

Função compara.rep

Página de ajuda da função

`compara.rep``package:unknown`

R Documentation

Teste de associação de dois conjuntos de dados.

Description

Faz a correlação de dois conjuntos de dados e produz o boxplot e gráfico de dispersão dos dados.

Usage:

```
compara.rep(x,y=NULL,type="Dados",sob=0.75,metodo="pearson",pvalor=0.05,te=0.7)
```

Arguments:

`x` Um data frame em que as linhas são os nomes de cada observação (ex: `gene1, gene2...`, `especie1, especie2...`) e as colunas são as diferentes amostras (ex: `replicata1, replicata2...`).

`y` Com este argumento, a função aceita dados em tabelas separadas (as duas tabelas serão mescladas para realizar os cálculos).

`type` Texto para adicionar aos gráficos (o que são seus dados? expressão gênica, espécies....?).

`sob` Valor mínimo que se deseja da proporção de sobreposição.

`método` Método do teste de correlação: "pearson", "spearman" ou "kendall".

`pvalor` Valor mínimo que se deseja do p-valor do teste de correlação.

`te` Valor mínimo que se deseja do tamanho de efeito.

Details:

A função `compara.rep`, testa a associação de dois conjuntos de dados através do p-valor, do tamanho de efeito (calculados no teste de correlação) e a proporção em que os dados se sobrepõem (quanto o conjunto 1 está dentro do conjunto 2 e vice-versa). Além disso, gera o boxplot com a distribuição dos dados e um gráfico de dispersão.

Esta função teve como motivação testar réplicas (biológicas ou técnicas) em quaisquer tipos de dados numéricos, por isso, apresenta um teste embutido

que, de acordo com o padrão da função ou com os argumentos fornecidos pelo usuário, retorna se a replicata foi aprovada ou não.

Value:

A função `compara.rep` cria três arquivos de saída:

`boxplots`: todos os boxplots gerados pela função serão adicionados em um arquivo pdf

`correlações`: todos os gráficos de dispersão serão adicionados em um pdf.

Resultados: um arquivo de texto com os resultados gerados que também são retornados na tela.

Warning(s):

Se fornecido o argumento `y`, a coluna com o tipo de dado deve ter o mesmo nome do argumento `x`, caso contrario a função para e solta uma mensagem de erro.

Note(s):

Valores faltantes (NAs) serão sempre substituídos por 0.

Author(s): Gisele Antoniazzi Cardoso (para dúvidas e perguntas:
`gi_antoni@yahoo.com.br`)

Example(s):

##Exemplo para dados em apenas uma tabela

```
Dados <- seq(1:20)
a <- c(20,53,1000,35,1,2548,20000,546,12,47,80,54,75,0,5,695,1,1,25,88)
b <- c(25,55,1050,33,1,2541,12487,540,10,42,61,525,73,1,3,690,1,1,27,95)
c <- c(40,60,78,15,45,78,12,58,12,25,95,63,547,874,12,45,7,45,74,62)
d <- c(34,70,85,20,46,79,9,57,11,23,93,60,540,880,14,48,8,52,70,65)
x <- data.frame(Dados,a,b,c,d)
```

```
compara.rep(x)
```

##Adicione o argumento `y` para mesclar tabelas

```
Dados <- seq(1:20)
e <- sample(1:20)
f <- sample(1:20)
y <- data.frame(Dados,e,f)
```

```
compara.rep(x,y)
```

Código da função

```
compara.rep <-  
function(x,y=NULL,type="Dados",sob=0.75,metodo="pearson",pvalor=0.05,te=0.7)  
{ #funcao com seus respectivos argumentos (y= para dados em tabelas  
separadas,type= nome para dar titulo aos graficos,sob=threshold de  
sobreposicao dos dados,metodo=metodo para o teste de correlacao, pvalor=  
threshold do pvalor do teste de correlacao,te= threshold do tamanho de  
efeito da correlacao dos dados)  
#Testando o argumento y  
if(is.object(y)) { #a presenca do argumento y e opcional, portanto devemos  
testar se ele existe ou nao  
  if(colnames(x[1]) != colnames(y[1])) { #se y existir precisamos mesclar as  
duas tabelas e para isso as colunas contendo o tipo de dado devem ter mesmo  
nome  
    stop("O nome da coluna de seus dados deve ser o mesmo nas duas tabelas")  
  } #se as colunas tiverem nomes diferentes a funcao solta uma mensagem de erro  
  } else { #se os nomes forem iguais as tabelas sao mescladas  
    x <- merge(x,y,by =  
colnames(x[1]),incomparables=NA,all.x=TRUE,all.y=TRUE) #mescla as tabelas(se  
uma linha existir em uma tabela e na outra nao teremos a insercao de NAs)  
    x[is.na(x)] <- 0 #os NAs sao trocados por 0  
  }  
} else { #se o y nao existir, devemos tirar os NAs tambem  
  
  x[is.na(x)] <- 0 #os NAs sao trocados por 0  
}  
  
#Criando boxplots dos dados  
  
pdf("boxplots.pdf") #todos os boxplots gerados serao guardados em um mesmo  
arquivo pdf  
  
for (i in seq(2,ncol(x),2)) { #loop para criar boxplot pegando de duas em  
duas colunas sem repeticao (ex: 1-2,3-4,5-6,etc); i sempre sera as colunas  
pares (a coluna 1 equivale ao "nome" dos dados, por isso todos os loops  
comecarao pela coluna 2)  
  j <- i+1 #j sempre sera as colunas impares  
  boxplot(x[,i],x[,j],names=c(colnames(x[i]),colnames(x[j])),main=paste("Distr  
ibuicao dos dados de",colnames(x[i]),"e",colnames(x[j])),ylab=type) #cria os  
boxplots de duas em duas colunas, colocando sempre o nome das colunas no  
titulo  
}  
  
dev.off()  
  
#Sobreposicao dos dados
```

```
sob.1 <- rep(NA,(ncol(x)/2)) #objeto para guardar a proporcao de
sobreposicao dos dados do conjunto 1 em relacao ao 2 dentro de um loop
sob.2 <- rep(NA,(ncol(x)/2)) #objeto para guardar a proporcao de
sobreposicao dos dados do conjunto 2 em relacao ao 1 dentro de um loop

d <- 1 #primeira posição dos resultados que guardam os valores da proporcao
de sobreposicao (sob.1 e sob.2)

for (i in seq(2,ncol(x),2)) { #loop2 para pegar as colunas de dois em dois
sem repeticao (ex: 1-2,3-4,5-6,etc); i sempre sera as colunas pares
  a <- 0 #objeto para contar quantos numeros de um conjunto 1 estao entre
o valor minimo e maximo do conjunto 2
  b <- 0 #objeto para contar quantos numeros de um conjunto 2 estao entre
o valor minimo e maximo do conjunto 1
  j <- i+1 #j sempre será as colunas impares
  for(c in 1:nrow(x)) { #loop para fazer os testes de sobreposicao dos
dados (conjunto 1 esta dentro dos limites do 2 e vice-versa)
    if(x[c,i] >= min(x[,j]) & x[c,i] <= max(x[,j])) { #testa se um valor
do conjunto 1 esta entre min e o max do conjunto 2
      a <- a+1 #para cada numero dentro da condicao acima, e acrescimo
um ao valor do objeto a
    }
    if(x[c,j] >= min(x[,i]) & x[c,j] <= max(x[,i])) { #testa se um valor
do conjunto 2 esta entre min e o max do conjunto 1
      b <- b+1 #para cada numero dentro da condicao acima, e acrescimo
um ao valor do objeto b
    }
  }
  sob.1[d] <- round(a/nrow(x),2) #calcula da proporcao de sobreposicao do
conjunto 1 em relacao ao 2
  sob.2[d] <- round(b/nrow(x),2) #calcula da proporcao de sobreposicao do
conjunto 2 em relacao ao 1
  d <- d+1 #todas vez que acaba o loop, os resultados de sob.1 e sob.2 sao
guardados em uma nova posicao dentro do objeto
}
```

#Teste de correlacao dos dados

```
cor.pvalue<-rep(NA,(ncol(x)/2)) #objeto para guardar os p-valores da
correlacao dentro de um loop
cor.estimate <-rep(NA,(ncol(x)/2)) #objeto para guardar os valores de te da
correlacao dentro de um loop
```

```
k <- 1 #primeira posicao dos resultados de correlacao (p-valor e tamanho de
efeito) dentro dos objetos
```

```
for (i in seq(2,ncol(x),2)) { #loop para fazer os testes de correlacao de
duas em duas colunas sem repeticao (ex: 1-2,3-4,5-6,etc); i sempre sera as
colunas pares
```

```
j<- i+1 #j sempre sera as colunas impares
if(metodo == "spearman" & metodo == "kendall") { #se o metodo de spearman
ou kendall for usado usaremos o argumento exact=FALSE para evitar o aviso
"Cannot compute exact p-values with ties"
  cor.pvalue[k] <- cor.test(x[,i],x[,j],method=metodo,exact=FALSE)$p.value
#faz o teste de correlacao e guarda no objeto apenas os p-valores
  cor.estimate[k] <-
cor.test(x[,i],x[,j],method=metodo,exact=FALSE)$estimate #faz o teste de
correlacao e guarda no objeto apenas os valores de te
} else { #se o metodo for pearson, nao precisamos inserir o argumento
exact=FALSE
  cor.pvalue[k] <- cor.test(x[,i],x[,j],method=metodo)$p.value #faz o
teste de correlacao e guarda no objeto apenas os p-valores
  cor.estimate[k] <- cor.test(x[,i],x[,j],method=metodo)$estimate #faz o
teste de correlacao e guarda no objeto apenas os valores de te
}
k <- k+1 #todas vez que acaba o loop, os resultados de correlacao sao
guardados em uma nova posicao dentro do objeto
}

#Criando os graficos de dispersao com suas respectivas ablines

pdf("correlacoes.pdf") #todas os graficos de dispersao gerados serao
guardados em um mesmo arquivo pdf

k <- 1 #valor da primeira posicao dos resultados de correlacao a serem
testados dentro do loop

for (i in seq(2,ncol(x),2)) { #loop para fazer os graficos de dispersao e
desenhar as linhas de duas em duas colunas sem repeticao (ex:
1-2,3-4,5-6,etc); i sempre sera as colunas pares
  j<- i+1 #j sempre sera as colunas impares
plot(x[,i],x[,j],xlab=colnames(x[i]),ylab=colnames(x[j]),main=paste("Correla
cao de",colnames(x[i]),"e",colnames(x[j]))) #cria os graficos de dispersao
de duas em duas colunas
  if (cor.pvalue[k] <= pvalor & cor.estimate[k] >= te) {#teste para criar
a linha no grafico de dispersao de acordo com o default ou os argumentos do
usuario
    abline(lm(x[,i]~x[,j])) #desenha a linha no grafico de dispersao
caso o teste acima seja verdadeiro
  }
  k<- k+1 #todas vez que acaba o loop, o objeto é acrescido de valor para
testar uma nova posicao dos resultados de correlacao (p-valor e tamanho de
efeito)
}

dev.off()

#Seu resultado foi como o usuario desejava?
```

```
Resultado <- rep(NA,length(cor.pvalue)) #objeto para guardar o resultado
"aprovado" ou "revisar" dentro de um loop

for(i in 1:length(cor.pvalue)) { #loop para testar se o resultado foi
aprovado ou nao
  if(sob.1[i] >= sob & sob.2[i] >= sob) { #testa se a sobreposicao dos
dados esta dentro do default ou sugerido pelo usuario
    if(cor.pvalue[i] <= pvalor & cor.estimate[i] >= te) { #testa se o pvalor
e o tamanho de efeito dos dados estao dentro do default ou sugerido pelo
usuario
      Resultado[i] <- "APROVADO" #caso as condicoes acima seja verdadeiras
as replicatas serao aprovadas
    } else { #se o pvalor e o tamanho de efeito nao satisfazerem a
condicao a replicata e reprovada
      Resultado[i] <- "REVISAR" #caso UMA das condicoes acima seja
falsa, as replicatas serao reprovadas e precisam ser revidas pelo usuario
    }
  } else { #se o os valores de sobreposicao nao satisfazerem a condicao a
replicata e reprovada
    Resultado[i] <- "REVISAR" #caso o threshold de sobreposicao dos dados
seja falso, as replicatas serao reprovadas e precisam ser revidas pelo
usuario
  }
}

#Saida dos resultados de acordo com o metodo utilizado de correlacao

if(metodo == "pearson") { #se o metodo de pearson foi escolhido a coluna de
tamanho de efeito sera intitulada r2
  resultados <-
list(p.value=cor.pvalue,r2=cor.estimate,Sobreposicao1=sob.1,Sobreposicao2=sob.2) #criar uma lista com resultados para transforma-los em tabela
}

if(metodo == "spearman") { #se o metodo de spearman foi escolhido a coluna
de tamanho de efeito sera intitulada rho
  resultados <-
list(p.value=cor.pvalue,rho=cor.estimate,Sobreposicao1=sob.1,Sobreposicao2=sob.2) #criar uma lista com resultados para transforma-los em tabela
}

if(metodo == "kendall") { #se o metodo de kendall foi escolhido a coluna de
tamanho de efeito sera intitulada tau
  resultados <-
list(p.value=cor.pvalue,tau=cor.estimate,Sobreposicao1=sob.1,Sobreposicao2=sob.2) #criar uma lista com resultados para transforma-los em tabela
}
}
```

```
names <- rep(NA,length(cor.pvalue)) #objeto para guardar os nomes das linhas
da tabela dentro de um loop

k<- 1 #valor da primeira posicao dos nomes das linhas da tabela que sera
criada dentro do loop

for (i in seq(2,ncol(x),2)) { #loop para pegar o nome das colunas par a par
sem repeticao (ex: 1-2,3-4,5-6,etc); i sempre sera as colunas pares
  j<- i+1 #j sempre sera as colunas impares
  names[k] <- paste0(colnames(x[i]),"- ",colnames(x[j])) #coloca os nomes dos
pares testados no objeto
  k <- k+1 #todas vez que acaba o loop, o objeto é acrescido de valor para
guardar um novo nome em uma nova posicao de names
}

tabela <- data.frame(resultados,Resultado) #cria a tabela de resultados
row.names(tabela) <- names #coloca o nome das linhas da tabela

write.table(tabela,file="resultados.txt",sep="\t") #cria um arquivo com a
tabela dos resultados

#Imprimindo avisos e a tabela de resultados

print("100% completado!") #imprimi o primeiro aviso na tela
print("Resultados finais no arquivo: resultados.txt") #imprimi o segundo
aviso na tela

return(tabela) #Retorna a tabela de resultados na tela

}
```

From:

<http://ecor.ib.usp.br/> - ecoR

Permanent link:

http://ecor.ib.usp.br/doku.php?id=05_curso_antigo:r2015:alunos:trabalho_final:gi_antoni:trabalho_final



Last update: **2020/08/12 06:04**