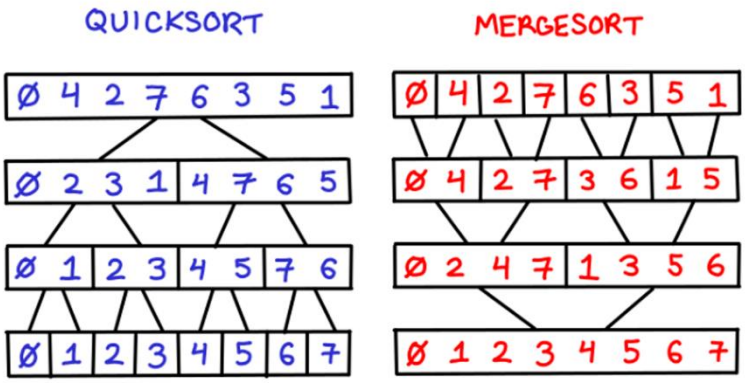
Què és més ràpid Mergesort o Quicksort?

Probabilitat i estadística



Marc Ordóñez
Christian Rivero
Jordi Soley

21 desembre 2021

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Índex

Resum

Objectiu: Comparar el temps d'ordenació de dos algorismes de l'ordre O(n · log(n)). Aquests dos algorismes són el *Quicksort* i el *Mergesort*. Volem saber quin dels dos convé utilitzar en els nostres codis per ordenar vectors d'enters.

Mètodes: Per fer la comparació generarem 50 vectors de mida aleatòria entre 100 i 500.000 (la mida segueix una distribució uniforme). Els vectors estaran formats per nombres, també generats aleatòriament i distribuïts uniformement, entre 0 i 10^6. L'ordre d'execució de l'algorisme d'ordenació també serà escollit de forma aleatòria. Hem transformat les dades amb el logaritme dels temps per evitar l'efecte multiplicatiu i aconseguir un efecte additiu. Hem fet la prova d'hipòtesis d'igualtat de la mitjana dels temps amb la t-Student per a dades aparellades.

Resultats: El temps mitjà d'execució del Mergesort ha estat de 190.6 ms. i pel *Quicksort* 118.5 ms. El p-valor i el punt crític ens han fet rebutjar la hipòtesis nul·la d'igualtat de temps.

Conclusió: L'algorisme *Mergesort* és un 56.754% més lent que el *Quicksort* amb un interval de confiança 95% [46.78% a 67.41%].

Introducció i objectius

En el món de la programació un dels aspectes que es té més en compte és l'eficiència dels codis. Durant aquest projecte volem comprovar quin algorisme d'ordenació entre *mergesort* (M) i *quicksort* (Q) seria millor utilitzar en els nostres codis a l'hora d'ordenar vectors de nombres enters. Considerarem que l'algorisme "millor" serà aquell que ordeni els vectors més ràpid.

També volem veure si la mida dels vectors condiciona a l'hora de veure quin dels dos algorismes ordenarà més ràpid. Tots aquests algorismes estaran implementats en el llenguatge de programació C++. Partirem de la hipòtesi de que el més ràpid és el *mergesort,* ja que té un cost en cas pitjor O(n·log(n)) respecte el O(n^2) del *quicksort* (sent n el nombre d'enters del vector a ordenar). Al final del treball veurem si és bona idea utilitzar l'algorisme *mergesort* en els nostres codis o si, per contra, fent servir l'algorisme *quicksort* podem trigar menys temps en ordenar. Per tant, intentarem

rebutjar la hipòtesi nul·la d'igualtat de temps a favor de la hipòtesi alternativa unilateral de que *quicksort*(Q) és més ràpid que *mergesort*(M).

Variables recollides

La nostra variable principal **T** serà el temps (en milisegons) d'execució de cada algorisme per cada vector. També tindrem dues variables explicatives que seran un natural **N** que és el nombre d'elements del vector i **A** el tipus d'algorisme (*mergesort* o *quicksort*).

Material i mètodes

Per poder comparar els temps hem utilitzat dades aparellades, concretament 50 vectors generats aleatòriament de grandària entre 100 i 500.000 amb números positius entre 0 i 10^6, on cada número té la mateixa probabilitat d'aparèixer (Distribució Uniforme). Havíem pensat en utilitzar també en la mostra vectors ordenats (creixentment i decreixentment) però vàrem rebutjar la idea ja que és poc comú que això succeeixi en un entorn real i volíem centrar-nos en el cas més genèric possible.

Vam recollir les dades amb un programa en C que guardava els temps en un arxiu *txt*. Per evitar errors causats per la memòria del PC i la pre-càrrega de dades a la memòria cache, l'ordre entre fer primer mergesort o quicksort era aleatori.

Farem ús sempre dels mateixos codis i del mateix compilador del programa escrit en C++.

Les proves es varen fer en una màquina virtual Linux Ubuntu 20.04 LTS 64 bits a un ordinador de sobretaula amb un processador AMD Ryzen 3600X de 3.8GHz (6 nuclis, 12 fils) i 16GB de memòria RAM DDR4 amb latència CL16 i freqüència a 3200MHz. La màquina virtual té assignada una memòria de RAM de 4096MB i 2 nuclis de CPU. Les dades van ser recollides sempre en el mateix entorn de treball per tal d'evitar que les dades extretes no siguin equiparables degut a les característiques de cada ordinador.

I. Prova d'hipòtesi unilateral i variable D

D = Diferència dels temps d'execució o bé dels seus algorismes = M – Q o bé log(M) – log(Q)

$$\begin{cases} H_0: \mu Q = \mu M \Rightarrow \mu D = 0 \\ H_1: \mu Q < \mu M \Rightarrow \mu D > 0 \end{cases}$$

II. Premisses

Tant si utilitzem els temps o els logaritmes dels temps les premisses són les mateixes:

- Normalitat de la variable diferència.
- Obtenció a l'atzar de les dades i ordre aleatori d'execució.
- Efecte additiu constant a la mostra aleatòria simple.

III. Càlcul i distribució de l'estadístic

$$\hat{t} = \frac{\bar{D} - \mu_D}{S_D/\sqrt{n}} = \frac{\bar{D}}{S_D/\sqrt{n}} \sim t_{n-1} \sim t_{49}$$

- $\bar{D}$ serà la mitjana de la variable diferència D
- S_D serà la desviació típica de D
- μ_D serà el valor a contrastar: 0 per la igualtat de les 2 opcions
- L'estadístic segueix una distribució t-Student amb n-1 graus de llibertat (49 graus)

IV. Interval de confiança per a la diferència

IC (μ_D, 1 - α) = $\mu_D \in \left[\bar{D} \pm t_{n-1,1-\frac{\alpha}{2}}\sqrt{\frac{S^2}{n}}\right]$ (si apliquem logaritme als temps, es pot calcular el IC95% del rati fent l'exponencial (operació contraria al logaritme) de l'interval obtingut pels logaritmes)

Resultats

Descriptiva numèrica

A la Taula 1 hem recollit les dades de la descriptiva dels temps d'execució en ms. de Q i de M i també de la mida dels vectors. En aquesta taula podem observar que, en promig, el mètode Q*uicksort* és més ràpid que el *Mergesort*.

	Mergesort	***Quicksort***	**Mida**
Mínim	3	1	4518
Mediana	190.6	110	260129
Màxim	370	230	489334
Mitjana aritmètica	190.6	118.5	255370
Desviació típica	109,441	64,39775	144376,9

Taula 1: Descriptiva de les dades

Per tal de mantenir la premissa d'efecte additiu constant de la mostra vam haver de recórrer a l'ús de logaritmes en els temps d'execució. En l'annex 1 es mostren les gràfiques QQ-norm i Bland-Altman i s'explica perquè vam decidir aplicar logaritmes als temps. En resum, si no aplicàvem logaritmes s'observava un efecte multiplicatiu que contradeia les premisses del treball. La variable D serà log(*Mergesort*) – log(*Quicksort*).

El valor de l'estadístic és 13.739. Els graus de llibertat són 49. El p-valor és $P(t_{49} > 13.739) < 2.2e\text{-}16$ (el *RStudio* no ho pot calcular amb precisió perquè es tracta d'un valor molt petit, però en tot cas sabem que és menor a 0.05). El punt crític amb un nivell de confiança del 95% val $t_{0.95,\,49} = 1.676551$.

IC (D, 95%) = IC (log(M) – log (Q), 95%) = IC $\left(\log\left(\frac{M}{Q}\right), 95\%\right) \Rightarrow IC\left(\frac{M}{Q}\right) =$

$= \exp\left[\overline{D} \pm t_{n-1,1-\frac{\alpha}{2}}\sqrt{\frac{S^2}{n}}\right] = \exp [0.3837624, 0.5152608] = [1.467796, 1.674075]$

Com el p-valor és més petit que 0.05 i el punt crític és més petit que l'estadístic, amb un nivell de significació del 5%, es rebutja la hipòtesi nul·la d'igualtat de temps d'execució ($P \approx 0$).

```
data:  dades$logMerge and dades$logQuick
t = 13.739, df = 49, p-value < 2.2e-16
alternative hypothesis: true difference in means is not
equal to 0
95 percent confidence interval:
 0.3837624 0.5152608
sample estimates:
mean of the differences
              0.4495116
```

La mitjana del logaritme de Q està 0.45 log{ms} per sota de M. Si desfem els logaritmes queda:

$\log(M) - \log(Q) = 0.4495116 \Rightarrow \log\left(\frac{M}{Q}\right) = 0.4495116 \Rightarrow \frac{M}{Q} = e^{0.4495116} = 1{,}56754640$

Eficiència relativa = $\left(1 - \frac{M}{Q}\right)\cdot 100 = -56.754$ %. Abans hem calculat l'interval de confiança del rati M/Q. Si ajuntem tots dos resultats ens queda que *Mergesort* triga en executar-se un 56.754% més que el temps del *Quicksort* amb un interval de confiança 95% [46.78% a 67.41%]. Equival a dir que l'algorisme *Mergesort* és, amb una certesa del 95%, entre un 46,78 i un 67,41% més lent que el *Quicksort*.

Descriptiva gràfica

A partir de la Figura 1 podem extreure que, en promig, *quicksort* és més ràpid que *mergesort* com també que té una variància més alta on comprèn un interval major de valors.

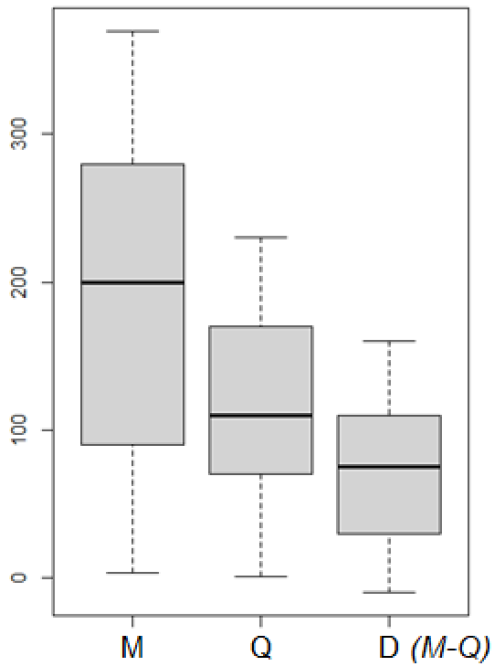


Figura 1: Boxplot del temps de *mergesort*, *quicksort* i de la diferència *merge-quick*.

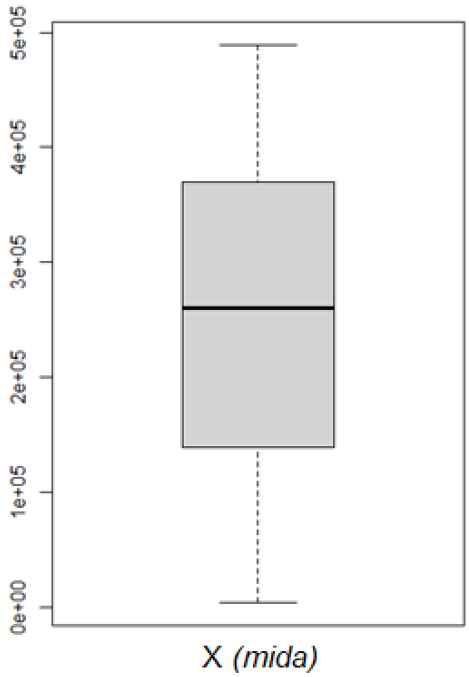


Figura 2: Boxplot de la variable X (mida dels vectors)

Com podem veure en la Figura 3 com més gran és la mida del vector l'algorisme d'ordenació *mergesort* tarda més que el *quicksort*. El creixement de temps es major en *mergesort* com podem veure en la recta d'aproximació del creixement a partir dels valors dels temps a la gràfica. Un cop feta aquesta observació és d'esperar que també com major és el la mida del vector la diferència de temps entre algorismes (*mergesort* menys *quicksort*) també augmenta (que es veu reflectit a la Figura 4, però confirmarem a la següent part del treball).

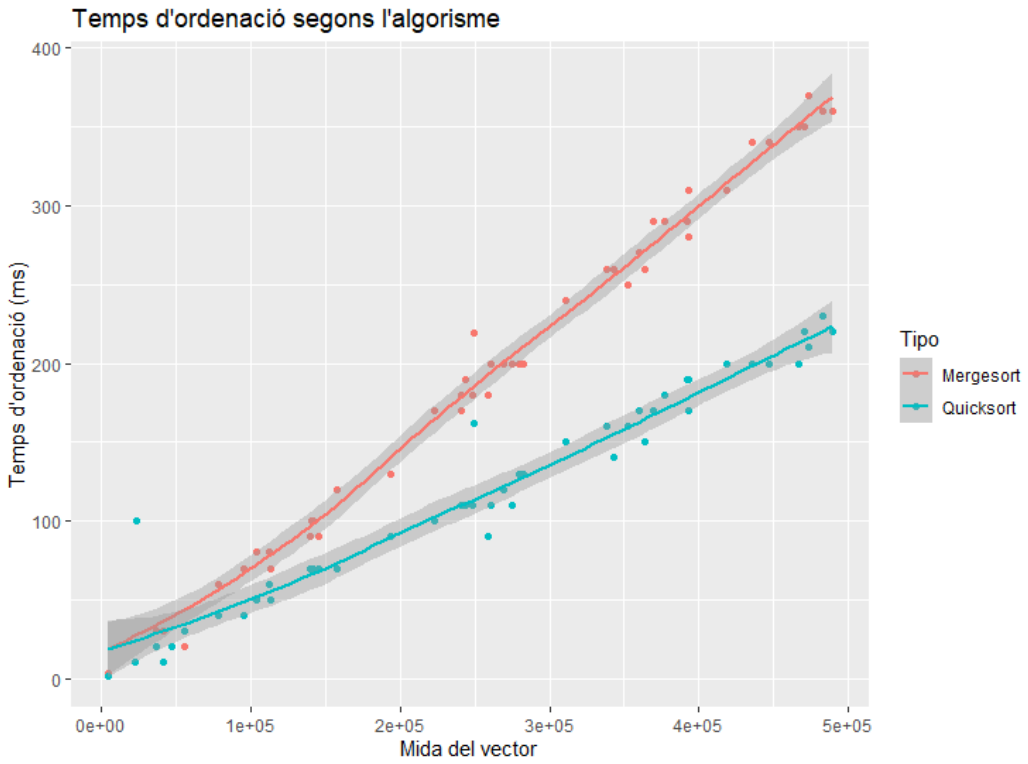


Figura 3: Gràfic de temps d'ordenació dels vectors segons l'algorisme utilitzat

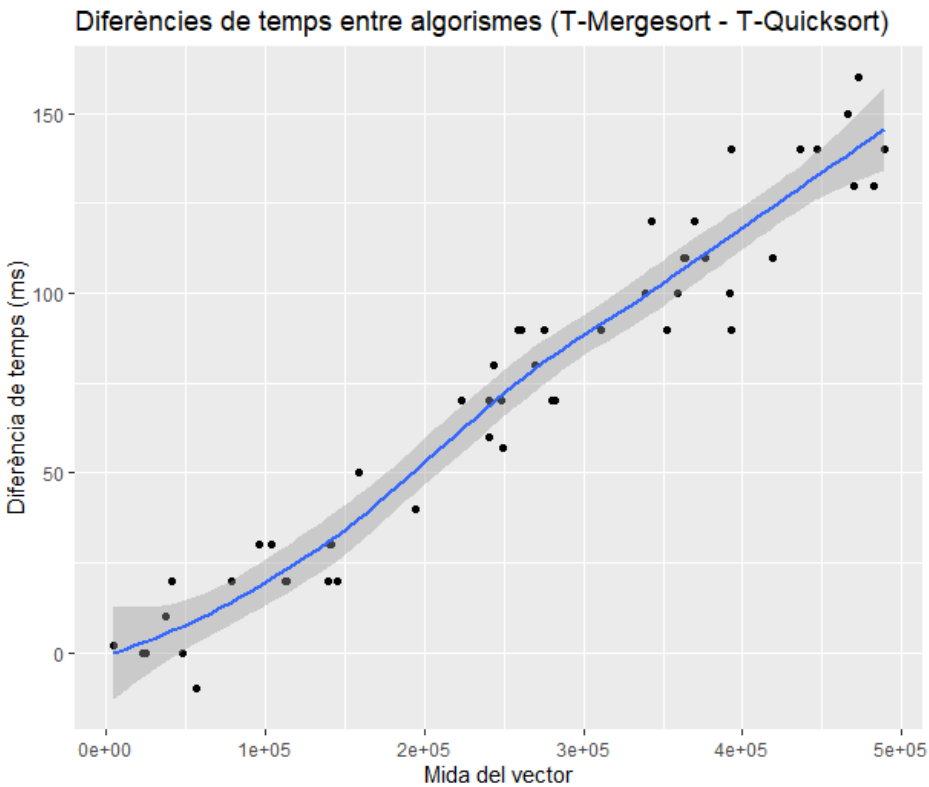


Figura 4: Gràfic de les diferències de temps entre algorismes

Estudi relació Diferències de temps ~ Mesura

En aquest estudi volem comprovar si la D (temps *Mergesort* – temps *Quicksort*, variable Y) varia en funció de la mida d els vectors (variable X). I en cas de que la resposta sigui sí, de quina forma.

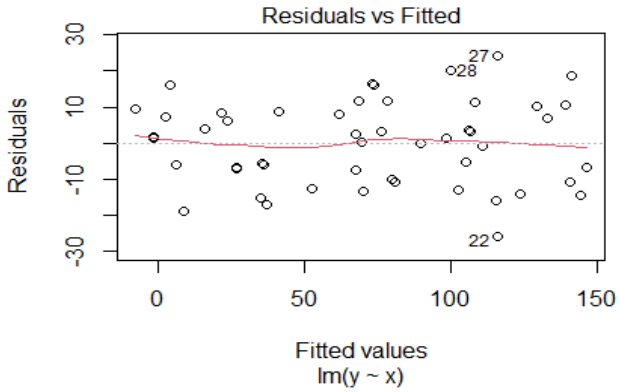


Figura 5: Linealitat i homoscedasticitat (D ~ Mida)

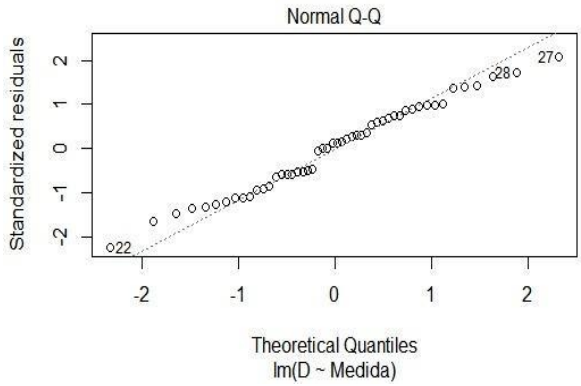


Figura 6: Normalitat

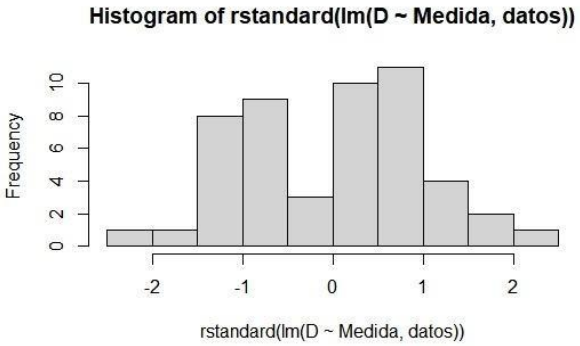


Figura 7: Normalitat

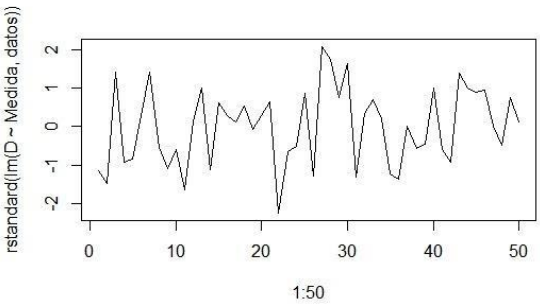


Figura 8: Independència

Anàlisi premisses:

- Linealitat: En la figura 5 podem veure que la línia s'ajusta molt a una recta, és a dir, el núvol de punts es mantenen estables a una mateixa alçada.

- Homoscedasticitat: En aquest mateix gràfic també es pot observar que els punts es mantenen bastant constants, sense que destaqui cap grup pel seu allunyament de la resta.

- Normalitat: En la figura 6 es veu que els residus es situen bastant bé sobre la recta en el qqnorm, només hi ha una petita desviació en les cues. Però no considerem aquesta suficientment gran com per dir que no compleix la premissa de normalitat. En la figura 7 es veu que es podria assemblar a una campana de Gauss però no es

pot assegurar actualment. Però creiem que amb infinits valors recollits aquesta tindria la forma de campana, ja que 50 no són dades suficients per aquest tipus de gràfic.

- Independència: En la figura 8 s'observa que els residus no mostren cap tipus de patró.

Per tant, direm que res s'oposa a validar cap de les 4 premisses.

Estimació de la recta:

Amb R calculem els estimadors de la recta:

$Y_i = \beta_0 + \beta_1 \cdot X_i + \varepsilon_i \rightarrow \hat{y}_i = b_0 + b_1 \cdot X_i$

b_0 = -9.0902084 , b_1 = 0.0003182 , S = 11.73

$\hat{y}_i$ = -9.0902084 + 0.0003182· X_i

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

Coefficients:
(Intercept)            x
-9.0902084    0.0003182
```

Interpretació de b_1: Per cada unitat més de mida, la diferència augmenta 3.182· 10^{-4} ms. És a dir, per cada 1000 unitats més de mida, la diferència augmenta 0.3182 ms. Com que la diferència de temps augmenta en funció de la mida, i sent aquesta *Merge - Quick*. Podem dir que com més gran sigui la mesura del vector més aconsellable és utilitzar el *Quicksort* en comptes de *Mergesort*.

Interpretació de S : És la desviació residual (ha sigut calculada amb el R amb summary(lm(y~x)), podem esperar fluctuacions d'uns 11.73 ms respecte la previsió del temps que ens dona el model.

Interpretació de b_0: Que sigui b_0 -9.0902084 vol dir que amb un vector de mida 0 ($X_i = 0$) s'espera que la diferencia de temps sigui de -9.1 segons, el qual és il·lògic ja que hauria de ser 0. Per aquest motiu ara farem un contrast d'hipòtesi sobre el terme independent.

Prova d'hipòtesi sobre el terme independent (β_0):

- Estadístic: $\hat{t} = \frac{b_0 - \beta'_0}{S_{b_0}}$, - Hipòtesis: H_0: $\beta_0 = 0$ vs H_1: $\beta_0 \neq 0$, - $\hat{t} = \frac{-9.09}{3.398}$ = -2.675

- P-valor: P($|t_{48}| > 2.675$) ≈ 0.0102 (punt crític = $t_{48,0.975}$ = 2.011)

- Conclusió: Com el P-valor < 0.05 o $\hat{t}$ (-2.675) < -2.011, sorprenentment, hem de rebutjar H_0. I per tant un vector de mida 0 no implica tenir una diferència de temps en la ordenació de 0 ms.

-$IC_{95\%}$: IC($\beta_0, 95\%$) = $b_0 \mp t_{n-2,0.975} \cdot S_{b_0}$ = -9.09 ∓ 2.011 · 3.398 = [-15.923, -2.257]

Sincerament, aquest resultat ens ha sorprès, ja que esperàvem no rebutjar H_0 , però els càlculs mostren el contrari. Amb un 95% de confiança, amb X_i = 0, D es trobaria en l'interval [-15.923, -2.257] el qual no només no seria 0, sinó que a més seria negatiu, el qual és impossible ja que estem parlant de temps. Creiem que aquest error ve degut a que la mida dels vectors de la mostra han sigut creats aleatòriament entre 100 i 10^6, però tot i així, el vector més petit és de mida 4518. Si en la mostra hi haguessin més vectors de mida petita aquest interval s'aproparia més al 0.

Prova d'hipòtesi sobre el pendent (β_1):

- Estadístic: $\hat{t} = \frac{b_1 - \beta'_1}{S_{b_1}}$, - Hipòtesis: H_0: $\beta_1 = 0$ vs H_1: $\beta_1 \neq 0$,

- $\hat{t} = \frac{3.182 \cdot 10^{-4}}{1.161 \cdot 10^{-5}}$ = 27.409

- P-valor: P($|t_{48}| > 27.409$) ≈ <2· 10^{-16} (punt crític = $t_{48,0.975}$ = 2.011)

- Conclusió: rebutgem H_0 (P-valor < 0.05 o que 27.409 > 2.011)

-$IC_{95\%}$: IC($\beta_1, 95\%$) = $b_1 \mp t_{n-2,0.975} \cdot S_{b_1}$ = $3.182 \cdot 10^{-4} \mp 2.011 \cdot 1.161 \cdot 10^{-5}$ =

= [$2.95 \cdot 10^{-4}$, $3.42 \cdot 10^{-4}$]

Per tant, no és versemblant que el coeficient del pendent sigui 0, això vol dir que la D varia segons la mida dels vectors, i com aquest es positiu, com més gran sigui la mida, més gran serà la diferència.

Discussió

Els resultats obtinguts amb aquest estudi recolzen que quan el vector no té gaires elements és indiferent quin algorisme usar per ordenar-lo. Però a mesura que aquest té més elements la diferència de temps que tarda el *mergesort* respecte *quicksort* cada vegada és més gran, on *quicksort* és més ràpid. Per tant queda vist que la mida del vector afecta a la durada del algorisme. A més hem fet un anàlisi sobre si quan la mida és 0, el temps també és nul, com seria lògic i és d'esperar però, la nostra mostra ens ha portat a que aquest és negatiu. Nosaltres creiem que és degut a que la nostra mostra recull molts pocs casos, 50 vectors per ser més exactes, on el més petit és de mida 4518 que ha fet portar-nos a aquest resultat. Si a la mostra hi haguessin molts més vectors i n'hi haguessin de mida molt petita també, pensem que obtindríem que el temps amb un vector de mida zero, seria pràcticament nul també tal i com és d'esperar (sense tenir en compte la part del codi que no ordena).

Annex 1: Anàlisis de normalitat i efecte additiu de la diferència

Amb el gràfic Bland-Altman s'observa un efecte multiplicatiu de D (figura 9) que no ens convé per al nostre estudi. Per aquest motiu hem de transformar les dades aplicant logaritme als temps d'execució i ens queda el gràfic de la figura 10 que sí que compleix la premissa d'efecte additiu constant. Per contra, quan analitzem la normalitat de les dades ens adonem que quan apliquem logaritmes (figura 12) en el gràfic qq-norm les cues estan més allunyades de la recta. Com la prova Bland-Altman ens indica que clarament hem d'evitar utilitzar les dades sense transformar si volem aconseguir un efecte additiu, tot i que les dades sense transformar es puguin ajustar millor a la normalitat, aplicarem logaritme als temps (en ms.) i assumirem la normalitat de la variable diferència de logaritmes ja que tots els valors observats s'ajusten prou bé als valors teòrics de la normal.

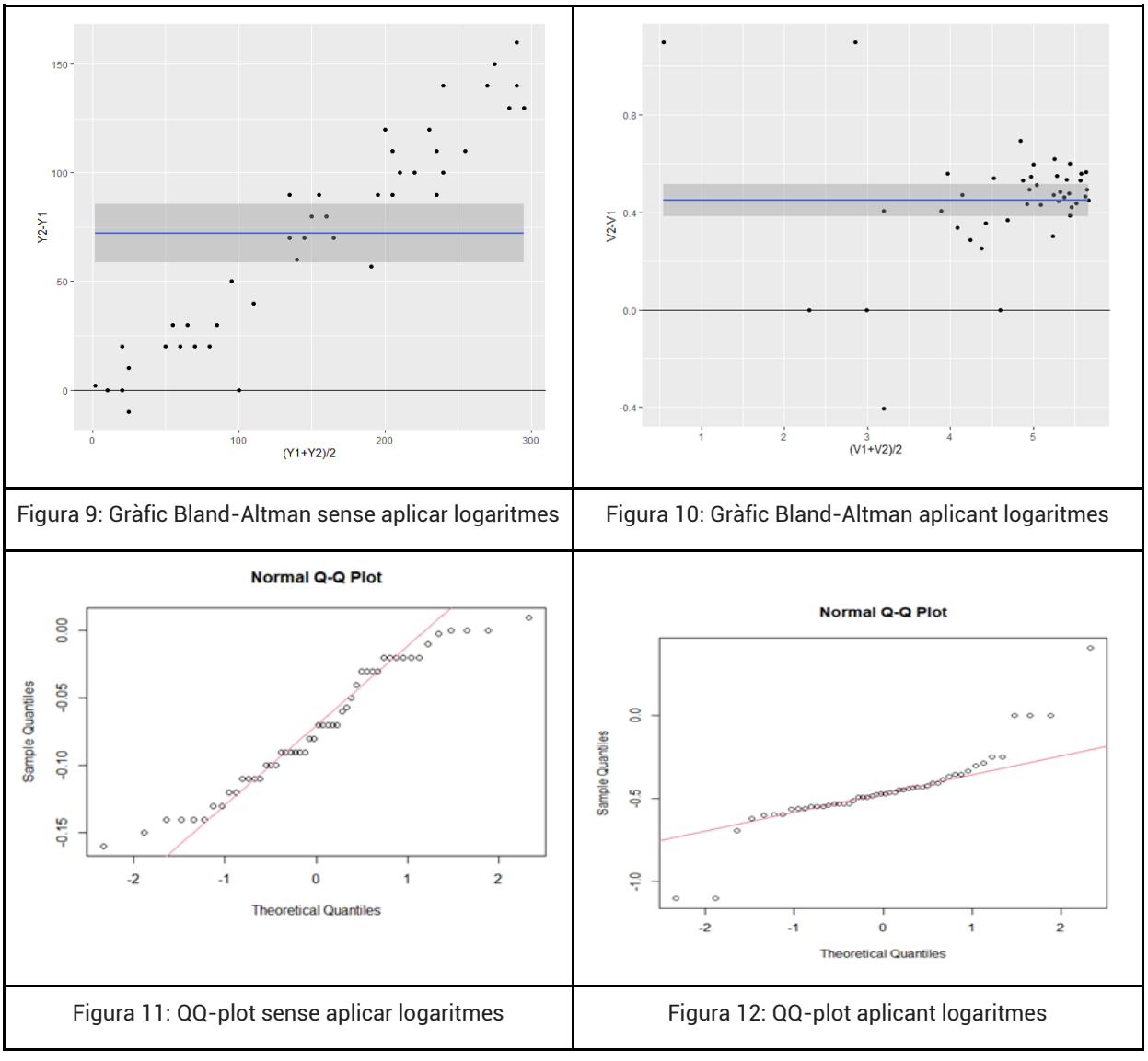


Figura 9: Gràfic Bland-Altman sense aplicar logaritmes

Figura 10: Gràfic Bland-Altman aplicant logaritmes

Figura 11: QQ-plot sense aplicar logaritmes

Figura 12: QQ-plot aplicant logaritmes

Annex 2: Codi en R

```
# Dades de la Taula de l'Annex 3
dades <- read.table("clipboard", header=TRUE)

#Descriptiva
summary(dades)

#Creació variable D (temps Mergesort - temps Quicksort)
dades$D <- dades$Mergesort - dades$Quicksort

#Mitjana D
mean(dades$D)

#Desviació estàndard D
sd(dades$D)

#Transformació logarítmica dels temps
dades$logMerge <- log(dades$Mergesort)
dades$logQuick <- log(dades$Quicksort)

#Creació variable D amb temps logaritmics
dades$logD = dades$logMerge - dades$logQuick

#Mitjana logD
mean(dades$logD)

#Desviació estàndard logD
sd(dades$logD)


# Boxplot del temps de Mergesort,Quicksort i de la diferència(Merge-Quick)
boxplot(dades$Mergesort,dades$Quicksort,D)


# Boxplot de la variable X (mida dels vectors)
boxplot(x)


# Llegir taula per per el següent gràfic (Taula dgràfic de l'Excel)
dgrafic <- read.table("clipboard", header=TRUE)

# Gràfica de Temps d'ordenació segons l'algorisme
ggplot(data = dgrafic,
       mapping = aes(x = Tamaño,
                     y = Tiempo,
                     color = Tipo)) +
  geom_point() +
  geom_smooth() +
  labs(title="Temps d'ordenació segons l'algorisme",
       x = "Mida del vector",
       y = "Temps d'ordenació (ms)")
```

```
# Gràfica de Diferències de Temps (T-Mergesort – T-Quicksort)
ggplot(data = dades,
       mapping = aes(x = Tamaño,
                     y = dades$Mergesort-dades$Quicksort)) +
  geom_point() + geom_smooth() + labs(title="Diferències de temps entre
algorismes (T-Mergesort - T-Quicksort)",
       x = "Mida del vector", y = "Diferència de temps (ms)")

#Test aparellat pels temps amb logaritmes
t.test(dades$logMerge, dades$logQuick, paired=TRUE)

#Gràfic Bland-Altman sense transformació
install.packages('PairedData')
library('PairedData')
x <- dades$Mergesort
y <- dades$Quicksort
p <- paired(y, x)
plot(p, type='BA')


#Gràfic Bland-Altman amb logaritmes
install.packages('PairedData')
library('PairedData')
x <- dades$logMerge
y <- dades$logQuick
p <- paired(y, x)
plot(p, type='BA')

#Grafic QQ-norm sense transformar
qqnorm(dades$D,pch=19)
qqline(dades$D,lwd=2,col=2)
#Gràfic QQ-norm amb logaritmes
qqnorm(dades$logD,pch=19)
qqline(dades$logD,lwd=2,col=2)

#Càlcul punt crític
qt(0.95, 49)


#Part B6, estudi relació D ~ Mida dels vectors
y = dades$D
x = dades$Tamaño

#Càlcul dels estimadors de la recta, PH i IC
mod.lm <- lm(y~x)
summary(mod.lm)

qt(0.975, 48) #Calcul del quantil per 0.975 amb 48 graus de #llibertat (n-
2), per la PH

#Premisses, validació model lineal
par(mfrow=c(2,2)) #Per veure les quatre a l'hora
plot(lm(y~x),c(2,1)) #QQ-Norm i Standard Residuals vs. Fitted, #linealitat,
homoscedasticitat i normalitat
hist(rstandard(lm(y~x))) #Histograma dels residus estandaritzats,
#normalitat
plot (1:50,rstandard(lm(y~x)),type="l") # Ordre dels residus,
#independència
```

Annex 3: Taula de dades

Quicksort	Mergesort	Tamaño
162	219	249161
70	90	145293
90	180	259324
220	350	470346
130	200	279784
150	260	363329
10	30	41265
20	20	47716
90	130	193993
50	70	113086
30	20	56165
10	10	22876
170	290	369738
160	250	351950
200	340	446758
120	200	269240
100	100	24126
50	80	103826
180	290	376728
150	260	363934
20	30	37313
190	280	393061
110	170	240910
70	100	141729
1	3	4518
230	360	482585
170	310	392818
140	260	342512
40	70	95948
210	370	473100
70	90	139523
40	60	78863
100	170	223303
110	180	240854
200	310	418425
190	290	392283
150	240	311213
60	80	112225
170	270	359597
110	200	275136
220	360	489334
130	200	282410
110	200	260934
110	190	243403
200	340	435930
200	350	466333
110	180	248085
70	100	140584
70	120	158237
160	260	338701