Raquel Dietsche Monfardini

Aluna de Graduação em Biologia, Instituto de Biociências, USP.

Pesquisa em Biologia (TCC): “Caracterização do Transcriptoma de *Hermetia illucens* (Diptera: Stratiomyidae)”, orientado pela Profa Dra Tatiana Teixeira Torres.

Meus Exercícios

Link para a página com meus exercícios resolvidos: exec

Trabalho Final

Proposta A

Em fitoquímica, é comum a medição de absorbâncias para avaliar a atividade antioxidante de uma amostra em relação à absorbância de um controle.

Existem vários tipos de experimentos (como o FRAP ou DPPH) que dividem-se em dois tipos de resultados: os que aumentam a absorbância com relação ao controle, e os que diminuem a absorbância. Neste contexto, o objetivo dessa função é avaliar a atividade de uma ou mais amostras em relação a um controle. Será possível dois tipos de análise, indicado pelo usuário: o primeiro considerando que os valores das amostras são menores que o do controle, e outro considerando o contrário. Sendo assim, a partir de um data.frame ou uma matriz como entrada de dados, informações sobre o teste escolhido e os valores do controle, a função realizará o teste equivalente e retornará um data.frame com as atividades e, para cada fator (amostra) existente, um gráfico de dispersão bem como a equação da reta.

Comentários Vitor Rios

No geral, é mais importante especificar o que a função do que o que os índices significam. Por favor, reescreva sua proposta com mais detalhes Além disso, há uma confusão nos termos: amostra não é fator. o que sua função faria é pegar os dados e jogar para uma outra função fazer o calculo? ou vc vai implementar o calculo na mão?

Revisão da proposta A

Em fitoquímica, é comum a medição de absorbâncias para avaliar a atividade antioxidante de uma amostra. Existem vários tipos de experimentos (como o FRAP ou DPPH) que dividem-se em dois tipos de resultados: os que aumentam a absorbância com relação a um controle negativo (sem nenhuma substância antioxidante), e os que diminuem a absorbância. Neste contexto, o objetivo desta função é calcular a atividade antioxidante das amostras. A partir da entrada de dados do usuário (data.frame ou matriz - figura como exemplo), a função irá calcular as atividades utilizando uma das seguintes

fórmulas, determinado como argumento na função:

(media.controle.negativo – média.amostra) / media.controle.negativo x 100

, quando espera-se que o controle tenha uma absorbância maior (ex. DPPH) ou

(media.amostra - media.controle.negativo) / media.controle.positivo x 100

, quando espera-se que as amostras tenham uma absorbância maior (ex. FRAP); neste caso, é necessário indicar o controle positivo (absorbância do experimento com uma substância antioxidante conhecida). Em ambos os casos, é utilizada a média dos valores uma vez que pode haver replicatas na entrada de dados.

A partir dos resultados calculados pela função, serão criados gráficos de dispersão para cada amostra em relação a um fator, que entrará como argumento (no caso do exemplo, o único fator possível é concentracao), bem como a regressão linear. Irá retornar também um data.frame com os resultados calculados.

Como resultado opcional, a função também poderá calcular o ICx (concentração para atingir x% de atividade antioxidante) para todas as amostras a partir das regressões lineares geradas.

Comentários Vitor Rios

Seria então apenas o calculo do índice e o gráfico?

Comentários Raquel

Sim, mas uma coisa que eu pensei agora (muito útil para quem mexe com esse tipo de coisa) é que ela poderia calcular o IC50 (concentração necessária para atingir 50% de atividade), caso o usuário peça. Para tanto, o cálculo utilizaria a regressão linear gerada anteriormente.

Comentários Vitor Rios

Raquel, apenas o calculo do índice ficaria um função muito simples, apenas uma ou duas linhas de código. Retornar o valor do IC, ou talvez comparar efeitos em diferentes amostras, seria um avanço melhor para a função

Comentários Raquel

Fiz a adição do IC na proposta. Com isso, eu posso começar a trabalhar nela? (tenho mais afinidade com esta proposta do que com a B)

Comentários Vitor Rios

Raquel, pode começar. Eu acho que a proposta B seria mais util pra vc em termos de aprendizado, mas a decisão é sua

Proposta B

O objetivo desta função é calcular o melhor teste estatístico entre os disponíveis pela função (paramétrico e não paramétrico) dependendo da distribuição dos dados de entrada. Para tanto, esta função realizará primeiro uma análise exploratória dos dados (podendo ser um vetor, data.frame ou matriz) e, a partir do resultado, irá calcular o teste propriamente dito. Existirão duas saídas para o usuário: a primeira (referente à análise exploratória) será na forma de lista e irá conter a justificativa para a escolha ou não do teste, e a segunda será o resultado do teste escolhido.

Comentário Vitor Rios

Defina “melhor teste estatistico”. Sua função seleciona entre que testes? do modo como vc descreveu, ela apenas verifica a normalidade dos dados, é isso? vc implementaria o tete de normalidade ou iria passar para uma outra função? Ambas suas propostas precisam de mais trabalho na descrição. descreva o que as funções fariam em detalhes, se possivel dê um prototipo do codigo, por exemplo: uma boa dica pra facilitar o entendimento da suas propostas é postar a interface das funções, ou seja, a linha de comando que vc usa pra chamar a função, um pseudocodigo do que acontece dentro e uma linha de retorno

```
contagem(dataframe.dados, salva.arquivo=FALSE){
confere dados
manipula.dados
conta.ocorrencias
return(dataframe.contagens)
}
```

Revisão da proposta B

O objetivo desta função é determinar o melhor teste estatístico (disponível pela função) a partir da distribuição dos dados. A função terá dois possíveis testes estatísticos: anova e Kruskal-Wallis. A partir da entrada de dados (data.frame ou matriz), a função irá fazer primeiro um teste de normalidade e a partir do resultado determinar qual o melhor teste. Caso o teste de negativo para a normalidade dos dados, a função irá seguir automaticamente para o teste de Kruskal-Wallis, e para o anova caso dê positivo. Em ambos os casos, o fator a ser testado entrará como argumento na função. Sendo assim, existirão dois outputs para o usuário: o primeiro referente ao teste de normalidade e o segundo contendo o resultado do teste final definido pelos dados.

Comentários Vitor Rios

Sua função selecionaria entre os diferentes tipos de ANOVA paramétricos e não paramétricos (Kruskal-Wallis é apenas um)? Vc implementaria os cálculos na mão? e as outras premissas da anova? Se você colocasse os testes das diferentes premissas da ANOVA (homocedasticidade, por exemplo), creio que sua função teria uma aplicabilidade maior do que apenas calcular normalidade. Ou uma opção de transformar os dados caso não seja normal (e recalcular os testes de premissa depois da transformação), e fazer um teste post-hoc de comparações múltiplas. E uma mensagem de erro caso os dados não possam ser analisados devido à violação de premissas

Comentários Raquel

Como eu não tenho uma formação muito aprofundada em estatística, eu não pensei em fazer algo tão complexo quanto você está sugerindo, como outros tipos de testes ou premissas. Apesar disso gostei da ideia de implementar o teste post-hoc, uma vez que refinaria melhor o resultado. Com relação aos cálculos, a princípio eu tentaria fazê-los a mão (o que me ajudaria muito a entender os princípios), mas se ficar muito complicado/gigante, eu passaria para funções pré-definidas.

Comentários Vitor Rios

Raquel, eu gosto da idéia de uma função que teste as premissas, mesmo que seja só para as análises mais comuns (ANOVA é um mundo de análises) como paramétrica / não-paramétrica. É um bom exercício de programação, e lhe ajudaria a entender as análises. os testes post-hoc são também muito úteis. Vc pode se inspirar no programa GraphPad InStat, que faz exatamente isso: analisar as premissas e verificar a quais testes se adequam. Claro que vc não vai fazer uma versao do GraphPad pro R, mas pra anova/kruskallWallis dá pra fazer de boa. GraphPad InStat

Página de Ajuda

```
atv.antiox              package:nenhum                R Documentation


Cálculo da Atividade Antioxidante e Concentração de Inibição

Description:

Irá calcular a taxa de atividade antioxidante de n amostras, em relação a um
padrão, a partir dos valores de absorbância. Calcula também a concentração
de inibição da amostra.

Usage:

atv.antiox(x, cn, cp=NULL, disp, rep=NULL, ic=NULL, n="output.pdf")

Arguments:

x   Data frame ou matriz contendo os valores de absorbância das amostras.
cn  Vetor contendo um ou mais valores de absorbâncias referentes ao controle
negativo do experimento.
cp  Vetor contendo um ou mais valores de absorbâncias referentes ao controle
positivo do experimento.
disp    Nome da coluna que será usada como variável independente nos
gráficos de dispersão.
rep Caso as amostras estejam em replicata, indicar o número das colunas.
ic  Valor para o calculo da concentração de inibição (IC).
```

```
n   Nome do pdf que conterá os gráficos.

Details:

Será realizada a média para ambos os controles (cn e cp). Na ausência de cp,
a função parte do princípio que as amostras tem uma absorbância menor que o
controle negativo. Já na presença do cp, parte-se do principio que as
amostras tem uma absorbância maior que o controle negativo.

Value:

Arquivo em pdf contendo os gráficos de dispersão com a equação da reta e o
R2 para todas as amostras.

Se ic não for nulo, uma lista contendo duas tabelas: uma com as atividades
calculadas e outra com os ICs. Se ic for nulo, apenas a tabela das
atividades.

Note:

Pode conter valores faltantes (NA).

Author(s):

Raquel Dietsche Monfardini

Examples:

## Exemplo 1
Concentracao <- c(0.25,0.5,0.75,1,1.25)
Amostra <- rep(c("Carqueja","Eucalipto","Alecrim"),each=5)
rep1 <-
c(0.874,0.802,0.742,0.637,NA,0.858,0.797,0.633,0.592,0.485,0.913,0.826,0.761
,0.69,0.625)
rep2 <-
c(0.908,0.802,0.755,0.628,0.584,0.867,0.734,0.659,0.519,0.487,0.924,0.858,0.
739,0.718,0.646)
rep3 <-
c(0.916,0.793,0.761,0.618,0.652,0.879,0.751,0.681,0.534,0.506,0.957,0.844,0.
776,0.713,0.618)
dados <- data.frame(Amostra,Concentracao,rep1,rep2,rep3)

atv.antiox(dados,cn=c(1.083,1.084,1.036),disp="Concentracao",rep=c(3,4,5),ic
=50,n="exemplo_1.pdf")



## Exemplo 2
Concentracao <- c(0.25,0.5,0.75,1,1.25)
Carqueja <- c(0.908,0.802,0.755,0.628,0.584)
Eucalipto <- c(0.867,0.734,0.659,0.519,0.487)
Alecrim <- c(0.924,0.858,0.739,0.718,0.646)
```

```
dados <- data.frame(Concentracao,Carqueja,Eucalipto,Alecrim)

atv.antiox(dados,cn=c(1.083,1.084,1.036),disp="Concentracao",n="exemplo_2.pd
f")



## Exemplo 3
Concentracao <- c(0.25,0.5,0.75,1,1.25)
Amostra <- rep(c("Carqueja","Eucalipto","Alecrim"),each=5)
Metanol <-
c(0.257,0.405,0.513,0.666,0.821,0.302,0.455,0.613,0.761,0.904,0.203,0.354,0.
614,0.669,0.801)
Etanol <-
c(0.312,0.421,0.514,0.618,0.723,0.201,0.323,0.411,0.517,0.605,0.218,0.233,0.
423,0.501,0.622)
Agua <-
c(0.401,0.462,0.527,0.578,0.641,0.256,0.302,0.367,0.423,0.479,0.116,0.302,0.
358,0.431,0.496)
dados <- data.frame(Amostra,Concentracao,Metanol,Etanol,Agua)

atv.antiox(dados,cn=c(0.111,0.115,0.112),cp=c(0.530,0.520,0.521),disp="Conce
ntracao",ic=90,n="exemplo_3.pdf")
```

Código da Função

```
atv.antiox <- function(x, cn, cp=NULL, disp, rep=NULL, ic=NULL,
n="output.pdf")
# x=tabela de dados; cn=valores do controle negativo; cp=valores do controle
positivo; disp=nome variavel independente dos graficos de dipersao;
# rep=numero das colunas em replicata, caso existentes; ic=valor para o
calculo da concentracano de inibicao; n=nome do arquivo de saida dos
graficos
{
    ## Checando se esta em replicata e calculando a media caso positivo:
    if(!is.null(rep)) {        # Se indicado pelo usuario:
        media <- matrix(NA, nrow(x),length(rep))    # Criando objeto para o
calculo das medias
        for(i in length(rep):1) {    # Para todas as colunas indicadas pelo
usuario:
            media[,i] <- as.matrix(x[rep[i]])    # Armazenando as colunas
replicadas no novo objeto
            x[rep[i]] <- NULL    # Atualizando a matriz de dados: retirada
das colunas em replicata
        }
        media <- apply(media,1,mean,na.rm=TRUE)    # Calculo da media
        x$media <- media    # Atualizando a matriz de dados: adicionando a
coluna media
```

```
    }
    x2 <- NULL    # Objeto para reorganizacao dos dados
    f <- NULL     # Objeto para armazenar o numero da coluna que tiver
fatores
    d <- NULL     # Objeto para armazenar o numero da coluna que o usuario
escolheu para gerar os graficos
    n.col <- NULL # Objeto para armazenar os numeros das colunas que serao
geradas a partir do calculo dos indices



    ## Calculo dos indices:
    for(i in 1:ncol(x)) {    # Para todas as colunas na entrada de dados
        if(class(x[,i]) == "factor") {    # Para colunas que forem fatores
(ex. especie1, especie2):

            f <- i    # Caso exista, armazena o numero da coluna
            x2 <- data.frame(c(x2,x[i]))    # Acidiona essa coluna no novo
objeto de dados (x2)

        }
        else if(colnames(x[i]) == disp) {    # Para a coluna escolhida para
gerar os graficos de dispersao:
            d <- i    # Armazena o numero da coluna
            x2 <- data.frame(c(x2,x[i]))    # Adiciona essa coluna no novo
objeto de dados (x2)

        }

        else {    # Para todas as colunas numericas, calculo dos indices:
            if(is.null(cp)) {    # Modelo 1: amostras possuem absorbancia
menor que o controle (ex. metodo de DPPH); nao exige o controle positivo
                count <- (mean(cn)-x[i])/mean(cn)*100               #
Calculo dos indices
                colnames(count) <- paste0("atv_",colnames(x[i]))    #
Mudança do nome da coluna
                x2 <- data.frame(c(x2,x[i],count))                  #
Adiciona a coluna das absorbancias seguida da coluna das atividades
calculadas
                n.col <- c(n.col,ncol(x2))                          #
Armazena o numero da coluna das atividades calculadas (para a construcao dos
graficos)

            }

            else {    # Modelo 2: amostras possuem absorbancia maior que o
controle (ex. metodo de FRAP); exige o controle positivo

                count <- (x[i]-mean(cn))/mean(cp-mean(cn))*100      #
Calculo dos indices
                colnames(count) <- paste0("atv_",colnames(x[i]))    #
Mudança do nome da coluna
```

```
                x2 <- data.frame(c(x2,x[i],count))                  #
Adiciona a coluna das absorbancias seguida da coluna das atividades
calculadas
                n.col <- c(n.col,ncol(x2))                          #
Armazena o numero da coluna das atividades calculadas (para a construcao dos
graficos)

            }
        }
    }



    ## Geracao dos graficos e calculo dos ICs:
    pdf(n)    # Abertura do arquivo em pdf, que ira conter os graficos
    if(!is.null(f)) {     # Geracao dos graficos por fator, caso exista
        if(!is.null(ic)){    # Se indicado pelo usuario:

            ICs <- data.frame(levels(x2[,f]))    # Objeto ira conter os ICs
calculados, ordenados pelos fatores
            colnames(ICs) <- "amostra"    # Mudando o nome da coluna

            for(i in n.col) {    # Para todos os indices (colunas)
calculados

                col <- data.frame(rep(NA,length(levels(x2[,f]))))    #
Criando uma coluna para cada indice
                colnames(col) <- paste0("IC",ic,"_",colnames(x2[i-1]))    #
Nomeando as colunas com o valor do IC e o nome da coluna de referencia
                ICs <- data.frame(c(ICs,col))    # Adicionando no objeto que
irá conter os ICs

            }

        }
        for(i in levels(x2[,f])) {    # Para cada fator existente
            for(j in n.col) {     # E indice (coluna) calculado
                # Gerando o grafico de dispersao:
                plot(x2[x2[,f]==i,d], x2[x2[,f]==i,j], main=paste(i,"-
",colnames(x2[j-1])), xlab=colnames(x2[d]), ylab="Atividade")
                # Gerando a regressao linear:
                rl <- lm(x2[x2[,f]==i,j] ~ x2[x2[,f]==i,d])
                # Plotando a regressao linear no grafico:
                abline(rl)
                if(round(summary(rl)$coefficients[1,1],5)>0) {    # Se o
intercepto foi positivo:
                # Gerando as equacoes da reta e o R-squared:
                eq.r2 <- paste0("y =
",round(summary(rl)$coefficients[2,1],5)," x +
",round(summary(rl)$coefficients[1,1],5),"\nR-squared =
",round(summary(rl)$adj,5))
```

```
            }
            else {    # Se o intercepto for negativo:
                # Gerando as equacoes da reta e o R-squared:
                eq.r2 <- paste0("y =
",round(summary(rl)$coefficients[2,1],5)," x
",round(summary(rl)$coefficients[1,1],5),"\nR-squared =
",round(summary(rl)$adj,5))
            }
            # Plotando as equacoes nos graficos:
            mtext(eq.r2,4)
                # Se indicado pelo usuario, ira calcular os ICs:
                if(!is.null(ic)) {

                    IC <- (ic-
round(summary(rl)$coefficients[1,1],5))/round(summary(rl)$coefficients[2,1],
5)    # Calculo utilizando a equacao da reta gerada
                    ICs[ICs[,1]==i,paste0("IC",ic,"_",colnames(x2[j-1]))] <-
IC    # Colocando no objeto

                }
            }
        }
    }
    else {    # Geracao dos graficos por coluna, caso nao tenha fatores
        if(!is.null(ic)){    # Se indicado pelo usuario:

            ICs <- data.frame(colnames(x2[n.col-1]),rep(NA,length(n.col)))
# Objeto ira conter os ICs calculados, ordenados pelos indices calculados
(colunas)
            colnames(ICs) <- c("amostra",paste0("IC",ic))    # Mudando o
nome da coluna
        }

        for(j in n.col) {    # Para cada indice (coluna) calculado
            # Gerando o grafico de dispersao:
            plot(x2[,d], x2[,j], main=colnames(x2[j-1]),
xlab=colnames(x2[d]), ylab="Atividade")
            # Gerando a regressao linear:
            rl <- lm(x2[,j] ~ x2[,d])
            # Plotando a regressao linear no grafico:
            abline(rl)
            if(round(summary(rl)$coefficients[1,1],5)>0) {    # Se o
intercepto foi positivo:
                # Gerando as equacoes da reta e o R-squared:
                eq.r2 <- paste0("y =
",round(summary(rl)$coefficients[2,1],5)," x +
",round(summary(rl)$coefficients[1,1],5),"\nR-squared =
",round(summary(rl)$adj,5))
            }
            else {    # Se o intercepto for negativo:
                # Gerando as equacoes da reta e o R-squared:
```

```
                eq.r2 <- paste0("y =
",round(summary(rl)$coefficients[2,1],5)," x
",round(summary(rl)$coefficients[1,1],5),"\nR-squared =
",round(summary(rl)$adj,5))
            }
            # Plotando as equacoes nos graficos:
            mtext(eq.r2,4)
            # Se indicado pelo usuario, ira calcular os ICs:
            if(!is.null(ic)) {

                IC <- (ic-
round(summary(rl)$coefficients[1,1],5))/round(summary(rl)$coefficients[2,1],
5)    # Calculo utilizando a equacao da reta gerada
                ICs[ICs[,1]==colnames(x2[j-1]),2] <- IC    # Colocando no
objeto
            }

        }
    }
    dev.off()    # Fecha o arquivo pdf
    output <- list(Atividade=x2)    # Gerando objeto de saida com a tabela
das atividades calculadas

    if(!is.null(ic)){    # Se indicado pelo usuario:

        output$ICs <- ICs    # Acrescentando a tabela dos ICs no objeto de
saida

    }

    return(output)    # Retornando os dados

}
```

From:
http://ecor.ib.usp.br/ - **ecoR**

Permanent link:
http://ecor.ib.usp.br/doku.php?id=05_curso_antigo:r2015:alunos:trabalho_final:raquel.monfardini:start

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