}$

Les commandes suivantes produisent successivement les réponses des questions a) à d) de l'exercice 6.3 page 183.

```
In [3]: qnorm(0.94062)       # question a). [Même résultat avec qnorm(0.05938,lower.tail=F)]
        qnorm(0.975)         # question b). [Même résultat avec qnorm(0.025,lower.tail=F)]
        qnorm(0.995)         # question c). [Même résultat avec qnorm(0.005,lower.tail=F)]
        qnorm(0.05)          # question d). [Même résultat avec qnorm(0.95,lower.tail=F)]
```

1.55999949723643
1.95996398454005
2.5758293035489
-1.64485362695147

2 Graphiques et calcul de statistiques descriptives

Afin d'illustrer ces concepts, nous générons 100 observations d'une loi normale $N(5,9)$.

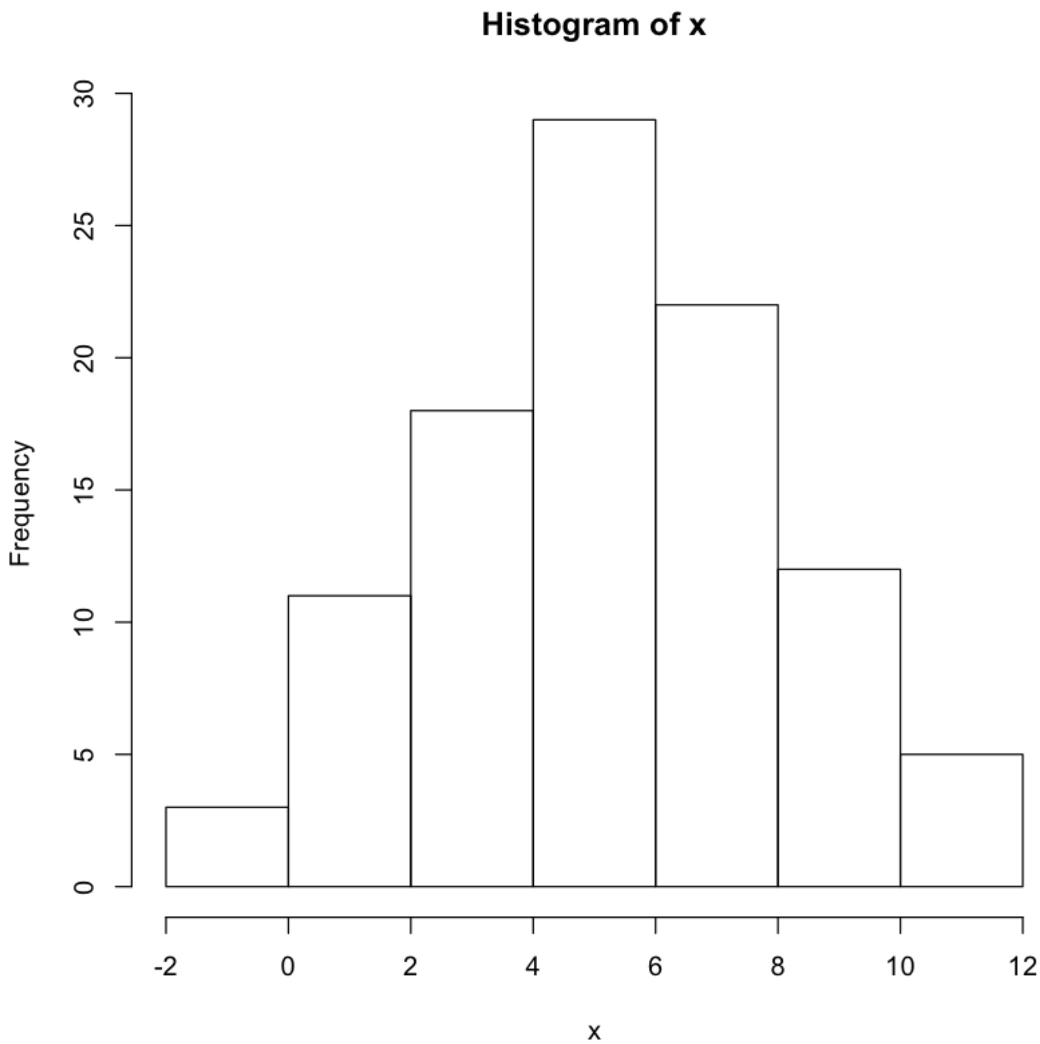
```
In [4]: set.seed(123) # on fixe le germe, ceci permet de générer toujours les mêmes données
        x <- rnorm(100, 5, 3)  # on génère des observations (ici 100).
        # Ici les données sont placées dans la variable x. Exemple :
        x
```

1. 3.31857306034336 2. 4.30946753155016 3. 9.67612494244737 4. 5.21152517427373
5. 5.38786320548284 6. 10.1451949606498 7. 6.38274861796761 8. 1.2048162961804
9. 2.93944144431942 10. 3.66301408970013 11. 8.67224539231838 12. 6.07944148117209
13. 6.20231435178216 14. 5.33204814783536 15. 3.33247659573778 16. 10.3607394104092
17. 6.49355143468772 18. -0.899851469888914 19. 7.10406770469106 20. 3.5816257768162
21. 1.79652888203946 22. 4.34607525602511 23. 1.92198665507828 24. 2.81332631212658
25. 3.12488219645223 26. -0.0600799322272403 27. 7.51336113348357 28. 5.46011935350955
29. 1.58558918896416 30. 8.76144476320978 31. 6.27939266443044 32. 4.11478555102319
33. 7.68537698313507 34. 7.63440046259913 35. 7.46474324491246 36. 7.06592076230027
37. 6.66175296061277 38. 4.81426486826984 39. 4.08211200878025 40. 3.85858699696285
41. 2.91587906323846 42. 4.3762481659412 43. 1.20381094529521 44. 11.5068678960155
45. 8.62388599491497 46. 1.63067425038995 47. 3.79134549410277 48. 3.60003393913034
49. 7.33989535500895 50. 4.74989280058451 51. 5.75995554198426 52. 4.91435973395389
53. 4.87138862812605 54. 9.10580685204337 55. 4.3226870430222 56. 9.54941181328862
57. 0.353741587309337 58. 6.75384124890821 59. 5.37156273153384 60. 5.64782470623192
61. 6.13891844827965 62. 3.49302964067209 63. 4.00037784899174 64. 1.94427385067873
65. 1.78462632057327 66. 5.91058592421277 67. 6.34462933588828 68. 5.15901268019151
69. 7.76680240363921 70. 11.1502540568814 71. 3.52690650183039 72. -1.92750662692244
73. 8.01721557338677 74. 2.87239771225282 75. 2.93597415059793 76. 8.0767141090901
77. 4.14568097884697 78. 1.33784686323639 79. 5.54391043924745 80. 4.58332591268287
81. 5.01729255769966 82. 6.15584120337899 83. 3.88801990462277 84. 6.9331296455565
85. 4.33854031454375 86. 5.99534589174709 87. 8.29051703944804 88. 6.30554447250141
89. 4.02220524340632 90. 8.44642285535328 91. 7.98051156788636 92. 6.64519087852421
93. 5.71619520533432 94. 3.11628177188189 95. 9.08195734559002 96. 3.19922123855862
97. 11.5619989790497 98. 9.59783187855557 99. 4.29289892269857 100. 1.92073729907966

2.1 Un histogramme

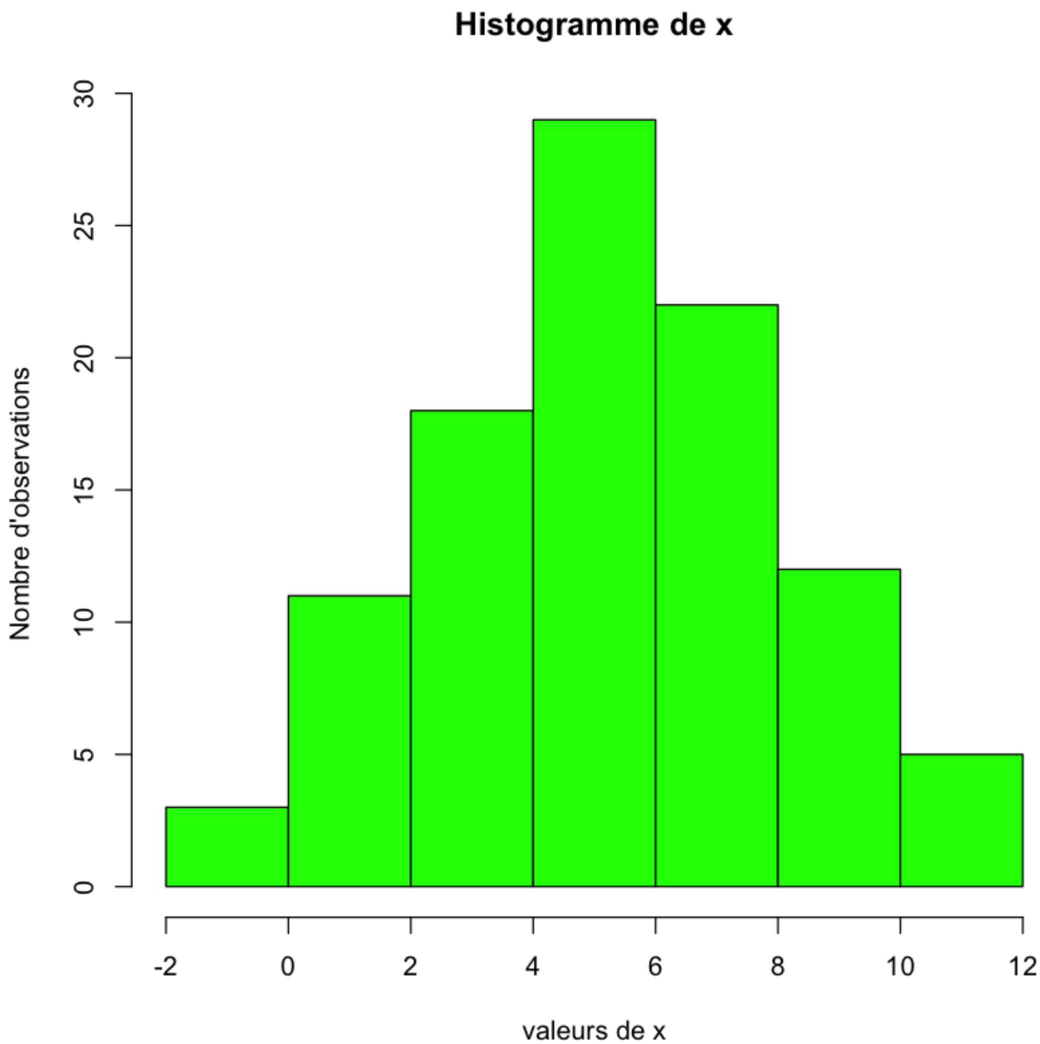
On trace un histogramme avec la commande `hist()`

```
In [5]: hist(x)
```



On peut produire l'histogramme avec de la couleur, des titres sur les axes, etc.

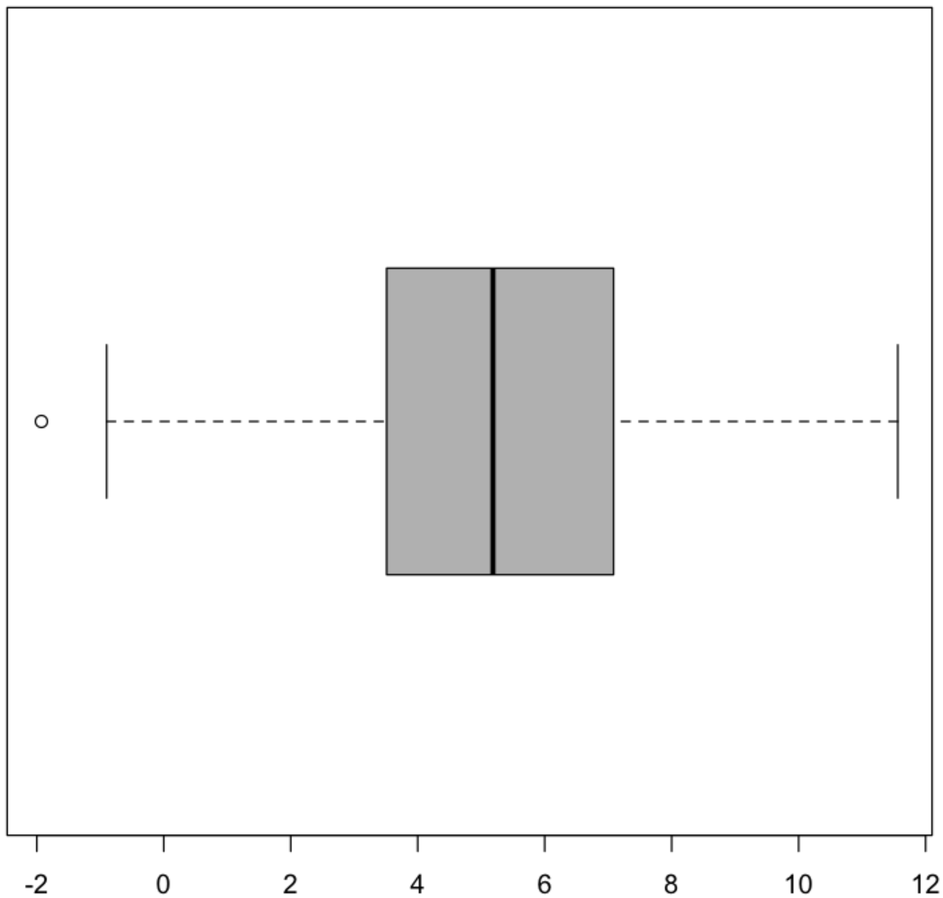
```
In [6]: hist(x, col="green", main="Histogramme de x", xlab="valeurs de x",
             ylab="Nombre d'observations")
```



2.2 Un diagramme en boîte (Box-Plot)

```
In [7]: boxplot(x, col="grey", horizontal=T)

        # par défaut la boîte est verticale.
        # L'option 'horizontal=TRUE' la place horizontalement
        # D'autre options sont disponibles.
        # Pour plus de détails, taper ?boxplot() dans la console de R
```



2.3 La moyenne, l'écart-type, la variance et la taille de l'échantillon

La commande `summary()`, permet d'obtenir les 3 quartiles, la moyenne, le minimum et le maximum de `x`.

In [8]: `summary(x)`

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 -1.928   3.518   5.185   5.271   7.075  11.562 
```

On peut également calculer certaines statistiques en invoquant leur noms

In [9]:
```
m=mean(x)   # la moyenne
s=sd(x)     # l'écart type
v=var(x)
n=length(x)
# et les écrire
cat('la moyenne =',m,', l\'écart-type = ', s,', la variance = ', v ,'\n',
    ' et la taille de l\'échatillon est n =',n,'.')
```

```
la moyenne = 5.271218 , l'écart-type =  2.738448 , la variance =  7.499095
  et la taille de l'échatillon est n = 100 .
```

Test de normalité :

La commande `shapiro.test(x)` permet d'effectuer le test de normalité de Shapiro pour les données de x

`qqnorm(x)` permet de tracer le graphique quantile-quantile des données de x

2.3.1 Le test de Shapiro-Wilk

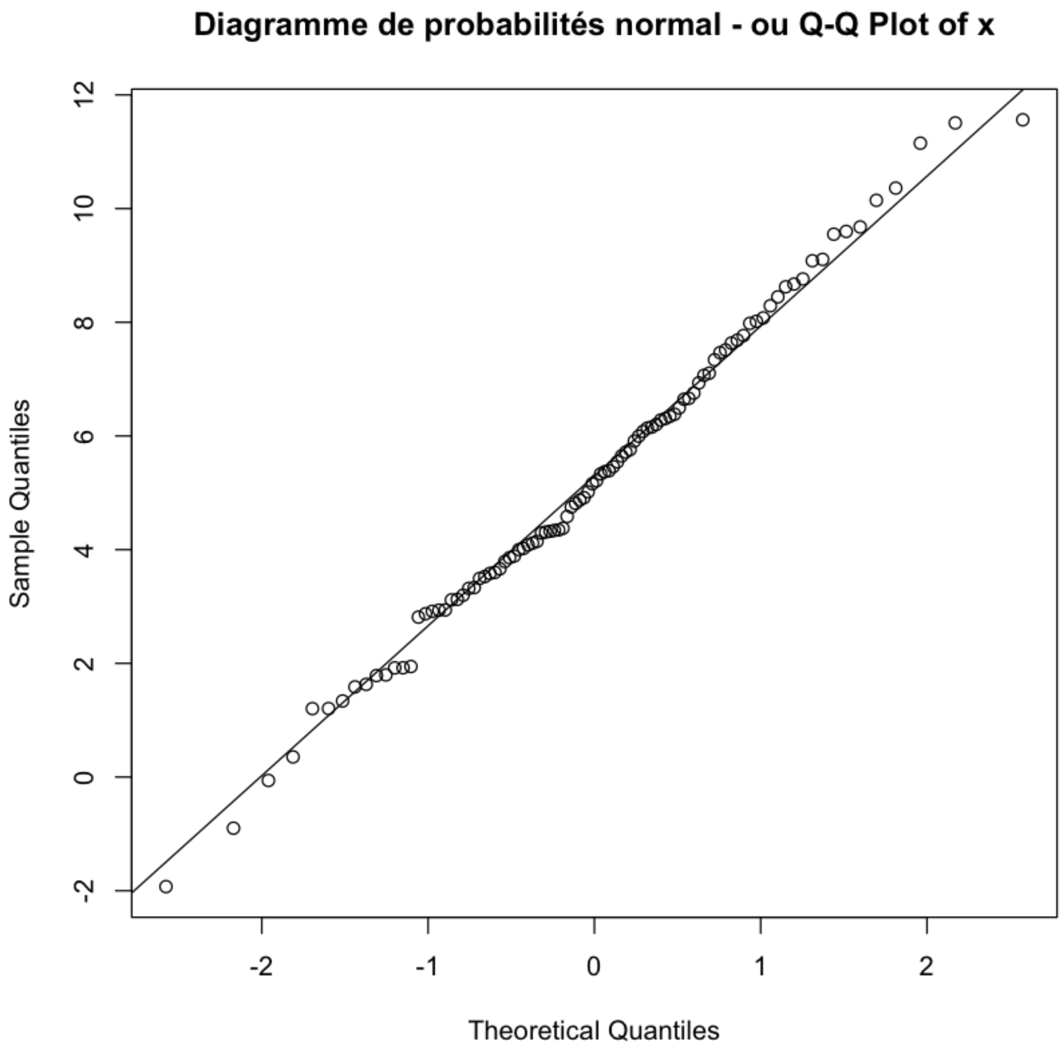
In [10]:
```
shapiro.test(x) # calcule et affiche les valeurs de W et p
```

```
Shapiro-Wilk normality test

data:  x
W = 0.99388, p-value = 0.9349
```

2.3.2 Le graphique Quantile-Quantile (pour la normale)

In [11]:
```
qqnorm(x, main ="Diagramme de probabilités normal - ou Q-Q Plot of x")
qqline(x)      # pour ajouter une droite
```



3 Tests d'hypothèses

On peut effectuer des tests sur une moyenne, une variace, une proportion, deux moyennes, deux variances, etc.

3.1 Tests sur une moyenne avec variance inconnue

Le test t est efectué avec la commande `t.test()`.

3.1.1 Exemple : exercice 10.6 page 335

```
In [12]: # On crée une variable, disons nbh, pour le nombre d'heures
         nbh <- c(8.8, 12.5, 5.4, 12.8, 8.8, 12.2, 13.3, 6.9, 9.1, 14.7, 2.2)
```

```
# on peut visualiser les données
nbh
```

1. 8.8 2. 12.5 3. 5.4 4. 12.8 5. 8.8 6. 12.2 7. 13.3 8. 6.9 9. 9.1 10. 14.7 11. 2.2

```
In [13]: # test de normalité si nécessaire
         shapiro.test(nbh)
```

```
Shapiro-Wilk normality test

data:  nbh
W = 0.93939, p-value = 0.5134
```

On peut voir que la normalité est plausible puique la p-value, 0.5134, est élevée
On peut donc tester la moyenne μ avec le test t.
Pour tester $H_0 : \mu = 8$ contre $H_1 : \mu > 8$,
la commande est `t.test()`.

```
In [14]: t.test(nbh, mu = 8, alternative = "greater")
         # alternative donne le sens du test de l'inégalité dans $H_1$.
         # On a donc aussi : "less" pour inférieur,
         # et "two.sided" pour bilatéral.
```

```
One Sample t-test

data:  nbh
t = 1.4745, df = 10, p-value = 0.08556
alternative hypothesis: true mean is greater than 8
95 percent confidence interval:
 7.610339      Inf
sample estimates:
mean of x
      9.7
```

On a $t = \dfrac{\bar{x} - 8}{s/\sqrt{n}} = 1{,}4745$, et valeur-P$= P(T_{10} > 1{,}4745) = 0{,}08556$. On ne rejette donc pas H_0.

L'intervalle de confiance unilatéral correspondant est [7.61, Infini], à 95% (par défaut). On peut changer le niveau de confiance en ajoutant `conf.level=1-alpha`. e.g., pour un niveau à 90% : `t.test(nbh, mu = 8, alternative = "greater", conf.level=0.90)`

Remarque : on peut calculer le centile $t_{0{,}05;10}$ avec la commande `qt(.95,10)` et la p-value avec la commande `1-pt(1.4745,10)`.

```
In [15]: qt(.95,10)
         1-pt(1.4745,10)
         # Si on veut, on peut toujours calculer soi-même la moyenne, l'écart type, etc.
         mean(nbh)
         sd(nbh)
         # ou la valeur de t
         (mean(nbh) - 8)/(sd(nbh)/sqrt(length(nbh)))
```

1.81246112281168
0.0855597545984924
9.7
3.82387238280777
1.47449014484739

3.2 Tests sur une moyenne avec variance connue

Dans le cas où la variance est connue et la distribution est normale, on peut calculer les quantités requises pour le test de $H_0 : \mu = \mu_0$ contre $H_1 : \mu \neq \mu_0$

3.2.1 Exemple : exemple 10.6 page 290

```
In [16]: alpha=0.05  # la valeur de alpha
         mu0=40      # la valeur de mu0
         xbar=41.25  # la valeur de la moyenne xbar
         sigma=2     # la valeur de l'écat type sigma
         n=25        # le nombre d'observations
         z0=(xbar - mu0)/(sigma/sqrt(n))
         pval=2*(1-pnorm(abs(z0)))
         zalpha2=qnorm(1-alpha/2)
         cat('Résultats :','\n')
         cat('z0 =',z0,';','zalpha2 =',zalpha2,';','p-value=',pval)
```

```
Résultats :
z0 = 3.125 ; zalpha2 = 1.959964 ; p-value= 0.001778051
```

3.3 Tests sur deux moyennes et deux variances

Dans le cas de 2 échantillons, on peut tester l'égalité des variances d'une part (test F)
et l'égalité des moyennes (test t) d'autre part. Illustrons ces deux tests par un exemple.

3.3.1 Exemple : exercice 10.20 page 338

Appelons x1 le rendement du procédé p1 et x2, celui du procédé p2.

Les données peuvent être lues directement d'un fichier csv (voir l'exemple sur la régression plus bas, mais ici enregistrons les manuellement.

```
In [17]: x1 <- c(24.2, 26.6, 25.7, 24.8, 25.9, 26.5)
         x2 <- c(21.0, 22.1, 21.8, 20.9, 22.4, 22.0)
```

```
In [18]: # on peut vérifier l'hypothèse de la normalité des données
         shapiro.test(x1)
         shapiro.test(x2)
```

```
Shapiro-Wilk normality test

data:  x1
W = 0.92127, p-value = 0.5146




Shapiro-Wilk normality test

data:  x2
W = 0.88638, p-value = 0.2997
```

Le test de l'égalité des variances, c'est-à-dire, le test des hypothèses
$H_0 : \sigma_1^2 = \sigma_2^2$ contre $H_1 : \sigma_1^2 \neq \sigma_2^2$
s'effectue avec la commande `var.test()` de la manière suivante :

```
In [19]: var.test(x1,x2)
```

```
F test to compare two variances

data:  x1 and x2
F = 2.398, num df = 5, denom df = 5, p-value = 0.3591
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
  0.3355614 17.1373784
sample estimates:
ratio of variances
          2.39805
```

Ici $f_0 = 2{,}398$ (formule 10.75 page 319); les degrés de liberté sont $\nu_1 = 5, \nu_2 = 5$. On ne rejette pas H_0 puisque valeur-p=0,3591.

Un intervalle de confiance du raport $\dfrac{\sigma_1^2}{\sigma_1^2}$ est $[0{,}336 \quad 17{,}14]$ à 95%.

On peut changer le niveau de confiance en ajoutant `con.level` : e.g. `var.test(x1,x2, conf.level=.90)`

On peut aussi conclure le test en considérant les percentiles $F_{0{,}975;5{,}5}$ et $F_{0{,}025;5{,}5}$ calculés comme suit :

```
In [20]: qf(.025,5,5)   # ou encore qf(.975,5,5, lower.tail=F)
         qf(.975,5,5)   # ou encore qf(.025,5,5, lower.tail=F)
```

0.139930950229862
7.14638182873283

Pour le test de deux moyennes $H_0 : \mu_1 = \mu_2$ contre $H_1 : \mu_1 \neq \mu_2$,

on utilise encore `t.test()`. Les paramètres sont les mêmes que dans le cas d'une moyenne, mais il faut fournir 2 variables et ajouter des options

- l'option `var.equal = T` (variances égales). Par défaut on a `var.equal=F` (les variances ne sont pas égales).
- l'option `paired = T` (données pairées ; attention x1 et x2 doivent avoir la même longueur).
- Par défaut on a `paired=F` (données non pairées) et `var.equal=F` (variances inégales).
- `conf.level = 1-alpha` (niveau de confiance pour l'intervalle de confiance de la différence des deux moyennes Par défaut `conf.level=0.05`.
- `alternative = two.sided` (ou `less`, ou `greater`) pour le sens de H_1. Taper `?t.test()` pour plus de détails.

```
In [21]: t.test(x1,x2, var.equal=T)
```

```
Two Sample t-test

data:  x1 and x2
t = 8.4876, df = 10, p-value = 6.987e-06
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 2.888472 4.944861
sample estimates:
mean of x mean of y
 25.61667  21.70000
```

On dispose d'un certain nombre de résultats ci-dessus :
$t_0 = 8.4876$, selon la formule (10.61) page 312
df =10, le nombre de degrés de liberté n1+n2-2
pvalue= 6x10^(-6) = 2*P(T>|t0|) avec 10 degrés de liberté (ici on rejette H_0)
L'intervalle de confiance est $\mu_1 - \mu_2 \in [2.89 \quad 4.94]$, à 95% (par défaut).

Remarque : dans le cas des variances connues, on peut faire les calculs similaires au cas d'une moyenne avec variance connue (voir plus haut).

On peut aussi installer un des 'packages' qui permettent d'effectuer les tests sur les moyennes et les variances dans différentes circonstances. Par exemple 'OneTwoSamples'.

3.4 Autres tests

On peut efectuer d'autres tests, tels que le test des proportions, les test di khi-deux (ajustement et indépendance).

3.4.1 Les tests des proportions

Les tests sur une ou deux proportions se font la commande prop.test(). Taper ?prop.test() pour plus de deétails et exemples.

3.4.2 Les tests du khi-deux (ou chi-carré)

Les tests du khi-deux (ajustement et indépendance) se font avec la commande chisq.test(). Taper ?chisq.test() pour plus de détails et exemples. Voici quelques exemples

pour le test d'ajustement (méthode du khi-deux):

- mettre les effectifs observés Oj dans un vecteur, disons O et les probabilités lorsque H_0 est vraie, p0j, dans un vecteur, disons p0 (leur somme doit être 1)
- taper chisq.test(O, p=p0)

exemple : on veut tester l'hypothèse H_0 selon laquelle les effectifs 967, 400 et 433 sont ceux d'une distribution de 3 valeurs (disons a, b et c) dont les probabilités sont : 1/2,1/4 et 1/4.

```
In [22]: O<-c(967,400,433)  # le ecteur des valeurs observées

In [23]: p0<-c(.5,.25,.25) # le vecteur des probabilités lorsque H0 est vraie

In [24]: chisq.test(O,p=p0)
```

```
Chi-squared test for given probabilities

data:  O
X-squared = 11.186, df = 2, p-value = 0.003725
```

La statistique du test est $U_0 = 11{,}19$ avec une p-value de 0,0037. L'hypothèse n'est donc pas plausible.

Il est préférable de mettre le test dans un objet pour aller chercher des résultats supplémentaire tels que les valeurs attendues Ej, la statistique, la p-value, etc.

```
In [25]: tch2<-chisq.test(O,p=p0)

In [26]: tch2$expected # les valeurs attendues Ej
```

1. 900 2. 450 3. 450

```
In [27]: tch2$statistic
```

X-squared: 11.1855555555556

Pour le test d'indépendance (méthode du khi-deux):

- mettre le tableau sous la forme d'une matrice, disons M

- taper 'chisq.test(M)'

exemple : exercice 10.35 page 343

```
In [28]: M<-matrix(c(216,226,114,245,409,297),nrow=3)
```

```
In [29]: M
```

216	245
226	409
114	297

Ceci nest pas necessaire, il est possible dajouter les noms de lignes et de colonnes.

```
In [30]: rownames(M)<-c("Faible","Moyen","Elevé")
```

```
In [31]: colnames(M)<-c("Inactive","Active")
```

```
In [32]: M
```

	Inactive	Active
Faible	216	245
Moyen	226	409
Elevé	114	297

```
In [33]: mt<-chisq.test(M)
```

```
In [34]: mt
```

```
Pearson's Chi-squared test

data:  M
X-squared = 34.909, df = 2, p-value = 2.627e-08
```

La statistique du test est $U_0 = 34{,}91$ avec une p-value de $2{,}6x10^{-8}$.

L'hypothèse d'indépendance des deux variablesn'est pas plausible. Les deux variables ne sont pas indépendantes.

Comme dans le test d'ajustement il est possible daller chercher des résultats supplémentaire tels que les valeurs attendues Eij, la statistique, la p-value, etc.

```
In [35]: mt$expected  # les valeurs attendues Eij
```

	Inactive	Active
Faible	170.0836	290.9164
Moyen	234.2800	400.7200
Elevé	151.6364	259.3636

4 Régression linéaire

4.1 Lecture des données et visualisation. Exemple : exercice 12.5 page 424

Afin d'illustrer l'anlyse de régression, considérons les données de l'exercice 12.5 page 424

Ici nous lisons les données d'un fichier au format .csv. Veuillez vous assurer que Le fichier Exerc12_5.csv est dans le répertoire de travail de R (ou de RStudio).

In [36]:
```
don <- read.csv2("Exerc12_5.csv")
# les données sont lues et placées dans une base de données appelée ici 'don'

# La commande read.csv2("fichier.csv") permet de lire les
# fichiers .csv dont la décimale des chiffres est une virgule
# dans le cas contraire on urilise read.csv("fichier.csv")

#attach(don) # cette commande permet d'attacher les données
# en mémoire et d'y accéder directement avec les noms des variables
```

Si l'instruction ci-dessus echoue, essayer celle qui suit après avoir enlevé le symbole # qui la précède

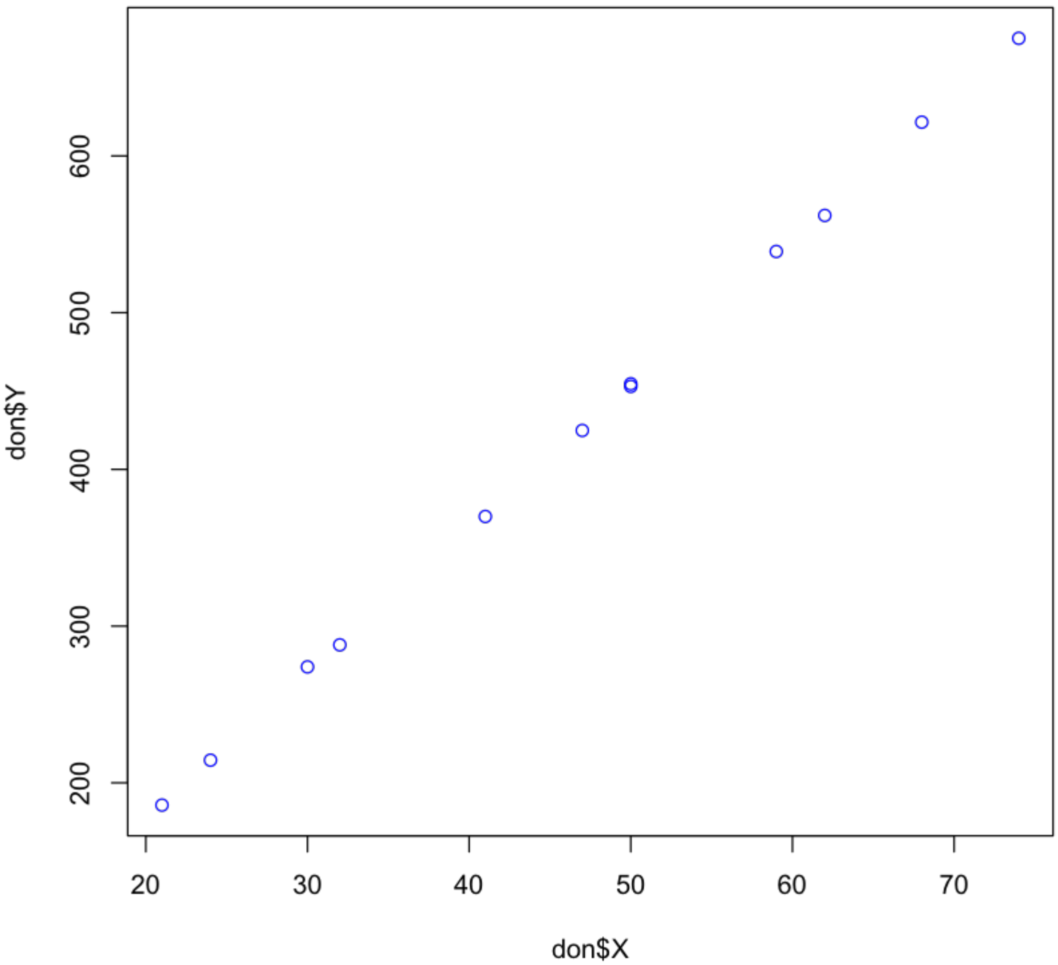
In [37]:
```
# don <- read.csv("Exerc12_5.csv", header = TRUE, sep = ";", dec = ".")
```

In [38]:
```
# On peut visualiserles les données en invoquant le nom de la base de données
don
```

X	Y
21	185.79
24	214.47
32	288.03
47	424.84
50	454.58
59	539.03
68	621.55
74	675.06
62	562.03
50	452.93
41	369.95
30	273.98

Le nuage de points

In [39]:
```
plot(don$X,don$Y,col="blue")
```



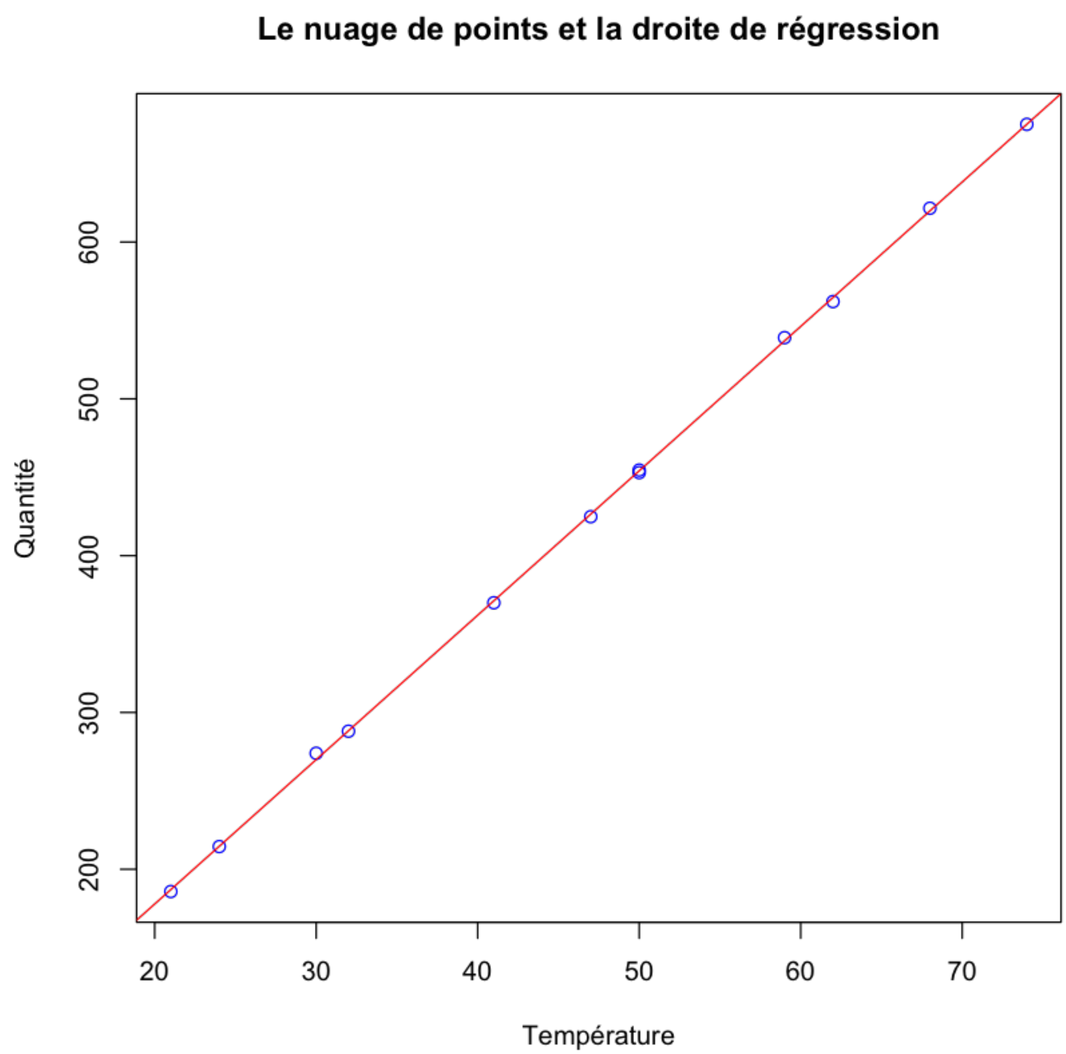
4.2 Estimation des coefficients β_1, β_2 tests et intervalles de confiance

La commande pour une régression linéaire de Y sur X est `lm(Y X)`
On crée un objet, ici 'reg.lin', qui contient tous les résultats de la régression

```
In [40]: reg.lin <- lm(Y~X, data=don)
```

Le nuage de points et la droite

```
In [41]: plot(don$X, don$Y, col="blue", main="Le nuage de points et la droite de régression", xl
         abline(reg.lin, col="red")
```



Les résultats s'obtiennent avec la commande summary()

```
In [42]: summary(reg.lin)


Call:
lm(formula = Y ~ X, data = don)

Residuals:
    Min      1Q  Median      3Q     Max
-2.5529 -1.2519 -0.2486  0.8023  4.0646

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept) -6.33550    1.66765  -3.799  0.00349 **
X            9.20836    0.03377 272.643  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.943 on 10 degrees of freedom
Multiple R-squared:  0.9999,Adjusted R-squared:  0.9999
F-statistic: 7.433e+04 on 1 and 10 DF,  p-value: < 2.2e-16
```

Le tableau ci-dessus montre que
$\hat{\beta}_0 = -6{,}3355$; $\hat{\beta}_1 = 9{,}208$

Le test de $H_0 : \beta_0 = 0$ contre $H_1 : \beta_0 \neq 0$. On a $t_0 = -6{,}3355/1{,}66765 = -3{,}799$. Puisque valeur-p$=$ 0,00349, on rejette H_0.

Le test de $H_0 : \beta_1 = 0$ contre $H_1 : \beta_1 \neq 0$. On a $t_0 = 9{,}208/0{,}03377 = 272{,}643$. Puisque valeur-p$=$ 0, on rejette H_0.

Les intervales de confiance pour β_0 et β_1 à 95% s’otiennent avec la commande `confint()`

```
In [43]: confint(reg.lin)
```

	2.5 %	97.5 %
(Intercept)	-10.051253	-2.619750
X	9.133108	9.283616

$\beta_0 \in [-10{,}05 \;\; -2{,}62]$ à 95% et $\beta_1 \in [9{,}13 \; 9{,}28]$ à 95%.

On peut modifier le niveau de confiance en ajoutant `level=1-alpha`. Par exemple

```
In [44]: confint(reg.lin, level=.90)
```

	5 %	95 %
(Intercept)	-9.358049	-3.312954
X	9.147147	9.269577

4.3 Tableau d’ananalyse de la variance

Le tableau d’analyse de la variance du modèle de régression s’obtient avec la commande `anova()`

```
In [45]: anova(reg.lin)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
X	1	280583.11938	2.805831e+05	74334.36	1.08364e-20
Residuals	10	37.74609	3.774609e+00	NA	NA

$SS_R = 280583.12$; $SS_E = 37.746$; $MS_R = 280583.12$; $MS_E = 3.775$
$F_0 = 74334.36$; $p - value = 10^{-20} \simeq 0$.

4.4 Calcul d'intervalles de prévision et de confiance

```
In [46]: newd <- data.frame(X = c(60)) # on détermine x0 (ici 60)
         predict(reg.lin, newd, interval = "prediction")
         # intervalle prédiction au point X=60 à 95% formule (12.25) page 409
         predict(reg.lin, newd, interval = "confidence")
         # intervalle de confiance pour la moyenne au point X=60 à 95%
         # formule (12.23) page 407
```

	fit	lwr	upr
1	546.1662	541.5474	550.785

	fit	lwr	upr
1	546.1662	544.5557	547.7767

fit représente $\hat{y}_0 = \hat{\beta}_0 + \hat{\beta}_1 \times 60 = 546{,}166$

lwr = borne inférieur de l'intervalle et **upr** = borne supérieure de l'intervalle.

Remarque : On peut calculer des intervalles pour plusieurs valeurs de x0. Par exemple, pour 3 valeurs 60, 65 et 70 de x,

```
In [47]: newd <- data.frame(X = c(60,65,70))
         predict(reg.lin, newd, interval = "prediction")
         predict(reg.lin, newd, interval = "confidence")
```

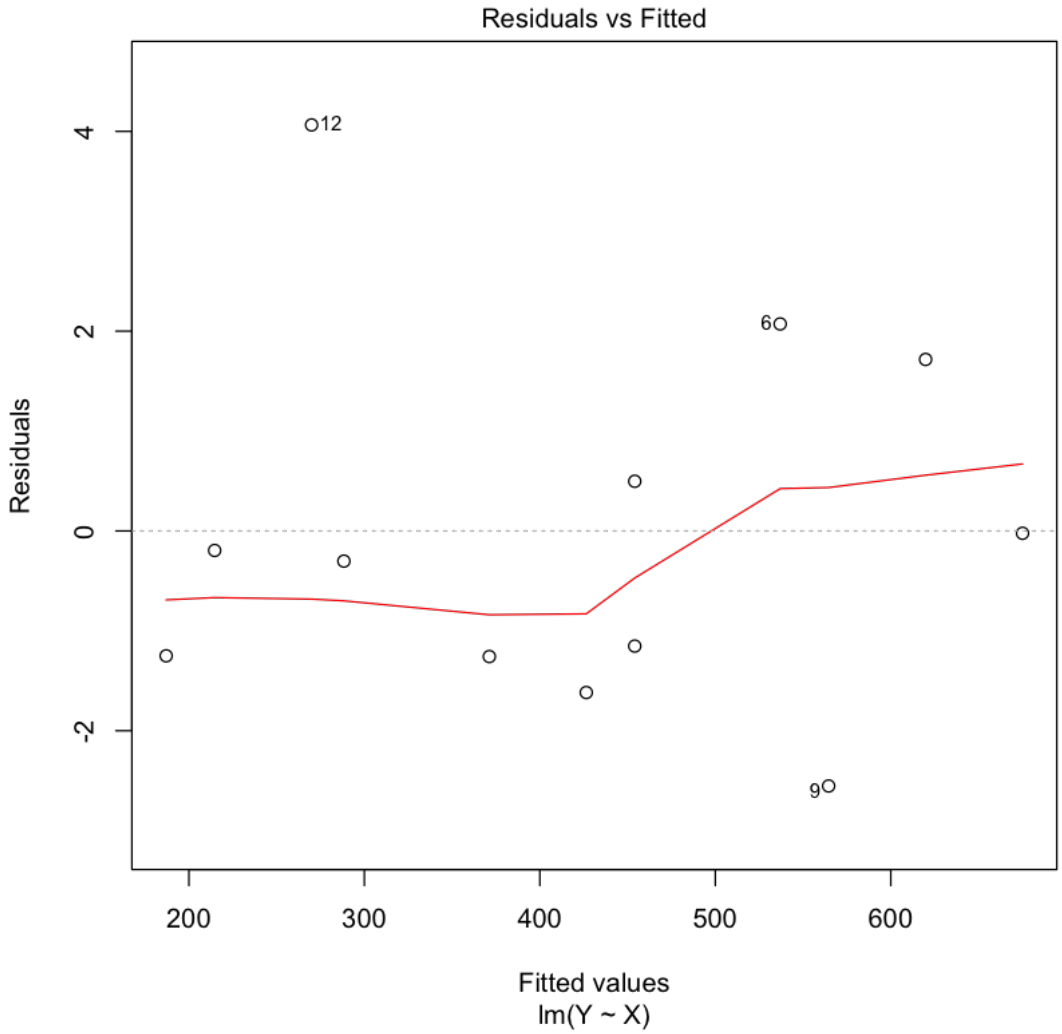
	fit	lwr	upr
1	546.1662	541.5474	550.7850
2	592.2080	587.4922	596.9239
3	638.2498	633.4095	643.0901

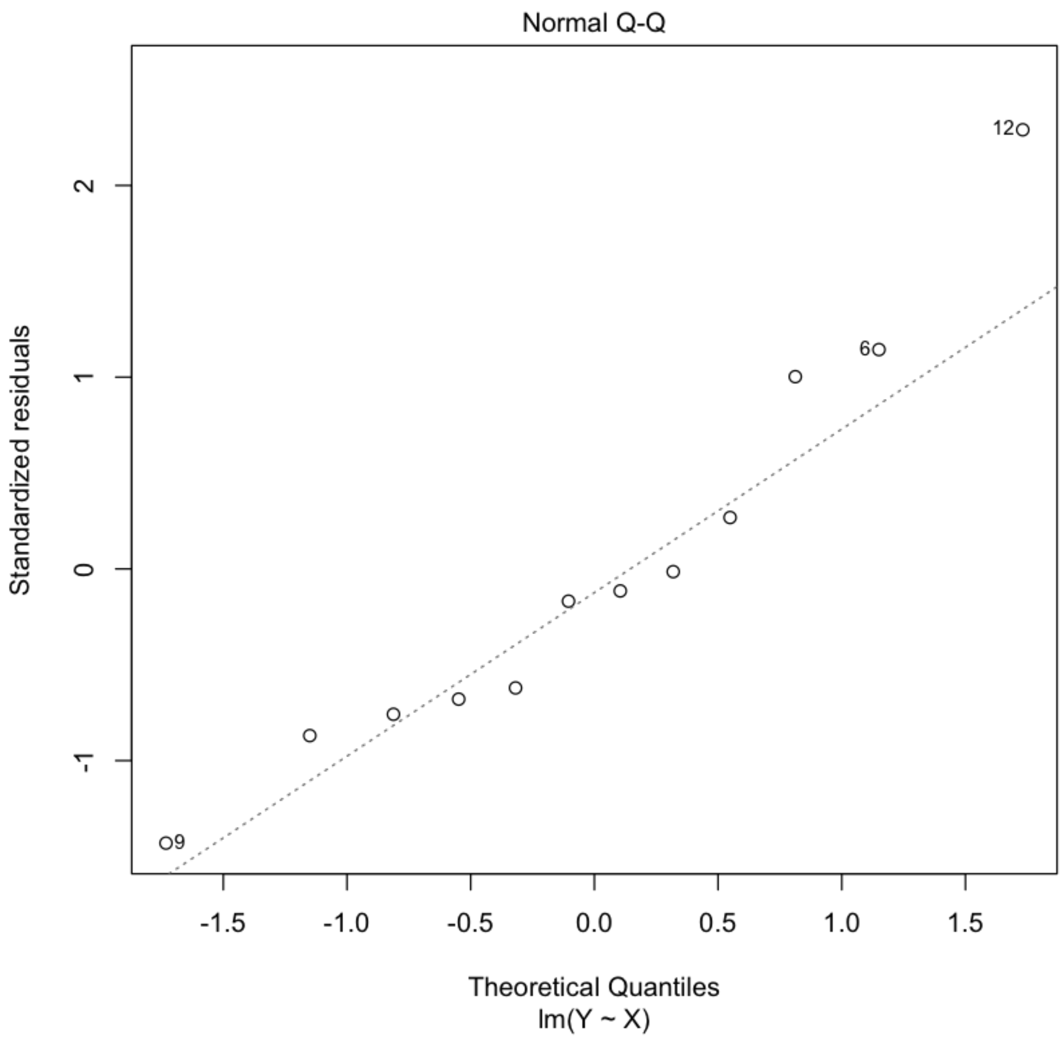
	fit	lwr	upr
1	546.1662	544.5557	547.7767
2	592.2080	590.3372	594.0788
3	638.2498	636.0844	640.4153

4.5 Quelques graphiques des résidus

Quelques graphiques des résidus (les deux premiers graphiques : résidus vs valeurs prédites et graphique de probabilité normal

```
In [48]: plot(reg.lin,1:2)
```


Residuals vs Fitted
Residuals
Fitted values
lm(Y ~ X)
4
2
0
-2
200
300
400
500
600
12
6
9


Normal Q-Q
Standardized residuals
2
1
0
-1
12
6
9
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
Theoretical Quantiles
lm(Y ~ X)