Processament de les dades de la llista TOP500

Tractament de les dades per preparar-les i obtenir-ne informació

Pep Turró Mauri

<2018-06-10 Sun>

Índex

1 Descripció del dataset

Les dades sobre les que treballem son un històric del llistat dels superordinadors més potents del món, en funció del seu rendiment a les proves LINPACK. Aquestes llistes les manté el projecte TOP500, que publica dos cops lany (cada Juny i Novembre) la llista actualitzada dels 500 ordinadors amb millor rendiment Linpack del moment.

El conjunt de dades conté les entrades de totes les llistes TOP500 publicades fins ara, amb un registre per cada entrada a una llista. Cal dir que un mateix

sistema sol aparèixer en més duna llista: des de la seva entrada al TOP500, en edicions posteriors pot anar baixant, és a dir, pot aparèixer de nou en una posició diferent.

L'anàlisi d'aquestes dades ens pot servir per obtenir informació diversa, però un tema comú serà l'estudi de l'evolució de la tecnologia de computació. En particular, les dades ens seran útils per observar les implicacions de la llei de Moore [7].

George Moore va predir [5] que el nombre de transistors que pot contenir un circuit integrat es doblaria cada dos anys. A la pràctica, aquest augment de capacitat dels processadors té conseqüències directes sobre la capacitat de càlcul, la quantitat de memòria disponible, i el cost, entre altres. Tot i que les dades de què disposem no permeten contrastar la llei de Moore tal i com està formulada, si que en podrem observar aquestes conseqüències directes, que són tan o més interessants.

Més enllà d'observar l'evolució del rendiment, podrem veure diferents aspectes d'aquesta evolució: els fabricants implicats, les àrees d'aplicació dels superordinadors, els canvis a nivell de sistema operatiu, quins països han apostat per la supercomputació, o si augmenta l'eficiència en consum d'energia. Un altre exemple el trobem a [4], que va utilitzar les mateixes dades per observar l'evolució de les arquitectures d'ordinadors als superordinadors i els països amb superordinadors instalůlats.

També ens podem plantejar estudiar quins factors contribueixen més a l'increment de la potència de càlcul: influeix gaire la quantitat de memòria disponible? i el nombre de nuclis?

2 Integració i selecció de les dades d'interès a analitzar

Carreguem les dades capturades anteriorment via web scraping:

```
# Carreguem les dades, corregint tipus d'atributs quan cal
col.classes <- c(
    'site_id'     = 'factor',
    'system_id'   = 'factor',
    'name'        = 'character',
    'site_name'   = 'character',
    'system_url'  = 'character',
    'site_url'    = 'character'
)
top500 <- read.csv('../data/top500.csv',
                   na.strings = '',
                   colClasses = col.classes)
str(top500)
```

El data frame conté 25000 registres amb 28 variables: `site_name`, `system_url`, `manufacturer`, `cores`, `memory`, `processor`, `interconnect`, `rmax`, `rpeak`, `nmax`, `nhalf`, `hpcg`, `power`, `os`, `compiler`, `math`, `mpi`, `gpu`, `country`, `site_id`, `system_id`, `name`, `site_url`, `city`, `segment`, `year`, `month`, `rank`.

L'estructura de les dades es troba detallada a l'informe de la captura de dades. En aquest document també s'explica l'origen i el contingut de cadascun dels camps. Cal destacar que dos dels camps (*name* i *gpu*) ja provenen d'un cert pre-processament de les dades d'origen: aquests s'han calculat a partir d'un camp original *System* que ja no és part del conjunt de dades amb què treballem, sinó que s'ha processat per calcular aquests dos.

A l'hora de fer anàlisi de les dades no utilitzarem cap de les variables de metadades (identificadors, noms i adreces web), sino que només treballarem amb les variables descriptives, tant quantitatives com qualitatives. Així doncs, eliminem les variables de metadades:

```
# Descartem variables que no són útils per l'anàlisi de les dades.
# Mantenim system_id per si ens cal identificar un mateix sistema
# a diverses edicions de la llista.
descarta <- c('site_id', 'name', 'site_name', 'system_url', 'site_url')
top500 <- top500[ , !(names(top500) %in% descarta) ]
```

Una nota sobre integració de dades: a l'hora de fer la captura ja es va notar que per fer un bon processament de les dades convindria creuar informació amb bases de dades específiques sobre processadors i coprocessadors. He estat cercant una font per aquesta informació que fós accessible i fàcil d'integrar però no l'he trobada; obtenir aquesta informació probablement implicaria un altre projecte de web scraping, i no hi ha hagut temps. A més, com veurem quan revisem els valors buits, hi ha una gran quantitat de registres que no tenen informació sobre coprocessador (variable *gpu*), fent-la per tant menys valuosa per a l'anàlisi; el considerable esforç que implicaria el creuament de dades millorar la qualitat d'aquest camp no es justifica donada la seva poca rellevància: tenim prou anàlisi per fer amb altres variables.

3 Neteja de les dades

3.1 Preparació

Abans de començar el procés de neteja prepararem una mica el terreny per poder facilitar la manipulació de les dades.

3.1.1 Funcions útils

Per algunes de les operacions que farem a continuació ens vindrà bé tenir una funció per ajuntar els valors semblants d'una variable categòrica sota un mateix nom:

```
# Creem una funcio per ajuntar els valors que coincideixen amb un patro.
# Tots els factors que coincideixin amb el patró seran substituïts pel
# nou valor
mergelevels <- function(x, pattern, newvalue, ...) {
  # aquesta funció només modifica valors categòrics
  if (class(x) == "factor") {
    sel <- grep(pattern, levels(x), ...)
    levels(x)[ sel ] <- newvalue
```

```
  }
  x
}
```

3.1.2 Número d'edició de la llista / data de la llista

Farem un pre-processament de les dades des d'un punt de vista pràctic que ens permeti consultar-les d'una forma una mica més còmoda.

L'edició de la llista està representada per dos camps: l'any i el mes. Ens pot resultar més pràctic poder referir-nos a una edició concreta utilitzant un únic camp, i potser ens sigui útil fer-ho per ordre (número d'edició de la llista) o per data (directament amb un camp). Per això crearem dos camps nous:

- *list*: data d'edició de la llista. Les edicions tenen mes i any, assumirem que són del dia 1 del mes.
- *edition*: número d'edició de la llista, en ordre seqüencial en que s'han publicat.

```
# Creem un nou camp "list" que contingui la data de la llista
top500$list <- as.Date(
    ISOdate(top500$year, top500$month, 1, c(0,12))
)
# Creem un nou camp "edition" amb el número d'edició de la llista
# (la primera llista que es va publicar serà la edició número 1, etc)
top500$edition <- factor(top500$list)
levels(top500$edition) <- order(levels(top500$edition))
top500$edition <- as.integer(top500$edition)
```

Un cop tenim aquestes noves variables, les originals *year* i *month* ens seran menys útils. L'any encara ens pot servir per visualitzar millor certes gràfiques, però la del mes ens és innecessària i l'eliminarem:

```
# Eliminem la variable month, que ja no farem servir
top500 <- subset(top500, select = -month)
```

3.2 Correcció d'unitats

Durant l'elaboració d'aquest anàlisi vaig adonar-me d'un problema que va passar desapercebut durant la captura de les dades: en un punt de la història de les llistes TOP500 les unitats de mesura de la potència de càlcul van passar de GFlop/s a TFlop/s!

Ho podem veure si mirem la progressió de *rmax* de l'ordinador més potent per anys:

```
# Veure com hi ha un canvi d'ordre de magnitud a rmax
top500[top500$rank == 1 & top500$month == 6, c('year', 'rmax')]
```

He verificat que a partir de la llista del Juny de 2005 (edició número 25) tots els valors de *rmax* i *rpeak* són en TFlop/s[1], mentre que a totes les edicions

[1] Edició de Novembre de 2004 mostrant unitats en TFlops: https://www.top500.org/lists/2005/06/

anteriors estaven en GFlop/s[2].

Corregirem doncs els valors perquè estiguin tots en la mateixa unitat (GFlop/s):

```
# Passem les mesures més recents, que estan en TFlops, a GFlops per
# tal que els valors de rmax i rpeak estiguin en la mateixa unitat
top500[top500$edition > 24, 'rmax'] <- 1000 *
    top500[top500$edition > 24, 'rmax']
top500[top500$edition > 24, 'rpeak'] <- 1000 *
    top500[top500$edition > 24, 'rpeak']
```

3.3 Duplicitats, simplificació d'informació

A les dades hi ha una certa redundància a les variables qualitatives: certs valors diferents però no prou, ja que les diferències no aporten informació útil. En aquests casos el que farem és unificar els valors semblants quan tingui sentit.

3.3.1 Mateix fabricant amb diferent nom

La variable *manufacturer* (el fabricant de l'ordinador) té uns quants casos de valors que són duplicats amb certes variacions, com per exemple *"Cray Computer"* i *"Cray Inc."*, que voldrem unir sota un únic valor.

En certs casos no és tan evident, ja que hi ha ordinadors que tenen parts de diferents fabricants. Per exmple, tenim *"Dell/Sun/IBM"* o *"IBM/HP"*. Per tal de simplificar, en aquests casos assumeixo que el primer dels fabricants mencionats és el principal i els unifico com a tals:

```
# Unifiquem noms de fabricants
top500$manufacturer <- mergelevels(top500$manufacturer,
                                   '^Cray', 'Cray')
top500$manufacturer <- mergelevels(top500$manufacturer,
                                   '^Dell', 'Dell', ignore.case = TRUE)
top500$manufacturer <- mergelevels(top500$manufacturer,
                                   '^(IBM|Lenovo)', 'IBM')
top500$manufacturer <- mergelevels(top500$manufacturer,
                                   '^(HP|Hewlett)', 'HP')
top500$manufacturer <- mergelevels(top500$manufacturer,
                                   'Fujitsu', 'Fujitsu')
top500$manufacturer <- mergelevels(top500$manufacturer,
                                   'NEC', 'NEC')
top500$manufacturer <- mergelevels(top500$manufacturer,
                                   'Hitachi', 'Hitachi')
top500$manufacturer <- mergelevels(top500$manufacturer,
                                   'ClusterVision', 'ClusterVision')
top500$manufacturer <- mergelevels(top500$manufacturer,
                                   'T-Platforms', 'T-Platforms')
top500$manufacturer <- mergelevels(top500$manufacturer,
                                   'NSSOL', 'NSSOL')
top500$manufacturer <- mergelevels(top500$manufacturer,
```

[2]Edició de Novembre de 2004 mostrant unitats en GFlops: https://www.top500.org/lists/2004/11/

```
                                    'SGI|Networx', 'SGI')
top500$manufacturer <- mergelevels(top500$manufacturer,
                                    'Kendall|KSR', 'KSR')
top500$manufacturer <- mergelevels(top500$manufacturer, 'Raytheon',
                                    'Raytheon-Aspen Systems')
top500$manufacturer <- mergelevels(top500$manufacturer,
                                    'supermicro', 'SuperMicro',
                                    ignore.case = TRUE)
# Unifiquem diversos dissenys propis del NRCPC a la Xina
top500$manufacturer <- mergelevels(top500$manufacturer,
                                    'NRCPC|National Research|University',
                                    'Self-made')
```

Després de d'aquestes correccions hem rebaixat el nombre de fabricants diferents de 124 a 82.

3.3.2 Sistemes operatius duplicats

La variable *os* (sistema operatiu) té 72 valors diferents, però hi ha força casos on la diferència correspòn a la versió del sistema operatiu (per exemple *"RHEL 7.2"* i *"RHEL 7.3"*). Aquestes diferències no aporten gaire informació als possibles anàlisis que poguem fer, o sigui que procedirem a agrupar els sistemes operatius repetits descartant-ne lleugeres variacions:

```
# Agrupem diferents versions d'un mateix sistema operatiu
top500$os <- mergelevels(top500$os, 'OSF/1', 'OSF/1')
top500$os <- mergelevels(top500$os, 'Windows', 'Windows')
top500$os <- mergelevels(top500$os, 'UNICOS', 'UNICOS')
top500$os <- mergelevels(top500$os, 'Ubuntu', 'Ubuntu')
top500$os <- mergelevels(top500$os, 'bullx', 'Bullx', ignore.case = TRUE)
top500$os <- mergelevels(top500$os, 'redhat|rhel',
                         'Red Hat Enterprise Linux',
                         ignore.case = TRUE)
top500$os <- mergelevels(top500$os, 'suse|SLES', 'SuSE Linux',
                         ignore.case = TRUE)
```

Després d'això la variable *os* ha passat a tenir 41 valors diferents.

3.4 Valors perduts, zeros o buits

La funció `summary` ens ajuda a trobar quines variables conenen valors buits (i quants) i fer-nos una idea del rang de valors cobert, i detectar per tant la presència de zeros que poden ser indicatius d'un error a les dades:

```
# El resum del data frame ens permet veure quines variables
# contenen NAs
summary(top500)
```

```
 manufacturer      cores              memory
HP     :7939   Min.   :       1   Min.   :      8
IBM    :7604   1st Qu.:      96   1st Qu.:  14976
```

```
Cray   :2991   Median :     960   Median :  48384
Oracle :1224   Mean   :   16853   Mean   : 117538
Fujitsu: 818   3rd Qu.:    9504   3rd Qu.: 125440
NEC    : 626   Max.   :19860000   Max.   :7685540
(Other):3798                      NA's   :21379
                processor                interconnect        rmax
Xeon E5-2670 8C 2.6GHz: 1120   Gigabit Ethernet: 5053   Min.   :       0
POWER3  375MHz        :  597   Infiniband      : 1399   1st Qu.:      48
POWER2  66MHz         :  553   SP Switch       : 1378   Median :    2500
Xeon E5450 4C 3GHz    :  529   10G Ethernet    : 1349   Mean   :  210688
R10000  195MHz        :  469   Infiniband FDR  : 1301   3rd Qu.:   85800
PowerPC 604e  332MHz  :  444   (Other)         :10293   Max.   :93014600
(Other)               :21288   NA's            : 4227
    rpeak                nmax              nhalf              hpcg
Min.   :        0   Min.   :    1900   Min.   :    112   Min.   :  1.70
1st Qu.:       64   1st Qu.:   23000   1st Qu.:   2400   1st Qu.: 31.50
Median :     4500   Median :  200000   Median :   5200   Median : 80.79
Mean   :   315621   Mean   :  983267   Mean   :  48915   Mean   :141.78
3rd Qu.:   130000   3rd Qu.: 1486952   3rd Qu.:  18560   3rd Qu.:167.05
Max.   :125435900   Max.   :16662804   Max.   :1834944   Max.   :602.74
                    NA's   :14068      NA's   :18993     NA's   :24610
    power                      os                            compiler
Min.   :   18.0   Linux          :10990   Intel                 :   90
1st Qu.:  172.0   AIX            : 2843   icc                   :   83
Median :  374.0   UNICOS         : 2047   Intel 12.1            :   82
Mean   :  862.5   IRIX           : 1314   GCC                   :   75
3rd Qu.:  831.0   HP Unix (HP-UX): 1294   intel parallel studio:   44
Max.   :19431.0   Solaris        : 1091   (Other)               :  653
NA's   :17970     (Other)        : 5421   NA's                  :23973
             math                                        mpi
MKL             :  228   Intel MPI                        :  248
Intel MKL       :  188   MVAPICH 1.2                      :   60
mkl             :   43   OPEN MPI                         :   40
MKL - 11.3.3.210:   33   MPICH2 with a custom GLEX channel:   36
MKL,CUDA6.5     :   23   Intel MPI 4.0                    :   34
(Other)         :  335   (Other)                          :  658
NA's            :24150   NA's                             :23924
             gpu                  country        system_id
GigE            :  448   United States :12675   174016 :   16
Gig-Ethernet    :  334   Japan         : 2346   176819 :   16
Custom          :  301   Germany       : 1924   176908 :   16
Xeon 54xx 3.0GHz:  263   China         : 1695   176899 :   15
Xeon 51xx 3.0GHz:  171   United Kingdom: 1446   176924 :   15
(Other)         : 5274   France        : 1026   176928 :   15
NA's            :18209   (Other)       : 3888   (Other):24907
        city              segment           year          rank
Tokyo     :  549   Academic  : 5010   Min.   :1993   Min.   :  1.0
Livermore :  368   Classified:  432   1st Qu.:1999   1st Qu.:125.8
Los Alamos:  336   Government: 1009   Median :2005   Median :250.5
Houston   :  331   Industry  :11601   Mean   :2005   Mean   :250.5
```

```
Beijing   :  317   Others    :    7   3rd Qu.:2011   3rd Qu.:375.2
(Other)   :12893   Research  : 6110   Max.   :2017   Max.   :500.0
NA's      :10206   Vendor    :  831
     list              edition
Min.   :1993-06-01   Min.   : 1.0
1st Qu.:1999-06-01   1st Qu.:13.0
Median :2005-08-16   Median :25.5
Mean   :2005-08-16   Mean   :25.5
3rd Qu.:2011-11-01   3rd Qu.:38.0
Max.   :2017-11-01   Max.   :50.0
```

De la sortida en podem destacar uns quants aspectes:

- Les variables *rpeak* i *rmax* no contenen valors buits (*NA*), però aparenten tenir zeros (Min == 0). Cal notar que per aquestes variables el zero no és un valor vàlid: qualsevol ordinador té una velocitat de càlcul superior a 0. Al verificar les dades, però, resulten ser els sistemes en les darreres posicions de les primeres llistes, amb velocitats de càlcul inferior a 1 GFlop/s (fraccions entre 0 i 1). És a dir: en aquestes dues variables, fonamentals per l'anàlisi, no hi ha valors buits o zeros.
- Altres variables sense valors buits o zeros i que poden ser útils per anàlisi són: *segment*, *country*, *manufacturer*, *cores*
- Hi ha algunes variables que contenen un nombre molt elevat de valors nuls (més de la meitat de les observacions, en alguns casos la pràctica totalitat): *math*, *hpcg*, *nmax*, *nhalf*, *power*, *compiler*, *mpi*, *gpu*, *memory*.
- Altres variables contenen valors buits però en una proporció menor: *interconnect*, *os*.

En resum: tenim algunes variables de bona qualitat, sense valors nuls o invàlids i que són rellevants per a l'anàlisi. D'altres contenen valors nuls, que són vàlids en el sentit que reflexen la manca d'informació.

No faré cap més tractament d'aquests valors perduts, els mantindrem tal i com estan. Les alternatives per tractar-los no tenen sentit en aquest cas: no volem descartar registres sencers només perquè els manca informació de, per exemple, quin compilador fan servir; tampoc tenim a l'abast omplir la informació que falta; no té gaire sentit tampoc en el nostre cas intentar calcular valors per les variables afectades, sobretot a les variables on la majoria dels valors són buits (i a les que no, com *os*, un valor calculat no ens aportaria més valor per a l'anàlisi que un valor buit). Simplement acceptarem aquesta presència de valors buits i ho tindrem en compte a l'hora de l'anàlisi. Amb un parell d'excepcions, però, que veurem a continuació.

3.4.1 Descartar variables

Comentava que no tractarem valors perduts a nivell de registres (files), però si que hi ha dues variables (columnes) que descartarem senceres.

La variable *gpu* a més de tenir un elevat nombre de valors nuls també conté valors invàlids: com comentàvem a l'apartat d'integració i selecció de les dades, sabem que el contingut de *gpu* és imprecís i no s'ha pogut corregir. Aquesta variable, doncs, no ens servirà i la descartarem:

```
# Descartem la vairable gpu pel seu elevat nombre de valors
# nuls i invàlids
top500 <- subset(top500, select = -gpu)
```

L'altre cas és la variable *hpcg*: és la variable que té més valors nuls, un 98.44 % dels seus valors són buits. A més, els pocs valors que té només es troben a les edicions més recents de la llista TOP500:

```
# Comprovem que només tenim valors d'hpcg per les edicions recents
table(top500[! is.na(top500$hpcg), "edition" ])
```

```
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
 1  1  4  5 10 13 15 17 21 23 32 35 43 52 58 60
```

Tot plegat, això vol dir que no ens servirà gaire per dur a terme el tipus d'estudi que volem fer aquí, basat en l'evolució al llarg dels anys. Per tant, també la descartem:

```
# Descartem la vairable hpcg pel seu elevat nombre de valors
# buits i la seva nulůla presència a les primeres 34 edicions
top500 <- subset(top500, select = -hpcg)
```

3.4.2 Valors "N/A"

Durant la feina de neteja amb sistemes operatius duplicats hem vist que hi ha alguns registres de la variable *os* que tenen el valor "N/A". No el valor nul *NA* sinó una cadena de caràcters (de fet, un factor en aquest cas) amb el valor textual "N/A". Per reflectir adequadament el seu significat (informació no disponible), substituirem aquests valors per el valor nul, que R representa com *NA*.

Repassant les altres variables també he trobat el mateix tipus de problema a la variable *compiler*, o sigui que hi aplicarem el mateix tipus de correcció:

```
# Corregir el valor del factor "N/A" reemplaçant-lo per NA
# a les variable os i compiler
top500[!is.na(top500$os) & top500$os == "N/A", "os"] <- NA
top500[!is.na(top500$compiler) & top500$compiler == "N/A", "compiler"] <- NA
# Desfer-nos de la categoria "N/A" ara que està buida
top500$os <- factor(top500$os)
top500$compiler <- factor(top500$compiler)
```

3.5 Valors extrems

Estem treballant amb dades de superordinadors, és a dir, ordinadors "extrems", i no serà sorprenent que ens trobem també amb valors extrems a les dades. Vegem-ho.

Tenint en compte que les dades contenen informació sobre la potència de càlcul dels superordinadors (a les variables *rmax* i *rpeak*) al llarg de 25 anys, i amb la llei de Moore al cap, ja podem suposar que ens toparem amb una distribució força esbiaixada dels valors numèrics: els ordinadors actuals són molt més potents que els de fa 25 anys.

[3] posa un exemple d'aquesta situació basat en [2] i suggereix transformar aquest tipus de dades utilitzant logaritmes per aconseguir una distribució més simètrica.

Aplicant-ho a les variables *rmax* i *cores*, en veiem el resultat a la figura 1.

```
# Boxplot de rmax i cores i els seu logaritme, un al costat de l'altre
par(mfrow=c(2,2))
boxplot(top500$rmax, xlab = "rmax")
boxplot(log(top500$rmax), xlab = "log(rmax)")
boxplot(top500$cores, xlab = "cores")
boxplot(log(top500$cores), xlab = "log(cores)")
```

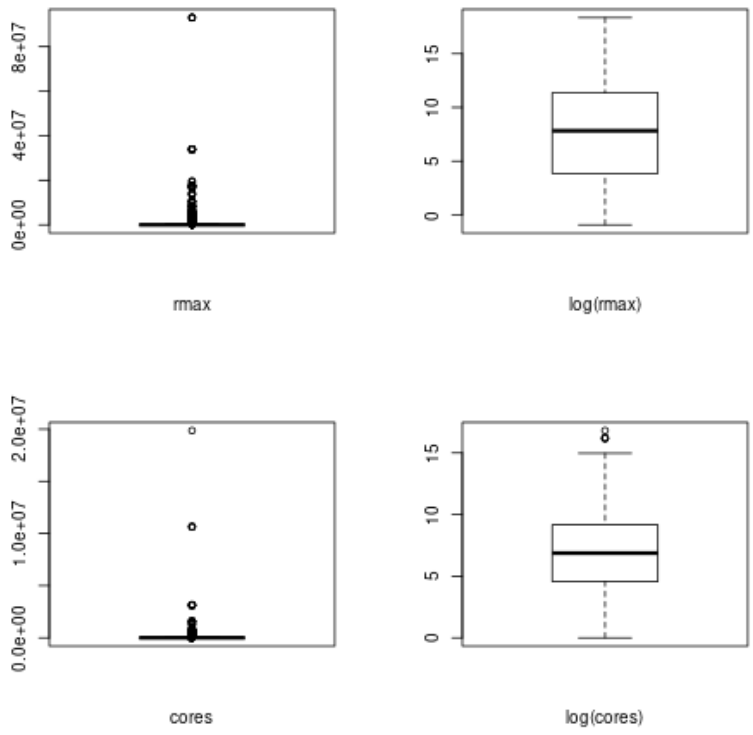


Figura 1: Boxplot de *rmax* i *cores* i els seus logaritmes

Es pot veure com el boxplot dels valors reals ens permet veure només valors extrems, mentre que el dels logaritmes dels valors ens presenta una distribució molt més *normal*. Això ens està mostrant la distribució exponencial del rendiment i del nombre de nuclis de còmput, coherent amb el que esperàvem.

Passa el mateix amb totes les variables numèriques que tenim: resulta que totes les variables numèriques presenten valors que normalment es considerarien extrems. Com deiem, però, en el nostre cas és una situació esperada. No es tracta d'errors de captura, registre o mostreig; simplement, les dades amb què

treballem són així: és precisament com han evolucionat els ordinadors, i és el que volem analitzar.

Per tant no hem de modificar, eliminar o amagar aquests valors, simplement hem de ser conscients de com són les dades amb les que estem treballant en aquest cas.

3.6 Atributs calculats

De cara a possibles anàlisi ens pot interessar preparar algun atribut addicional calculat a partir de les dades existents.

Abans ja hem creat el número d'edició i la data de cada llista amb finalitats pràctiques, per facilitar-nos referenciar les dades. Més enllà d'això també podem generar nous atributs que ens serveixin directament per a l'anàlisi.

3.6.1 Familia de Sistema Operatiu

Anteriorment hem simplificat la informació de sistemes operatius eliminant detalls de versió que no són rellevants per als anàlisi. Tot i això, encara ens queden 40 sistemes operatius diferents, i de cara a analitzar l'evolució dels sistemes operatius dels superordinadors ens pot anar bé tenir-los agrupats per famílies o tipus de sistema operatiu.

Crearem un nou atribut *osfamily* agrupant els sistemes operatius en 3 grans families: "Unix", "Linux" i "Windows".

```
# Agrupació de sistemes operatius per families
top500$osfamily <- top500$os

linux <- c(
    'Linux', 'Ubuntu', 'CentOS', 'Bullx', 'RaiseOS', 'TOSS', 'CNL'
)
top500$osfamily <- mergelevels(top500$osfamily,
                               paste0(linux, collapse="|"),
                               'Linux', ignore.case = TRUE)

unix <- c(
    'AIX', 'IRIX', 'HP', 'Unix', 'CMOST', 'Solaris', 'SunOS',
    'MacOS', 'HI-UX', 'Ultrix', 'PARIX', 'Super-UX', 'UNICOS',
    'ConvexOS', 'SPP-UX', 'Tru64', 'OSF/1', 'KSR', 'EWS', 'UXP'
)
top500$osfamily <- mergelevels(top500$osfamily,
                               paste0(unix, collapse = "|"),
                               'Unix', ignore.case = TRUE)

other <- c('Cell OS', 'CRS-OS', 'NX/2', 'Paragon')
top500$osfamily <- mergelevels(top500$osfamily,
                               paste0(other, collapse="|"),
                               'Other')
table(top500$osfamily)
```

Unix	11457
Linux	13280
Other	126
Windows	50

3.7 Desar les dades netejades

Un cop tenim les dades a punt per fer anàlisi, en desem una còpia com a referència per poder carregar-les ja processades quan convingui:

```
# Desem les dades pre-processades
write.csv(top500, '../data/top500-clean.csv', row.names = FALSE)
```

4 Anàlisi de les dades

A la descripció del dataset platejàvem unes quantes preguntes que ens podríem plantejar a partir de les dades de què disposem:

1. Es poden veure els efectes de la predicció/llei de Moore?
2. Com han evolucionat els fabricants o els sistemes operatius utilitzats pels superordinadors?
3. Quins països disposen de superordinadors?
4. Podem establir quins factors contribueixen més a l'augment de la potència de càlcul? (memòria, CPU, potència consumida)

Per respondre a les tres primeres preguntes el millor és representar adequadament la informació de què disposem, no cal dur a terme càlculs específics, i per tant ho afrontarem a la secció de representació dels resultats.

En canvi, per estudiar com afecten els processadors, memòria o potència a la velocitat de càlcul si que ens caldrà obtenir models estadístics, que és el que farem en aquest apartat.

4.1 Selecció dels grups de dades a analitzar/comparar

Ens disposem a analitzar quins factors contribueixen més a l'augment de velocitat de càlcul dels superordinadors (representada per la variable *rmax*).

Començarem per buscar possibles factors entre les altres variables numèriques de què disposem. Les variables *rpeak*, *nmax* i *nhalf* estan relacionades directament amb *rmax*: totes són mesures relacionades amb la velocitat de càlcul. Podría ser interessant especialistes, sobretot pels fabricants de superordinadors, estudiar relacions entre elles, però a nosaltres el que ens interessa és comparar el rendiment real assolit (*rmax*) amb altres factors.

Mirarem, doncs, les altres variables numèriques que ens queden: *cores*, *memory* i *power*.

Com a primer pas farem una visualització per parelles per veure si s'aprecia alguna aparent correlació (figura 2). Tal i com hem comentat quan hem revisat els valors extrems, treballem amb els logaritmes dels valors per tal de visualitzar de forma lineal el seu comportament exponencial.

```
# Diagrames de punts per parelles de les variables numèriques
# candidates a comparar
pairs(~ rmax + cores + memory + power, data = top500, log = "xy")
```

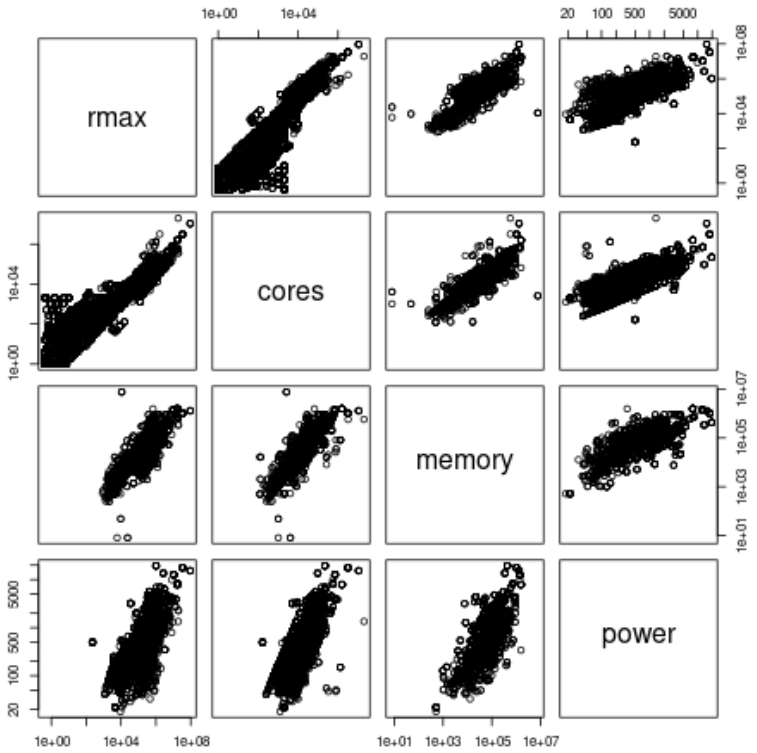


Figura 2: Diagrames de punts per parelles de les variables numèriques candidates a comparar amb el rendiment

Aparentment s'observen relacions entre el rendiment i la resta de variables, especialment amb el nombre de nuclis.

Ho confirmarem amb les proves estadístiques.

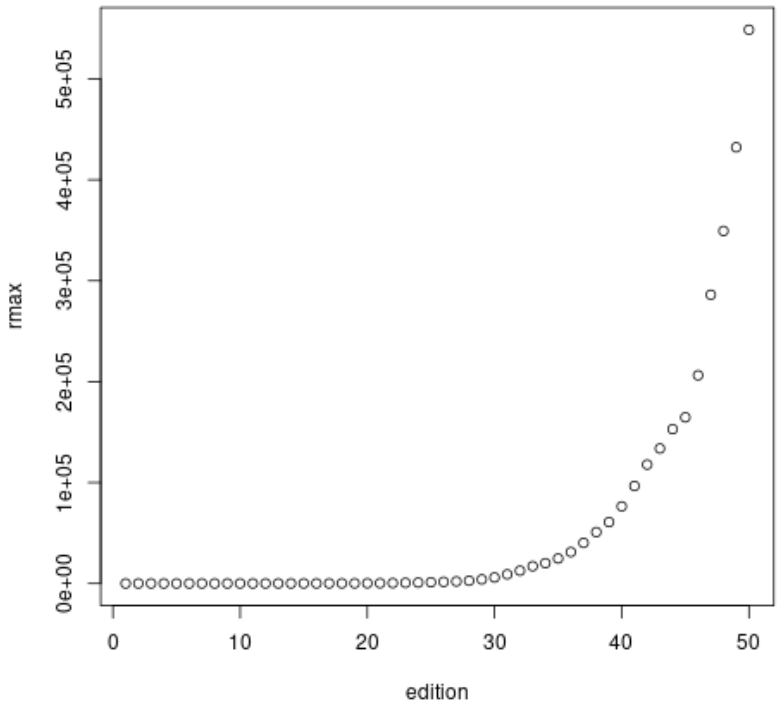
4.2 Comprovació de la normalitat i homogeneïtat de la variància

Tal i com hem comentat durant la revisió de valors extrems, les dades amb què treballem no segueixen una distribució normal, com es veu clarament si representem els valors de rmax al llarg de les diferents edicons.

Per fer-ho utilitzarem només el darrer ordinador de cada llista (el que ocupa la posició número 500); això és perquè aquest representa l'evolució de la potència de càlcul d'una manera més suau i realista: accedir a la posició número 1 és molt difícil i ens trobem que només hi ha canvis a la posició 1 de la llista quan al món algú desenvolupa l'ordinador més potent del moment, cosa que no succeeix

amb facilitat; en canvi, quan més avall a la llista, més disputada està la plaça i per tant l'evolució de la última plaça és la que representa millor l'evolució global de la velocitat de càlcul[3].

```
# Visualitzem la potència de càlcul de l'últim ordinador de cada
# edició de la llista TOP500
bottom500 <- top500[top500$rank == 500, ]
with(bottom500, plot(edition, rmax))
```



4.3 Aplicació de proves estadístiques

Anem a veure quina de les tres variables *memory*, *cores* i *power* té més relació amb el rendiment dels ordinadors (*rmax*).

Per fer-ho aplicarem un model de regressió entre *rmax* i cadascuna d'elles, compararem quin encaixa millor i quins són els respectius índex de correlació.

```
# Regressió entre rmax i cores
rmaxcores <- lm(log10(top500$rmax) ~ log10(top500$cores))
summary(rmaxcores)
```

[3]recordem que no ens interessa tant el valor absolut de rendiment com l'estudi de la seva evolució all llarg del temps

```
Call:
lm(formula = log10(top500$rmax) ~ log10(top500$cores))

Residuals:
    Min      1Q  Median      3Q     Max
-4.2254 -0.2339  0.0703  0.3124  2.2617

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         -0.873142   0.009574   -91.2   <2e-16 ***
log10(top500$cores)  1.419542   0.003015   470.9   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.5879 on 24998 degrees of freedom
Multiple R-squared:  0.8987,Adjusted R-squared:  0.8987
F-statistic: 2.217e+05 on 1 and 24998 DF,  p-value: < 2.2e-16
```

```r
# Regressió entre rmax i memory
rmaxmem <- lm(log10(top500$rmax) ~ log10(top500$memory))
summary(rmaxmem)
```

```
Call:
lm(formula = log10(top500$rmax) ~ log10(top500$memory))

Residuals:
    Min      1Q  Median      3Q     Max
-3.4195 -0.2610 -0.0287  0.2525  2.3887

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)           1.16229    0.04825   24.09   <2e-16 ***
log10(top500$memory)  0.91416    0.01036   88.20   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.4453 on 3619 degrees of freedom
  (21379 observations deleted due to missingness)
Multiple R-squared:  0.6825,Adjusted R-squared:  0.6824
F-statistic:  7780 on 1 and 3619 DF,  p-value: < 2.2e-16
```

```r
# Regressió entre rmax i power
rmaxpower <- lm(log10(top500$rmax) ~ log10(top500$power))
summary(rmaxpower)
```

```
Call:
lm(formula = log10(top500$rmax) ~ log10(top500$power))

Residuals:
```

```
     Min       1Q   Median       3Q      Max
-2.86498 -0.35771 -0.06859  0.43162  1.82276

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)          2.19657    0.03621   60.66   <2e-16 ***
log10(top500$power)  1.12273    0.01362   82.43   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.5766 on 7028 degrees of freedom
  (17970 observations deleted due to missingness)
Multiple R-squared:  0.4915,Adjusted R-squared:  0.4915
F-statistic:  6794 on 1 and 7028 DF,  p-value: < 2.2e-16
```

Comparant els índex de correlació R^2 veiem que la variable *cores* (número de nuclis) és la que presenta una correlació més elevada, de 0.899, mentre que en el cas de *power* (consum) baixa fins a 0.49.

A l'apartat següent representarem gràficament aquestes relacions entre potència i les altres variables i ho confirmarem de manera més visual.

Comentar també que aquí tornem a veure la presència de valors buits a les dades: a part de *rmax*, només *cores* conté valors a tots els registres, mentre que per a *power* i sobretot *memory* s'han descartat automàticament un gran nombre d'entrades per tenir valor buit.

5 Representació dels resultats

Representant gràficament les dades n'obtenim informació important. De fet, és la millor manera de resoldre un dels objectius que hem plantejat al principi: el d'observar les conseqüències de la llei de Moore.

5.1 Verificació de la llei de Moore

Anem a visualitzar l'evolució de la capacitat de càlcul dels superordinadors durant els darrers 25 anys, representada pel seu rendiment Linpak mesurat en milers de milions d'operacions de càlcul per segon (GFlop/s), que a les dades que tenim correspòn a la variable *rmax*.

Quan hem comentat al parlar de la distribució de les dades hem explicat que fixar-nos en l'ordinador a la posició 500 ens permet observar millor l'evolució en global. D'altra banda, també pot resultar interessant veure l'evolució del superordinador més potent (a la posició 1 de la llist). Representarem, doncs, l'evolució de tots dos ordinadors (primer i últim) per a cada edició.

A [6] es suggereix utilitzar el paquet *ggplot2* de R, i l'utilitzo en alguns dels gràfics a continuació.

```
library(ggplot2)
# Dibuixem l'evolució del rendiment del primer i últim ordinador de
# cada llista al llarg del temps. Segons la llei de Moore aquesta
# progressió és exponencial, i per tant fem servir una escala logarítmica
# a l'eix del rendiment. Hi dibuixem també models lineals per
```

```
# visualitzar millor l'evolució
ggplot(data = top500[top500$rank == 1 | top500$rank == 500, ],
       aes(x = edition, y = rmax, colour = factor(rank))) +
    labs(x = 'Edició de la llista', y = 'Rendiment (rmax GFlop/s)',
         colour = 'Posició') +
    geom_point() +
    geom_smooth(data = top500[top500$rank == 1,], method = 'lm') +
    geom_smooth(data = top500[top500$rank == 500,], method = 'lm') +
    scale_y_log10()
```

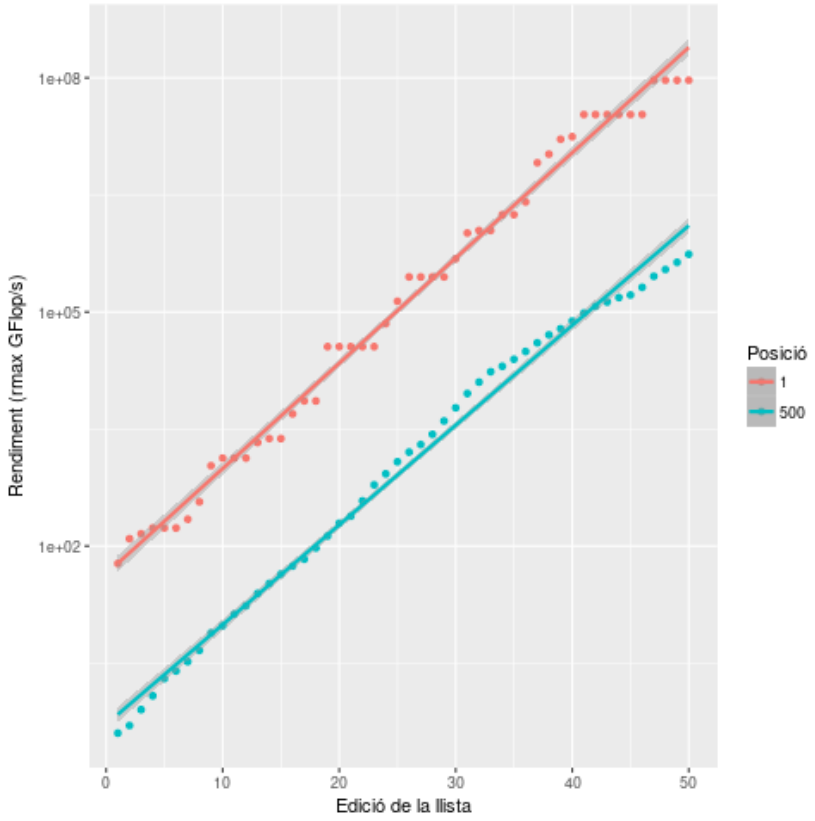


Figura 3: Evolució del rendiment amb el temps

Efectivament, a la figura 3 veiem la progressió exponencial (recordem que la representació és en escala logarítmica) que han seguit fidelment els superordinadors, conseqüència directa de la predicció de Moore.

Com a curiositat, també veiem a la gràfica que al voltant de la 17a edició de la llista (o sigui, en un interval d'entre 8 i 9 anys) l'ordinador menys potent de la llista ja era més potent que el primer de la primera edició.

5.2 Relacions entre potència, memòria, nuclis i consum

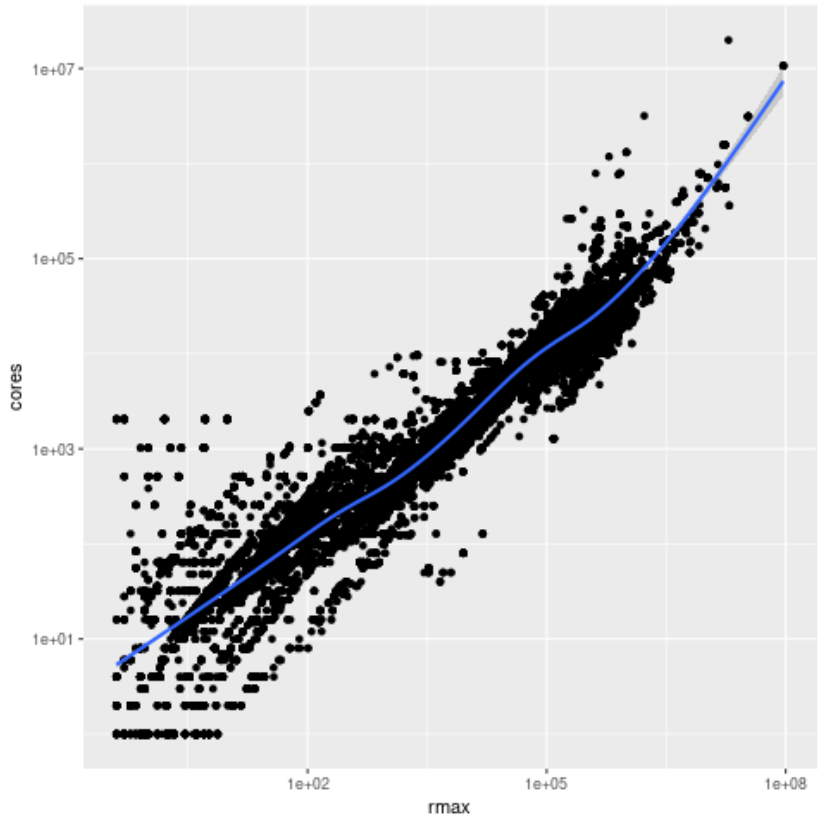
Aquí visualitzarem gràficament les proves estadístiques que hem fet per comparar el número de nuclis de computació (*cores*), la quantitat de memòria (*mem-*

ory) i el consum energètic (*power*) dels superordinadors amb el seu rendiment (*rmax*).

5.2.1 Potència vs nuclis

Començarem per la variable que hem vist que està més relacionada amb el rendiment: el número de nuclis.

```
# rmax vs cores
ggplot(data = top500, aes(x = rmax, y = cores)) +
    geom_point() + geom_smooth() +
    scale_x_log10() + scale_y_log10()
```

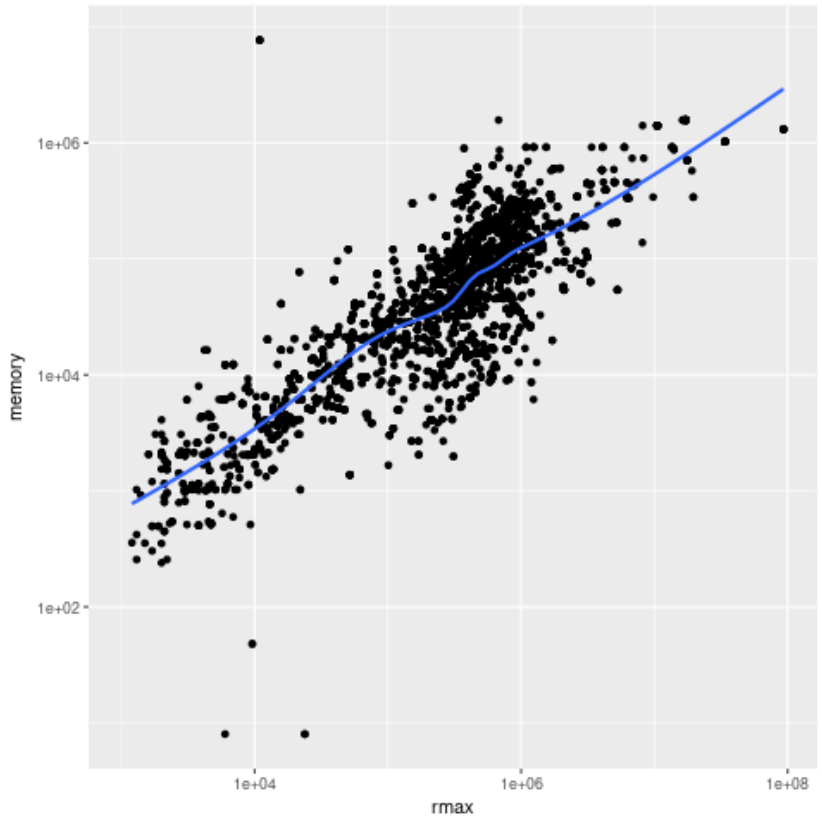


Podem veure la relació pràcticament lineal (en escala logarítmica) amb claredat.

5.2.2 Potència vs memòria

Representem ara la relació amb la memòria. Recordem que en aquest cas disposem de moltes menys dades, i per tant el gràfic conté molts menys punts:

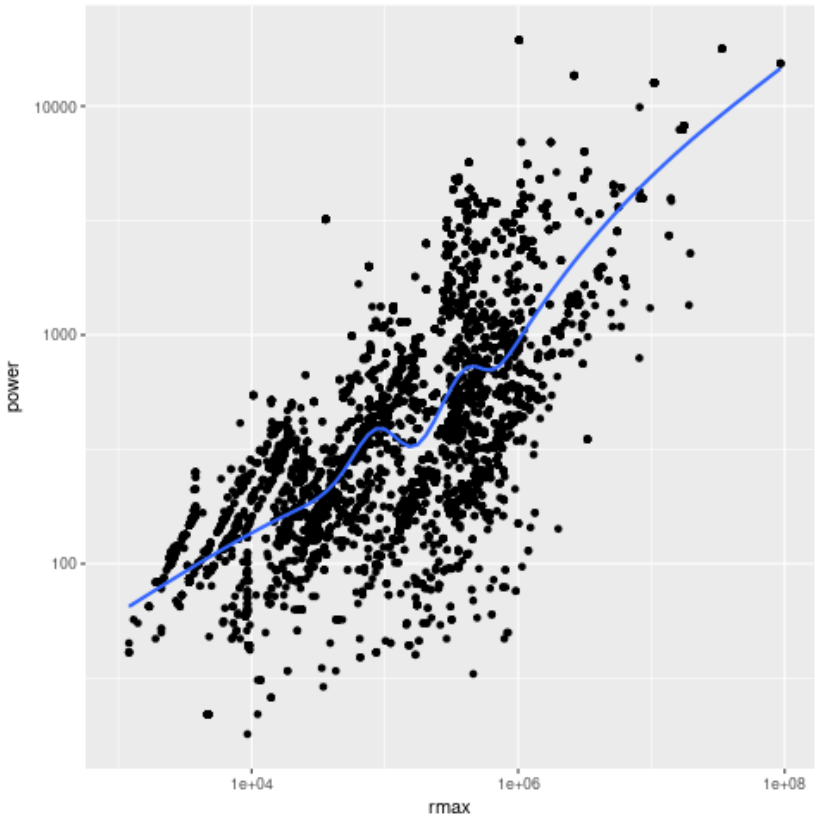
```
# rmax vs memory
ggplot(data = top500[top500$rmax > 1000,], aes(x = rmax, y = memory)) +
    geom_point() + geom_smooth(se = FALSE) +
    scale_x_log10() + scale_y_log10()
```



Comparant-lo amb el gràfic del número de nuclis veiem que, efectivament, el núvol de punts és més dispers al voltant de la possible recta de regressió, indicant-nos que no és un factor tant determinant.

5.2.3 Consum per potència

```
# rmax vs power
ggplot(data = top500[top500$rmax > 1000,],
       aes(x = rmax, y = power)) +
    geom_point() + geom_smooth(se = FALSE) +
    scale_x_log10() + scale_y_log10()
```

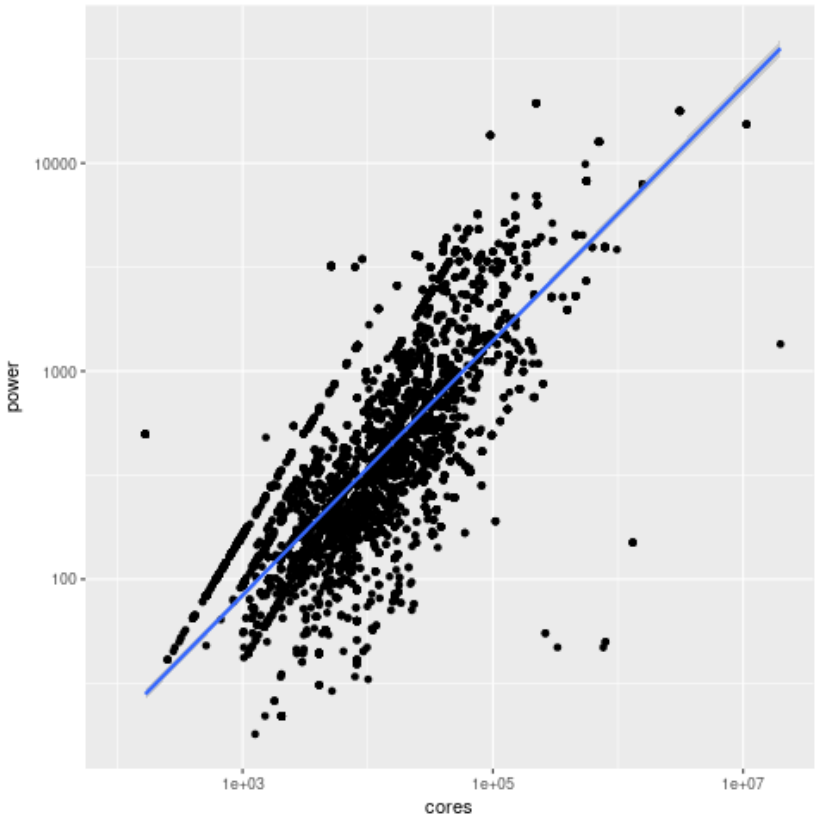


En el cas del consum energètic encara ens trobem amb més dispersió. De fet, ens caldria forçar que la visualització utilitzi un model lineal de regressió, ja que altrament el mètode d'aproximació automàtic no ens aproxima tan bé una recta com els anteriors.

5.2.4 Consum per nuclis

Com a visualització addicional no relacionada amb la cerca dels factors que contribueixen al rendiment, representarem també el consum en funció del nombre de nuclis. La idea és intentar veure si el consum energètic augmenta de forma proporcional al factor que hem vist que contribueix més al rendiment.

```
# power vs cores
ggplot(data = top500[top500$cores > 100,], aes(x = cores, y = power)) +
    geom_point() + geom_smooth(method = "lm") +
    scale_x_log10() + scale_y_log10()
```



6 Resolució del problema

Resumim les conclusions del que hem observat a les dades.

D'entrada hem pogut observar l'efecte de la llei de Moore, veient que la potència/velocitat de càlcul dels superordinadors ha progressat de forma exponencial durant els darrers 25 anys, seguint la predicció.

També hem vist que el factor que més contribueix a la potència de càlcul és el nombre de nuclis de computació.

Potser aquestes observacions no ens resultin gaire sorprenents, però l'exercici ens ha permès verificar que la història dels superordinadors segueix els paràmetres que se n'esperava.

7 Codi

Aquest anàlisi ha estat desenvolupat amb Emacs org-mode i ESS [1], i el codi R que es troba contingut en l'arxiu font de l'anàlisi s'ha extret automàticament i es pot consultar de forma independent a l'arxiu analisi.R.

References

[1] K. Hornik M. Maechler R.A. Sparapani S.J. Eglen S.P. Luque H. Redestig V. Spinu A.J. Rossini, R.M. Heiberger and L. Henry. Ess - emacs speaks statistics, 2016.

[2] D. G. Altman. *Practical Statistics for Medical Research*. Chapman Hall, 1991.

[3] Peter Dalgaard. *Introductory statistics with R*. Springer, 2002.

[4] Dan Lenski. TOP500 data for Wikipedia graphs. https://github.com/dlenski/top500, 2015.

[5] Gordon Moore. Progress in digital integrated electronics, 1975.

[6] Hadley Wickham and Garrett Grolemund. *R for Data Science*. O'Reilly, 2017.

[7] Wikipedia contributors. Moore's law — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Moore%27s_law&oldid=843169096, 2018. [Online; accessed 2-June-2018].