Fernanda de Vasconcellos Barros

Doutoranda em Ecologia pela Unicamp, Laboratório de Ecologia Funcional de Plantas.

```
 vul.curve                package:nenhum                R Documentation

~~ function to ajustar curva de vulnerabilidade do xilema à cavitação ... ~~
  Description:
  ~~ A vul.curve é uma função para aplicação em estudos de hidráulica de
plantas. Ela ajusta dados de perda de conditividade em função do potencial
hídrico do xilema.
A função gera parâmetros importantes para a entender o funcionamento
hidráulico de plantas (P50 e P88) e o R2 do ajuste.
Além disso a função produz um gráfico com o ajuste escolhido ou com o melhor
ajuste.

  Usage:
  ~~ vul.curve (data, fit) ~~
  Arguments:
  ~  data      caráter. Referente ao nome do arquivo de entrada dos dados.
                        Deve ser um data frame com duas variáveis fatores
(espécie e ramo amostrado) e duas variáveir numéricas (x e y) nessa ordem.
                        x e y correspondem as variáveis de interesse
utilizadas no ajuste da curva, sendo x o potencial hídrico e y o porcentual
de perda de condutividade do xilema.

     fit       caráter. Referente ao tipo de ajste que você pretende
utilizar. As opções para esse argumento são o ajuste sigmoide (sig),  o
ajuste de Welbull (wbl) e o uma comparação entre esses ajustes, resultando o
melhor deles (select).
                        O argumento "select" é o defaut da função. Ver em
"details" as equações utilizadas em cada ajuste.
  Details:
  ~~ Os valores de PLC são plotados em relação aos valores de potencial
hídrico. A função vul.curve ajusta essa relação, que geramlmente na
literatura é citada como sendo sigmoidal.
Os dados de potencial hídrico para gerar a curva devem ser valores negativos
(variam entre -1 e -10) e os valores de PLC (perda de condutividade do
xilema) correspondem a valores de porcentagem (1 a 100).
Ver warning para verificar o formato do data frame a ser utilizado.

As equações utilizadas nos ajustes são:
Ajuste sigmoide =  sig.
Equação: PLC =  100 / (1 + exp (S/25*(pot.hid-P50))), onde:
  P50 (Mpa) = pressão do xilema (potencial hídrico) responsável por perda de
50% da sua condutividade;
S (% MPa−1) = inclinação da curva no ponto de inflexão
```

```
Ajuste Weibull = wbl
Equação: PLC = 100*(1-exp(-((pot.hid/b)^c))), onde:
  b e c, são constantes de Weibull.

Comparação entre os dois ajustes acima = select
Utiliza o critério de Akaike para determinar o melhor ajuste para os dados.
Opta pelo ajuste que tiver o menor AIC.
~~
  Value:

~ Se o argumento utilizado for fit = "sig", retorna:

  Um gráfico com o ajuste sigmoide e uma lista com:
    P50.sig: Valor do potencial hídrico em que a planta perde 50 % de sua
condutividade hidráulica, ajustado pela função sigmoide.
    P88.sig: Valor do potencial hídrico em que a planta perde 88 % de sua
condutividade hidráulica, ajustado  pela função sigmoide.
    R2.sig: Valor do R^2 para o ajuste sigmoide.

Se o argumento utilizado for fit = "wlb", retorna:
Um gráfico com o ajuste de Weilbull e uma lista com:

  P50.wbl: Valor do potencial hídrico em que a planta perde 50 % de sua
condutividade hidráulica, ajustado pelo modelo de Weilbull.
  P88.wbl: Valor do potencial hídrico em que a planta perde 88 % de sua
condutividade hidráulica, ajustado pelo modelo de Weilbull.
  R2.wbl: Valor do R^2 para o ajuste de Weilbull.

Se o argumento utilizado for fit = "select", retorna:
Um gráfico com o melhor ajuste e uma lista com:
  Nome do melhor ajuste (Sigmoid Fit ou Weilbull Fit)
  P50 do melhor ajuste (P50.sig ou P50.wbl)
  P88 do melhor ajuste (P88.sig ou P88.wbl)
  R^2 do melhor ajuste (R2.sig ou R2.wbl)
...

  Warning:
A função deve ser aplicada apenas a data.frame.
O data frame deve conter 4 variáveis, sendo as duas primeiras:fatores
(espécie e ramo, respectivamente);
e duas outras: variáveis numéricas x e y, onde x corresponde ao potencial
hídrico e y ao porcentual de perda de condutividade do xilema.

  Author(s):
  ~~ Fernanda V. Barros ~~
  References:
~ Sperry JS., Donnelly JR., Tyree MT. (1988). A method for measuring
hydraulic conductivity and embolism in xylem. Plant, Cell and Environment,
11: 35–40.
  Urli M., Porté AJ., Cochard H., Guengant Y., Burlett R., Delzon S. (2013).
```

```
Xylem embolism threshold for catastrophic hydraulic failure
in angiosperm trees. Tree Physiology 33:672-683.
  Cai J., Li S., Zhang H., Zhang S., Tyree MT. (2014). Recalcitrant
vulnerability curves: methods of analysis and the
concept of fibre bridges for enhanced cavitation resistance. Plant, Cell and
Environment 37: 35-44.
~
  See Also:
  ~~ nls() ~~~
  Examples:
  vul.curve (data,"sig") # para ajuste sigmoide

  vul.curve (data, "wbl") # para ajuste de Weilbull

  vul.curve(data,"select") # para selecionar o melhor ajuste

  vul.curve (data) # utiliza o defaut do argumento fit, que no caso é o
select.
```

```
 #### Função curva.vul --- Fernanda de Vasconcellos Barros #####

# A função curva.vul faz o ajuste de curvas de vulnerabilidade do xilema à
cavitação e gera parâmetros importantes para a entender o funcionamento
hidráulico de plantas (P50 e P88).

                                          ### Atenção! ###
# A entrada de dados deve ser um data frame com 4 variáveis, sendo as duas
primeiras fatores (espécie e ramo, respectivamente) e duas variáveis
numéricas,  x e y, onde x corresponde ao potencial hídrico e y ao porcentual
de perda de condutividade do xilema.



# Para o ajuste sigmoide utilizaremos a equação:

# PLC =  100 / (1 + exp (S/25*(pot.hid-P50))), onde:

# P50 (Mpa) = pressão do xilema (potencial hídrico) responsável por perda de
50% da sua condutividade;
# S (% MPa−1) = inclinação da curva no ponto de inflexão

# Para o ajuste "Weibull" (cumulative distribution function - CDF),
utilizaremos a equação:

# PLC = 100*(1-exp(-((pot.hid/b)^c))), onde:

# b e c, são constantes de Weibull.

                            ## Construindo a função curva.vul ##

vul.curve<- function (data, fit="select")
  # data = nome do objeto data.frame com os dados
```

```
  # fit = argumento que seleciona o ajuste para os dados do data.frame
"data" (fit == "sig" = ajuste sigmoide; fit== "wbl" = ajuste Weibull; fit==
"select" = ajusta o melhor modelo selecionado por critério de akaike).
{ # para abrir a função
    if ((dim(data)[2])!=4) # Se o número de variáveis/colunas do data.frame
for maior que 2, a função irá gerar aviso de erro
          {
           cat ("Warning: O objeto utilizado por essa função deve ser um
data.frame com duas variáveis")
          }
  colnames(data)<- (c("Species", "branch","water.pot","PLC")) # Para
transformar o nome das colunas do data.frame
                                    ## Fazendo o ajuste sigmoide ##
  p50.par<-round(mean(data$water.pot)) # estimando um valor para o start do
parâmetro p50
  S.par<-round(mean(data$PLC)) # estimando um valor para o start do
parâmetro S
  sig.fit<- nls(PLC ~ 100/(1+exp(S/25*(water.pot - p50))), data=data,
start=list (S=S.par,p50=p50.par)) # para fazer o ajuste sigmoide dos dados
  #Calculando o R2 do ajuste sigmoide
  RSS<-sum((residuals(sig.fit))^2) #Cálculo da soma dos quadrados residuais
(quadrado da diferença entre os valores observados e valores preditos).
  TSS<-sum((data$PLC-mean(data$PLC))^2) #
  R2.sig<-1-RSS/TSS
                                    ## Fazendo o ajuste de Weilbull ##
  b.par<-round(mean(data$water.pot)) # estimando um valor para o start do
parâmetro b
  c.par<-(mean(data$PLC))/3 # estimando um valor para o start do parâmetro c
  weilbull.fit<- nls(PLC~100*(1-exp(-((water.pot/b)^c))), data = data,
start=list (b=b.par, c=c.par)) # Ajuste de weilbull com o conjunto de dados.
  #Calculando o R2 do ajuste de Weilbull
  RSS<-sum((residuals(weilbull.fit))^2) # Cálculo da soma dos quadrados
residuais (quadrado da diferença entre os valores observados e valores
preditos).
  TSS<-sum((data$PLC-mean(data$PLC))^2)
  R2.wbl<-1-RSS/TSS
                                   ## Especificar os argumentos ##
  if (fit == "sig") # se o argumento ajuste for igual a sig deve gerar os
seguintes comandos listados entre as {}
            {
                 sig.fit # chama o objeto sig.fit, já criado anteriormente
                 P50.sig<-summary(sig.fit)$parameters[2,1] # Para obter o
valor do parâmetro de interesse; o P50.
                 S<-summary(sig.fit)$parameters[1,1] # Para obter o valor do
slope da curva. Esse valor será utilizado no cálculo do P88 no próximo
comando.
                 P88.sig<-(-50/S)+P50.sig # Calculando o valor do P88, a
partir do valor do P50
                 ## Plotando o gráfico ##
                 quartz() # Comando para abrir a janela gráfica
```

```
                min.pot<- min(data$water.pot)-0.5 # Calcular o menor valor
de pot. hid e reduzir o limite por mais 0.5 para os pontos não ficarem
colados no eixo y
                plot(PLC ~ water.pot,data=data, xlab = "ψ xylem (MPa)",
ylab = "PLC (%)", xlim=c(min.pot,0),ylim=c(0,100)) # plotando o gráfico
                ## Traçando linha de tendência do ajuste sigmoide ##
                x<-seq(0,min.pot,-0.1) # listando uma sequencia de valores
sobre os quais será traçada a linha.
                y<-predict(sig.fit,list(water.pot=x)) # valores de y
previstos pela equação (modelo especificado na função), a partir dos valores
de x (limitados na sequencia anterior).
                linha.sig<-lines(x,y, lty = 1) # manda traçar a linha entre
os pontos x e y do intervalo calculado acima, a partir do ajuste da curva.
                return(list(P50.sig,P88.sig,R2.sig)) # para retornar os
valores que eu quero a partir do ajuste sigmoide.
            }
  if (fit == "wbl") # se o argumento ajuste for igual a wbl deve gerar os
seguintes comandos listados entre as {}
            {
                weilbull.fit #chama o objeto weilbull.fit, já criado
anteriormente
                            ## Calculando o P50 a partir do ajuste de
Weilbull ##
                lower.pot<-min(data$water.pot) # valor mínimo do potencial
hídrico
                higher.pot<-max(data$water.pot) # valor máximo do potencial
hídrico
                data.range<- seq (higher.pot, lower.pot, -0.01) # dados
entre o maior e menor potencial hídrico, com intervalo de 0.01, entre cada
dado.
                data.pot<-predict(weilbull.fit,list(water.pot=data.range))
# valores preditos de PLC para cada valor dentro do conjunto de dados criado
no comando anterior.
                list.50<-which(data.pot > 49 & data.pot< 51) # lista a
posição de todos os dados em que o valor de ajuste do PLC está próximo de 50
(maior que 49 e menor que 51). Utilizei argumento lógica para isso.
                list.p50<-c(data.range[list.50]) # lista, dentro do
intervalo de dados de potencial hídrico, os valores que estão na mesma
posição da lista anterior. Ou seja os valores de potencial que quando
ajustados tiveram o PLC próximo à 50.
                P50.wbl <- mean(list.p50) # faz a média dos valores
listados no comando anterior.
                             ## Calculando o P88 a partir do ajuste de
Weilbull ##
                lower.pot2<-3*lower.pot # três vezes o valor mínimo do
potencial hídrico
                data.range2<-seq(higher.pot, lower.pot2,-0.01) # dados
entre o maior e menor potencial hídrico, com intervalo de 0.01, entre cada
dado. Entretanto nesse caso, o limite mínimo foi duplicado, para que quando
ajustados no modelo obtenhamos um valor de PLC igual ou maior que 88.
                data.pot2<-
```

```
predict(weilbull.fit,list(water.pot=data.range2)) # valores preditos de PLC para cada valor dentro do conjunto de dados criado no comando anterior
                list.88<-which(data.pot2 > 87 & data.pot2< 89)  # lista a posição de todos os dados em que o valor de ajuste do PLC está próximo de 88 (maior que 87 e menor que 89). Utilizei argumento lógica para isso.
                list.p88<-c(data.range2[list.88]) #lista, dentro do intervalo de dados de potencial hídrico, os valores que estão na mesma posição da lista anterior. Ou seja os valores de potencial que quando ajustados tiveram o PLC próximo à 88.
                P88.wbl <- mean(list.p88) # faz a média dos valores listados no comando anterior.
                                ## Plotando o gráfico e linha de tendência do ajuste de Weilbull ##
                quartz() # Comando para abrir a janela gráfica
                min.pot<- min(data$water.pot)-0.5 # Calcular o menor valor de pot. hid e reduzir o limite por mais 0.5 para os pontos não ficarem colados no eixo y
                plot(PLC ~ water.pot,data=data, xlab = "ψ xylem (MPa)", ylab = "PLC (%)", xlim=c(min.pot,0),ylim=c(0,100)) # plotando o gráfico
                x<-seq(0,min.pot,-0.1) # lista sequência de valores sobre os quais será traçada a linha.
                y<-predict(weilbull.fit,list(water.pot=x)) # valores de y previstos pela equação (modelo especificado na função), a partir dos valores de x (limitados na sequencia anterior).
                linha.wbl<-lines(x,y, lty = 1) # traça a linha entre os pontos x e y do intervalo calculado acima, a partir do ajuste da curva. A linha será traçada sobre o gráfico criado anteriormente.
                return(list(P50.wbl,P88.wbl,R2.wbl)) # retornar os valores de P50, P88 e R2 do modelo de Weilbull.
            }
  if (fit == "select") # se o argumento ajuste for igual a select deve gerar os seguintes comandos listados entre as {}. Entretanto, caso não coloque nada o defaut da função irá fazer o select.
            {
                sig.mod<-AIC(sig.fit) #Critério de Akaike do ajuste sigmoide
                wbl.mod<-AIC(weilbull.fit) #Critério de Akaike do ajuste Weibull
                if (sig.mod < wbl.mod) # se o modelo sig tiver um valor de Akaike menor que o modelo de weilbull ele deve ser utilizado.Fazer os comenado listados entre {}.
                            {
                                # Plotando o gráfico e a linha de tendência a partir dos resultados previstos pela equação ajustada.
                                quartz() # Comando para abrir a janela gráfica
                                min.pot<- min(data$water.pot)-0.5 # Calcular o menor valor de pot. hid e reduzir o limite por mais 0.5 para os pontos não ficarem colados no eixo y
                                plot(PLC ~ water.pot,data=data, xlab = "ψ
```

```
xylem (MPa)", ylab = "PLC (%)", xlim=c(min.pot,0),ylim=c(0,100)) # plotando
o gráfico
                                    x<-seq(0,min.pot,-0.1) # listando uma
sequencia de valores sobre os quais será traçada a linha.
                                    y<-predict(sig.fit,list(water.pot=x)) #
valores de y previstos pela equação (modelo especificado na função), a
partir dos valores de x (limitados na sequencia anterior).
                                    linha.sig<-lines(x,y, lty = 1) # manda
traçar a linha entre os pontos x e y do intervalo calculado acima, a partir
do melhor ajuste da curva, no caso o sigmoide.
                                    P50.sig<-summary(sig.fit)$parameters[2,1]
# Para obter o valor do parâmetro de interesse; o P50.
                                    S<-summary(sig.fit)$parameters[1,1] # Para
obter o valor do slope da curva. Esse valor será utilizado no cálculo do P88
posteriormente.
                                    P88.sig<-(-50/S)+P50.sig # Calculando o
valor do P88, a partir do valor do P50
                                    return(list("Sigmoid
fit",P50.sig,P88.sig,R2.sig)) # Retorna o nome do melhor modelo ajustado, o
valor do P50, P88 e o Rˆ2, nesse caso seria o ajuste Sigmoide.
                               }
                  else
                               {
                                   # Plotando o gráfico e a linha de tendência
a partir dos resultados previstos pela equação ajustada.
                                    quartz() # Comando para abrir a janela
gráfica
                                    min.pot<- min(data$water.pot)-0.5 #
Calcular o menor valor de pot. hid e reduzir o limite por mais 0.5 para os
pontos não ficarem colados no eixo y
                                    plot(PLC ~ water.pot,data=data, xlab = "ψ
xylem (MPa)", ylab = "PLC (%)", xlim=c(min.pot,0),ylim=c(0,100)) # plotando
o gráfico
                                    x<-seq(0,min.pot,-0.1) # listando uma
sequencia de valores sobre os quais será traçada a linha.
                                    y<-predict(weilbull.fit,list(water.pot=x))
# valores de y previstos pela equação (modelo especificado na função), a
partir dos valores de x (limitados na sequencia anterior).
                                    linha.wbl<-lines(x,y, lty = 1) # manda
traçar a linha entre os pontos x e y do intervalo calculado acima, a partir
do melhor ajuste da curva, no caso o weilbull.
                                    lower.pot<-min(data$water.pot) # valor
mínimo do potencial hídrico
                                    higher.pot<-max(data$water.pot) # valor
máximo do potencial hídrico
                                    data.range<- seq (higher.pot, lower.pot,
-0.01) # dados entre o maior e menor potencial hídrico, com intervalo de
0.01, entre cada dado.
                                    data.pot<-
predict(weilbull.fit,list(water.pot=data.range)) # valores preditos de PLC
para cada valor dentro do conjunto de dados criado no comando anterior.
```

```
                                    list.50<-which(data.pot > 49 & data.pot<
51) # lista a posição de todos os dados em que o valor de ajuste do PLC está
próximo de 50 (maior que 49 e menor que 51). Utilizei argumento lógica para
isso.
                                    list.p50<-c(data.range[list.50]) # lista,
dentro do intervalo de dados de potencial hídrico, os valores que estão na
mesma posição da lista anterior. Ou seja os valores de potencial que quando
ajustados tiveram o PLC próximo à 50.
                                    P50.wbl <- mean(list.p50) # faz a média
dos valores listados no comando anterior.
                                    lower.pot2<-3*lower.pot # três vezes o
valor mínimo do potencial hídrico
                                    data.range2<-seq(higher.pot,
lower.pot2,-0.01) # dados entre o maior e menor potencial hídrico, com
intervalo de 0.01, entre cada dado. Entretanto nesse caso, o limite mínimo
foi duplicado, para que quando ajustados no modelo obtenhamos um valor de
PLC igual ou maior que 88.
                                    data.pot2<-
predict(weilbull.fit,list(water.pot=data.range2)) # valores preditos de PLC
para cada valor dentro do conjunto de dados criado no comando anterior
                                    list.88<-which(data.pot2 > 87 & data.pot2<
89)  # lista a posição de todos os dados em que o valor de ajuste do PLC
está próximo de 88 (maior que 87 e menor que 89). Utilizei argumento lógica
para isso.
                                    list.p88<-c(data.range2[list.88]) #lista,
dentro do intervalo de dados de potencial hídrico, os valores que estão na
mesma posição da lista anterior. Ou seja os valores de potencial que quando
ajustados tiveram o PLC próximo à 88.
                                    P88.wbl <- mean(list.p88) # faz a média
dos valores listados no comando anterior.
                                    return(list("Weilbull Fit",
P50.wbl,P88.wbl,R2.wbl)) # Retorna o nome do melhor modelo ajustado, o valor
do P50, P88 e o Rˆ2, nesse caso seria o ajuste de Weilbull.
                                }
  }  # Para fechar a função if
} # Para fechar a função
```

exec

Proposta Trabalho Final

A curva de vulnerabilidade a cavitação gera parâmetros importantes para a entender o funcionamento hidráulico de plantas. A curva é construída a partir da porcentagem de perda de condutividade do xilema (PLC) em função do potencial hídrico (Pi). No geral as curvas de vulnerabilidade recebem um ajuste sigmoide pela equação:

PLC = 100 / (1 + exp (S/25*(Pi-P50))), onde o P50 (Mpa) a pressão (potencial hídrico) do xilema responsável por perda de 50% da sua condutividade; o S (% MPa−1) a inclinação da curva no ponto de inflexão.

Nessa proposta, pretendo desenvolver uma função que faça o ajuste sigmoide dessa curva e determine o valor do P50, para um determinado conjunto de dados. Além disso, a função incluirá a representação gráfica dessa curva e o valor do P88, outro parâmetro importante, que será calculado pela equação:

P88 = (-50/S) + P50.

Plano B

Desenvolver uma função que calcule a plasticidade fenotípica de uma característica para uma espécie/ população/genótipo. Essa plasticidade é estimada a partir de um índice de distâncias fenotípicas relativas (RDPI), como descrito por Valladares et al. (2006). As distâncias relativas são calculadas para pares de amostras (indivíduo) em diferentes condições ambientais, a partir da fórmula:

RDij ⇒i`j` = d ij ⇒i`j` / (xi`j` + xij),

onde d ij ⇒i`j` representa a distância entre a característica para todos os pares de amostras, sendo j e j`, indivíduos dos ambientes distintos, i e i`. xi`j` representa o valor da característica para um indivíduo i` no ambiente j`; e xij , o valor da característica para um indivíduo i no ambiente j. Com os valores de RD será calculado o RDPI, a partir da fórmula:

RDPI = Σ (d ij ⇒i`j` / (xi`j` + xij))/ n, onde n corresponde ao número total de RD. O RDPI pode variar entre 0 (nenhuma plasticidade) e 1 (máxima plasticidade).

> A duas propostas me parecerem viáveis. Acho que a proposta A pode ser incrementada. Talvez você possa ajustar diferentes curvas aos dados e dar como um dos outputs o ajuste dos modelos. Se for fazer a proposta B, deixe claro no help da função quais são os possíveis dados de input.
>
> —- *Cristiane*

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

Permanent link:
http://ecor.ib.usp.br/doku.php?id=05_curso_antigo:r2014:alunos:trabalho_final:nandavascon:start

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