Bruna de Oliveira Cassettari

Doutoranda no programa de pós-graduação em Ciências (Fisiologia Geral) do IB-USP. Membro do laboratório de Comportamento e Fisiologia Evolutiva, com interesse em pesquisas em ecofisiologia de anfíbios anuros.

Meus exercícios

Link para exec

Proposta de Trabalho Final

Principal

Função: aposentR

A função pretende simular e comparar datas previstas de aposentadoria de acordo com as regras vigentes e pós-reforma da previdência que está sendo discutida no Brasil (considerando o formato em que está atualmente, antes de votação no plenário da Câmara e Senado). A função irá considerar as regras gerais, para contribuintes do INSS (contribuição contínua desde a data de início), incluindo casos de transição.

Argumentos:

- sexo: vetor de caracteres ou fator (F ou M)
- in.contr: vetor de datas no formato dd/mm/aaaa, contendo a data de início da contribuição
- nasc: vetor de datas no formato dd/mm/aaaa, contendo a data de nascimento
- porc: porcentagem a ser definida pelo usuário para saber a data em que poderá aposentar, segundo a proposta de reforma atual, recebendo X% da média de salários de contribuição (X = 70 a 100%; default=100%)

Entrada: data.frame com colunas: sexo, data de início de contribuição, data de nascimento

Saída: A função retornará quatro colunas adicionadas ao data.frame inicial:

- (I) Data mínima para aposentadoria (com 70% da média de salários de contribuição), segundo a regra vigente;
- (II) Data prevista para aposentadoria com X% da média de salários de contribuição, segundo a regra vigente;
- (III) Data mínima prevista para aposentadoria (com 70% da média de salários de contribuição), de acordo com a proposta de reforma e
- (IV) Data prevista para aposentadoria com X% da média de salários de contribuição, de acordo com a proposta de reforma

Olá Bruna,

Muito legal essa sua proposta, acho que vai dar pra gente se deprimir simulando esses resultados...rsrsrs. No argumento 'porc', você permitirá que o usuário estabeleça qualquer valor contínuo entre 70 e 100%? Outra coisa, seria legal que a função testasse logo de cara se os vetores de data são de classe 'Date' mesmo, e parasse se não forem. Seria interessante retornar uma coluna com a idade do contribuinte em cada uma das datas estimadas, tanto para a regra vigente quanto para a proposta pela reforma? Por fim, que tal calcular as diferenças entre as datas calculadas pela reforma e pela regra vigente para cada contribuinte? Com esse último o impacto da reforma poderia ser enxergado mais fácil (e triste), né? Eu aconselharia você continuar com essa proposta. — *Gustavo Agudelo 2017/06/02 19:38*

Oi, Gustavo! Muito obrigada pelas dicas! Como conversamos, vou continuar com essa proposta. Quanto ao argumento porc, eu penso que deve se restringir a valores inteiros ou de x.5%, uma vez que a base de cálculos da proposta é feita contando 1, 1.5, 2 ou 2.5%, dependendodo caso. Gostei das sugestões de incluir as idades e as diferenças entre as datas e incluí isso na proposta! Como também conversamos, decidi restringir um pouco mais proposta: apenas para casos de aposentadoria por idade pela regra vigente e não incluir casos de transição da PEC. Isso porque os casos de transição se aplicam principalmente a pessoas que poderiam se aposentar por tempo de contribuição pela regra vigente. Com isso, penso que aumentaria muito a complexidade a ser incluída em uma única função, uma vez que teriam que ser incluídos os cálculos para a aposentadoria por tempo de contribuição, além dos cálculos da transição. Estou colocando a proposta com ajustes e o que pensei até agora como pseudo-código. Estou pensando ainda sobre como vou calcular alguns objetos, como do X.Ref por exemplo. Obrigada :) — *Bruna*

Proposta de função com ajustes

Função: aposentR

A função pretende simular e comparar datas previstas de aposentadoria por idade para trabalhadores urbanos vinculados ao INSS, de acordo com as regras vigentes e pós-reforma da previdência que está sendo discutida no Brasil (PEC 287/16)*.

* A PEC ainda está em discussão e, portanto, até a aprovação e sanção da lei, os termos das novas regras podem mudar. Será utilizado para o desenvolvimento da função, o texto base aprovado com as alterações realizadas até Maio/2017 (Inclusa proposta do relator da reforma da Previdência, deputado Arthur Oliveira Maia)

Argumentos:

- sexo: vetor de caracteres ou fator (H ou M)
- in.contr: vetor de datas no formato dd/mm/aaaa, contendo a data de início da contribuição
- nasc: vetor de datas no formato dd/mm/aaaa, contendo a data de nascimento
- porc: porcentagem a ser definida pelo usuário para saber a data em que poderá aposentar, segundo a proposta de reforma atual, recebendo X% da média de salários de contribuição (X = 70 a 100%; default=100%)

Entrada: data.frame com colunas: sexo, data de início de contribuição, data de nascimento

Saída: A função retornará as seguintes colunas, adicionadas ao data.frame inicial:

- 1. Data mínima para aposentadoria por idade, segundo a regra vigente;
- 2. Idade do contribuinte em 1
- 3. Data prevista para aposentadoria com X% do salário de benefício (SB), segundo a regra vigente;
- 4. Idade do contribuinte em 3
- 5. Data mínima prevista para aposentadoria, de acordo com a proposta de reforma
- 6. Idade do contribuinte em 5
- 7. Data prevista para aposentadoria com X% do SB, de acordo com a proposta de reforma
- 8. Idade do contribuinte em 7
- 9. Diferença de tempo entre cenários mínimos
- 10. Diferença de tempo entre os cenários melhores (X)

Referências

Pseudocódigo I

> Oi Bruna,
>
> Achei muito legal seu pseudo-código, muito instrutivo e dá para ver que você entende de fato o que está querendo fazer. Fiquei pensando em outra restrição da sua função, e é que ela calcularia as datas e idade de aposentadoria do contribuinte (para ambas as normativas) assumindo que el@ não parou de contribuir desde o início, né? Obviamente é necessário restringir a função porque não dá para incluir todos os cenários possíveis, mas deixa o mais claro possível o tipo de simulação que você irá fazer. Qualquer dúvida, é só escrever. Abraço, — *Gustavo Agudelo 2017/06/09 12:38*

Plano B

A função pretende facilitar a obtenção de resultados de capacidade antimicrobiana a partir de leituras de densidade óptica (DO). Em geral, ensaios deste tipo são realizados em placas de ELISA, contendo controles positivo e negativo e amostras testadas em duplicatas ou triplicatas. Em geral são feitas várias leituras da mesma placa, e costuma-se considerar para análises posteriores os dados obtidos no momento de crescimento exponencial do microorganismo contra o qual as amostras estão sendo testadas. Apesar de a função pretender ser útil a experimentos de avaliação de capacidade antibiótica, poderia ser utilizada para outros fins, já que se trata de um protocolo comum em ciências biológicas. Os objetos de entrada seriam uma ou mais matrizes com as DOs obtidas. Os objetos de saída seriam vetores com a capacidade antibiótica de cada amostra (%) e possivelmente uma figura da curva de crescimento bacteriano do controle positivo. Os argumentos seriam:

- x = Objetos que podem ser uma ou mais matrizes ou data frames contendo os dados de DO em cada momento
- rep = indica se o ensaio foi realizado com amostras em duplicata ou triplicata (“2” ou“3”)
- cont.pos = indexação da posição do controle positivo no objeto com os valores de DO
- cont.neg = indexação da posição do controle negativo no objeto com os valores de DO
- plot = argumento lógico que indica se deve ser incluído o gráfico de crescimento bacteriano (TRUE / FALSE)

O cálculo de capacidade antimicrobiana de cada amostra é: (1 - (DO média de cada amostra / DO do controle positivo)). A curva de crescimento deve ser feita utilizando as DOs médias do controle positivo em cada leitura. Referência: Assis et al. 2013. DOI: http://dx.doi.org/10.2994/SAJH-D-13-00007.1

> Bruna, essa proposta também está interessante. Talvez seria mais útil se o objeto de entrada dos dados fosse um df no qual você pudesse adicionar as colunas dos resultados, no caso de serem do mesmo comprimento. Não ficou claro como seria o gráfico, e se consideraria a identidade de cada amostra.— *Gustavo Agudelo 2017/06/02 19:54*

Trabalho Final

Help

```
aposentR              package:unknown                R Documentation

Função para calcular e comparar datas de aposentadoria por idade.

Description:

    Simula e compara datas previstas de aposentadoria por idade  para
```

```
trabalhadores urbanos vinculados ao INSS, de acordo com as regras vigentes e
pós-reforma da previdência que está sendo discutida no Brasil (PEC 287/16).

Usage:

    aposentR(sexo="F", nasc, in.contr, porc=100, result="dataframe")

Arguments:

 sexo: objeto de caracter ou fator, 'F' (Feminino) ou 'm' (mulher), 'M'
(Masculino) ou 'h'(homem).
 nasc: objeto de classe Date, POSIXlt ou POSIXct, no formato dd/mm/aaaa,
contendo a data de nascimento
 in.contr: objeto de classe Date, POSIXlt ou POSIXct, no formato dd/mm/aaaa,
contendo a data de início da contribuição
 porc: objeto numérico, entre 70 e 100 e múltiplo de 0.5, contem a
porcentagem do salário benefício (SB) para o qual deseja saber a data de
aposentadoria,
segundo a proposta de reforma (default=100)

Details:

     Se porc=100, também será calculada data estimada para aposentadoria por
tempo de contribuição pela regra vigente (regra 85/95).
     A função pressupões contribuição ininterrupta a partir de data de
início.

Value:

    Retorna um dataframe ou uma lista com os valores correspondentes a:
    Data mínima para aposentadoria por idade, segundo a regra vigente; Idade
do contribuinte nesta data;
    Data prevista para aposentadoria com porc% do SB, segundo a regra
vigente; Idade do contribuinte nesta data;
    Data mínima prevista para aposentadoria, de acordo com a proposta de
reforma; Idade do contribuinte nesta data;
    Data prevista para aposentadoria com porc% do SB, de acordo com a
proposta de reforma; Idade do contribuinte nesta data;
    Se porc=100: Data prevista para aposentadoria por tempo de contribuição
pela regra vigente; Idade do contribuinte nesta data;
    Diferença de tempo entre cenários mínimos; Diferença de tempo entre os
cenários melhores; Diferença de tempo entre cenário por tempo de
contribuição e pós-reforma
Warning:
     A função para se dados de entrada não forem das classes determinadas.
     Se sexo for nulo ou diferente de 'F', 'm', 'M' ou 'h', será considerado
feminino.

Note:

     A PEC ainda está em discussão e, portanto, até a aprovação e sanção da
```

```
lei, os termos das novas regras podem mudar. A função considera o texto base
aprovado com as alterações realizadas até Maio/2017 (Inclusa proposta do
relator da reforma da Previdência, deputado Arthur Oliveira Maia).

Author(s):

     Bruna de Oliveira Cassettari
     brucassettari@gmail.com

References:

     Brasil. Lei Nº 8.213, 24 de Julho de 1991.
     Brasil. PEC 287/2016. Disponível em:
http://www.camara.gov.br/proposicoesWeb/fichadetramitacao?idProposicao=21198
81. Acesso em Junho de 2017.

Examples:
  aposentR(sexo="M", nasc=as.Date("20/03/1990", format="%d/%m/%Y"),
in.contr= as.Date("02/08/2020", format="%d/%m/%Y"), porc=75, result =
"lista")

  n <- as.Date("20/03/1990", format="%d/%m/%Y")
  c <- as.Date("02/08/2015", format="%d/%m/%Y")
  aposentR(nasc=n, in.contr=c)
```

Código da Função

```
aposentR <- function (sexo="F", nasc, in.contr, porc=100,
result="dataframe") ## Iniciar função com argumentos de entrada que serão
necessários
{
  if ((class (nasc) != "Date")  & (class (nasc) != "POSIXlt") & (class
(nasc) != "POSIXt"))
  {stop("nasc deve ser data da classe Date ou POSIX,  no formato %d/%m/%Y")}
#verificar se dado de entrada de data de nascimento é data (classes Date ou
POSIX) e parar em caso negativo
  if ((class (nasc) == "POSIXlt") |  (class (nasc) == "POSIXt"))
  {nasc <- as.Date(nasc, format="%d/%m/%Y")}  #Transforma data da classe
POSIXlt ou POSIXt em classe "Date" no formato dd/mm/aaaa para garantir que
cálculos posteriores serão feitos corretamente
  if ((class (in.contr) != "Date")  & (class (in.contr) != "POSIXlt") &
(class (in.contr) != "POSIXt"))
  {stop("in.contr deve ser data ou POSIX  no formato %d/%m/%Y")} #verificar
se dado de entrada de data de início de contribuição é data (classes Date ou
POSIX) e parar em caso negativo
  if ((class (in.contr) == "POSIXlt") |  (class (in.contr) == "POSIXt"))
  {in.contr <- as.Date(in.contr, format="%d/%m/%Y")} #Transforma data da
```

```
classe POSIXlt ou POSIXt em classe "Date" no formato dd/mm/aaaa para
garantir que cálculos posteriores serão feitos corretamente
  if   (( class(sexo) != "character") & (class(sexo) != "factor" ))
  {stop("classe de sexo deve ser caracter ou fator")} #verificar se dado de
entrade de sexo é caracter ou fator e parar em caso negativo
  sexo <- as.character(sexo) #define classe de sexo como caracter
  if(((sexo != "h") & (sexo != "m") & (sexo != "F") & (sexo != "M")))
  {warning("sexo deve ser 'F' (Feminino), 'M' (Masculino), 'h'(homem) ou 'm'
(mulher). Sexo feminino por default.") }  #verificar se dado de entrada de
sexo é "h" (homem) ou "m" (mulher)  ou "F" (feminino) ou "M" (masculino) e
dar mensagem de alerta em caso negativo
  pa <- seq(from=70, to=100, by=0.5) # criar objeto com sequencia de números
entre 70 e 100, a cada 0.5, para ser usada para controle do argumento porc
  if (sum(!(porc %in% pa)) > 0)
  {stop("porcentagem deve ser valor entre 70 e 100 e ser múltiplo de 0.5")}
# teste para saber se os valor de porc incluído pelo usuário está entre 70 e
100 e se é número inteiro ou múltiplo de 0.5. Parar em caso negativo.
  data <- as.Date(seq(as.Date("01/01/1900", format="%d/%m/%Y"),
as.Date("31/12/2200", format="%d/%m/%Y"), by="day")) #cria vetor de muitas
datas hipotéticas para serem utilizadas nos cálculos seguintes
  idade <- floor(as.numeric(difftime(data[data>nasc], nasc,
units="days")/365.25)) #calcula idade, em anos, para cada data hipotética a
partir da data de nascimento
  temp.contr <-  floor(as.numeric(difftime(data[data>nasc], in.contr,
units="days")/365.25)) #calcula tempo de contribuição, em anos, para cada
data hipotética do vetor data, a partir da data de nascimento (para garantir
vetor do mesmo tamanho do vetor idade)
  v1 <- data.frame(data[data>nasc], idade, temp.contr) #data frame com os 2
vetores criados
  if(sexo == "h" | sexo == "M") #se sexo = homem ou masculino
  {
    v.MA <-   as.Date(v1$data [v1$idade >= 65 & v1$temp.contr >= 15],
format="%d/%m/%Y") # vetor de datas em que idade é maior que 65 anos e tempo
de contribuição é maior que 15 anos
    Min.Atual <- v.MA[1] # Primeira data do vetor V.MA = Data mínima para
aposentadoria pela regra vigente
    Id.MA <- (v1$idade[v1$data == Min.Atual]) #Idade em Min.Atual
    v.MR <-   v1$data [v1$idade >= 65 & v1$temp.contr >= 25]  # vetor de
datas em que idade é maior que 65 anos e tempo de contribuição é maior que
25 anos
    Min.Ref <- v.MR[1] # Primeira data do vetor V.MR = Data mínima para
aposentadoria pela proposta de reforma
    Id.MR <- (v1$idade[v1$data == Min.Ref] ) #Idade em Min.Ref
  }
  else #se sexo = mulher ou feminino ou outro caracter { cálculos de
Min.Atual e Id.MA}
  {
    v.MA <-   v1$data [v1$idade >= 60 & v1$temp.contr >= 15] # vetor de
datas em que idade é maior que 60 anos e tempo de contribuição é maior que
15 anos
    Min.Atual <- v.MA[1]  # Primeira data do vetor V.MA = Data mínima para
```

```
aposentadoria pela regra vigente
    Id.MA <- (v1$idade[v1$data == Min.Atual]) #Idade em Min.Atual
    v.MR <-   v1$data [v1$idade >= 62 & v1$temp.contr >= 25] # vetor de
datas em que idade é maior que 62 anos e tempo de contribuição é maior que
25 anos
    Min.Ref <- v.MR[1] # Primeira data do vetor V.MR = Data mínima para
aposentadoria pela proposta de reforma
    Id.MR <- (v1$idade[v1$data == Min.Ref]) #Idade em Min.Ref
  }
  ## criar data frame com um vetor com datas: 30 anos a partir do Min.Atual
e um vetor com porcentagens correspondente a cada ano (70 a 100)
  data.v <- seq(as.Date(Min.Atual, format="%d/%m/%Y"), by="month",
length=372) #cria vetor de datas a partir de Min.Atual
  porc.v <- rep(70:100, each=12) #cria vetor com percentuais correspondentes
a cada ano
  v2<- data.frame(data.v, porc.v)
  #criar objeto com data em que porcentagem seja aquela definida pelo
usuário
  x.a <- v2$data.v[v2$porc.v == floor(porc)] # cria vetor de datas a partir
de porc
  X.Atual <- x.a[1]  # primeira posição de x.a corresponde a X.Atual
  #adicionar mensagem se porc for x.5% pois pela regra vigente aumenta 1pp
do SB a cada ano
  p.5 <- seq(from=70.5, to=99.5, by=1) #cria sequencia de 70.5 a 99.5, de
meio em meio
  if(porc %in% p.5)  #mensagem se porc não estiver contido em p.5
  {message("Porcentagem arredondada para menor valor inteiro para cálculo de
data \n de aposentadoria pelas regras vigentes")}
  Id.XA <- floor(as.numeric(difftime(X.Atual, nasc, units="days")/365.25))
#idade em X.Atual
  data.r <- seq(as.Date(Min.Ref, format="%d/%m/%Y"), by="month", length=192)
#cria vetor de datas com 16 anos a partir do Min.Ref
  porc.r <- rep((c(seq(70, 77.5, by=1.5), seq(79.5, 87.5, by=2), seq(90,
100, by=2.5))), each=12) #cria vetor com porcentagens de SB correspondentes
a cada mês do vetor data.r
  if (!(porc %in% porc.r))
  { cat(paste("X.Ref retornará data em que será posível aposentar por idade
recebendo, no mínimo,", porc, "% do SB")) } #retorna mensagem caso a
porcentagem escolhida pelo usuário não esteja contida em porc.r. Neste caso
porc será considerado como mínimo.
  v3 <- data.frame(data.r, porc.r) #cria data frame com vetores de datas e
porcentagens de SB correspondentes
  x.r <- v3$data.r[v3$porc.r >= porc] # vetor de datas em que porcentagem de
SB é maior ou igual porc
  X.Ref  <- x.r[1] #retorna primeira data em porc.r >= porc
  Id.XR <- floor(as.numeric(difftime(X.Ref, nasc, units="days")/365.25)) #
Idade em X.Ref, em anos
  Dif.Min <- Id.MR - Id.MA  # cria objeto contendo valor de tempo, em anos,
correspondente à diferença entre cenários mínimos
  Dif.X <- Id.XR - Id.XA    # cria objeto contendo valor de tempo, em anos,
```

```
correspondente à diferença entre cenários X
  if (porc==100) #Se porc for 100%, calcula também data de aposentadoria por
tempo de contribuição pela regra vigente
  {
    v1$stc <- v1$idade + v1$temp.contr #cria vetor com soma de idade e tempo
de contribuição para cáluclo de pontos para aposentadoria integral por tempo
de contribuição
    if(sexo == "h" | sexo == "M")
    {   Min.tc1 <- as.Date(v1$data[v1$stc >= 95] , format= "%d/%m/%Y") #data
em que soma de idade e tempo de contribuição é igual ou maior que 95 pontos
    Min.tc <- Min.tc1[1]  #data para aposentadoria integral por tempo de
contribuição pela regra 85/95 vigente, se homem
    }
    else
    {  Min.tc2 <- as.Date(v1$data[v1$stc >= 85], format= "%d/%m/%Y")  #data
em que soma de idade e tempo de contribuição é igual ou maior que 85 pontos
    Min.tc <- Min.tc2[1]
    } #data para aposentadoria integral por tempo de contribuição pela regra
85/95 vigente, se mulher
    Id.tc <- v1$idade[Min.tc] #idade em Min.tc
    Dif.tc <- Id.XR - Id.tc #diferença, em anos entre cenários de
aposentadoria por tempo de contribuiçao atual e pós reforma
    result.l <- list(Min.Atual, Id.MA, Min.Ref, Id.MR, Dif.Min, X.Atual,
Id.XA, X.Ref, Id.XR, Dif.X, Min.tc, Id.tc, Dif.tc)
    names(result.l) <- c("Data mínima pela regra vigente", "Idade mínima
pela regra vigente", "Data mínima pós reforma", "Idade mínima pós reforma",
"Diferença em anos entre cenários mínimos","Data simulada para regra
vigente", "Idade simulada para regra vigente", "Data simulada pós reforma",
"Idade simulada pós reforma",  "Diferença em anos entre cenários simulados",
"Data para aposentadoria integral por tempo de contribuição pela regra
vigente", "Idade para aposentadoria integral por tempo de contribuição pela
regra vigente", "Diferença em anos entre cenários de aposentadoria por tempo
de contribuição e pós-reforma") #nomeia conteúdos da lista
    result.df <- data.frame(Min.Atual, Id.MA, Min.Ref, Id.MR, Dif.Min,
X.Atual, Id.XA, X.Ref, Id.XR, Dif.X, Min.tc, Id.tc, Dif.tc)
  }
  else
  {
    result.l <- list(Min.Atual, Id.MA, Min.Ref, Id.MR, Dif.Min, X.Atual,
Id.XA, X.Ref, Id.XR, Dif.X) # cria lista com os resultados relevantes
    names(result.l) <- c("Data mínima pela regra vigente", "Idade mínima
pela regra vigente", "Data mínima pós reforma", "Idade mínima pós reforma",
"Diferença em anos entre cenários mínimos","Data simulada para regra
vigente", "Idade simulada para regra vigente", "Data simulada pós reforma",
"Idade simulada pós reforma",  "Diferença em anos entre cenários simulados")
#nomeia conteúdos da lista
    result.df <- data.frame(Min.Atual, Id.MA, Min.Ref, Id.MR, Dif.Min,
X.Atual, Id.XA, X.Ref, Id.XR, Dif.X) # cria dataframe com os resultados
relevantes
  }
  if (result == "lista") { return(result.l) } #retorna lista de resultados
```

```
  else {return (result.df)} #retorna dataframe de resultados
}
```

Código da função