Projeto de Experimentos

Rafael Tenfen

Data de entrega: 11/06/2021

Descrição da atividade

O objetivo desta atividade é aplicar a técnica de projeto de experimentos. A atividade é dividida em duas partes:

1. Definição de projetos de experimentos
2. ANOVA fatorial

Algumas recomendações:

- Se você não estiver habituado com R Markdown, acostume-se a processar com frequência o documento, usando o botão **Knit**. Isso permitirá que eventuais erros no documento ou no código R sejam identificados rapidamente, pouco depois de terem sido cometidos, o que facilitará sua correção. Na verdade, é uma boa ideia você fazer isso **agora**, para garantir que seu ambiente esteja configurado corretamente. Se você receber uma mensagem de erro do tipo *Error in library(foo)*, isso significa que o pacote `foo` não está instalado. Para instalar um pacote, execute o comando `install.packages("foo")` no Console, ou clique em *Tools -> Install Packages*.
- Após concluir a atividade, você deverá submeter no Moodle um arquivo ZIP contendo:
  - o arquivo fonte .Rmd;
  - a saída processada (PDF ou HTML) do arquivo .Rmd;
  - arquivos necessários para o processamento do .Rmd.

Uma forma simples de enviar uma submissão correta é exportar o seu projeto no RStudio Cloud e submeter o arquivo ZIP gerado.

Configuração

Nesta atividade, nenhuma configuração é necessária por padrão, mas você pode usar o bloco abaixo para incluir o que julgar necessário.

```
# insira seus comandos de configuração aqui
```

Parte 1: Definição de Projetos de Experimentos

Nesta parte você irá considerar como aplicar o projeto de experimentos na análise de desempenho de um sistema ou aplicação de seu interesse.

1. Descreva brevemente o sistema.
2. Identifique uma ou mais métricas de desempenho que podem ser usadas como variável de resposta. O tempo de execução/resposta pode ser uma das métricas, mas não a única.
3. Identifique pelo menos 3 fatores que podem influenciar o desempenho do sistema. Para cada fator, descreva os níveis que você adotaria. Justifique brevemente a escolha dos fatores e dos níveis; não precisa discorrer sobre o assunto, apenas explicar porque você supõe que os fatores podem influenciar no desempenho e porque os níveis escolhidos são razoáveis.
4. Calcule o número de experimentos de um projeto fatorial completo usando os fatores e níveis descritos no item 3. Não se esqueça de considerar as replicações (defina quantas replicações seriam realizadas).
5. Partindo dos fatores e níveis definidos no item 3, monte um projeto 2^k para esse experimento, especificando quais fatores e níveis seriam usados no projeto. Calcule o número de experimentos do projeto, considerando o mesmo número de replicações do item 4.
6. Suponha que a análise dos resultados do projeto 2^k definido no item 5 revele que o fator com mais níveis proposto no item 3 não é estatisticamente significativo. Calcule quantos experimentos adicionais precisarão ser realizados para um projeto fatorial completo com os fatores significativos (considere replicações).

Respostas aqui

1

- Uma aplicação em C utilizando OpenMp, para verificação de números perfeitos.
- Números perfeitos são números inteiros positivos em que a soma de seus divisores positivos excluindo o próprio número, é igual ao seu valor. Exemplos de Números Perfeitos:
  - O número 6 é um número perfeito porque a soma de seus divisores são iguais ao número seis. (3 + 2 + 1) = 6.
  - 28: (14 + 7 + 4 + 2 + 1) = 28
  - 496
  - 8128
  - 33550336
  - 8589869056
  - 137438691328
  - 2305843008139952128
- Para verificar se o número é perfeito, foi utilizado "Brute force": Dividir números antecessores um a um para verificar seus possíveis divisores, realizar a soma desses divisores e verificar se o resultado da soma é igual ao número, definindo assim o número como perfeito.
- Hardware utilizado nos testes ens5 8 CPU's Intel(R) Xeon(R) CPU @ 3.50GHz 48 GB Memory

Open MP A execução com OpenMP utilizou os seguintes alcances de números para verificação de números perfeitos: 0-100000, 0-300000 e 0-600000. Além das seguintes quantidades de threads: 1, 2, 4, 6 e 8, para cada alcance, e por fim foram testados os seguintes schedules: static, dynamic e guided para cada alcance e número de threads definidas. Todas as execuções foram testadas na mesma maquina: ens5.

MPI A execução com o MPI utilizou os seguintes alcances de números para verificação de números perfeitos: 0-100000, 0-300000 e 0-600000. Além das seguintes quantidades de threads: 1, 2, 4, 8. Todas as execuções foram testadas na mesma maquina: ens5.

2

- “seconds”: Tempo de execução
- “memory”: Quantidade de memoria utilizada em kb

3

- “limit”: Alcance dos números a serem verificados se são perfeitos, como 100000, o sistema irá pesquisar todos os números de 1 até 100000 a fim de encontrar os números perfeitos nesse intervalo [100000, 300000, 600000]
- “threads”: Quantidade de threads utilizadas para realizar o paralelismo [1, 2, 4, 6, 8]
- “schedule”: Modo de paralelismo utilizado [“static”, “dynamic”, “guided”, “auto_open_mp”, “auto_mpi”]

4

- (“limit”) 3 x (“threads”) 5 x (“schedule”) 5 = 75 experimentos sem replicação, com 5 replicações teríamos 375 experimentos

5

- “limit”: [300000, 600000]
- “threads”: [2, 8]
- “schedule”: [“auto_open_mp”, “auto_mpi”]
- 8 experimentos sem replicaçao, com 5 replicaçoes teriamos 40 experimentos

6

- Tanto as threads quanto o schedule possuem 5 níveis, removendo apenas um deles, ficaria com (“limit”) 3 x (“threads” ou “schedule”) 5 = 15 experimentos sem replicação, com 5 replicações teríamos 75 experimentos, descontando 2ˆ2=4 experimentos já realizados no item 5, ficam 11 experimentos sem replicação, e 55 experimentos com 5 replicações

Parte 2: ANOVA Fatorial

Nesta parte da atividade você irá usar ANOVA fatorial para analisar os resultados de um conjunto de experimentos. Esses experimentos tinham por objetivo avaliar o impacto de processador e memória no tempo de execução de dois *benchmarks* A e B. Os fatores e níveis considerados foram:

fator	níveis
CPU	AMD, Intel
no. de núcleos	4, 8
memória	2 GB, 4 GB, 8 GB

Foi usado um projeto de experimentos fatorial completo, com 5 replicações de cada experimento. Os dados coletados estão contidos no arquivo `doe-benchmarks.dat`, que contém as seguintes colunas:

- `benchmark`: qual o *benchmark* (A ou B);
- `cpu`: tipo de CPU;
- `cores`: quantidade de núcleos;
- `mem`: quantidade de memória;
- `tempo`: tempo de execução do *benchmark*.

Com base nesses dados, analise separadamente cada *benchmark*, seguindo os passos abaixo:

1. Analise graficamente como cada fator influencia o tempo de exemplo de execução.
2. Faça uma ANOVA fatorial, e identifique os fatores e interações estatisticamente significativos.
3. Faça a alocação de variação, descrevendo qual a porcentagem de variação explicada pelos fatores e interações significativos e pelos resíduos.
4. Use o teste de Tukey para verificar quais são as diferenças significativas entre os níveis dos fatores/interações significativos.
5. Faça o diagnóstico dos resíduos da ANOVA.
6. Verifique a premissa de homogeneidade de variâncias.
7. Descreva como os fatores significativos influenciam o tempo de execução. Leve em consideração as interações, se houver. (Sugestão: use gráficos de interação para visualização.)

ATENÇÂO: Os passos 1 a 7 deverão ser realizados duas vezes, uma para o *benchmark* A e outra para o *benchmark* B.

```
benchmarks <- read.table("doe-benchmarks.dat", head=T)
str(benchmarks)
```

```
## 'data.frame':    120 obs. of  5 variables:
##  $ benchmark: Factor w/ 2 levels "A","B": 1 1 1 1 1 1 1 1 1 1 ...
##  $ cpu      : Factor w/ 2 levels "AMD","Intel": 1 2 1 2 1 2 1 2 1 2 ...
##  $ cores    : int  4 4 4 4 4 4 4 4 4 4 ...
##  $ mem      : Factor w/ 3 levels "2 GB","4 GB",..: 1 1 2 2 3 3 1 1 2 2 ...
##  $ tempo    : num  70.2 64.6 50.5 44.9 39.8 ...
```

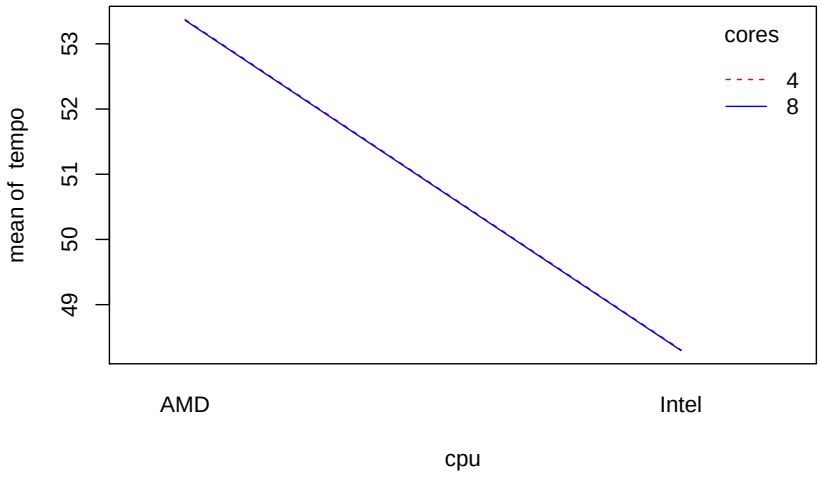
```
benchmarks.split <- split(benchmarks, factor(benchmarks$benchmark))
benchmarks.split$A$cores = as.factor(benchmarks.split$A$cores)
benchmarks.split$B$cores = as.factor(benchmarks.split$B$cores)
str(benchmarks.split$A)
```

```
## 'data.frame':    60 obs. of  5 variables:
##  $ benchmark: Factor w/ 2 levels "A","B": 1 1 1 1 1 1 1 1 1 1 ...
##  $ cpu      : Factor w/ 2 levels "AMD","Intel": 1 2 1 2 1 2 1 2 1 2 ...
##  $ cores    : Factor w/ 2 levels "4","8": 1 1 1 1 1 1 1 1 1 1 ...
##  $ mem      : Factor w/ 3 levels "2 GB","4 GB",..: 1 1 2 2 3 3 1 1 2 2 ...
##  $ tempo    : num  70.2 64.6 50.5 44.9 39.8 ...
```

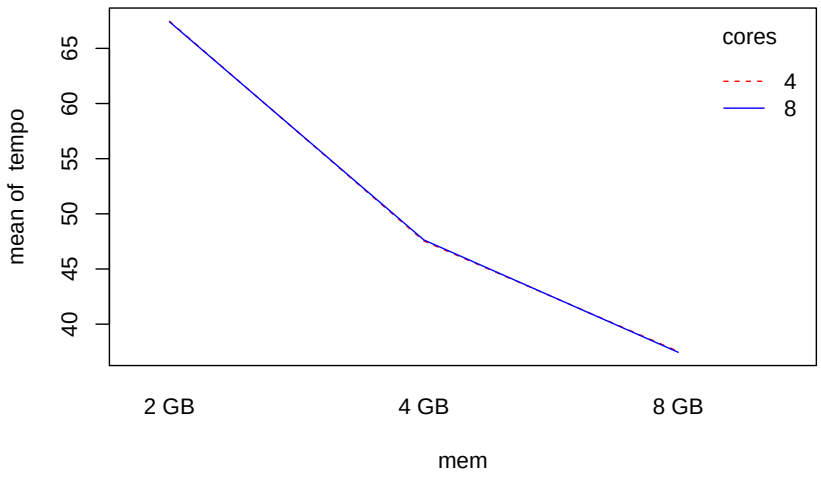
```
str(benchmarks.split$B)
```

```
## 'data.frame':    60 obs. of  5 variables:
##  $ benchmark: Factor w/ 2 levels "A","B": 2 2 2 2 2 2 2 2 2 2 ...
##  $ cpu      : Factor w/ 2 levels "AMD","Intel": 1 2 1 2 1 2 1 2 1 2 ...
##  $ cores    : Factor w/ 2 levels "4","8": 1 1 1 1 1 1 1 1 1 1 ...
##  $ mem      : Factor w/ 3 levels "2 GB","4 GB",..: 1 1 2 2 3 3 1 1 2 2 ...
##  $ tempo    : num  31 35.5 24.4 25.7 24.6 ...
```

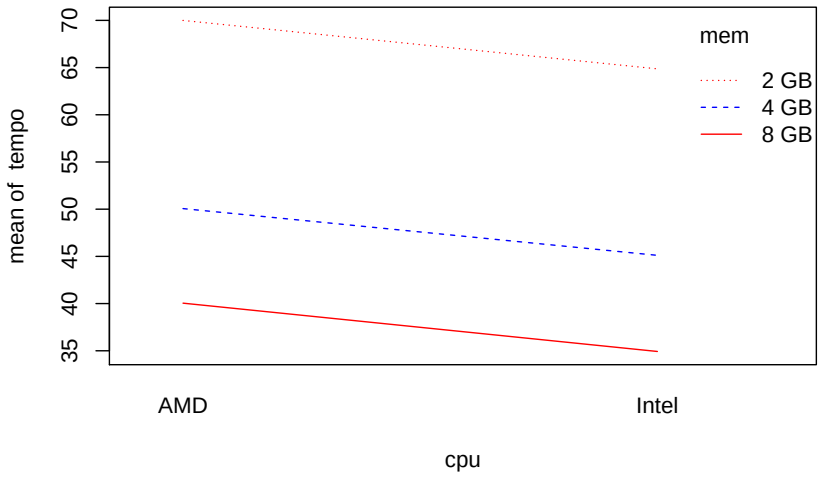
```
with(benchmarks.split$A, interaction.plot(cpu, cores, tempo, col = c("red", "blue")))
```




```
with(benchmarks.split$A, interaction.plot(mem, cores, tempo, col = c("red", "blue")))
```




```
with(benchmarks.split$A, interaction.plot(cpu, mem, tempo, col = c("red", "blue")))
```



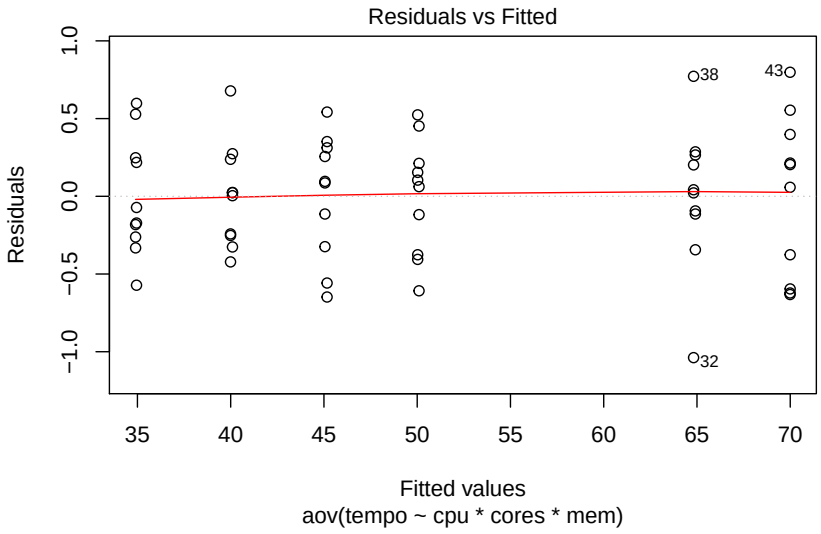

```
(benchmarks.split$A.aov = aov(tempo ~ cpu * cores * mem, benchmarks.split$A))
```

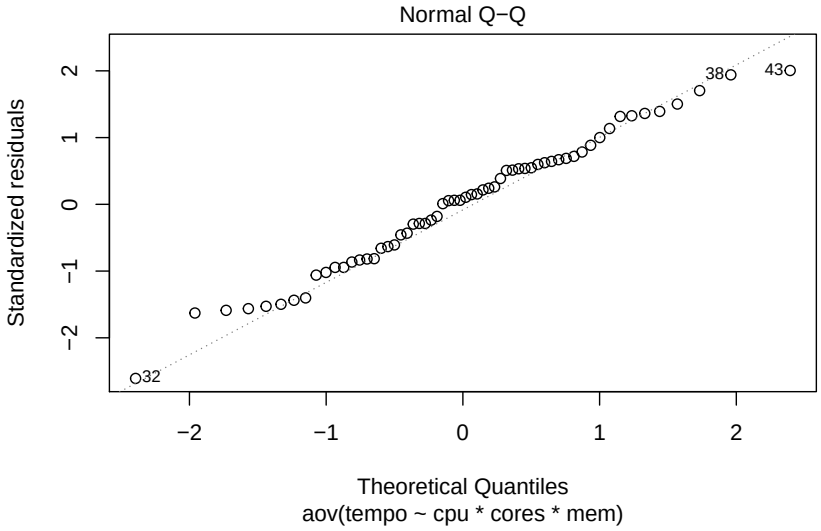
```
## Call:
##    aov(formula = tempo ~ cpu * cores * mem, data = benchmarks.split$A)
##
## Terms:
##                       cpu    cores      mem cpu:cores  cpu:mem cores:mem
## Sum of Squares   384.864    0.002 9283.118     0.000    0.098     0.083
## Deg. of Freedom        1        1        2         1        2         2
##                  cpu:cores:mem Residuals
## Sum of Squares           0.016     9.502
## Deg. of Freedom              2        48
##
## Residual standard error: 0.4449195
## Estimated effects may be unbalanced
```

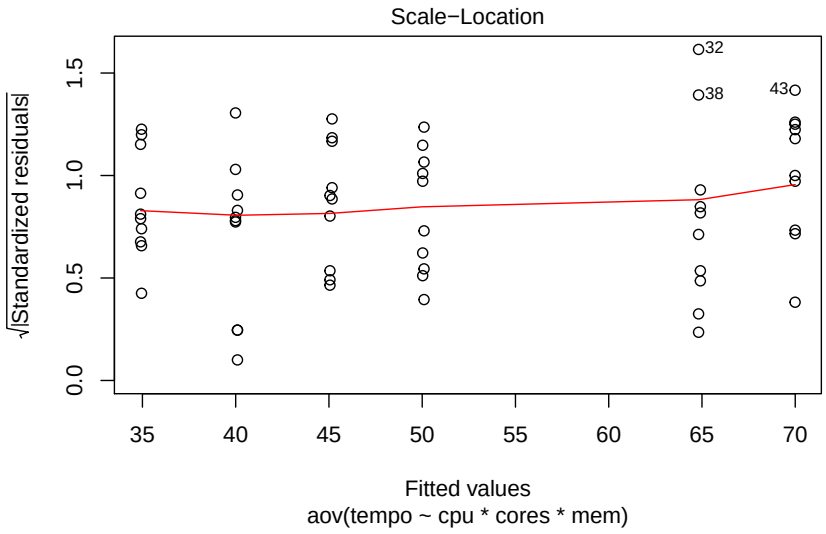
```
summary(benchmarks.split$A.aov)
```

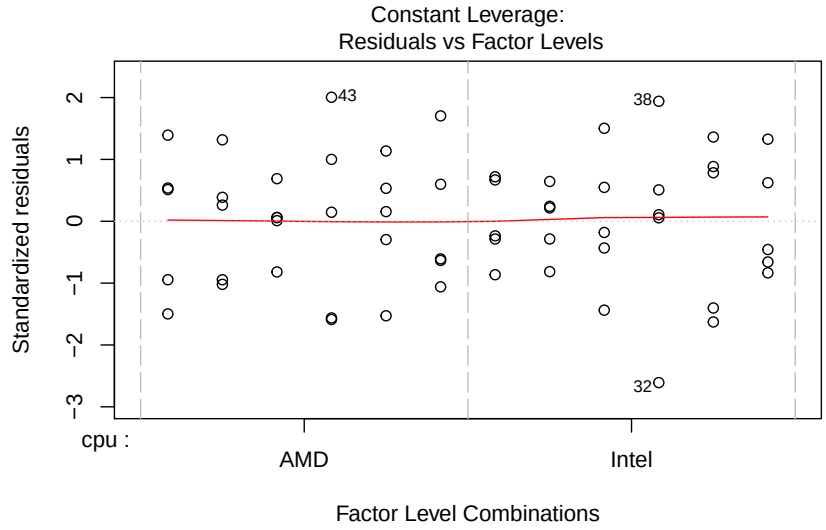
```
##               Df Sum Sq Mean Sq   F value Pr(>F)
## cpu            1    385     385  1944.216 <2e-16 ***
## cores          1      0       0     0.010  0.922
## mem            2   9283    4642 23447.745 <2e-16 ***
## cpu:cores      1      0       0     0.000  0.995
## cpu:mem        2      0       0     0.248  0.781
## cores:mem      2      0       0     0.211  0.811
## cpu:cores:mem  2      0       0     0.041  0.959
## Residuals     48     10       0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
plot(benchmarks.split$A.aov)
```


Residuals vs Fitted
Residuals
1.0
0.5
0.0
−0.5
−1.0
38
43
32
35
40
45
50
55
60
65
70
Fitted values
aov(tempo ~ cpu * cores * mem)


Normal Q−Q
Standardized residuals
2
1
0
−1
−2
38
43
32
−2
−1
0
1
2
Theoretical Quantiles
aov(tempo ~ cpu * cores * mem)






```
(SS = anova(benchmarks.split$A.aov)['Sum Sq'])
```

```
##                Sum Sq
## cpu             384.9
## cores             0.0
## mem            9283.1
## cpu:cores         0.0
## cpu:mem           0.1
## cores:mem         0.1
## cpu:cores:mem     0.0
## Residuals         9.5
```

```r
SST = sum(SS)
(benchmarks.split$A.pv = round(100*SS/SST, 2))
```

```
##                Sum Sq
## cpu              3.98
## cores            0.00
## mem             95.92
## cpu:cores        0.00
## cpu:mem          0.00
## cores:mem        0.00
## cpu:cores:mem    0.00
## Residuals        0.10
```

```r
(benchmarks.split$A.flig = fligner.test(tempo ~ interaction(cpu, cores, mem), benchmarks.split$A))
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  tempo by interaction(cpu, cores, mem)
## Fligner-Killeen:med chi-squared = 6.5785, df = 11, p-value = 0.8321
```

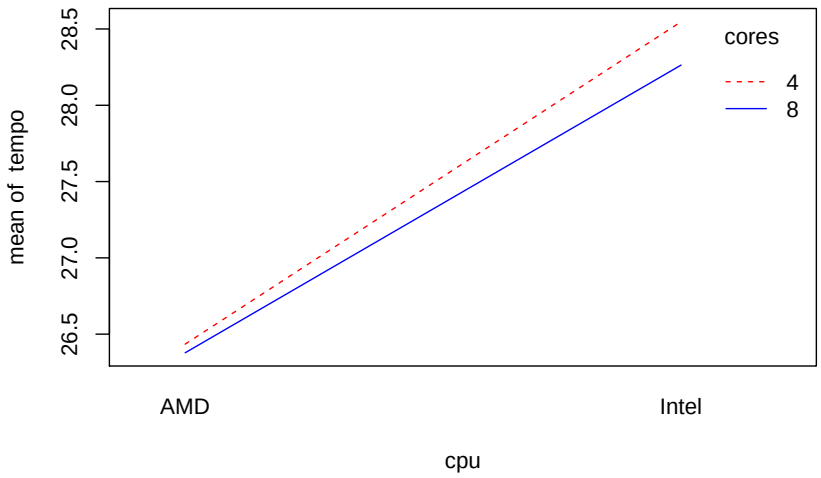
```r
# p-value > alfa (assumindo 95% de nivel de confiança), então é válido


TukeyHSD(benchmarks.split$A.aov)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = tempo ~ cpu * cores * mem, data = benchmarks.split$A)
##
## $cpu
##                diff      lwr       upr p adj
## Intel-AMD -5.065333 -5.29631 -4.834356     0
##
## $cores
##            diff        lwr       upr     p adj
## 8-4 -0.01133333 -0.2423105 0.2196438 0.9218225
##
## $mem
##               diff       lwr        upr p adj
## 4 GB-2 GB -19.8435 -20.18377 -19.503229     0
## 8 GB-2 GB -29.9445 -30.28477 -29.604229     0
## 8 GB-4 GB -10.1010 -10.44127  -9.760729     0
##
## $`cpu:cores`
##                         diff        lwr        upr     p adj
## Intel:4-AMD:4   -5.06600000 -5.4983709 -4.6336291 0.0000000
## AMD:8-AMD:4     -0.01200000 -0.4443709  0.4203709 0.9998533
## Intel:8-AMD:4   -5.07666667 -5.5090375 -4.6442958 0.0000000
## AMD:8-Intel:4    5.05400000  4.6216291  5.4863709 0.0000000
```

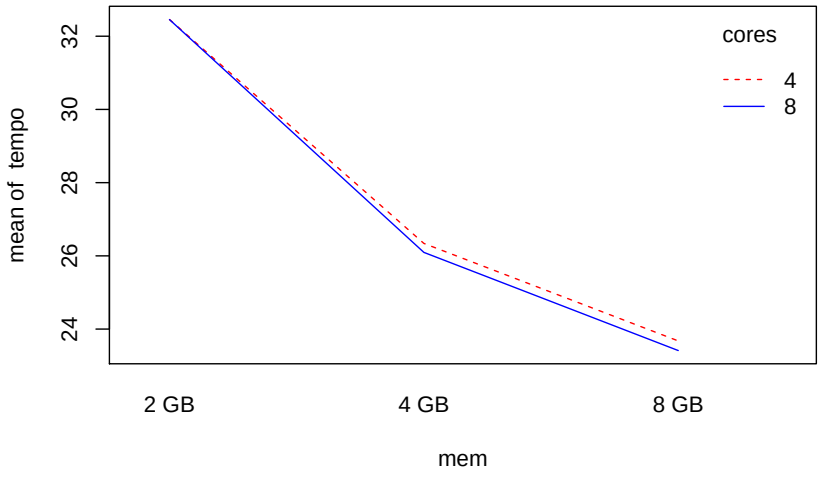
```
## Intel:8-Intel:4 -0.01066667 -0.4430375  0.4217042 0.9998969
## Intel:8-AMD:8   -5.06466667 -5.4970375 -4.6322958 0.0000000
##
## $`cpu:mem`
##                            diff        lwr        upr p adj
## Intel:2 GB-AMD:2 GB    -5.128  -5.718534  -4.537466     0
## AMD:4 GB-AMD:2 GB     -19.932 -20.522534 -19.341466     0
## Intel:4 GB-AMD:2 GB   -24.883 -25.473534 -24.292466     0
## AMD:8 GB-AMD:2 GB     -29.950 -30.540534 -29.359466     0
## Intel:8 GB-AMD:2 GB   -35.067 -35.657534 -34.476466     0
## AMD:4 GB-Intel:2 GB   -14.804 -15.394534 -14.213466     0
## Intel:4 GB-Intel:2 GB -19.755 -20.345534 -19.164466     0
## AMD:8 GB-Intel:2 GB   -24.822 -25.412534 -24.231466     0
## Intel:8 GB-Intel:2 GB -29.939 -30.529534 -29.348466     0
## Intel:4 GB-AMD:4 GB    -4.951  -5.541534  -4.360466     0
## AMD:8 GB-AMD:4 GB     -10.018 -10.608534  -9.427466     0
## Intel:8 GB-AMD:4 GB   -15.135 -15.725534 -14.544466     0
## AMD:8 GB-Intel:4 GB    -5.067  -5.657534  -4.476466     0
## Intel:8 GB-Intel:4 GB -10.184 -10.774534  -9.593466     0
## Intel:8 GB-AMD:8 GB    -5.117  -5.707534  -4.526466     0
##
## $`cores:mem`
##                   diff        lwr        upr     p adj
## 8:2 GB-4:2 GB  -0.050  -0.640534   0.540534 0.9998543
## 4:4 GB-4:2 GB -19.915 -20.505534 -19.324466 0.0000000
## 8:4 GB-4:2 GB -19.822 -20.412534 -19.231466 0.0000000
## 4:8 GB-4:2 GB -29.931 -30.521534 -29.340466 0.0000000
## 8:8 GB-4:2 GB -30.008 -30.598534 -29.417466 0.0000000
## 4:4 GB-8:2 GB -19.865 -20.455534 -19.274466 0.0000000
## 8:4 GB-8:2 GB -19.772 -20.362534 -19.181466 0.0000000
## 4:8 GB-8:2 GB -29.881 -30.471534 -29.290466 0.0000000
## 8:8 GB-8:2 GB -29.958 -30.548534 -29.367466 0.0000000
## 8:4 GB-4:4 GB   0.093  -0.497534   0.683534 0.9970481
## 4:8 GB-4:4 GB -10.016 -10.606534  -9.425466 0.0000000
## 8:8 GB-4:4 GB -10.093 -10.683534  -9.502466 0.0000000
## 4:8 GB-8:4 GB -10.109 -10.699534  -9.518466 0.0000000
## 8:8 GB-8:4 GB -10.186 -10.776534  -9.595466 0.0000000
## 8:8 GB-4:8 GB  -0.077  -0.667534   0.513534 0.9988030
##
## $`cpu:cores:mem`
##                                diff         lwr         upr     p adj
## Intel:4:2 GB-AMD:4:2 GB    -5.082  -6.0482238  -4.1157762 0.0000000
## AMD:8:2 GB-AMD:4:2 GB      -0.004  -0.9702238   0.9622238 1.0000000
## Intel:8:2 GB-AMD:4:2 GB    -5.178  -6.1442238  -4.2117762 0.0000000
## AMD:4:4 GB-AMD:4:2 GB     -19.970 -20.9362238 -19.0037762 0.0000000
## Intel:4:4 GB-AMD:4:2 GB   -24.942 -25.9082238 -23.9757762 0.0000000
## AMD:8:4 GB-AMD:4:2 GB     -19.898 -20.8642238 -18.9317762 0.0000000
## Intel:8:4 GB-AMD:4:2 GB   -24.828 -25.7942238 -23.8617762 0.0000000
## AMD:4:8 GB-AMD:4:2 GB     -29.900 -30.8662238 -28.9337762 0.0000000
## Intel:4:8 GB-AMD:4:2 GB   -35.044 -36.0102238 -34.0777762 0.0000000
## AMD:8:8 GB-AMD:4:2 GB     -30.004 -30.9702238 -29.0377762 0.0000000
## Intel:8:8 GB-AMD:4:2 GB   -35.094 -36.0602238 -34.1277762 0.0000000
## AMD:8:2 GB-Intel:4:2 GB     5.078   4.1117762   6.0442238 0.0000000
## Intel:8:2 GB-Intel:4:2 GB  -0.096  -1.0622238   0.8702238 0.9999999
```

```
## AMD:4:4 GB-Intel:4:2 GB   -14.888 -15.8542238 -13.9217762 0.0000000
## Intel:4:4 GB-Intel:4:2 GB -19.860 -20.8262238 -18.8937762 0.0000000
## AMD:8:4 GB-Intel:4:2 GB   -14.816 -15.7822238 -13.8497762 0.0000000
## Intel:8:4 GB-Intel:4:2 GB -19.746 -20.7122238 -18.7797762 0.0000000
## AMD:4:8 GB-Intel:4:2 GB   -24.818 -25.7842238 -23.8517762 0.0000000
## Intel:4:8 GB-Intel:4:2 GB -29.962 -30.9282238 -28.9957762 0.0000000
## AMD:8:8 GB-Intel:4:2 GB   -24.922 -25.8882238 -23.9557762 0.0000000
## Intel:8:8 GB-Intel:4:2 GB -30.012 -30.9782238 -29.0457762 0.0000000
## Intel:8:2 GB-AMD:8:2 GB    -5.174  -6.1402238  -4.2077762 0.0000000
## AMD:4:4 GB-AMD:8:2 GB     -19.966 -20.9322238 -18.9997762 0.0000000
## Intel:4:4 GB-AMD:8:2 GB   -24.938 -25.9042238 -23.9717762 0.0000000
## AMD:8:4 GB-AMD:8:2 GB     -19.894 -20.8602238 -18.9277762 0.0000000
## Intel:8:4 GB-AMD:8:2 GB   -24.824 -25.7902238 -23.8577762 0.0000000
## AMD:4:8 GB-AMD:8:2 GB     -29.896 -30.8622238 -28.9297762 0.0000000
## Intel:4:8 GB-AMD:8:2 GB   -35.040 -36.0062238 -34.0737762 0.0000000
## AMD:8:8 GB-AMD:8:2 GB     -30.000 -30.9662238 -29.0337762 0.0000000
## Intel:8:8 GB-AMD:8:2 GB   -35.090 -36.0562238 -34.1237762 0.0000000
## AMD:4:4 GB-Intel:8:2 GB   -14.792 -15.7582238 -13.8257762 0.0000000
## Intel:4:4 GB-Intel:8:2 GB -19.764 -20.7302238 -18.7977762 0.0000000
## AMD:8:4 GB-Intel:8:2 GB   -14.720 -15.6862238 -13.7537762 0.0000000
## Intel:8:4 GB-Intel:8:2 GB -19.650 -20.6162238 -18.6837762 0.0000000
## AMD:4:8 GB-Intel:8:2 GB   -24.722 -25.6882238 -23.7557762 0.0000000
## Intel:4:8 GB-Intel:8:2 GB -29.866 -30.8322238 -28.8997762 0.0000000
## AMD:8:8 GB-Intel:8:2 GB   -24.826 -25.7922238 -23.8597762 0.0000000
## Intel:8:8 GB-Intel:8:2 GB -29.916 -30.8822238 -28.9497762 0.0000000
## Intel:4:4 GB-AMD:4:4 GB    -4.972  -5.9382238  -4.0057762 0.0000000
## AMD:8:4 GB-AMD:4:4 GB       0.072  -0.8942238   1.0382238 1.0000000
## Intel:8:4 GB-AMD:4:4 GB    -4.858  -5.8242238  -3.8917762 0.0000000
## AMD:4:8 GB-AMD:4:4 GB      -9.930 -10.8962238  -8.9637762 0.0000000
## Intel:4:8 GB-AMD:4:4 GB   -15.074 -16.0402238 -14.1077762 0.0000000
## AMD:8:8 GB-AMD:4:4 GB     -10.034 -11.0002238  -9.0677762 0.0000000
## Intel:8:8 GB-AMD:4:4 GB   -15.124 -16.0902238 -14.1577762 0.0000000
## AMD:8:4 GB-Intel:4:4 GB     5.044   4.0777762   6.0102238 0.0000000
## Intel:8:4 GB-Intel:4:4 GB   0.114  -0.8522238   1.0802238 0.9999996
## AMD:4:8 GB-Intel:4:4 GB    -4.958  -5.9242238  -3.9917762 0.0000000
## Intel:4:8 GB-Intel:4:4 GB -10.102 -11.0682238  -9.1357762 0.0000000
## AMD:8:8 GB-Intel:4:4 GB    -5.062  -6.0282238  -4.0957762 0.0000000
## Intel:8:8 GB-Intel:4:4 GB -10.152 -11.1182238  -9.1857762 0.0000000
## Intel:8:4 GB-AMD:8:4 GB    -4.930  -5.8962238  -3.9637762 0.0000000
## AMD:4:8 GB-AMD:8:4 GB     -10.002 -10.9682238  -9.0357762 0.0000000
## Intel:4:8 GB-AMD:8:4 GB   -15.146 -16.1122238 -14.1797762 0.0000000
## AMD:8:8 GB-AMD:8:4 GB     -10.106 -11.0722238  -9.1397762 0.0000000
## Intel:8:8 GB-AMD:8:4 GB   -15.196 -16.1622238 -14.2297762 0.0000000
## AMD:4:8 GB-Intel:8:4 GB    -5.072  -6.0382238  -4.1057762 0.0000000
## Intel:4:8 GB-Intel:8:4 GB -10.216 -11.1822238  -9.2497762 0.0000000
## AMD:8:8 GB-Intel:8:4 GB    -5.176  -6.1422238  -4.2097762 0.0000000
## Intel:8:8 GB-Intel:8:4 GB -10.266 -11.2322238  -9.2997762 0.0000000
## Intel:4:8 GB-AMD:4:8 GB    -5.144  -6.1102238  -4.1777762 0.0000000
## AMD:8:8 GB-AMD:4:8 GB      -0.104  -1.0702238   0.8622238 0.9999998
## Intel:8:8 GB-AMD:4:8 GB    -5.194  -6.1602238  -4.2277762 0.0000000
## AMD:8:8 GB-Intel:4:8 GB     5.040   4.0737762   6.0062238 0.0000000
## Intel:8:8 GB-Intel:4:8 GB  -0.050  -1.0162238   0.9162238 1.0000000
## Intel:8:8 GB-AMD:8:8 GB    -5.090  -6.0562238  -4.1237762 0.0000000
```

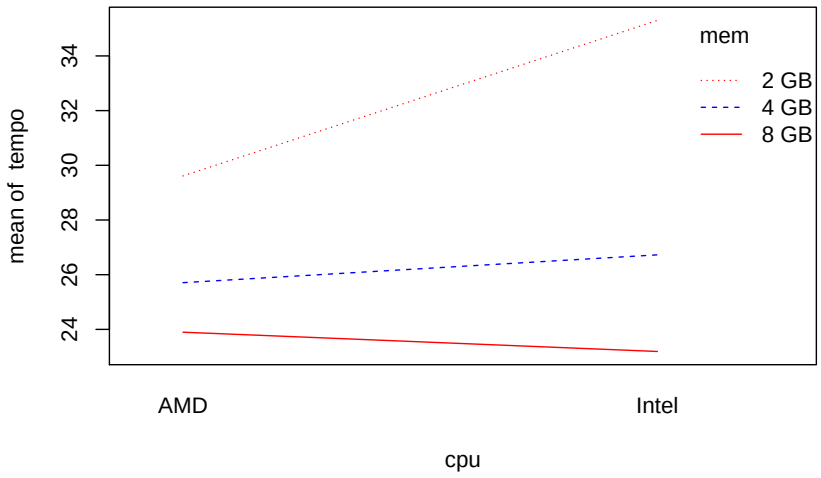
```
# TESTE BENCHMARK B

with(benchmarks.split$B, interaction.plot(cpu, cores, tempo, col = c("red", "blue")))
```




```
with(benchmarks.split$B, interaction.plot(mem, cores, tempo, col = c("red", "blue")))
```




```
with(benchmarks.split$B, interaction.plot(cpu, mem, tempo, col = c("red", "blue")))
```



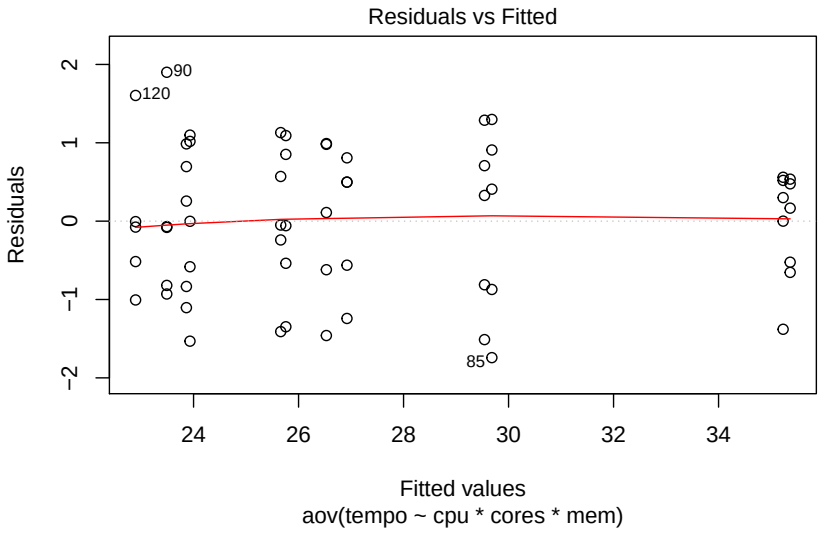

```
(benchmarks.split$B.aov = aov(tempo ~ cpu * cores * mem, benchmarks.split$B))
```

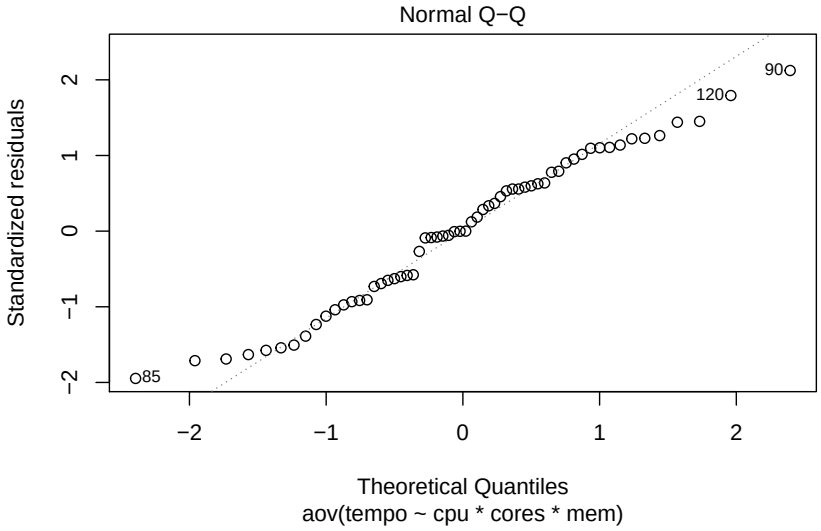
```
## Call:
##    aov(formula = tempo ~ cpu * cores * mem, data = benchmarks.split$B)
##
## Terms:
##                       cpu    cores      mem cpu:cores  cpu:mem cores:mem
## Sum of Squares   59.9400   0.4352 836.0669    0.1938 109.3127    0.2108
## Deg. of Freedom        1        1        2         1        2         2
##                 cpu:cores:mem Residuals
## Sum of Squares         0.5559   48.0400
## Deg. of Freedom             2        48
##
## Residual standard error: 1.000416
## Estimated effects may be unbalanced
```

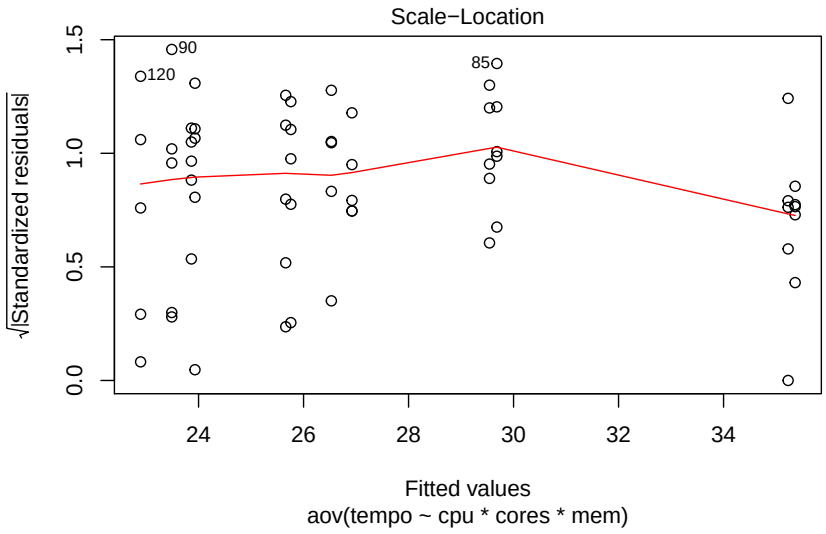
```
summary(benchmarks.split$B.aov)
```

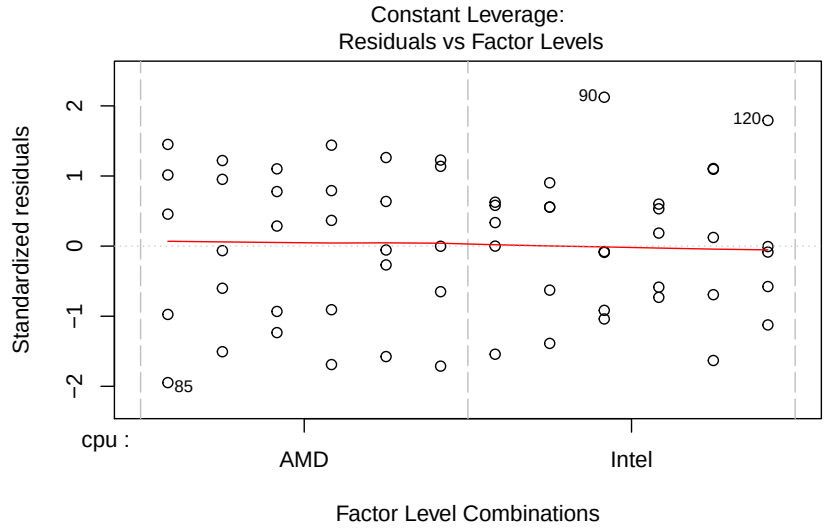
```
##               Df Sum Sq Mean Sq F value   Pr(>F)
## cpu            1   59.9    59.9  59.890 5.47e-10 ***
## cores          1    0.4     0.4   0.435    0.513
## mem            2  836.1   418.0 417.686  < 2e-16 ***
## cpu:cores      1    0.2     0.2   0.194    0.662
## cpu:mem        2  109.3    54.7  54.611 4.30e-13 ***
## cores:mem      2    0.2     0.1   0.105    0.900
## cpu:cores:mem  2    0.6     0.3   0.278    0.759
## Residuals     48   48.0     1.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
plot(benchmarks.split$B.aov)
```


Residuals vs Fitted
Residuals
Fitted values
aov(tempo ~ cpu * cores * mem)
90
120
85
−2
−1
0
1
2
24
26
28
30
32
34


Normal Q−Q
Standardized residuals
Theoretical Quantiles
aov(tempo ~ cpu * cores * mem)
90
120
85
−2
−1
0
1
2






```
(SS = anova(benchmarks.split$B.aov)['Sum Sq'])
```

```
##                Sum Sq
## cpu             59.94
## cores            0.44
## mem            836.07
## cpu:cores        0.19
## cpu:mem        109.31
## cores:mem        0.21
## cpu:cores:mem    0.56
## Residuals       48.04
```

```r
SST = sum(SS)
(benchmarks.split$B.pv = round(100*SS/SST, 2))
```

```
##                Sum Sq
## cpu              5.68
## cores            0.04
## mem             79.27
## cpu:cores        0.02
## cpu:mem         10.36
## cores:mem        0.02
## cpu:cores:mem    0.05
## Residuals        4.55
```

```r
(benchmarks.split$B.flig = fligner.test(tempo ~ interaction(cpu, cores, mem), benchmarks.split$B))
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  tempo by interaction(cpu, cores, mem)
## Fligner-Killeen:med chi-squared = 3.7271, df = 11, p-value = 0.9772
```

```r
# p-value > alfa (assumindo 95% de nivel de confiança), então é válido


TukeyHSD(benchmarks.split$B.aov)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = tempo ~ cpu * cores * mem, data = benchmarks.split$B)
##
## $cpu
##            diff     lwr     upr p adj
## Intel-AMD 1.999 1.47964 2.51836     0
##
## $cores
##           diff       lwr       upr     p adj
## 8-4 -0.1703333 -0.689693 0.3490264 0.5127752
##
## $mem
##             diff       lwr       upr p adj
## 4 GB-2 GB -6.237 -7.002111 -5.471889     0
## 8 GB-2 GB -8.909 -9.674111 -8.143889     0
## 8 GB-4 GB -2.672 -3.437111 -1.906889     0
##
## $`cpu:cores`
##                         diff        lwr        upr     p adj
## Intel:4-AMD:4     2.11266667  1.1404665  3.0848669 0.0000032
## AMD:8-AMD:4      -0.05666667 -1.0288669  0.9155335 0.9986514
## Intel:8-AMD:4     1.82866667  0.8564665  2.8008669 0.0000458
## AMD:8-Intel:4    -2.16933333 -3.1415335 -1.1971331 0.0000018
```

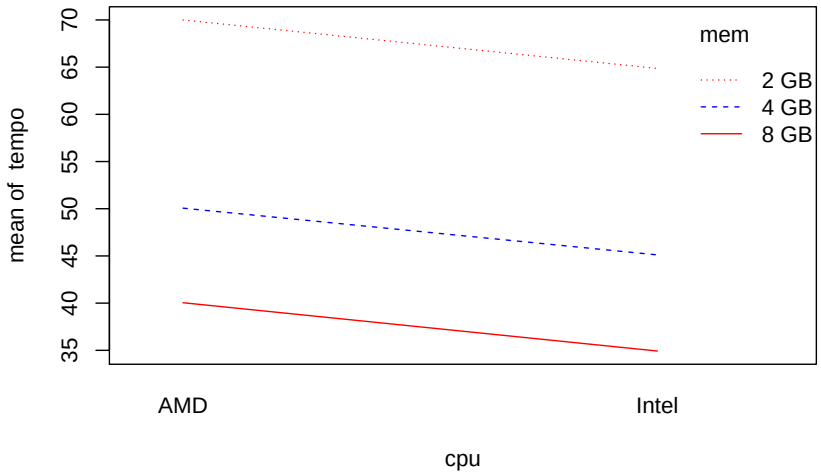
```
## Intel:8-Intel:4 -0.28400000 -1.2562002  0.6882002 0.8642730
## Intel:8-AMD:8    1.88533333  0.9131331  2.8575335 0.0000270
##
## $`cpu:mem`
##                            diff         lwr         upr     p adj
## Intel:2 GB-AMD:2 GB     5.685   4.3571647   7.0128353 0.0000000
## AMD:4 GB-AMD:2 GB      -3.903  -5.2308353  -2.5751647 0.0000000
## Intel:4 GB-AMD:2 GB    -2.886  -4.2138353  -1.5581647 0.0000007
## AMD:8 GB-AMD:2 GB      -5.714  -7.0418353  -4.3861647 0.0000000
## Intel:8 GB-AMD:2 GB    -6.419  -7.7468353  -5.0911647 0.0000000
## AMD:4 GB-Intel:2 GB    -9.588 -10.9158353  -8.2601647 0.0000000
## Intel:4 GB-Intel:2 GB  -8.571  -9.8988353  -7.2431647 0.0000000
## AMD:8 GB-Intel:2 GB   -11.399 -12.7268353 -10.0711647 0.0000000
## Intel:8 GB-Intel:2 GB -12.104 -13.4318353 -10.7761647 0.0000000
## Intel:4 GB-AMD:4 GB     1.017  -0.3108353   2.3448353 0.2250650
## AMD:8 GB-AMD:4 GB      -1.811  -3.1388353  -0.4831647 0.0024465
## Intel:8 GB-AMD:4 GB    -2.516  -3.8438353  -1.1881647 0.0000134
## AMD:8 GB-Intel:4 GB    -2.828  -4.1558353  -1.5001647 0.0000012
## Intel:8 GB-Intel:4 GB  -3.533  -4.8608353  -2.2051647 0.0000000
## Intel:8 GB-AMD:8 GB    -0.705  -2.0328353   0.6228353 0.6180355
##
## $`cores:mem`
##                  diff        lwr       upr     p adj
## 8:2 GB-4:2 GB -0.003  -1.330835  1.324835 1.0000000
## 4:4 GB-4:2 GB -6.116  -7.443835 -4.788165 0.0000000
## 8:4 GB-4:2 GB -6.361  -7.688835 -5.033165 0.0000000
## 4:8 GB-4:2 GB -8.779 -10.106835 -7.451165 0.0000000
## 8:8 GB-4:2 GB -9.042 -10.369835 -7.714165 0.0000000
## 4:4 GB-8:2 GB -6.113  -7.440835 -4.785165 0.0000000
## 8:4 GB-8:2 GB -6.358  -7.685835 -5.030165 0.0000000
## 4:8 GB-8:2 GB -8.776 -10.103835 -7.448165 0.0000000
## 8:8 GB-8:2 GB -9.039 -10.366835 -7.711165 0.0000000
## 8:4 GB-4:4 GB -0.245  -1.572835  1.082835 0.9937927
## 4:8 GB-4:4 GB -2.663  -3.990835 -1.335165 0.0000043
## 8:8 GB-4:4 GB -2.926  -4.253835 -1.598165 0.0000005
## 4:8 GB-8:4 GB -2.418  -3.745835 -1.090165 0.0000286
## 8:8 GB-8:4 GB -2.681  -4.008835 -1.353165 0.0000037
## 8:8 GB-4:8 GB -0.263  -1.590835  1.064835 0.9913886
##
## $`cpu:cores:mem`
##                                diff        lwr           upr     p adj
## Intel:4:2 GB-AMD:4:2 GB     5.548   3.375414   7.720586139 0.0000000
## AMD:8:2 GB-AMD:4:2 GB      -0.140  -2.312586   2.032586139 1.0000000
## Intel:8:2 GB-AMD:4:2 GB     5.682   3.509414   7.854586139 0.0000000
## AMD:4:4 GB-AMD:4:2 GB      -3.924  -6.096586  -1.751413861 0.0000076
## Intel:4:4 GB-AMD:4:2 GB    -2.760  -4.932586  -0.587413861 0.0035370
## AMD:8:4 GB-AMD:4:2 GB      -4.022  -6.194586  -1.849413861 0.0000044
## Intel:8:4 GB-AMD:4:2 GB    -3.152  -5.324586  -0.979413861 0.0004862
## AMD:4:8 GB-AMD:4:2 GB      -5.818  -7.990586  -3.645413861 0.0000000
## Intel:4:8 GB-AMD:4:2 GB    -6.192  -8.364586  -4.019413861 0.0000000
## AMD:8:8 GB-AMD:4:2 GB      -5.750  -7.922586  -3.577413861 0.0000000
## Intel:8:8 GB-AMD:4:2 GB    -6.786  -8.958586  -4.613413861 0.0000000
## AMD:8:2 GB-Intel:4:2 GB    -5.688  -7.860586  -3.515413861 0.0000000
## Intel:8:2 GB-Intel:4:2 GB   0.134  -2.038586   2.306586139 1.0000000
```

```
## AMD:4:4 GB-Intel:4:2 GB    -9.472 -11.644586  -7.299413861 0.0000000
## Intel:4:4 GB-Intel:4:2 GB  -8.308 -10.480586  -6.135413861 0.0000000
## AMD:8:4 GB-Intel:4:2 GB    -9.570 -11.742586  -7.397413861 0.0000000
## Intel:8:4 GB-Intel:4:2 GB  -8.700 -10.872586  -6.527413861 0.0000000
## AMD:4:8 GB-Intel:4:2 GB   -11.366 -13.538586  -9.193413861 0.0000000
## Intel:4:8 GB-Intel:4:2 GB -11.740 -13.912586  -9.567413861 0.0000000
## AMD:8:8 GB-Intel:4:2 GB   -11.298 -13.470586  -9.125413861 0.0000000
## Intel:8:8 GB-Intel:4:2 GB -12.334 -14.506586 -10.161413861 0.0000000
## Intel:8:2 GB-AMD:8:2 GB     5.822   3.649414   7.994586139 0.0000000
## AMD:4:4 GB-AMD:8:2 GB      -3.784  -5.956586  -1.611413861 0.0000164
## Intel:4:4 GB-AMD:8:2 GB    -2.620  -4.792586  -0.447413861 0.0069369
## AMD:8:4 GB-AMD:8:2 GB      -3.882  -6.054586  -1.709413861 0.0000095
## Intel:8:4 GB-AMD:8:2 GB    -3.012  -5.184586  -0.839413861 0.0010018
## AMD:4:8 GB-AMD:8:2 GB      -5.678  -7.850586  -3.505413861 0.0000000
## Intel:4:8 GB-AMD:8:2 GB    -6.052  -8.224586  -3.879413861 0.0000000
## AMD:8:8 GB-AMD:8:2 GB      -5.610  -7.782586  -3.437413861 0.0000000
## Intel:8:8 GB-AMD:8:2 GB    -6.646  -8.818586  -4.473413861 0.0000000
## AMD:4:4 GB-Intel:8:2 GB    -9.606 -11.778586  -7.433413861 0.0000000
## Intel:4:4 GB-Intel:8:2 GB  -8.442 -10.614586  -6.269413861 0.0000000
## AMD:8:4 GB-Intel:8:2 GB    -9.704 -11.876586  -7.531413861 0.0000000
## Intel:8:4 GB-Intel:8:2 GB  -8.834 -11.006586  -6.661413861 0.0000000
## AMD:4:8 GB-Intel:8:2 GB   -11.500 -13.672586  -9.327413861 0.0000000
## Intel:4:8 GB-Intel:8:2 GB -11.874 -14.046586  -9.701413861 0.0000000
## AMD:8:8 GB-Intel:8:2 GB   -11.432 -13.604586  -9.259413861 0.0000000
## Intel:8:8 GB-Intel:8:2 GB -12.468 -14.640586 -10.295413861 0.0000000
## Intel:4:4 GB-AMD:4:4 GB     1.164  -1.008586   3.336586139 0.7882374
## AMD:8:4 GB-AMD:4:4 GB      -0.098  -2.270586   2.074586139 1.0000000
## Intel:8:4 GB-AMD:4:4 GB     0.772  -1.400586   2.944586139 0.9846078
## AMD:4:8 GB-AMD:4:4 GB      -1.894  -4.066586   0.278586139 0.1423069
## Intel:4:8 GB-AMD:4:4 GB    -2.268  -4.440586  -0.095413861 0.0337036
## AMD:8:8 GB-AMD:4:4 GB      -1.826  -3.998586   0.346586139 0.1787218
## Intel:8:8 GB-AMD:4:4 GB    -2.862  -5.034586  -0.689413861 0.0021374
## AMD:8:4 GB-Intel:4:4 GB    -1.262  -3.434586   0.910586139 0.6945610
## Intel:8:4 GB-Intel:4:4 GB  -0.392  -2.564586   1.780586139 0.9999673
## AMD:4:8 GB-Intel:4:4 GB    -3.058  -5.230586  -0.885413861 0.0007912
## Intel:4:8 GB-Intel:4:4 GB  -3.432  -5.604586  -1.259413861 0.0001105
## AMD:8:8 GB-Intel:4:4 GB    -2.990  -5.162586  -0.817413861 0.0011209
## Intel:8:8 GB-Intel:4:4 GB  -4.026  -6.198586  -1.853413861 0.0000043
## Intel:8:4 GB-AMD:8:4 GB     0.870  -1.302586   3.042586139 0.9628686
## AMD:4:8 GB-AMD:8:4 GB      -1.796  -3.968586   0.376586139 0.1968653
## Intel:4:8 GB-AMD:8:4 GB    -2.170  -4.342586   0.002586139 0.0505255
## AMD:8:8 GB-AMD:8:4 GB      -1.728  -3.900586   0.444586139 0.2428830
## Intel:8:8 GB-AMD:8:4 GB    -2.764  -4.936586  -0.591413861 0.0034685
## AMD:4:8 GB-Intel:8:4 GB    -2.666  -4.838586  -0.493413861 0.0055732
## Intel:4:8 GB-Intel:8:4 GB  -3.040  -5.212586  -0.867413861 0.0008680
## AMD:8:8 GB-Intel:8:4 GB    -2.598  -4.770586  -0.425413861 0.0076956
## Intel:8:8 GB-Intel:8:4 GB  -3.634  -5.806586  -1.461413861 0.0000371
## Intel:4:8 GB-AMD:4:8 GB    -0.374  -2.546586   1.798586139 0.9999797
## AMD:8:8 GB-AMD:4:8 GB       0.068  -2.104586   2.240586139 1.0000000
## Intel:8:8 GB-AMD:4:8 GB    -0.968  -3.140586   1.204586139 0.9244740
## AMD:8:8 GB-Intel:4:8 GB     0.442  -1.730586   2.614586139 0.9998926
## Intel:8:8 GB-Intel:4:8 GB  -0.594  -2.766586   1.578586139 0.9982765
## Intel:8:8 GB-AMD:8:8 GB    -1.036  -3.208586   1.136586139 0.8862082
```

Análise para o *benchmark* A

- 1: Analisando graficamente o benchmark A, é possível observar sobreposição com o gráfico de interação dos dados “cpu, cores, tempo” e “mem, cores, tempo” dos cores, apresentando que os cores: quantidade de núcles não são significativos para o tempo de execução. Já para o gráfico “cpu, mem, tempo” aparenta que a memoria é um fator significativo, assim como o tipo de “cpu” que aparenta ser significativo.
- 2: Para 95% de nível de confiança, a anova indica que os fatores “cpu” e “mem” são significativos, já para os demais fatores e interações, os valores são estatisticamente não significativos.
- 3: A alocação de variação apresenta 3.98% para o fator cpu e 95.92% para a memória, os demais fatores e interações possuem valor insignificantes. Além disso, os resíduos é 0.1, próximo a zero.
- 4: Para os fatores significativos “cpu” e “mem” e a sua interaçao [“cpu”, “mem”] todos passam no teste de Tukey, pois o alfa 0.05 é menor que o valor p. Para as interações e fatores, apresentam no mínimo um valor que não é significativos.
- 5: Os resíduos estão próximos ao valor 0 o que é bom, não possuem tendência, e tem poucos pontos fora que podem atrapalhar a homogeneidade de variâncias. O teste de fligner apresenta um valor p de 0.83 que é suficiente para um nível de confiança de 95%, aprovando o modelo.
- 6: As variâncias devem ser aproximadamente iguais, e é aprovado pelo teste de filgner, que apresenta um valor p de 0.83, o que seria válido até para um nível de confiança de 20%, aprovando o modelo utilizando 95% de confiança.
- 7: O desempenho é bem influenciado pela memória, então quanto maior a memória melhor para o programa, e que o tipo de cpu favorece o tipo da Intel.

```
with(benchmarks.split$A, interaction.plot(cpu, mem, tempo, col = c("red", "blue")))
```



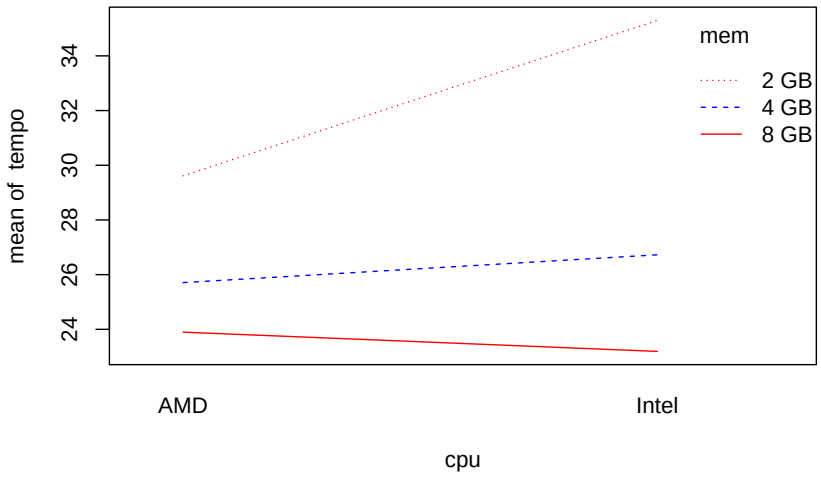
Análise para o *benchmark* B

- 1: Analisando graficamente o benchmark B, é possível observar que dessa vez os cores modificam os gráficos, talvez não seja estatisticamente significativo, mas não existe mais sobreposição com o gráfico

de interação dos dados “cpu, cores, tempo” e “mem, cores, tempo” para os cores. Para o fator “mem”, novamente acredito ser bastante significante para o tempo de execução, e o tipo de cpu apresenta a maior variância para 2 GB de “mem” e parece inverter do benchmark A e favorecer o tipo de cpu “AMD”.

- 2: Para 95% de nível de confiança, a anova indica que os fatores “cpu” e “mem” e a interaçao entre eles [“cpu”, “mem”] são significativos, já para os demais fatores e interações, os valores são estatisticamente não significativos.
- 3: A alocação de variação apresenta 5.68% para o fator cpu e 79.27% para a mem (a memória é a que mais contribui para a alocação de variação), para a interação entre eles [“cpu”, “mem”] apresenta 10.36%, os demais fatores e interações possuem valores próximos a zero. Dessa vez, os resíduos são maiores que o benchmark A, apresentando 4.55 de resíduo, mas ainda parece bom.
- 4: Para os fatores significativos “cpu” e “mem” todos passam no teste de Tukey, pois o alfa 0.05 é menor que o valor p. Já a sua interação [“cpu”, “mem”] contem 2 valores que não passam no teste de Tukey, significando que para os tipos de “cpu” as “mem” 4 GB e 8 GB não possuem diferença estatística.
- 5: Os resíduos não possuem tendência, mas fica próximo a 0, e tem alguns pontos distantes que podem atrapalhar a homogeneidade de variâncias. O teste de fligner apresenta um valor p de 0.98 que é suficiente para um nível de confiança de 95%, aprovando o modelo.
- 6: As variâncias devem ser aproximadamente iguais, e é aprovado pelo teste de filgner, que apresenta um valor p de 0.98, o que seria válido até para um nível de confiança de 5%, aprovando o modelo utilizando 95% de confiança.
- 7: O desempenho é bem influenciado pela memória, mas nesse benchmark para os casos de 4 GB e 8 GB o tempo de execução não apresenta grandes diferenças. O tipo de cpu pode se inverter, mas é difícil dizer qual se favorece, pois a diferença do tempo de execução é pouca entre os dados, mas se fosse obrigado a escolher um, escolheira o AMD, pois mesmo que para 4 e 8 GB não exista diferença significativa, para 2 GB é possível observar que a AMD ganha, tendo a média próxima a 30 e a Intel passando dos 34.

```
with(benchmarks.split$B, interaction.plot(cpu, mem, tempo, col = c("red", "blue")))
```



```
with(benchmarks.split$B, interaction.plot(cpu, cores, tempo, col = c("red", "blue")))
```