Comparação de Alternativas

Beatriz Bento Martins

Data de entrega: 01/05/2022

Descrição da atividade

O objetivo desta atividade é aplicar as técnicas de comparação de alternativas. A atividade é dividida em duas partes:

1. Comparação usando ICs vs. teste t
2. Comparação de múltiplas alternativas

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;
  - o arquivo de dados referente à Parte 2, que é necessário para o processamento do .Rmd.

Configuração

Nesta atividade, a única configuração necessária consiste em carregar o pacote `ggplot2` e o arquivo `compar-altern.R`, que são usados na Parte 1 da atividade.

```
library(ggplot2)
source("compar-altern.R")
```

Parte 1: Comparação usando ICs vs. teste t

Uma das formas de determinar se duas variáveis são estatisticamente diferentes é observando os seus intervalos de confiança. Existem três resultados possíveis para essa comparação:

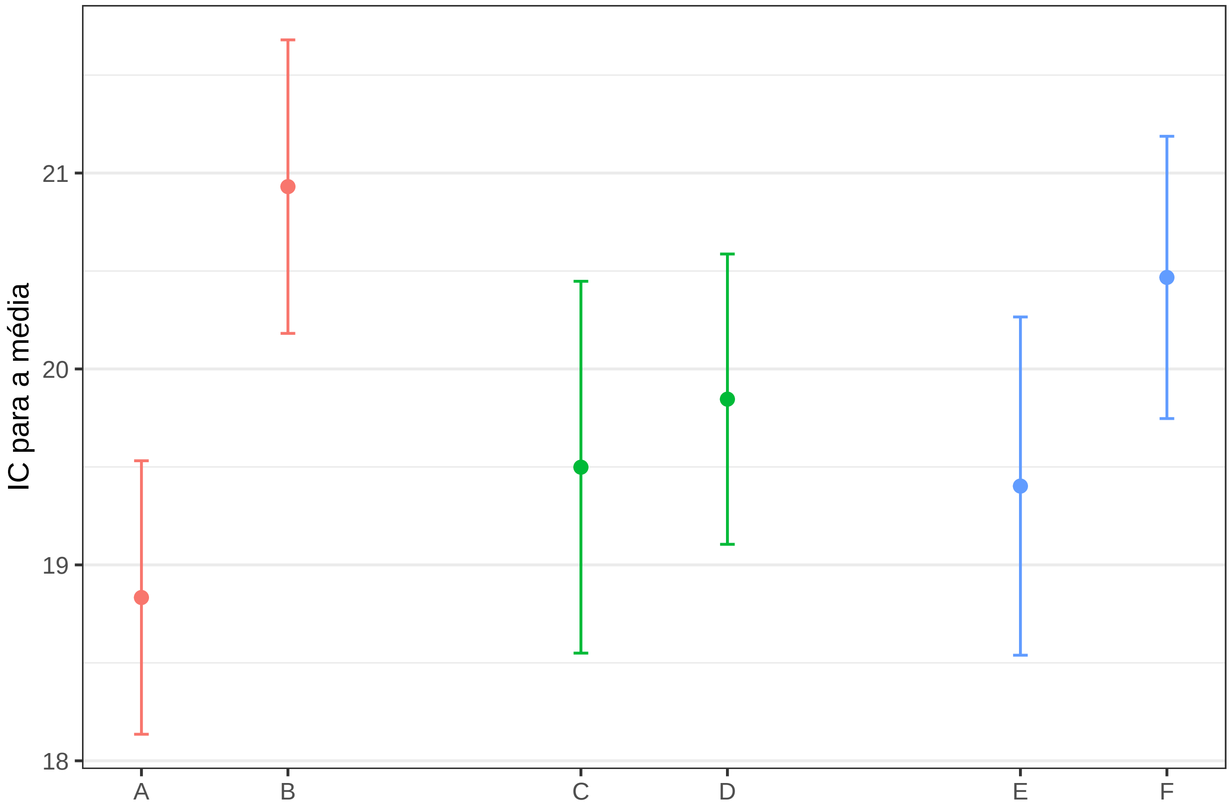
1. *Não existe sobreposição entre os ICs.* Nesse caso, as variáveis são estatisticamente diferentes.

2. *Existe sobreposição entre os ICs, e ao menos um deles inclui a média da outra variável.* Nesse caso, as variáveis são estatisticamente equivalentes.
3. *Existe sobreposição entre os ICs, mas nenhum deles inclui a média da outra variável.* Nesse caso não é possível afirmar nada, sendo necessário realizar um teste t (ou equivalente) para determinar se a diferença é estatisticamente significativa.

O gráfico abaixo ilustra os três resultados. As variáveis comparadas são as colunas A–F do conjunto de dados contido no arquivo `comparacao-ic.dat`, e os ICs têm um nível de confiança de 95%. As conclusões visuais são as seguintes:

1. As variáveis A e B são estatisticamente diferentes, e A < B.
2. As variáveis C e D são estatisticamente equivalentes.
3. Não é possível afirmar se E < F ou não, é preciso realizar um teste t para ver se a diferença é estatisticamente significativa.

```
dados <- read.table("comparacao-ic.dat", head=TRUE)
dados.ic <- geraIC(dados)
plotaIC(dados.ic)
```



Para esta primeira parte, você deve comparar os pares de variáveis representados no gráfico (A/B, C/D, E/F) usando o teste t com um nível de confiança de 95% (o mesmo usado para gerar os ICs). Para cada par de variáveis, indique claramente (a) o resultado da comparação (ou seja, se as variáveis são ou não estatisticamente diferentes) e (b) se esse resultado é idêntico ao obtido pela comparação visual dos ICs. Considere que as observações não são pareadas.

Análise e respostas

```
# seu código R aqui

dados
```

```
##          A        B        C        D        E        F
## 1  21.67424 21.48488 17.58587 21.26818 18.64442 21.78631
## 2  17.89704 24.32349 20.55486 20.01863 18.66001 19.56320
## 3  18.21695 20.44786 22.16888 20.11890 16.25540 20.91498
## 4  19.07907 19.49555 15.30860 21.91918 18.65243 19.49381
```

```
## 5  18.00849 20.48859 20.85825 19.61256 20.70046 18.68500
## 6  18.04723 23.16563 21.01211 18.10359 20.39522 21.16747
## 7  17.83902 18.79209 18.85052 22.14951 20.09999 18.50934
## 8  17.62202 22.55174 18.90674 18.95269 18.19454 21.17046
## 9  18.71426 22.49435 18.87110 20.96972 18.61681 21.63742
## 10 18.75473 21.00471 18.21992 19.12810 16.61094 20.61974
## 11 16.79095 20.51034 19.04561 23.20460 18.89368 20.26016
## 12 18.62689 19.81680 18.00323 20.04881 19.51039 19.12733
## 13 17.83667 19.95021 18.44749 19.58112 22.41193 17.97558
## 14 17.97756 21.47243 20.12892 19.99748 21.00303 22.87319
## 15 19.25654 23.60541 21.91899 17.74181 18.00883 20.80591
## 16 20.34458 20.26990 19.77943 18.66476 19.71110 22.82785
## 17 21.97173 18.41395 18.97798 16.63992 16.73078 17.63928
## 18 18.33997 19.41463 18.17761 18.31801 20.75641 21.17257
## 19 21.90886 20.88739 18.32566 20.41141 20.94583 22.43292
## 20 17.76329 20.02441 24.83167 20.06820 23.24223 20.68431
```

```r
nc = 0.95
alfa = 1- nc
(dados.A.shap = shapiro.test(dados$A))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  dados$A
## W = 0.82545, p-value = 0.002124
```

```r
dados.A.shap$p.value > alfa
```

```
## [1] FALSE
```

```r
(dados.B.shap = shapiro.test(dados$B))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  dados$B
## W = 0.95462, p-value = 0.4428
```

```r
dados.B.shap$p.value > alfa
```

```
## [1] TRUE
```

```r
(dados.C.shap = shapiro.test(dados$C))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  dados$C
## W = 0.93847, p-value = 0.2243
```

```r
dados.C.shap$p.value > alfa
```

```
## [1] TRUE
```

```r
(dados.D.shap = shapiro.test(dados$D))
```

```
##
##  Shapiro-Wilk normality test
```

```
##
## data:  dados$D
## W = 0.98634, p-value = 0.9886
```

```
dados.D.shap$p.value > alfa
```

```
## [1] TRUE
```

```
(dados.E.shap = shapiro.test(dados$E))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  dados$E
## W = 0.9709, p-value = 0.7738
```

```
dados.E.shap$p.value > alfa
```

```
## [1] TRUE
```

```
(dados.F.shap = shapiro.test(dados$F))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  dados$F
## W = 0.96094, p-value = 0.5628
```

```
dados.F.shap$p.value > alfa
```

```
## [1] TRUE
```

```
#O shapiro tem como objetivo avaliar se uma distribuição é semelhante a uma distribuição normal.
#Quando p < alfa indica que tem diferença e quando p > alfa não há diferença.

(dados.AB.test = t.test(dados$A, dados$B, conf.level=nc, paired=FALSE))
```

```
##
##  Welch Two Sample t-test
##
## data:  dados$A and dados$B
## t = -4.2872, df = 37.812, p-value = 0.0001202
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.087673 -1.106755
## sample estimates:
## mean of x mean of y
##  18.83350  20.93072
```

```
dados.AB.test$p.value < alfa
```

```
## [1] TRUE
```

```
(dados.CD.test = t.test(dados$C, dados$D, conf.level=nc, paired=FALSE))
```

```
##
##  Welch Two Sample t-test
##
## data:  dados$C and dados$D
## t = -0.60358, df = 35.889, p-value = 0.5499
```

```
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.5139020  0.8195255
## sample estimates:
## mean of x mean of y
##  19.49867  19.84586
```

```
dados.CD.test$p.value < alfa
```

```
## [1] FALSE
```

```
(dados.EF.test = t.test(dados$E, dados$F, conf.level=nc, paired=FALSE))
```

```
##
##  Welch Two Sample t-test
##
## data:  dados$E and dados$F
## t = -1.9827, df = 36.821, p-value = 0.0549
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.15377092  0.02353298
## sample estimates:
## mean of x mean of y
##  19.40222  20.46734
```

```
dados.EF.test$p.value < alfa
```

```
## [1] FALSE
```

```
dados.AB.test$conf.int
```

```
## [1] -3.087673 -1.106755
## attr(,"conf.level")
## [1] 0.95
```

```
dados.CD.test$conf.int
```

```
## [1] -1.5139020  0.8195255
## attr(,"conf.level")
## [1] 0.95
```

```
dados.EF.test$conf.int
```

```
## [1] -2.15377092  0.02353298
## attr(,"conf.level")
## [1] 0.95
```

Respostas aqui - No par AB podemos identificar que não há valor 0 em seu intervalo de confiança e por isso eles são estatisticamente diferentes e seu valor p é menor que alfa.

- Os pares CD e EF são estatisticamente iguais/equivalentes pois o valor 0 está contido no intervalo de confiança e seus valores p é maior que alfa.

Parte 2: Comparação de três algoritmos de ordenação

Na segunda parte iremos comparar o tempo de execução de três algoritmos de ordenação, *QuickSort*, *MergeSort* e *HeapSort*. Esses três algoritmos têm complexidade $O(n \log n)$ no caso médio, e são considerados eficientes. Para essa comparação iremos usar tempos de execução medidos pelo script Python `sortcomp3.py`. Esse script mede o tempo que cada algoritmo leva para ordenar um vetor de `n` elementos (em uma rodada, cada algoritmo ordena um vetor diferente, sempre de tamanho `n`). O número de rodadas pode ser passado como parâmetro na linha de comando (por *default* são realizadas 3 rodadas). A cada rodada os elementos do vetor sofrem uma permutação aleatória; logo, é possível (mas pouco provável) que o vetor esteja (quase) em ordem (de)crescente.

O script pode ser executado no RStudio Cloud. Na janela inferior esquerda, normalmente usada para o console, há uma aba Terminal, na qual você pode executar comandos do Linux.

Neste experimento, primeiro execute o script usando o comando `python sortcomp3.py 2`. O número de rodadas (2, no exemplo) fica a seu critério.

A seguir, faça uma análise de variância adotando um nível de confiança de 95%, e responda aos seguintes itens:

1. Qual a porcentagem de variação que pode ser explicada pelas alternativas e qual a porcentagem explicada pelo ruído das medições?
2. Mostre a tabela ANOVA (conforme o esquema abaixo) e determine se existem diferenças estatisticamente significativas entre os tempos médios de resposta de cada algoritmo.

Fonte de variação	Alternativas	Erros	Total
Soma de quadrados			
Graus de liberdade			
Média quadrática			
F calculado			
F crítico			

3. Caso a ANOVA indique que há diferenças estatisticamente significativas, ranqueie os algoritmos de acordo com o seu tempo médio de resposta (use o teste de Tukey).

Lembre-se que os tempos de execução dos algoritmos devem ser salvos em um arquivo de dados para que sua análise seja reproduzível. Para facilitar essa tarefa, o script já gera a saída em um formato apropriado; você pode redirecionar a saída do script para um arquivo (por exemplo, `python sortcomp3.py 2 >parte2.dat`) ou simplesmente criar o arquivo de dados no próprio editor do RStudio (crie um novo arquivo texto e cole a saída do script).

Análise e respostas

```
p2.dados <- read.table("parte2.dat", header = TRUE)
summary(p2.dados)
```

```
##      merge             heap            quick       
##  Min.   :0.4923   Min.   :0.8196   Min.   :0.2270  
##  1st Qu.:0.4945   1st Qu.:0.8236   1st Qu.:0.2351  
##  Median :0.4957   Median :0.8251   Median :0.2386  
##  Mean   :0.4964   Mean   :0.8260   Mean   :0.2405  
##  3rd Qu.:0.4973   3rd Qu.:0.8280   3rd Qu.:0.2448  
##  Max.   :0.5290   Max.   :0.8368   Max.   :0.2667
```

```
p2.stack = stack(p2.dados)

# os dados na forma correta podemos então invocar aov

(p2.aov = aov(values~ind, p2.stack))
```

```
## Call:
```

```
##    aov(formula = values ~ ind, data = p2.stack)
##
## Terms:
##                       ind Residuals
## Sum of Squares  17.229913  0.009173
## Deg. of Freedom         2       297
##
## Residual standard error: 0.005557574
## Estimated effects may be unbalanced
```

```r
ssa = 17.230
sse = 0.009
sst = ssa + sse
(alt = ssa/sst * 100)
```

```
## [1] 99.94779
```

```r
(ruido = sse/sst * 100)
```

```
## [1] 0.0522072
```

```r
(hsd = TukeyHSD(p2.aov, conf.level = 0.95))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = values ~ ind, data = p2.stack)
##
## $ind
##                   diff        lwr        upr p adj
## heap-merge   0.3295611  0.3277098  0.3314124     0
## quick-merge -0.2559222 -0.2577735 -0.2540709     0
## quick-heap  -0.5854833 -0.5873346 -0.5836320     0
```

```r
#critico

(Fcrit = qf(0.95, p2.aov$rank - 1, p2.aov$df.residual))
```

```
## [1] 3.026153
```

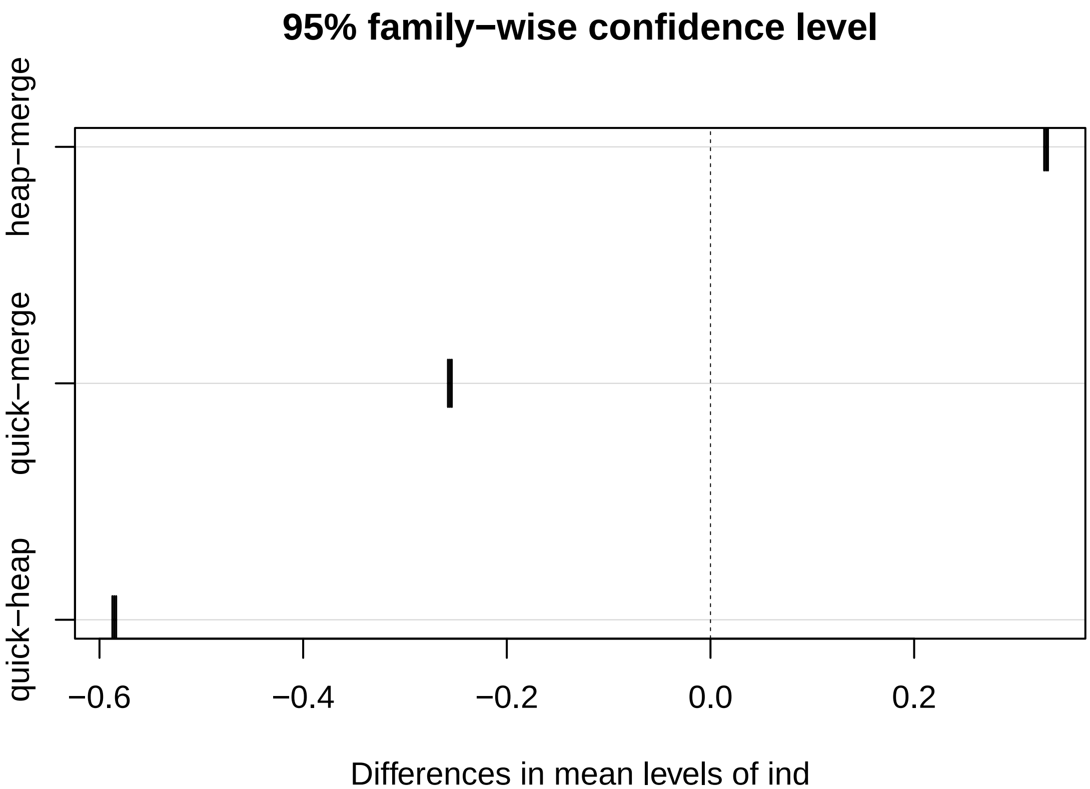
```r
k <- 3
n <- 100
sa2 <- ssa/(k-1)
se2 <- sse/(k*(n-1))
fcalc <- sa2/se2
fcalc > Fcrit
```

```
## [1] TRUE
```

```r
apply(p2.dados, 2, mean)
```

```
##     merge      heap     quick
## 0.4964351 0.8259962 0.2405129
```

```
plot(hsd)
```



- *Respostas*
- 1- A porcentagem de variação entre as diferenças das alternativas apresentadas é de 99.95%.
- A variação por ruído de 0.05%.
- Tabela ANOVA

Fonte de variação	Alternativas	Erros	Total
Soma de quadrados	17.23	0.009	17.239
Graus de liberdade	2	297	299
Média quadrática	8.615	3×10^{-5}	
F calculado	2.84295×10^5		
F crítico	3.03		

- 3- Podemos observar que fcalc > fcrit, por isso o ANOVA indica que existe uma diferença significativa. Para ranquear os algoritmos utilizando o teste de Turkey, temos quick como o melhor, merge o seguinte e heap o pior, levando em conta o tempo de resposta como métrica.