Valéria Conversani

Mestranda em Oceanografia Biológica, Instituto Oceanográfico da USP, no Laboratório de Biologia da Conservação de Mamíferos Aquáticos

Título da minha dissertação: Idade e crescimento da toninha, *Pontoporia blainvillei* (Gervais & D'Orbigny, 1844) e do boto-cinza, *Sotalia guianensis* (Van Bénéden, 1864) de águas costeiras do Sudeste-Sul do Brasil. Orientador: Prof. Dr. Marcos César de Oliveira Santos.

e-mail: valconversani@usp.br

exec

Trabalho Final

Proposta A

PROPOSTA REVISADA. O tamanho corporal é a característica mais fundamental de um organismo, pois é responsável por limitar seus aspectos morfológicos e suas dimensões estruturais. Portanto, a estimativa dos parâmetros de crescimento se tornou fundamental em estudos de dinâmica populacional de determinada espécie utilizando modelos de crescimento que descrevam a variação do comprimento dessa espécie ao longo do tempo. Um dos modelos mais utilizados em estudos de crescimento (ex. mamíferos aquáticos, peixes, moluscos e crustáceos), são os modelos não lineares de von Bertalanffy e o de Gompertz.

Modelo de von Bertallanfy:

$L(t) = L\infty*[1- exp(-K*t-t0))]$

onde, L(t): comprimento do indivíduo em cm para uma dada idade t; L∞ (L – infinito): comprimento máximo teórico que pode ser atingido por uma espécie; K: constante de crescimento (cm/mês) (velocidade com que o individuo alcança L∞) e, t0: idade em que o indivíduo teria comprimento igual a zero.

Modelo de Gompertz:

Ct = a * Exp (- Exp (b – c * Idade))

onde, a: comprimento assintótico; b: fator de correção e c: taxa constante de crescimento.

Assim sendo, minha função terá como objetivo inicial, estimar os parâmetros de crescimento a partir do ajuste destes dois modelos e fazer uma representação gráfica das curvas de crescimento ajustadas para machos e fêmeas de uma determinada espécie. Minha ideia é de que a função retorne curvas de crescimento distintas para machos e fêmeas e compare-as por meio do teste de verossimilhança de Kimura (1980), a fim de testar as hipóteses de que as curvas são semelhantes ou independentes.

- A primeira parte seria um “ajuste dos modelos matemáticos” que irá retornar apenas os valores das estimativas dos parâmetros de crescimento, os valores dos intervalos de confiança e coeficiente de determinação (R2) ajustado de cada modelo. Essa parte não tem como objetivo ser um instrumento de decisão sobre qual modelo se ajustou melhor aos dados, pois para a utilização desta função, pressupõe-se que o usuário tenha um conhecimento prévio do assunto. Aqui terá um argumento para o usuário indicar qual dos dois modelos irá usar.

- A segunda parte irá verificar por meio do teste de Kimura, a semelhança de duas curvas de crescimento (machos e fêmeas), a partir dos parâmetros estimados das curvas que se pretende comparar, desde que sigam um mesmo modelo.

O objeto de entrada deverá conter as seguintes variáveis: idade, comprimento total e sexo.

O objeto de saída irá retornar: 1) uma tabela com os valores dos parâmetros estimados por cada modelo (ex: L infinito, valor de k, t0); 2) um sumário com os valores dos intervalos de confiança e coeficiente de determinação (R2) para cada um dos modelos; 3) representação gráfica das respectivas curvas ajustadas, sendo um gráfico para machos e outro para fêmeas e 4) o resultado do teste de verossimilhança.

A idéia inicial parece interessante, mas você precisa rever o retor no da função.

Não é possível retornar gráficos como elementos de uma lista,e de qualquer modo, esses histogramas seriam muito simples de produzir a partir dos dados iniciais. O que é possível é fazer a função produzir um gráfico além do objeto de saída.

Além disso, se o principal objetivo é estimar os parâmetros de crescimento, é melhor focar em calcular essas estimativas, e produzir gráfico que comparem os diferentes modelos. Que gráficos mostrariam melhor quão bem as curvas de crescimento se ajustam aos dados reais?

Você já sabe como fazer a estimação dos parâmetros e intervalos de confiança? Receio que essa função fique muito simples se for só o caso de usar funções que estimam parâmetros, mas se você conseguir mostrar os resultados num bom gráfico, pode ficar bom!

—-*Mali*

Resposta: Oi Mali, obrigada pelas sugestões. Minha ideia na verdade é que o retorno da função sejam os gráficos das curvas de crescimento, um para machos e outro para fêmeas e verificar a semelhança entre elas usando um teste de verossimilhança, mas acho que não deixei isso muito claro. O gráfico gerado pela função seria mais ou menos como esse:

Os parâmetros de crescimento serão estimados a partir da fórmula do modelo que o usuário escolher, com esses valores (ex: L infinito, valor de k, t0) serão gerados os gráficos das curvas e aí é que elas serão comparadas entre si. Não quero comparar os modelos, quero comparar as curvas. Ficou mais claro? Vou reescrever a proposta e tentar deixar o

mais claro possível. —-*Valéria*

Olá Valéria, Está um pouco mais claro, mas acho que você está confundindo alguns conceitos ainda.

O retorno da função é o objeto que você pode guardar em uma variável (usando um comando como “var ← lm(x,y)”). Esse objeto pode ser uma lista com vários valores e tabelas dentro, mas não pode conter gráficos. Por outro lado, gráficos podem ser gerados pela função, mas não podem ser guardados em variáveis. Como a sua função vai gerar gráficos e também retornar valores, é importante você ter em mente a diferença entre essas duas coisas que sua função vai fazer.

Você fala em duas partes da função. O que exatamente separa essas duas partes? Uma mesma chamada da função fará as duas partes? A segunda parte parece tratar de apenas um modelo por vez. Isso significa fazer esses gráficos para cada um dos modelos, ou determinar na chamada da função qual dos dois modelos utilizar? Eu sugiro que você deixe que o usuário determine qual modelo quer ajustar, através de um argumento da função.

Acho que está mais claro o que você pretende fazer e que já dá pra trabalhar na função.

—-*Mali*

Proposta B

Em busca do hambúrguer perfeito!

PROPOSTA REVISADA. O hambúrguer é um dos pratos mais populares do mundo, e a dica de ouro para quem quer preparar um bom hambúrguer é a quantidade correta e equilibrada de gordura no blend (composição de diferentes carnes). Se for muito acima ou muito abaixo, vai dar errado. A quantidade mínima de gordura em um hambúrguer é de 15%, mas o recomendado é que tenha em média 20% de gordura do total do peso do blend.

Portanto, como uma boa apreciadora de hambúrguer, pretendo criar uma função que calcule o melhor blend para um hambúrguer usando três tipos de carne, levando em conta sua porcentagem de gordura.

Meus argumentos serão: tipo de carne (neste argumento o usuário poderá indicar no mínimo 1 tipo de carne que ele deseja usar, ainda estou decidindo se ele escolherá os 3 tipos ou pelo menos um), tamanho do hambúrguer (pequeno= 150g, médio= 180g e grande= 200g), custo (barato, médio e caro), porcentagem de gordura do blend (magro= 15% ou gordo= 20%) e o número de hambúrgueres pretendido (opção de 1 a 10).

O próprio script da função já contém os objetos com os tipos de carne, a porcentagem de gordura de cada uma e o preço. O script também já contém um data frame cruzando essas informações.

Meu objeto de saída será um data frame com três colunas: 1) tipos de carnes calculadas para o blend (3 tipos); 2) quantidade de cada uma delas (em grama); 3) o preço equivalente a quantidade de cada uma.

Caso o usuário queira um hambúrguer mais “saudável” ou mais “suculento” e coloque no argumento “porcentagem de gordura” valores diferentes de 15 ou 20% de gordura, aparecerá a seguinte mensagem de erro: “Essa quantidade de gordura está diferente da quantidade ideal, por favor, tente novamente e escolha uma das duas opções”.

> Pelo tom do texto eu tenho a impressão de que você vai se divertir mais com esta proposta do que com a outra :) Porém, eu não entendi exatamente como funcionaria sua função.
>
> Em primeiro lugar, não entendi bem qual é a entrada da função. Você diz que seria a lista de carnes, mas também diz que o usuário teria que indicar quais carnes irá usar, a quantidade de hambúrgueres, etc. Então, essas coisas têm que ser argumentos da função também. É bom você pensar em como você gostaria de usar essa função, quer dizer, qual o comando exato pra executar a função, para ter uma idéia clara de quais são os argumentos de entrada.
>
> Por exemplo, não entendi o que significaria o usuário “tentar usar” uma porcentagem abaixo ou acima do ideal. O usuário pode determinar qual porcentagem de gordura usar?
>
> Mas o problema mesmo é como calcular o melhor blend. Imagino que se só houverem duas carnes, é fácil calcular a proporção etre elas que dá a proporção de gordura mais próxima da ideal. Mas com três ou mais carnes, podem haver blends diferentes que dão a mesma proporção de gordura. Esse é um problema com muitas soluções. Uma delas é só mostrar as opções usando apenas duas carnes. Outra é adicionar um critério de seleção, por exemplo: preço da carne, ou uma ordem de preferência.
>
> Pergunta final: o que acontece se todas as carnes listada estiverem acima (ou abaixo) da proporção desejada de gordura?
>
> Enfim, estou fazendo muitas perguntas porque seu texto está pouco claro, mas acho que essa função tem um bom potencial de ser divertida de fazer e envolver muito aprendizado em R. Mas tente definir melhor como exatamente sua função é usada e que tipo de retorno ela dá. Além disso, que tal mostrar um gráfico representando o resultado?
>
> —-*Mali*

> As duas propostas têm potencial, mas precisam ser melhor definidas e reescritas. Ainda dá tempo de reescrever as propostas e receber mais feedback! Projetar bem a função é uma parte importante do aprendizado.
>
> —-*Mali*

> Valéria, esperamos a re-elaboração das propostas para melhor orientá-la. Apresente de maneira mais clara as entradas e saídas das funções e tente pensar no passo a passo das

tarefas a serem executadas. Mas as propostas que você apresentou podem se tornar interessantes funções! —-*Sara*

Resposta: Olá Mali e Sara, muito obrigada pelas dicas e sugestões. Realmente acho que me equivoquei ao descrever a função do hambúrguer. Minha ideia é fazer uma função que calcule o melhor blend para um hambúrguer usando três tipos de carne.

Meus argumentos serão: tipo de carne (neste argumento o usuário poderá indicar no mínimo 1 tipo de carne que ele deseja usar, ainda estou decidindo se ele escolherá os 3 tipos ou pelo menos um), tamanho do hambúrguer (pequeno= 150g, médio= 180g e grande= 200g), custo (barato, médio e caro), porcentagem de gordura do blend (magro= 15% ou gordo= 20%) e o número de hambúrgueres pretendido (opção de 1 a 10).

O próprio script da função já contém os objetos com os tipos de carne, a porcentagem de gordura de cada uma e o preço. O script também já contém um data frame cruzando essas informações.

Meu objeto de saída seria um data frame com três colunas, uma com as carnes calculadas para o blend (3 tipos) outra com a quantidade de cada uma delas em grama e a última com o preço.

A minha ideia das mensagens é que caso o usuário coloque no argumento porcentagem de gordura valores diferentes de 15 ou 20% de gordura, apareceria uma das mensagens de erro.

Quanto ao comentário da Mali sobre eu colocar um gráfico que represente o resultado, posso tentar incluir sim, não vou prometer isso na proposta ainda, mas com certeza irei testar!

Não sei se consegui deixar mais claro! O que acham? —-*Valéria*

Olá Valéria. Está bem mais claro quais os argumentos e saída da função, mas você poderia esclarecer como vai usar esses argumentos exatamente. Quer dizer, como cada opção afetará o resultado?

Também acho que algumas dessas opções são desnecessariamente restritivas. Imagino que você poderia permitir um tamanho qualquer de hambúrguer (sem precisar restringir a três opções), um número qualquer de hambúrgueres (qual o sentido de limitar de um a dez?), e uma porcentagem de gordura decidida pelo usuário (sem limitar a duas opções – por exemplo, o usuário poderia querer 17% de gordura). Como esses valores são numéricos, colocar as restrições dá mais trabalho pra você e menos liberdade para o usuário.

A opção que mais precisa de explicações é a opção de preço. Como essa opção vai influenciar o resultado da função?

Sobre a tabela de carnes e preços ser interna à função, acho isso extremamente problemático. Isso torna a utilidade da sua função extremamente limitada, já que as informações nas quais ela se baseia serão limitadas. É importante que a sua base de dados seja atualizada. Você poderia escrever uma função que baixa essas informações de algum site da internet (em anos anteriores alguns alunos fizeram coisas parecidas), ou exigir esses dados como entrada. Esse é um ponto central da sua função, muito mais importante que a variedade de opções que você oferece ao usuário.

Eu gostaria que você explicasse como funciona esse cálculo do blend, especificando como você usa os argumentos pra influenciar o cálculo.

—-*Mali*

Help da função

```
growth               package: nenhum                R Documentation

Descrição:

Estima os parâmetros de crescimento a partir do ajuste de dois modelos não
lineares
a escolha do usuário (modelo de von Bertalanffy ou Gompertz). Retorna dois
gráficos
com curvas de crescimento ajustadas para machos e fêmeas e uma lista com os
valores
de R2 calculados para machos e fêmeas e o resultado da comparação das curvas
por meio
do teste de verossimilhança (Kimura, 1980).


Uso:

growth <- function(dados, modelo= "Bertalanffy", Linf=500, k=0.2, t0=0,
grafico= TRUE)


Argumentos:

dados:   data frame com três colunas contendo dados de idade na primeira
coluna, comprimento na segunda e sexo na terceira. As colunas devem ser
nessa ordem, independente ao nome que se dê a elas.
modelo:  modelo de crescimento “von Bertalanffy” ou “Gompertz” à escolha do
usuário.
Linf:    comprimento máximo teórico que pode ser atingido por uma espécie. O
default deste argumento é “500”.
k:       taxa constante de crescimento (velocidade com que o individuo
alcança Linf). O default deste argumento é “0.2”.
t0:      idade em que o indivíduo teria comprimento igual a zero. O default
```

```
deste argumento é “0”.
grafico: argumento lógico, por padrão TRUE, retorna a curva de crescimento
ajustada para ambos os sexos. Se FALSE, não retorna gráfico.

Detalhes:

O nome do objeto de entrada deve ser “dados”.
Se houver NA’s, estes serão omitidos.
Quando a janela gráfica for aberta, o usuário terá que indicar o local
desejado para inserir a legenda de cada gráfico.

Valores:

Retorna uma janela gráfica com as curvas de crescimento para machos e
fêmeas.
O objeto de saída é uma lista que contém um sumário dos parâmetros
calculados para machos e fêmeas, o valor de R2, os valores ajustados de cada
parâmetro e o resultado do teste de verossimilhança ajustado para cada
modelo.

Avisos:

Mensagens de erro serão geradas se o objeto de entrada não possuir três
colunas, se o modelo escolhido pelo usuário for diferente de “Bertalanffy”
ou “Gompertz” ou se os parâmetros de crescimento não forem caracteres
numéricos.

Nota:

Esta função requer instalação do pacote “qpcR”.

Autora:

Valéria Conversani
valconversani@usp.br

Referências:

Kimura, D. K. 1980. Likelihood methods for the von Bertalanffy growth curve.
U. S. Fish. Bull. 77(4): 765-776.

Exemplos:

Exemplo 1
idade <-
c(1,2,3.3,4.3,5.3,6.3,7.3,8.3,9.3,10,11.3,1,2,3.3,4.3,5.3,6.3,7.3,8.3,9.3,10
,11.3,12.3,13.3)
comp <-
c(15.4,26.9,42.2,44.6,47.6,49.7,50.9,52.3,54.8,56.4,55.9,15.4,28,41.2,46.2,4
8.2,50.3,51.8,54.3,57,58.9,59,60.9,61.8)
sexo <-
```

```
c("M","M","M","M","M","M","M","M","M","M","M","F","F","F","F","F","F","F","F
","F","F","F","F","F")
dados <- data.frame(idade, comp, sexo)

growth(dados, "Bertalanffy", 100, 0.2, 0)

Exemplo 2
age <- c(1, 10, 13, 2, 5, 7, 9, 12, 20, 21, 18, 1, 3, 18, 5, 10, 12)
lt <- c(100.4, 305.5, 398.6, 134.9, 265.7, 295.4, 301.3, 365.8, 406.3,
435.2, 398.5, 101.3, 143.6, 378.9, 245.2, 302.9, 334.2)
sex <- c("M", "F", "F", "M", "M", "F", "F", "M", "M", "F", "F", "F", "F",
"M", "F", "F", "F")
dados <- data.frame(age, lt, sex)

growth(dados, "Gompertz") #neste exemplo, os parâmetros são o default da
função
```

Código da função

```
growth <- function(dados, modelo = NULL, Linf=500, k=0.2, t0=0, grafico =
TRUE)
{

require(qpcR) #carrega o pacote "qpcR", partindo do pressuposto que o
usuário o tenha instalado
remove.na <- na.omit(dados) #omite de todos os NAs do objeto de entrada
  if (!is.data.frame(dados) && ncol!= 3) #cria uma condição que verifica se
o data frame tem três colunas
  {
    stop("'dados' tem que ser um data.frame de 3 colunas")
  }
    if (modelo!= "Bertalanffy" && modelo!= "Gompertz") #cria uma condição
que verifica se o usuário escolheu um modelo válido
   {
     stop("O modelo escolhido não é válido.\nSuas opções são: 'Bertalanffy'
ou 'Gompertz'")
    }
     if (!is.numeric(Linf) | !is.numeric(k) | !is.numeric(t0)) #cria uma
condição que verifica se o usuário indicou os valores dos parâmetros
     {
        stop("Os parâmetros de crescimento (Linf, k, t0) devem conter
caracteres numéricos.")
     }
t <- dados[,1] #cria um objeto que recebe os dados de idade
Lt <- dados[,2] #cria um objeto que recebe os dados de comprimento
sex <- dados[,3] #cria um objeto que recebe os dados de sexo

  if(modelo=="Bertalanffy") #se este for o modelo escolhido, ajusta os dados
de comprimento e idade
  {
```

```
    macho.b <- nls(Lt~Linf*(1-exp(-k*(t-
t0))),subset=sex=="M",start=list(Linf=500,k=0.2,t0=0)) #ajusta os dados de
comprimento e idade para machos
    sum.macho.b <- summary(macho.b) #sumário dos parâmetros estimados
    coef.macho.b <- Rsq(macho.b) #R2 ajustado pelo pacote qpcR
    names(coef.macho.b) <- "R2 machos"
    femea.b <-nls(Lt~Linf*(1-exp(-k*(t-
t0))),subset=sex=="F",start=list(Linf=500,k=0.2,t0=0)) #ajusta os dados de
comprimento e idade para fêmeas
    sum.femea.b <- summary(femea.b) #sumário dos parâmetros estimados
    coef.femea.b <- Rsq(femea.b) #R2 ajustado pelo pacote qpcR
    names(coef.femea.b) <- "R2 femeas"
    if (grafico==TRUE)
    {
      x11() #abre uma nova janela gráfica
      par(mfrow=c(1,2)) #divide a janela gráfica em uma linha, duas colunas
      plot.macho.b <- plot(Lt~t,xlab="idade",ylab="comprimento total",
xlim=range(0,max(t)),ylim=range(0,max(Lt)),cex.lab=1.2,main="Machos",cex.mai
n=1.2) #plota o gráfico para machos
      curve(coef(macho.b)[1]*(1-exp(-coef(macho.b)[2]*(x-
coef(macho.b)[3]))),add=T,col="blue") #desenha a curva
      legend(locator(1),bty="n",legend=substitute(L[i]==Linf%*%"["*1-e^{-
k%*%(t-t0)}*"]",
list(Linf=round(coef(macho.b)[1],1),k=round(coef(macho.b)[2],2),t0=-
round(coef(macho.b)[3],2))),cex=1.5) # insere a legenda (deve-se clicar no
gráfico para indicar o local desejado)
      plot.femea.b <- plot(Lt~t,xlab="idade",ylab="comprimento total",
xlim=range(0,max(t)),ylim=range(0,max(Lt)),cex.lab=1.2,main="Fêmeas",cex.mai
n=1.2) #plota o gráfico para fêmeas
      curve(coef(femea.b)[1]*(1-exp(-coef(femea.b)[2]*(x-
coef(femea.b)[3]))),add=T,col="red") #desenha a curva
      legend(locator(1),bty="n",legend=substitute(L[i]==Linf%*%"["*1-e^{-
k%*%(t-t0)}*"]",
list(Linf=round(coef(femea.b)[1],1),k=round(coef(femea.b)[2],2),t0=-
round(coef(femea.b)[3],2))),cex=1.5) # insere a legenda (deve-se clicar no
gráfico para indicar o local desejado)
    }
    resulta <- list(sum.macho.b, coef.macho.b, sum.femea.b, coef.femea.b)
#cria uma lista para guardar os resultados gerados no modelo
# ========Compara curvas de machos e fêmeas por verossimilhança========
# =======Adaptado de Kimura 1980 para o modelo de von Bertalanffy======
# número médio de observações
par.med <-
mean(c(nrow(subset(dados,sex=="M")),nrow(subset(dados,sex=="F"))))
# cria objetos com os parâmetros das curvas de machos e fêmeas
par.macho.b <- c(coef(macho.b),deviance(macho.b))
names(par.macho.b)<- c("Linf","k","t0","SSQ")
par.femea.b <- c(coef(femea.b),deviance(femea.b))
names(par.femea.b)<- c("Linf","k","t0","SSQ")
# calcula a soma dos quadrados residual total
par.SSQ <- par.macho.b[c("SSQ")]+ par.femea.b[c("SSQ")]
```

```
# ajuste do modelo com Linf comum
b.Linf <- nls(Lt ~ Linf *(1-exp(-ifelse(sex=="M",kM,kF)*(age-
ifelse(sex=="M",t0M,t0F)))),start=list(Linf=739,kM=0.2362,kF=0.1359,t0M=2E-0
1,t0F=1E-01))
par.Linf <- c(coef(b.Linf),deviance(b.Linf))
names(par.Linf)<- c("Linf","kM","kF","t0M","t0F","SSQ")
# ajuste do modelo com k comum
b.k <- nls(Lt ~ ifelse(sex=="M",LiM,LiF)*(1-exp(-k* (age-
ifelse(sex=="M",t0M,t0F)))),start=list(LiM=555.15,LiF=732.35,k=0.1359,t0M=1E
-01,t0F=1E-01))
par.k <- c(coef(b.k),deviance(b.k))
names(par.k)<- c("LinfM","LinfF","k","t0M","t0F","SSQ")
# ajuste do modelo com t0 comum
b.t0 <- nls(Lt ~ ifelse(sex=="M",LiM,LiF)*(1-exp(-ifelse(sex=="M",kM,kF)*
(age-t0))),start=list(LiM=555.15,LiF=732,kM=0.2362,kF=0.1359,t0=1E-01))
par.t0 <- c(coef(b.t0),deviance(b.t0))
names(par.t0)<- c("LinfM","LinfF","kM","kF","t0","SSQ")

par.total <- list(par.Linf, par.k, par.t0)

# teste Chi2 (Qui quadrado) e p-valor para cada parâmetro
par.ChiLinf <- -2*log((par.SSQ/par.Linf[c("SSQ")])^par.med)
names(par.ChiLinf) <- "Chi2 Linf"
p.chiLinf <- 1-pchisq(par.ChiLinf, 1) # p-valor
names(p.chiLinf) <- "p-valor Linf"

par.Chik <- -2*log((par.SSQ/par.k[c("SSQ")])^par.med)
names(par.Chik) <- "Chi2 k"
p.chik <- 1-pchisq(par.Chik, 1) # p-valor
names(p.chik) <- "p-valor k"
par.Chit0 <- -2*log((par.SSQ/par.t0[c("SSQ")])^par.med)
names(par.Chit0) <- "Chi2 t0"
p.chit0 <- 1-pchisq(par.Chit0, 1) # p-valor
names(p.chit0) <- "p-valor t0"
teste.vero <- c(par.ChiLinf,p.chiLinf,par.Chik,p.chik,par.Chit0,p.chit0)
return(list(resulta, par.total, teste.vero))
  }
  if (modelo=="Gompertz") #se este for o modelo escolhido, ajusta os dados
de comprimento e idade
  {
    macho.g <- nls(Lt~Linf*exp(-exp(-k*(t-
t0))),subset=sex=="M",start=list(Linf=500,k=0.2,t0=0)) #ajusta os dados de
comprimento e idade para machos
    sum.macho.g <- summary(macho.g)
    coef.macho.g <- Rsq(macho.g) #R2 ajustado pelo pacote qpcR
    names(coef.macho.g) <- "R2 machos"
    femea.g <- nls(Lt~Linf*exp(-exp(-k*(t-
t0))),subset=sex=="F",start=list(Linf=500,k=0.2,t0=0)) #ajusta os dados de
comprimento e idade para fêmeas
    sum.femea.g <- summary(femea.g)
```

```
    coef.femea.g <- Rsq(femea.g) #R2 ajustado pelo pacote qpcR
    names(coef.femea.g) <- "R2 femeas"
    if (grafico==TRUE)
    {
      x11() #abre uma nova janela gráfica
      par(mfrow=c(1,2))
      plot.macho.g <- plot(Lt~t,xlab="idade",ylab="comprimento total",
xlim=range(0,max(t)),ylim=range(0,max(Lt)),cex.lab=1.2,main="Machos",cex.mai
n=1.2)
      curve(coef(macho.g)[1]*(1-exp(-coef(macho.g)[2]*(x-
coef(macho.g)[3]))),add=T,col="blue")
      legend(locator(1),bty="n",legend=substitute(L[i]==Linf%*%e*"["*1-e^{-
k%*%(t-t0)}*"]",
list(Linf=round(coef(macho.g)[1],1),k=round(coef(macho.g)[2],2),t0=-
round(coef(macho.g)[3],2))),cex=1.5)
      plot.femea.b <- plot(Lt~t,xlab="idade",ylab="comprimento total",
xlim=range(0,max(t)),ylim=range(0,max(Lt)),cex.lab=1.2,main="Fêmeas",cex.mai
n=1.2)
      curve(coef(femea.g)[1]*(1-exp(-coef(femea.g)[2]*(x-
coef(femea.g)[3]))),add=T,col="red")
      legend(locator(1),bty="n",legend=substitute(L[i]==Linf%*%e*"["*1-e^{-
k%*%(t-t0)}*"]",
list(Linf=round(coef(femea.g)[1],1),k=round(coef(femea.g)[2],2),t0=-
round(coef(femea.g)[3],2))),cex=1.5)
    }
    resulta <- list(sum.macho.g, coef.macho.g, sum.femea.g, coef.femea.g)
#cria uma lista para guardar os resultados gerados no modelo
# ========Compara curvas de machos e fêmeas por verossimilhança========
# ==========Adaptado de Kimura 1980 para o modelo de Gompertz==========
# número médio de observações
par.med <-
mean(c(nrow(subset(dados,sex=="M")),nrow(subset(dados,sex=="F"))))
# cria objetos com os parâmetros das curvas de machos e fêmeas
par.macho.g <- c(coef(macho.g),deviance(macho.g))
names(par.macho.g)<- c("Linf","k","t0","SSQ")
par.femea.g <- c(coef(femea.g),deviance(femea.g))
names(par.femea.g)<- c("Linf","k","t0","SSQ")
# calcula a soma dos quadrados residual total
par.SSQ <- par.macho.g[c("SSQ")]+ par.femea.g[c("SSQ")]
# ajuste do modelo com Linf comum
g.Linf <- nls(Lt ~ Linf *exp(-exp(-ifelse(sex=="M",kM,kF)*(age-
ifelse(sex=="M",t0M,t0F)))),start=list(Linf=700,kM=0.2,kF=0.1,t0M=2E-01,t0F=
1E-01))
par.Linf <- c(coef(g.Linf),deviance(g.Linf))
names(par.Linf)<- c("Linf","kM","kF","t0M","t0F","SSQ")
# ajuste do modelo com k comum
g.k <- nls(Lt ~ ifelse(sex=="M",LiM,LiF)*(1-exp(-k*(age-
ifelse(sex=="M",t0M,t0F)))),start=list(LiM=500,LiF=500,k=0.1,t0M=1E-01,t0F=1
E-01))
par.k <- c(coef(g.k),deviance(g.k))
names(par.k)<- c("LinfM","LinfF","k","t0M","t0F","SSQ")
```

```
# ajuste do modelo com to comum
g.t0 <- nls(Lt ~ ifelse(sex=="M",LiM,LiF)*(1-exp(-
ifelse(sex=="M",kM,kF)*(age-
t0))),start=list(LiM=500,LiF=500,kM=0.2,kF=0.1,t0=1E-01))
par.t0 <- c(coef(g.t0),deviance(g.t0))
names(par.t0)<- c("LinfM","LinfF","kM","kF","t0","SSQ")

par.total <- list(par.Linf, par.k, par.t0)
# teste Chi2 (Qui quadrado) e p-valor para cada parâmetro
par.ChiLinf <- -2*log((par.SSQ/par.Linf[c("SSQ")])^par.med)
names(par.ChiLinf) <- "Chi2 Linf"
p.chiLinf <- 1-pchisq(par.ChiLinf, 1) # p-valor
names(p.chiLinf) <- "p-valor Linf"

par.Chik <- -2*log((par.SSQ/par.k[c("SSQ")])^par.med)
names(par.Chik) <- "Chi2 k"
p.chik <- 1-pchisq(par.Chik, 1) # p-valor
names(p.chik) <- "p-valor k"

par.Chit0 <- -2*log((par.SSQ/par.t0[c("SSQ")])^par.med)
names(par.Chit0) <- "Chi2 t0"
p.chit0 <- 1-pchisq(par.Chit0, 1) # p-valor
names(p.chit0) <- "p-valor t0"

teste.vero <- c(par.ChiLinf,p.chiLinf,par.Chik,p.chik,par.Chit0,p.chit0)
return(list(resulta, par.total, teste.vero))
  }
}

### Fim da função ###
```

Arquivos da função growth()

Arquivo da Função: functiongrowth

Help da Função: helpgrowth