Modelagem de Carga de Trabalho

Rafael Tenfen

Data de entrega: 18/06/2021

Descrição da atividade

O objetivo do exercício é obter um modelo da carga de trabalho de uma determinada aplicação, a partir de um *trace* coletado durante um período de execução.

Descrição do *trace*

A aplicação de interesse processa requisições de escrita e leitura. Cada requisição transfere um determinado número de bytes. O arquivo `wl-rdwr.dat` contém o registro de 10.000 requisições processadas pela aplicação. Cada requisição é representada pelos seguintes campos:

- `tstamp`: o *timestamp* da requisição, em milissegundos (o *timestamp* 0 corresponde ao início da monitoração);
- `tipo`: o tipo da requisição, que pode ser leitura (`read`) ou escrita (`write`);
- `bytes`: o número de bytes lidos ou escritos.

Modelagem da carga de trabalho

A obtenção do modelo da carga de trabalho pode ser subdividida em duas etapas:

1. Modelagem das classes de requisições;
2. Modelagem dos tempos entre chegadas para cada classe.

```
reqs <- read.table("wl-rdwr.dat", head=T)


str(reqs)
```

```
## 'data.frame':    10000 obs. of  3 variables:
##  $ tstamp: num  3.23 6.17 12.97 27.78 31.65 ...
##  $ tipo  : Factor w/ 2 levels "read","write": 1 1 2 2 1 1 2 1 1 1 ...
##  $ nbytes: int  130 1172 224 304 127 540 272 940 1060 129 ...
```

```
head(reqs)
```

```
##     tstamp  tipo nbytes
## 1  3.22806  read    130
## 2  6.16755  read   1172
## 3 12.96876 write    224
## 4 27.77754 write    304
## 5 31.64822  read    127
## 6 33.24480  read    540
```

```
reqs.diff = diff(reqs$tstamp)
head(reqs.diff)
```

```
## [1]  2.93949  6.80121 14.80878  3.87068  1.59658  7.18747
```

```
str(reqs.diff)
```

```
##  num [1:9999] 2.94 6.8 14.81 3.87 1.6 ...
```

```
#reqs$tstamp[1]
reqs$texec = c(reqs$tstamp[1], reqs.diff)
head(reqs)
```

```
##     tstamp  tipo nbytes    texec
## 1  3.22806  read    130  3.22806
## 2  6.16755  read   1172  2.93949
## 3 12.96876 write    224  6.80121
## 4 27.77754 write    304 14.80878
## 5 31.64822  read    127  3.87068
## 6 33.24480  read    540  1.59658
```

```
tail(reqs)
```

```
##         tstamp tipo nbytes     texec
## 9995  71652.04 read    988   8.95539
## 9996  71655.33 read   1006   3.28926
## 9997  71733.98 read   1124  78.64909
## 9998  71795.01 read    922  61.02969
## 9999  71888.78 read    972  93.76428
## 10000 72009.72 read   1051 120.94287
```

```
#reqs.std <- scale(reqs[-1])

#reqs$diff = diff(reqs$tstamp)
#cbind(reqs, diff(reqs$tstamp))

reqs.split <- split(reqs, reqs$tipo)
str(reqs.split)
```

```
## List of 2
##  $ read :'data.frame':   7700 obs. of  4 variables:
##   ..$ tstamp: num [1:7700] 3.23 6.17 31.65 33.24 41.79 ...
```

```
##   ..$ tipo  : Factor w/ 2 levels "read","write": 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ nbytes: int [1:7700] 130 1172 127 540 940 1060 129 522 915 4234 ...
##   ..$ texec : num [1:7700] 3.23 2.94 3.87 1.6 1.35 ...
##  $ write:'data.frame':   2300 obs. of  4 variables:
##   ..$ tstamp: num [1:2300] 13 27.8 40.4 55.9 80.2 ...
##   ..$ tipo  : Factor w/ 2 levels "read","write": 2 2 2 2 2 2 2 2 2 2 ...
##   ..$ nbytes: int [1:2300] 224 304 272 279 2452 4280 274 2077 258 3984 ...
##   ..$ texec : num [1:2300] 6.8 14.81 7.19 8.03 2.61 ...
```

```
reqs.split$read$tipo = 1
reqs.split$read.std = scale(reqs.split$read[c('nbytes')])

reqs.split$write$tipo = 2
reqs.split$write.std = scale(reqs.split$write[c('nbytes')])

head(reqs.split$write)
```

```
##      tstamp tipo nbytes    texec
## 3  12.96876    2    224  6.80121
## 4  27.77754    2    304 14.80878
## 7  40.43227    2    272  7.18747
## 12 55.86225    2    279  8.02818
## 18 80.15284    2   2452  2.61057
## 21 98.98646    2   4280  0.78036
```

```
head(reqs.split$write.std)
```

```
##        nbytes
## 3  -0.9294396
## 4  -0.8767325
## 7  -0.8978154
## 12 -0.8932035
## 18  0.5384541
## 21  1.7428122
```

```
length(reqs.split$read)
```

```
## [1] 4
```

```
length(reqs.split$write)
```

```
## [1] 4
```

```
# Commented to avoid execute again
library(NbClust)

#reqs.split$read.nc = NbClust(reqs.split$read.std, min.nc=2, max.nc=7, method="kmeans")
#table(reqs.split$read.nc$Best.n[1,])

## ********************************************************************
```

```
## * Among all indices:
## * 1 proposed 2 as the best number of clusters
## * 5 proposed 4 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  4
##
##
## *******************************************************************
```

```
#reqs.split$write.nc = NbClust(reqs.split$write.std, min.nc=2, max.nc=7, method="kmeans")
#table(reqs.split$write.nc$Best.n[1,])
```

```
## *******************************************************************
## * Among all indices:
## * 5 proposed 3 as the best number of clusters
## * 1 proposed 7 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  3
##
##
## *******************************************************************
```

```
# K-means
set.seed(1130)

reqs.split$read.km <- kmeans(reqs.split$read.std, 4, nstart=25)

reqs.split$read.km$centers
```

```
##       nbytes
## 1 -0.6350665
## 2 -0.3191951
## 3  2.6250326
## 4  0.1005891
```

```
str(reqs.split$read)
```

```
## 'data.frame':    7700 obs. of  4 variables:
##  $ tstamp: num  3.23 6.17 31.65 33.24 41.79 ...
##  $ tipo  : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ nbytes: int  130 1172 127 540 940 1060 129 522 915 4234 ...
##  $ texec : num  3.23 2.94 3.87 1.6 1.35 ...
```

```
length(reqs.split$read$nbytes)
```

```
## [1] 7700
```

```
length(reqs.split$read.km$cluster)
```

```
## [1] 7700
```

```
head(reqs.split$read.km$cluster)
```

```
## 1 2 5 6 8 9
## 1 4 1 2 4 4
```

```
head(reqs.split$read)
```

```
##     tstamp tipo nbytes   texec
## 1  3.22806    1    130 3.22806
## 2  6.16755    1   1172 2.93949
## 5 31.64822    1    127 3.87068
## 6 33.24480    1    540 1.59658
## 8 41.78642    1    940 1.35415
## 9 44.68380    1   1060 2.89738
```

```
#reqs.split$read.km$cluster


str(reqs.split$read)
```

```
## 'data.frame':    7700 obs. of  4 variables:
##  $ tstamp: num  3.23 6.17 31.65 33.24 41.79 ...
##  $ tipo  : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ nbytes: int  130 1172 127 540 940 1060 129 522 915 4234 ...
##  $ texec : num  3.23 2.94 3.87 1.6 1.35 ...
```

```
head(reqs.split$read[c('nbytes', 'texec')])
```

```
##   nbytes   texec
## 1    130 3.22806
## 2   1172 2.93949
## 5    127 3.87068
## 6    540 1.59658
## 8    940 1.35415
## 9   1060 2.89738
```

```
head(reqs.split$read[3])
```

```
##   nbytes
## 1    130
## 2   1172
## 5    127
## 6    540
## 8    940
## 9   1060
```

```
head(reqs.split$read[4])
```

```
##     texec
## 1 3.22806
## 2 2.93949
## 5 3.87068
## 6 1.59658
## 8 1.35415
## 9 2.89738
```

```
(reqs.split$read.group.nbytes.mean <- aggregate(reqs.split$read[c('nbytes', 'texec')], by=list(cluster=
```

```
##   cluster    nbytes     texec
## 1       1  127.8100  6.210843
## 2       2  512.9415  6.338221
## 3       3 4102.7411  6.655949
## 4       4 1024.7706 11.254394
```

```
reqs.split$read.group.nbytes.sd <- aggregate(reqs.split$read[c('nbytes')], by=list(cluster=reqs.split$r
reqs.split$read.group.texec.sum <- aggregate(reqs.split$read[c('texec')], by=list(cluster=reqs.split$rea
table(reqs.split$read.km$cluster)
```

```
##
##    1    2    3    4
## 3000 2000  900 1800
```

```
reqs.split$write.km <- kmeans(reqs.split$write.std, 3, nstart=25)

(reqs.split$write.group.nbytes.mean <- aggregate(reqs.split$write[c('nbytes', 'texec')], by=list(cluste
```

```
##   cluster    nbytes    texec
## 1       1 4101.9580 6.218365
## 2       2 2038.5114 6.434570
## 3       3  256.2964 6.217368
```

```
reqs.split$write.group.nbytes.sd <- aggregate(reqs.split$write[c('nbytes')], by=list(cluster=reqs.split
reqs.split$write.group.texec.sum <- aggregate(reqs.split$write[c('texec')], by=list(cluster=reqs.split$
table(reqs.split$write.km$cluster)
```

```
##
##    1    2    3
##  500  700 1100
```

```
reqs.split$write.group.texec.sum$texec
```

```
## [1] 3109.183 4504.199 6839.104
```

```r
# Modelo da carga de trabalho
reqs.split$read.prop <- reqs.split$read.km$size/sum(reqs.split$read.km$size)
#sum()
(reqs.split$read.wlmod <- data.frame(

  texecpercentage=round(reqs.split$read.group.texec.sum$texec * 100 / sum(reqs.split$read.group.texec.su
  texec=round(reqs.split$read.group.nbytes.mean$texec, 2),
  nbytes=round(reqs.split$read.group.nbytes.mean$nbytes, 1),
  prop=reqs.split$read.prop)
)
```

```
##   texecpercentage texec nbytes      prop
## 1           32.37  6.21  127.8 0.3896104
## 2           22.02  6.34  512.9 0.2597403
## 3           10.41  6.66 4102.7 0.1168831
## 4           35.20 11.25 1024.8 0.2337662
```

```r
reqs.split$read.wlmod$prop = round(reqs.split$read.wlmod$prop * 100, 2)

head(reqs.split$read.wlmod)
```

```
##   texecpercentage texec nbytes  prop
## 1           32.37  6.21  127.8 38.96
## 2           22.02  6.34  512.9 25.97
## 3           10.41  6.66 4102.7 11.69
## 4           35.20 11.25 1024.8 23.38
```

```r
reqs.split$read.wlmod.print.i = data.frame(
  'Número de bytes Lidos'=reqs.split$read.wlmod$nbytes,
  "Frequência%"=reqs.split$read.wlmod$prop
)

reqs.split$read.wlmod.print.ii = data.frame(
  'Porcentagem do Tempo de Execução do Grupo'=reqs.split$read.wlmod$texecpercentage,
  'Média Tempo de Execução (ms)'=reqs.split$read.wlmod$texec,
  'Número de bytes Lidos'=reqs.split$read.wlmod$nbytes,
  "Frequência%"=reqs.split$read.wlmod$prop
)

reqs.split$write.prop <- reqs.split$write.km$size/sum(reqs.split$write.km$size)
(reqs.split$write.wlmod <- data.frame(
  texecpercentage=round(reqs.split$write.group.texec.sum$texec * 100 / sum(reqs.split$write.group.texec
  texec=round(reqs.split$write.group.nbytes.mean$texec, 2),
  nbytes=round(reqs.split$write.group.nbytes.mean$nbytes, 1),
  prop=reqs.split$write.prop)
)
```

```
##   texecpercentage texec nbytes      prop
## 1           21.51  6.22 4102.0 0.2173913
## 2           31.17  6.43 2038.5 0.3043478
## 3           47.32  6.22  256.3 0.4782609
```

```
reqs.split$write.wlmod$prop = round(reqs.split$write.wlmod$prop * 100, 2)
#reqs.split$write.wlmod
#print(reqs.split$write.wlmod, quote = FALSE, row.names = FALSE, column.names=c('tao', 'tao'))

reqs.split$write.wlmod.print.i = data.frame(
  'Número de bytes Escritos'=reqs.split$write.wlmod$nbytes,
  "Frequência%"=reqs.split$write.wlmod$prop
)

reqs.split$write.wlmod.print.ii = data.frame(
  'Porcentagem do Tempo de Execução do Grupo'=reqs.split$write.wlmod$texecpercentage,
  'Média Tempo de Execução (ms)'=reqs.split$write.wlmod$texec,
  'Número de Bytes Escritos'=reqs.split$write.wlmod$nbytes,
  "Frequência%"=reqs.split$write.wlmod$prop
)

reqs.general.km.size = c(reqs.split$read.km$size, reqs.split$write.km$size)
reqs.general.prop <- reqs.general.km.size/sum(reqs.general.km.size)

reqs.general.texec.sum.texec = c(reqs.split$read.group.texec.sum$texec, reqs.split$write.group.texec.su

reqs.group.nbytes.mean.nbytes = c(reqs.split$read.group.nbytes.mean$nbytes, reqs.split$write.group.nbyt
reqs.group.nbytes.sd.nbytes = c(reqs.split$read.group.nbytes.sd$nbytes, reqs.split$write.group.nbytes.s
reqs.group.tipo=c(rep("Leitura", length(reqs.split$read.group.nbytes.sd$nbytes)), rep("Escrita", length

(reqs.general.wlmod <- data.frame(
  texecpercentage=round(reqs.general.texec.sum.texec * 100 / sum(reqs.general.texec.sum.texec), 2),
  texec=round(reqs.general.texec.sum.texec, 2),
  nbytesmean=round(reqs.group.nbytes.mean.nbytes, 1),
  nbytessd=round(reqs.group.nbytes.sd.nbytes, 1),
  tipo=reqs.group.tipo,
  prop=reqs.general.prop)
)
```

```
##   texecpercentage    texec nbytesmean nbytessd    tipo prop
## 1           25.88 18632.53      127.8     13.0 Leitura 0.30
## 2           17.60 12676.44      512.9     40.2 Leitura 0.20
## 3            8.32  5990.35     4102.7    304.0 Leitura 0.09
## 4           28.13 20257.91     1024.8     75.9 Leitura 0.18
## 5            4.32  3109.18     4102.0    194.7 Escrita 0.05
## 6            6.25  4504.20     2038.5    203.6 Escrita 0.07
## 7            9.50  6839.10      256.3     25.0 Escrita 0.11
```

```
length(reqs.split$read.group.nbytes.sd$nbytes)
```

```
## [1] 4
```

```
reqs.general.wlmod$prop = round(reqs.general.wlmod$prop * 100, 2)

str(reqs.general.wlmod)
```

```
## 'data.frame':    7 obs. of  6 variables:
```

```
##  $ texecpercentage: num  25.88 17.6 8.32 28.13 4.32 ...
##  $ texec          : num  18633 12676 5990 20258 3109 ...
##  $ nbytesmean     : num  128 513 4103 1025 4102 ...
##  $ nbytessd       : num  13 40.2 304 75.9 194.7 ...
##  $ tipo           : Factor w/ 2 levels "Escrita","Leitura": 2 2 2 2 1 1 1
##  $ prop           : num  30 20 9 18 5 7 11
```

```
reqs.general.wlmod = reqs.general.wlmod[order(reqs.general.wlmod$prop),]

reqs.general.print.iii = data.frame(
  'Classe'=seq(7),
  'Frequência (%)'=reqs.general.wlmod$prop,
  'Tipo'=reqs.general.wlmod$tipo,
  'Número de Bytes Média'=reqs.general.wlmod$nbytesmean,
  'Número de Bytes Desvio'=reqs.general.wlmod$nbytessd,
  "Porcentagem do Tempo de Execução"=reqs.general.wlmod$texecpercentage
)


library(lemon)
#knit_print.data.frame <- lemon_print
```

```
reqs.split$read.wlmod.print.i
```

Table 1: Leitura

Número.de.bytes.Lidos	Frequência.
127.8	38.96
512.9	25.97
4102.7	11.69
1024.8	23.38

```
reqs.split$write.wlmod.print.i
```

Table 2: Escrita

Número.de.bytes.Escritos	Frequência.
4102.0	21.74
2038.5	30.43
256.3	47.83

```
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'

## The following objects are masked from 'package:lemon':
##
##     CoordCartesian, element_render
```

```
#str(reqs.split$read)

#clusters <- factor(reqs.split$read.km$cluster)
#ggplot(data=reqs.split$read, aes(x=nbytes, color=clusters, shape=clusters))
```

Etapa 1: Modelagem das classes de requisições

O primeiro passo é descobrir as classes de requisições processadas pela aplicação. Para isso, deve-se (i) obter as requisições representativas de cada classe usando *clustering* e (ii) descobrir a frequência relativa de cada classe.

Requisições de leitura e de escrita devem ficar em grupos separados. Para realizar o agrupamento das requisições, recomenda-se separar as requisições de leitura das requisições de escrita, e processar cada subconjunto individualmente. Supondo que sejam encontradas k_r classes de requisições de leitura e k_w classes de requisições de escrita, os números das classes de escrita $(1, 2, \ldots, k_w)$ podem ser mapeados em $k_r + 1, k_r + 2, \ldots, k_r + k_w$.

Uma vez encontradas as classes de requisições, é necessário associar cada requisição a sua respectiva classe e calcular a frequência relativa de cada classe, isto é, a porcentagem de requisições em cada uma.

Respostas

(i)

Obter as requisições representativas de cada classe usando clustering

- Os centroides são as requisiçoes representativas, porém não estão na escala original, então será apresentado a média dos grupos

```
round(reqs.split$read.group.nbytes.mean[c('cluster', 'nbytes')], 1)
```

Table 3: Leitura

cluster	nbytes
1	127.8
2	512.9
3	4102.7
4	1024.8

```
round(reqs.split$write.group.nbytes.mean[c('cluster', 'nbytes')], 1)
```

Table 4: Escrita

cluster	nbytes
1	4102.0
2	2038.5
3	256.3

(ii)

Descobrir a frequência relativa de cada classe Obs.: Frequência em %

```
reqs.split$read.wlmod.print.i
```

Table 5: Leitura

Número.de.bytes.Lidos	Frequência.
127.8	38.96
512.9	25.97
4102.7	11.69
1024.8	23.38

```
reqs.split$write.wlmod.print.i
```

Table 6: Escrita

Número.de.bytes.Escritos	Frequência.
4102.0	21.74
2038.5	30.43
256.3	47.83

Etapa 2: Modelagem dos tempos entre chegadas

A caracterização das classes de requisições é o primeiro passo para modelar a carga. O segundo passo consiste em encontrar a distribuição dos tempos entre chegadas das requisições em cada classe. Sabe-se que, para a aplicação de interesse, esses tempos podem ser distribuídos normalmente ou exponencialmente.

Neste ponto, ressalta-se que o arquivo de dados não contém tempos entre chegadas, mas *timestamps*. Os tempos entre chegadas τ_i são as diferenças entre os *timestamps* t_i e t_{i-1} de requisições consecutivas:

$$\tau_i = t_i - t_{i-1}$$

A função `diff()` do R pode ser útil para obter os tempos entre chegadas a partir dos *timestamps*.

Uma vez obtidos os tempos entre chegadas de uma classe de requisições, é possível ajustar uma distribuição de probabilidade para esse parâmetro.

Respostas

- Os tempos de chegada, acredito que seria como se fosse o tempo de execução de cada requisição, então adicionei a coluna ao data frame e em todo o processo
- Obs.: Frequência e Porcentagem do Tempo de Execução do Grupo em %
- Porcentagem do Tempo de Execução do Grupo: Seria a porcentagem do tempo de execução levado para executar aquele grupo, separado por leitura e escrita

```
reqs.split$read.wlmod.print.ii
```

Table 7: Leitura

Porcentagem.do.Tempo.de.Execução.do.Grupo	Média.Tempo.de.Execução..ms	Número.de.bytes.Lidos	Frequência.
32.37	6.21	127.8	38.96
22.02	6.34	512.9	25.97
10.41	6.66	4102.7	11.69
35.20	11.25	1024.8	23.38

```
reqs.split$write.wlmod.print.ii
```

Table 8: Escrita

Porcentagem.do.Tempo.de.Execução.do.Grupo	Média.Tempo.de.Execução..ms	Número.de.Bytes.Escritos	Frequência.
21.51	6.22	4102.0	21.74
31.17	6.43	2038.5	30.43
47.32	6.22	256.3	47.83

Etapa 3

Ao final do processo, será obtido um modelo de carga, que poderá ser resumido em uma tabela que combina as anteriores, conforme o exemplo abaixo (com dados fictícios):

classe	frequência (%)	tipo	no. bytes (média)	no. bytes (desvio)	distribuição de TEC
1	31.6	leitura	514.08	49.52	Exp(0.098)
2	45.9	leitura	2001.32	125.48	Exp(0.021)
3	22.5	escrita	1034.67	262.73	N(99.6, 18.3)

Respostas

- Os tempos de chegada, acredito que seria como se fosse o tempo de execução de cada requisição, então adicionei a coluna ao data frame e em todo o processo
- Obs.: Frequência e Porcentagem do Tempo de Execução em %

```
reqs.general.print.iii
```

Table 10: Geral Combinado

Classe	Frequência. . . .	Tipo	Número.de.Bytes.Média	Número.de.Bytes.Desvio	Porcentagem.do.Tempo.de.Execução
1	5	Escrita	4102.0	194.7	4.32
2	7	Escrita	2038.5	203.6	6.25
3	9	Leitura	4102.7	304.0	8.32
4	11	Escrita	256.3	25.0	9.50
5	18	Leitura	1024.8	75.9	28.13

Classe	Frequência. . . .	Tipo	Número.de.Bytes.Média	Número.de.Bytes.Desvio	Porcentagem.do.Tempo.de.Execução
6	20	Leitura	512.9	40.2	17.60
7	30	Leitura	127.8	13.0	25.88

Fim respostas

- Bom, é necessário fazer outro para todo o conjunto, para que as frequências aparecam de modo correto. # negative. . .

```
head(reqs)
```

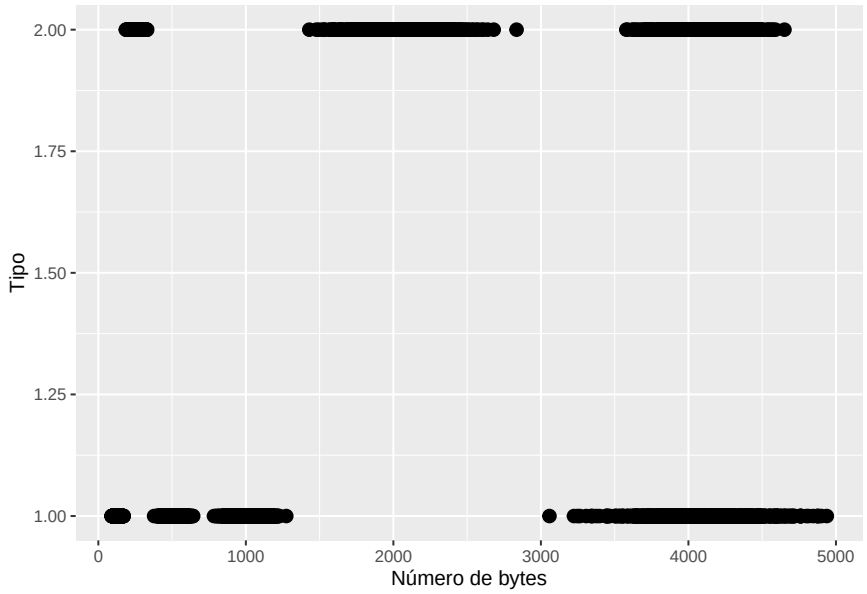
```
##     tstamp  tipo nbytes    texec
## 1  3.22806  read    130  3.22806
## 2  6.16755  read   1172  2.93949
## 3 12.96876 write    224  6.80121
## 4 27.77754 write    304 14.80878
## 5 31.64822  read    127  3.87068
## 6 33.24480  read    540  1.59658
```

```
reqs$tiponum = as.numeric(reqs$tipo)

reqs.std = scale(reqs[c('nbytes', 'tiponum', 'texec')])
#reqs.nc = NbClust(reqs.std, min.nc=2, max.nc=10, method="kmeans")
#table(reqs.nc$Best.n[1,])
## *******************************************************************
## * Among all indices:  only nbytes
## * 3 proposed 3 as the best number of clusters
## * 3 proposed 5 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  3
##
##
## *******************************************************************
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:  'nbytes', 'tiponum', 'texec'
## * 1 proposed 2 as the best number of clusters
## * 6 proposed 3 as the best number of clusters
## * 8 proposed 4 as the best number of clusters
## * 2 proposed 6 as the best number of clusters
## * 3 proposed 7 as the best number of clusters
## * 3 proposed 9 as the best number of clusters
##
##                    ***** Conclusion *****
##
```

```
## * According to the majority rule, the best number of clusters is  4
##
##
## ****************************************************************

# Notei que assim não conseguiria distinguir se era escrita ou leitura, então de fato devo combinas os
ggplot(
  data = reqs,
  aes(x=nbytes, y=tiponum)
) + geom_point( size = 3) + labs(
  x = "Número de bytes",
  y = "Tipo"
) + theme(legend.position = "none")
```




```
reqs.km <- kmeans(reqs.std, 3, nstart=25)

(reqs.group.nbytes.mean <- aggregate(reqs[c('nbytes')], by=list(cluster=reqs.km$cluster), mean))
```

```
##   cluster    nbytes
## 1       1 4102.4614
## 2       2  478.5147
## 3       3  949.3800
```

```
(reqs.group.nbytes.sd <- aggregate(reqs[c('nbytes')], by=list(cluster=reqs.km$cluster), sd))
```

```
##   cluster   nbytes
## 1       1 270.0215
## 2       2 368.3625
## 3       3 878.4988
```

```
(reqs.group.texec.sum <- aggregate(reqs[c('texec')], by=list(cluster=reqs.km$cluster), sum))
```

```
##   cluster     texec
## 1       1  9099.537
## 2       2 51566.879
## 3       3 11343.303
```

```
reqs.prop <- reqs.km$size/sum(reqs.km$size)

#(reqs.wlmod <- data.frame(
#    texecpercentage=round(reqs.group.texec.sum$texec * 100 / sum(reqs.group.texec.sum$texec), 2),
#    texec=round(reqs.group.nbytes.mean$texec, 2),
#    nbytesmean=round(reqs.group.nbytes.mean$nbytes, 1),
#    nbytessd=round(reqs.group.nbytes.sd$nbytes, 1),
#    prop=reqs.prop
#  )
#)
```