Patricia Marques Moralejo Bermudi

Graduada e mestranda em Saúde Pública, Faculdade de Saúde Pública, USP.

Projeto de pesquisa: Análise espacial e espaço-temporal dos óbitos por câncer de mama e de colo de útero, município de São Paulo, 2000 a 2016.

Orientador: Francisco Chiaravalloti Neto.

Co-orientadora: Alessandra Cristina Guedes Pellini

Meus exercícios

Link para acesso aos meus Exercícios.

Trabalho Final

Código da função: Padronização direta de taxas

```
## Função para Padronização direta de taxas

stRate  <-  function (df.event, df.pop, df.pop.st = NULL, mult = 100000,
                      crude.rt = TRUE) { #Atribui ao objeto "stRate" a
função
  #com os argumentos que estão listados entre virgulas, dentro dos
parênteses.
#### Realizando os controles de fluxo
  ## CONTROLE DE FLUXO: Os objetos de entrada df.event; df.pop; (e
df.pop.st,
  #se for o caso) são da classe data.frame?
  if (class(df.event) != "data.frame" | class(df.pop) != "data.frame") { #Se
a
    #classe do objeto 'df.event' for diferente de "data.frame" ou a classe
do
    #objeto "df.pop" for diferente da classe "data.frame".
    stop ("Os objetos 'df.event' e 'df.pop' devem possuir a classe
data.frame. Faça as correções necessárias e rode a função novamente.")}
  # Então, a função para e retorna uma mensagem de erro.
  if (!is.null(df.pop.st) & class(df.pop.st) != "data.frame") { # Se o
usuário
    #escolheu colocar um objeto no argumento 'df.pop.st', ou seja, se este
for
    #diferente de nulo no argumento da função  "E" a classe do objeto
inserido
```

```
    #pelo usuário para esse argumento for diferente de "data.frame."
    stop ("O objeto 'df.pop.st' deve possuir a classe data.frame. Faça as
correções necessárias e rode a função novamente.")} #Então, a
  #função para e retorna uma mensagem de erro.
  ## CONTROLE DE FLUXO: Os objetos de entrada df.event; df.pop; ( e
df.pop.st,
  #se for o caso) contém na posição [1] uma coluna de identificação? Se não,
a
  #função para e retorna uma mensagem ao usuário para correção.
  ANSWER_1 <- readline(prompt = " \n PERGUNTA 01/02: DIGITE SUA RESPOSTA
AQUI NO CONSOLE!
       \n A PRIMEIRA COLUNA EM TODOS OS SEUS OBJETOS DE ENTRADA
          CORRESPONDE À UMA COLUNA DE IDENTIFICAÇÃO (ID) ? \n
          O ID DEVE APRESENTAR A MESMA SEQUÊNCIA EM TODOS OS OBJETOS.
                      DIGITE s para SIM ou n para NÃO: s / n ? ") #Atribui
ao
  #objeto "ANSWER_1" a função 'readline' que irá exibir um prompt na tela do
  #console com uma pergunta que deverá ser respondida pelo usuário digitando
  #"s" para sim ou "n" para não.
  if(ANSWER_1 == "n") { #Se a resposta for igual a não
    stop ("Insira uma coluna de identificação na posição 1 [, 1] em todos os
objetos de entrada. Esta coluna não precisa ser numérica, mas deve
apresentar a mesma sequência entre todos os objeto de entrada. Faça as
correções necessárias e rode a função novamente.")} # Então, a função para e
retorna uma mensagem de erro.
  # Observação: como o campo de ID sempre é aconselhável estar presente em
um
  #banco de dados. A função foi construída para realizar as operações
  #considerando este ID e realizando as operações sempre a partir da coluna
de
  #posição 2. Desde modo, o usuário é notificado disso e, ao digitar sim
  #, assume a responsabilidade de que o não cumprimento deste critério
  #compromete os resultados finais da função.
  #Essa coluna de identificação não precisa ser numérica. Pode ser nome do
  #município, por exemplo.
  ## CONTROLE DE FLUXO: Os objetos de entrada df.event; df.pop; ( e
df.pop.st,
  #se for o caso) contém nas posições [2] em diante de colunas, variáveis
  #numéricas com as respectivas contagens? Se não, a função para e retorna
uma
  #mensagem ao usuário para correção.
  if(any (apply(df.event [, 2: ncol(df.event)], 2, is.character)) #Se
qualquer
     #observação for caractere em qualquer linha, entre a coluna  dois até a
     #última coluna (especificado por meio do n° total de colunas) do objeto
     #'df.event'
     | any (apply(df.pop[, 2: ncol(df.pop)], 2, is.character)) # OU se
qualquer
     #observação for caractere em qualquer linha, entre a coluna  dois até a
     #última coluna (especificado por meio do n° total de colunas) do objeto
```

```
     #'df.pop'
     | any (apply(df.event [, 2: ncol(df.event)], 2, is.logical)) #'OU' se
     #qualquer observação for lógica em qualquer linha, entre a coluna  dois
     #até a última coluna (especificado por meio do n° total de colunas) do
     #objeto 'df.event'
     | any (apply(df.pop[, 2: ncol(df.pop)], 2, is.logical))) { # OU se
    #qualquer observação for lógica em qualquer linha, entre a coluna  dois
até
    #a última coluna (especificado por meio do n° total de colunas) do
objeto
    #'df.pop'
    ( stop("Em todas as colunas (exceto a primeira, de identificação (ID))
dos objetos de entrada, as observações contidas devem ser números
(numéricas)! Faça as correções necessárias e rode a função novamente.")) } #
Então, a função para e retorna uma mensagem de erro.
  #Observação: o ideal seria utilizar "!is.numeric", porém a função apply
não
  #aceita esta opção.
  if (!is.null(df.pop.st)) { # Se o usuário escolheu colocar um objeto no
    #argumento 'df.pop.st', ou seja, se este for diferente de nulo no
argumento
    #da função
    if(any (apply(df.pop.st [, 2: ncol(df.pop.st)], 2, is.character)) # Se o
       #usuário escolheu colocar um objeto no argumento 'df.pop.st', ou
seja,
       #se este for diferente de nulo no argumento da função
       | any (apply(df.pop.st [, 2: ncol(df.pop.st)], 2,  is.logical))){ #OU
se
      #qualquer observação for lógica em qualquer linha, entre a coluna
dois
      #até a última coluna (especificado por meio do n° total de colunas) do
      #objeto 'df.pop.st'
    stop("Em todas as colunas (exceto a primeira, de identificação) do
objeto de entrada 'df.pop.st', as observações contidas devem ser números
(numéricas)! Faça as correções necessárias e rode a função novamente.") #
Então, a função para e retorna uma mensagem de erro.
  } }
  ## CONTROLE DE FLUXO: Os objetos de entrada df.event; df.pop; ( e
df.pop.st,
  #se for o caso) contém o mesmo número de linhas (locais ou ano) e colunas
  #(variáveis)? Se não, a função para e retorna uma mensagem ao usuário para
  #correção.
  if (ncol(df.event) != ncol(df.pop) | nrow(df.event) != nrow(df.pop)) { #Se
o
    #número de colunas do objeto 'df.event' for diferente do número de
colunas
    #do objeto 'df.pop' "OU" se o número de linhas do objeto 'df.event' for
    #diferente do número de linhas do objeto 'df.pop'
    stop ("Os objetos 'df.event' e 'df.pop' devem possuir o mesmo número de
linhas (localidade) e o mesmo número de colunas (id e colunas dos estratos
que serão utilizadas na padronização), as observações contidas devem ser
```

```
números (numéricas)! Faça as correções necessárias e rode a função
novamente.") # Então, a função para e retorna uma mensagem de erro.
  }
  if (!is.null(df.pop.st)) { #Se o usuário escolheu colocar um objeto no
    #argumento 'df.pop.st', ou seja, se este for diferente de nulo no
argumento
    #da função
    if (ncol(df.pop.st) != ncol(df.event) | nrow(df.pop.st) !=
nrow(df.event)){
      #Se o número de colunas do objeto 'df.pop.st' for diferente do número
de
      #colunas do objeto 'df.event' "OU" se o número de linhas do objeto
      #'df.pop.st' for diferente do número de linhas do objeto 'df.event'
    stop ("Os objetos 'df.pop.st', 'df.event' e 'df.pop' devem possuir o
mesmo
          número de linhas (localidade) e o mesmo número de colunas (id e
        colunas dos estratos que serão utilizadas na padronização). Faça as
correções necessárias e rode a função novamente.") #Então, a função
      #para e retorna uma mensagem de erro.
   } }
  ## CONTROLE DE FLUXO: Os objetos de entrada df.pop; ( e df.pop.st, se for
o
  #caso) contém o valor zero ou NA? Se sim, a função para e retorna uma
  #mensagem ao usuário para correção.
  if(any (apply(df.event [, 2: ncol(df.event)], 2, is.na)) #Se qualquer
     #observação for NA em qualquer linha, entre a coluna dois até a última
     #coluna (especificado por meio do n° total de colunas) do objeto
     #'df.event'
     | any (apply(df.pop [, 2: ncol(df.pop)], 2, is.na))){ # OU se qualquer
    #observação for NA em qualquer linha, entre a coluna dois até a última
    #coluna (especificado por meio do n° total de colunas) do objeto
'df.pop'
    stop("Os objetos de entrada 'de.event'e 'df.pop' NÃO devem possuir
nenhum valor NA. Faça as correções necessárias e rode a função novamente.")
# Então, a função para e retorna uma mensagem de
    #erro.
  } else if (any (df.pop [, 2:ncol(df.pop)] == 0)) { # Se não e se qualquer
    #observação for ZERO em qualquer linha, entre a coluna dois até a última
    #coluna (especificado por meio do n° total de colunas) do objeto
'df.pop'
    stop("O objeto 'df.pop' NÃO deve possuir nenhum valor zero. Faça as
observações contidas devem ser números (numéricas)! Faça as correções
necessárias e rode a função novamente.") # Então, a função para e retorna
    #uma mensagem de erro.
  }
  #Observação: A conferência para valores zero precisa ser posterior a
  #conferência de NAs para não ocorrer erro. Além disso, a conferência para
  #valores zero só é necessária para o denominador, ou seja, população, uma
vez
  #que não se pode dividir por zero.
```

```
 #Há diferentes maneiras de lidar com valores zero no banco, optou-se por
  #deixar o usuário escolher sua maneira, resolver fora da função e
  #posteriormente retornar a função.

  if (!is.null(df.pop.st)) { #Se o usuário escolheu colocar um objeto no
    #argumento 'df.pop.st', ou seja, se este for diferente de nulo no
argumento
    #da função
    if(any (apply(df.pop.st [, 2: ncol(df.pop.st)], 2, is.na))) { #Se
qualquer observação for NA em qualquer linha, entre a coluna dois até a
última coluna (especificado por meio do n° total de colunas) do objeto
'df.pop.st'
      stop("O objeto de entrada 'df.pop.st'NÃO deve possuir nenhum valor NA.
Faça as correções necessárias e rode a função novamente.")  #Então, a função
para e retorna uma mensagem de erro.
    } else if (any (df.pop.st[, 2:ncol(df.pop.st)] == 0)) #Se não e se
qualquer
      #observação for ZERO em qualquer linha, entre a coluna dois até a
última
      #coluna (especificado por meio do n° total de colunas) do objeto
      #'df.pop.st'
    stop("O objeto de entrada 'df.pop.st'NÃO deve possuir nenhum valor zero.
Faça as correções necessárias e rode a função novamente.") #Então, a função
para e retorna
    #uma mensagem de erro.
  }
  ## CONTROLE DE FLUXO: O usuário respondeu (por meio da função readline)
que
  #seus dataframes fornecidos NÃO possuem a mesma sequência em relação aos
  #extratos? Se não, a função para e retorna uma mensagem ao usuário para
  #correção.

ANSWER_2 <- readline(prompt = " \n PERGUNTA 02/02: \n
AS DEMAIS COLUNAS APRESENTAM A MESMA ORDEM (MESMA POSIÇÃO DE COLUNAS)
EM TODOS OS OBJETOS DE ENTRADA?\n
                     DIGITE s para SIM ou n para NÃO: s / n ? ") #Atribui ao
#objeto "ANSWER_2" a função 'readline' que irá exibir um prompt na tela do
#console com uma pergunta que deverá ser respondida pelo usuário digitando
"s"
#para sim ou "n" para não.
if(ANSWER_2 == "n") { #Se a resposta for igual a não
  stop ("Os objetos de entrada devem apresentar a mesma sequência em relação
aos estratos. Exemplo: Se você colocou a coluna referente a contagem pela
faixa etária de 00 a 14 anos de idades na posição 2 de coluna [, 2] no
objeto 'df.event', então, esta coluna deve ocupar a mesma posição, posição 2
de coluna [, 2] no objeto 'df.pop'. Faça as correções necessárias e rode a
função novamente.") #Então, a função para e retorna uma mensagem de erro.
}

#Observação: este controle evita que o usuário insira objetos com diferentes
#ordens das sequencias de estratos entre os mesmos. Assim, o usuário recebe
```

```
a
#informação que seus estratos precisam estar na mesma de sequência em
relação
#as colunas e digitando "sim", assume a responsabilidade, uma vez que ordens
#diferentes irão resultar em resultados errôneos.

  ## CONTROLE DE FLUXO: O usuário deixou a opção df.pop.st = NULL? Se sim,
#verifica se os objetos de entrada df.event; df.pop contém quatro colunas.
#Se sim, mostrar uma mensagem que será utilizada a população padrão da OMS.
#Se não, a função para e retorna uma mensagem ao usuário para correção.

if (is.null(df.pop.st) & ((ncol(df.event) != 4)  #Se o usuário deixou a
opção
    #'df.pop.st' como default , ou seja, igual a nulo
    #no argumento da função "E" o  número de colunas do objeto 'df.event'
for
    #diferente de 4
                          | is.null(df.pop.st) & (ncol(df.pop) != 4))) { #OU
se
  #o usuário escolheu colocar um objeto no argumento 'df.pop.st', ou seja,
se
  #este for diferente de nulo no argumento da função "E" o número de colunas
do
  #objeto 'df.pop' for diferente de 4
  stop ("O banco utilizado para a padronização será o da OMS, com um ID e
três colunas de estratos etários (estrato 1: 00 a 14 anos de idade; estrato
2: 15 a 59 anos de idade; e estrato 3: 60 anos e mais de idade). Nesta
ordem! Portanto, os objetos 'df.event' e 'df.pop' devem também ter quatro
colunas. A primeira correspondente ao ID e as demais com a mesma ordem de
estratos que o banco da OMS. Faça as correções necessárias e rode a função
novamente.") #Então, a função para e retorna uma mensagem de erro.
}

if (is.null(df.pop.st)) { #Se o usuário deixou a opção 'df.pop.st' como
default
  #ou seja, igual a nulo
  cat(" \n \t A população padrão que está sendo utilizada é a da OMS,
conforme consta no HELP \n \n")
  #Se não, é apresentado no console uma mensagem de que a população padrão
  #utilizada é a da OMS
}

#Observação: este controle é necessário, porque a opção default de
df.pop.st'
#apresenta 4 colunas. Logo, os objetos de entrada também devem apresentar
este
#número.

#### Construindo o banco de dados com a população padrão da OMS para ser
utilizada
```

```
#caso o usuário mantenha a opção default 'df.pop.st' = NULL.

if(is.null(df.pop.st)) { # Se o usuário escolheu manter a opção default para
  #'df.pop.st', ou seja, 'df.pop.st' = NULL
  df.pop.st <- data.frame(cod = df.event[1], #Então é atribuído ao objeto
        #'df.pop.st' um data.frame, com a primeira coluna correspondente a
        #'#primeira coluna do objeto 'df.event', com o nome "cod"
                          poppadr00_14 = rep(26139, nrow(df.event)), #A
segunda
        #coluna corresponde a repetição do valor da população padrão da OMS
de
        #00 a 14 anos de idade. O número de repetições é equivalente ao
número
        #de linhas do objeto 'df.event', com o nome da coluna correspondente
a
        #"poppadr00_14"
                          poppadr15_59 = rep(61920, nrow(df.event)), # A
terceira
        #coluna corresponde a repetição do valor da população padrão da OMS
de
        #15 a 59 anos de idade. O número de repetições é equivalente ao
número
        #de linhas do objeto 'df.event', com o nome da coluna correspondente
a
        #"poppadr15_59"
                          poppadr60_em = rep(11941, nrow(df.event))) } # A
quarta
        #coluna corresponde a repetição do valor da população padrão da OMS
de
        #60 anos e mais de idade. O número de repetições é equivalente ao
número
        #de linhas do objeto 'df.event', com o nome da coluna correspondente
a
        #poppadr60_em"

#### Criando uma matriz vazia para receber o valor de eventos que seriam
esperados
#caso a população observada correspondesse ao valor da população
padronizada.

ob_esp <- matrix(NA, ncol = ncol(df.event), nrow = nrow(df.event)) #Atribui
ao
#objeto "ob_esp" uma matriz com valores NA, com o número de colunas
correspondente
#ao número de colunas do objeto 'df.event' e com o número de linhas
correspondente
#ao número de linhas do objeto 'df.event'

#### Criando um ciclo para preencher os valores de eventos esperados.

#Regra de três para cada estrato: "Evento observado" está para "População
```

```
observada",
#assim como "Evento esperado" está para a "População padrão"

# "Evento esperado" é igual a multiplicação do "Evento observado" pela
#"População padrão" correspondente, dividido pela "População observada"
também
#correspondente

for(i in 1:nrow(df.event)){ # Para o contador i indo de 1 até o número de
linhas
  #do objeto 'df.event'
  for (j in 2:ncol(df.event)){ #Para o contador j indo de 2 até o número de
    #colunas do objeto 'df.event' ( a coluna 1 ficará vazia)
    ob_esp[i, j] <- (df.event[i, j] * df.pop.st[i, j]) / df.pop[i, j]
#Atribui
    #ao objeto 'ob_esp' na posição i de linha, que no primeiro ciclo é 1, e
na
    #posição j de coluna, que no primeiro ciclo é 2, a multiplicação do
valor do
    #objeto 'df.event', na mesma posição de linha e coluna, com o valor do
objeto
    #'df.pop.st', na mesma posição de linha e coluna, com o resultado
dividido
    #pelo valor do objeto 'df.pop' na mesma posição de linha e coluna
  }
}

ob_esp <- data.frame(ob_esp) # Faz a coerção do objeto "ob_esp" para
data.frame
#### Calculando a taxa padronizada

st_rate <- data.frame (cod = df.event[1], #Atribui ao objeto 'st_rate' um
                       #data.frame com a primeira coluna correspondente à
primeira
                       #coluna do objeto 'df.event', com o nome "cod"
            st_rt = round((apply(ob_esp [, 2:ncol(ob_esp)], 1, sum) * mult )
# A
                      #segunda coluna corresponde a soma, por linha, dos
valores
                      #do objeto 'ob_esp' entre a coluna 2 até a última
coluna
                      #(especificado por meio do n° total de colunas do
objeto
                      #''ob_esp'). Este resultado multiplicado pelo objeto
'mult'
            / apply(df.pop.st[, 2:ncol(df.pop.st)], 1, sum), 2)) # Este
resultado
                     #dividido pela soma, por linha, dos valores do objeto
                      #'df.pop.st' entre a coluna 2 até a última coluna
(especificado
```

```
                        #'#por meio do n° total de colunas do objeto
'df.pop.st').
                        #'#Este resultado arredondado para duas casas decimais

#### Calculando a taxa não ajustada (Bruta)

if(crude.rt ==  TRUE) { #  Se o usuário escolheu manter a opção default para
  #'crude.rt', ou seja, 'crude.rt' = TRUE
  crude.rt <- data.frame (cod = df.event[1], # Então é atribuído ao objeto
                          #"crude.rt" um data.frame com a primeira coluna
correspondente
                          #rimeira coluna do objeto 'df.event', com o nome
"cod"
      crude.rt  =  round((apply(df.event [, 2:ncol(df.event)], 1, sum) * # A
segunda
                            #coluna corresponde a soma, por linha, dos
valores do objeto
                            #'df.event' entre a coluna 2 até a última coluna
                            #(especificado por meio do n° total de colunas
                            #do objeto ''df.event).
                        mult ) / apply(df.pop [, 2:ncol(df.pop)], 1, sum),
2))# Este
  #resultado multiplicado pelo objeto 'mult'. Em seguida, o novo resultado é
dividido
  #pela soma, por linha, dos valores do objeto 'df.pop' entre a coluna 2 até
a última
  #coluna (especificado por meio do n° total de colunas do objeto 'df.pop').
  #Este resultado é arredondado para duas casas decimais
  exit <- cbind(crude.rt , st_rate[, -1]) #Atribui ao objeto "exit" a junção
por
  #colunas dos objetos 'crude.rt' e 'st_rate[, -1]' com exceção da primeira
coluna,
  #para que no banco não tenham dois campos de identificação
  names(exit) <- c("ID", "Taxa Bruta", "Taxa Padronizada") #Renomeia os
nomes
  #das colunas do objeto 'exit'
} else { # Se não, se o usuário selecionou a opção 'crude.rt' = FALSE
  exit <- st_rate # Então, atribui ao objeto 'exit' apenas a taxa
padronizada,
  #objeto 'st_rate'
  names(exit) <- c("ID",  "Taxa Padronizada") # Renomeia os nomes das
colunas
  #do objeto 'exit'
}

return (exit) # Retorna o objeto 'exit'
}
```

Texto de ajuda

```
stRate             package:unknown                     R Documentation

 ~~ Função para padronizar taxas pelo método direto~~

Description

 ~~ A função calcula as taxas não ajustadas e faz a padronização direta por
idade das taxas de um evento por 100 mil habitantes, utilizando como
população padrão a da Organização Mundial da Saúde. Também permite
padronizar por outras variáveis de interesse e utilizar outra população
padrão. ~~

Usage

~~ stRate(df.event, df.pop, df.pop.st = NULL, mult = 100000, crude.rt = T)~~

Arguments


~~''df.event''  Um data.frame com uma coluna de identificação na primeira
posição de coluna [, 1] e com a contagem de eventos observados por estratos,
nas demais posições de colunas. Se ''df.pop.st'' = NULL, os estratos de
''df.event'' deverão ser os mesmos que os contidos no argumento
''df.pop.st''. ~~

~~''df.pop''    Um data.frame com a mesma coluna de identificação do objeto
''df.event'', na primeira posição de coluna [, 1], e com a contagem de
populações observadas, pelos mesmos estratos e posições de coluna do objeto
''df.event''. ~~

~~''df.pop.st'' Um data.frame com a contagem de populações padrão com a
mesma coluna de identificação e mesma posição de estratos fornecidos no
objeto ''df.event''. Se ''df.pop.st'' = NULL, a contagem de populações
padrão será a da Organização Mundial da Saúde, com três estratos etários
(ver detalhes).~~

~~''mult''  Um vetor numérico utilizado como multiplicador para evitar
valores fracionados de difícil interpretação. ~~

~~''crude.rt''  Um vetor lógico para apresentação das taxas não ajustadas
(brutas). ~~

Details

~~ A coluna de identificação em todos os objetos de entrada pode ser um
local, um ano, código de setor, nome de município ou outro. Os eventos
observados podem ser casos, óbitos, agravos ou outros. Os estratos podem ser
```

```
categorias por faixa etária, categorias por sexo ou outros. ~~
~~  No objeto ''df.pop.st'', quando este é nulo (default), a população padrão utilizada como é a da Organização Mundial da Saúde (OMS), que está dividida nos seguintes estratos: segunda coluna correspondente a contagem de 00 a 14 anos de idade; terceira coluna equivalente a contagem de 15 a 59 anos de idade; e quarta coluna referente a contagem de 60 anos e mais de idade.[1] ~~
~~  O método utilizado para a padronização, é o método direto.[2] ~~

Value

~~ Se ''crude.rt'' = TRUE (defalut), retorna um data.frame, com uma coluna com a mesma identificação fornecida pelo usuário, uma segunda coluna com as taxas padronizadas por pelo método direto e uma terceira coluna com as taxas não ajustadas (brutas).~~

~~ Se ''crude.rt'' = FALSE retorna um data.frame de saída com uma coluna com a mesma identificação fornecida pelo usuário e uma segunda coluna com as taxas padronizadas pelo método direto.~~

~~ Se ''mult'' = 100000, os resultados das taxas serão multiplicados por 100 mil habitantes. ~~

~~ Se ''df.pop.st'' = NULL, a população utilizada para a padronização das taxas será a da Organização Mundial da Saúde.~~

Warning:

~~ Os objetos de entrada devem:~~
~~ - ser da classe data.frame ~~
~~ - apresentar uma primeira coluna comum de identificação ~~
~~ - apresentar colunas por categorias de estrados comum entre todos os objetos, constando mesma ordem de posição de coluna e com observações numéricas. ~~
~~ - não apresentar valores NAs ou valores zero (este último, exceto para o objeto de entrada 'df.event'). ~~

Note:

...

References


~~ 1.   National Cancer Institute. World (WHO 2000-2025) Standard. Acesso em: 18 jun 2019. Disponível em <https://seer.cancer.gov/stdpopulations/world.who.html> ~~

~~ 2.   National Cancer Institute. SEER*Stat Tutorials: Calculating Age-adjusted Rates. Acesso em: 18 jun 2019. Disponível em<https://seer.cancer.gov/seerstat/tutorials/aarates/definition.html> ~~
```

```
Author

 ~~ Patricia Marques Moralejo Bermudi patricia.bermudi@usp.br~~

See Also

...

Examples

##Exemplo 1
## df.event <- data.frame(cod_local = seq(1:3),
##                        c0_14 = c(3, 200, 500),
##                        c15_60 = c(8, 400, 1090),
##                        c60_em = c(50, 4800, 19200))
## df.pop <-  data.frame(cod_local = seq(1:3),
##                       p00_14 = c(603500, 225957, 150367),
##                       p15_60 = c(272373, 573743, 541965),
##                       p60_em = c(41630, 167600, 107313))
##x <- stRate(df.event, df.pop)
##x
## A população padrão que está sendo utilizada é a da OMS, conforme consta
no HELP
## ID Taxa Bruta Taxa Padronizada
## 1  1      17.00            43.00
## 2  2     544.09           393.15
## 3  3    1589.58          1430.01

##Exemplo 2
##df.event <- data.frame(cod_local = seq(1:3),
##                        c1 = c(3, 200, 500),
##                        c2 = c(50, 4800, 19200))

##df.pop <-  data.frame(cod_local = seq(1:3),
##                       p1 = c(60350, 22595, 15036),
##                       p2 = c(4163, 16760, 10731))

##df_st <- data.frame(cod_local = seq(1:3),
##                     pp1 = rep(58080, 3),
##                     pp2 = rep(81920, 3))

##y <- stRate(df.event, df.pop, df.pop.st = df_st, mult = 10000, crude.rt =
F)

##y

##  ID Taxa Padronizada
##1  1            70.49
##2  2          1712.55
```

```
##3  3         10607.38
```

Proposta: Padronização direta de taxas

CONTEXTUALIZAÇÃO:

Comparar taxas sem nenhum ajuste (taxas brutas), entre locais ou entre diferentes períodos de tempo no mesmo local, pode induzir à interpretações errôneas. É necessário considerar que diferentes populações podem apresentar diferentes distribuições de determinantes que estão relacionados ao evento de estudo. Deste modo, a padronização direta de taxas é um método utilizado para retirar esse efeito de confusão, fazendo um ajuste por uma ou mais variáveis presentes na população que estão causando confundimento.

Nesta padronização, as populações observadas que serão comparadas (separadas em estratos segundo a variável de interrese) são matematicamente ajustadas para a estrutura de uma população escolhida ('população padrão'). Assim, a medida obtida representa a taxa [1)] de um local, considerando que este teria a mesma distribuição da população adotada como “padrão” em relação a variável escolhida para o ajuste.

Na área da saúde, é comum realizar a padronização pela variável idade [2)] e usar, como população padrão, a estrutura de estratos da população elaborada pela da Organização Mundial da Saúde, que nesta função será agrupada em três estratos (jovens[3)], adultos[4)] e idosos[5)]) e deixada como default.

Cálculo de uma taxa não ajustada (bruta) ‘crude.rt’:

Cálculo dos eventos esperados de um extrato z ‘espct_event’ de um estrato apenas:

Cálculo da taxa padronizada ‘st_rate’:

MINHA FUNÇÃO:

Título: stRate (standardised rate).

Entrada: stRate (`df.event`, `df.pop`, `df.pop.st` = NULL, `mult` = 100000, `crude.rt` = TRUE)

ARGUMENTOS:

- `df.event` Um dataframe com a contagem de eventos observados[6)] por estratos[7)], segundo uma variável de identificação.[8)]
- `df.pop` Um dataframe com a contagem de populações observadas, pelos mesmos estratos e variável de identificação fornecidos no objeto `df.event`.
- `df.pop.st` Um dataframe com a contagem de populações padrão pelos mesmos estratos e variável de identificação fornecidos no objeto `df.event`. Se `df.pop.st` = NULL, a contagem de populações padrão será a da Organização Mundial da Saúde.
- `mult` Um vetor númerico utilizado como multiplicador para evitar valores fracionados de difícil

interpretação. Se `mult` = 100000, os resultados das taxas serão multiplicados por 100 mil habitantes.
- `crude.rt` Um vetor lógico. Se `crude.rt` = TRUE, realiza e apresenta o cálculo das taxas não ajustadas (brutas).

CONTROLES DE FLUXO:

- Os objetos de entrada `df.event`; `df.pop`; (e `df.pop.st`, se for o caso) são da classe data.frame? Se não, a função para e retorna uma mensagem ao usuário para correção.
- Os objetos de entrada `df.event`; `df.pop`; (e `df.pop.st`, se for o caso) contém na posição [1] uma coluna de identificação? Se não, a função para e retorna uma mensagem ao usuário para correção.
- Os objetos de entrada `df.event`; `df.pop`; (e `df.pop.st`, se for o caso) contém nas posições [2] em diante de colunas, variáveis numéricas com as respectivas contagens? Se não, a função para e retorna uma mensagem ao usuário para correção.
- Os objetos de entrada `df.event`; `df.pop`; (e `df.pop.st`, se for o caso) contém o mesmo número de linhas (locais ou ano) e colunas (variáveis)? Se não, a função para e retorna uma mensagem ao usuário para correção.
- Os objetos de entrada `df.pop`; (e `df.pop.st`, se for o caso) contém o valor zero ou NA? Se sim, a função para e retorna uma mensagem ao usuário para correção.
- O usuário respondeu (por meio da função readline) que seus dataframes fornecidos NÃO possuem a mesma sequência em relação aos extratos? Se não, a função para e retorna uma mensagem ao usuário para correção.
- O usuário selecionou `crude.rt` = FALSE? Se sim, a função não realiza o cálculo e não apresenta os valores das taxas não ajustadas (brutas).
- O usuário deixou a opção `df.pop.st` = NULL? Se sim, verifica se os objetos de entrada `df.event`; `df.pop` contém quatro colunas. Se sim, mostrar uma mensagem que será utilizada a população padrão da OMS. Se não, a função para e retorna uma mensagem ao usuário para correção.

PSEUDO-CÓDIGO:

1. Aplica uma condição:
    1. Se 'is.null(`df.pop.st`)': cria um objeto `df.pop.st`, da classe 'data.frame' que contém:
        1. A primeira coluna de identificação do objeto `df.cases`
        2. As três demais colunas correspondentes as repetições dos respectivos três valores da população padrão da organização mundial da saúde, com número de linhas equivalente ao objeto `df.cases`.
2. Cria um objeto vazio de eventos esperados `expct_event`.
    1. Este recebe uma matriz com valores NA e o mesmo número de linhas e colunas do objeto `df.cases`
3. Transforma `expct_event` em data.frame
4. Cria dois ciclos `for` para preencher `expct_event`
    1. O primeiro com contador `i` de 1 até o número de linhas do objeto `df.cases`
    2. O segundo com contador `k` de 2 até o número de colunas do objeto `df.cases`
    3. Atribui ao objeto `expct_event` nas posições `i` e `k`:
        1. A multiplicação dos eventos observados de `df.cases` nas posições `i` e `k`, pela população padrão `df.pop.st` nas posições `i` e `k`

        2. Dividindo pela população observadas `df.pop` nas posições `i` e k.
5. Cria um objeto que recebe a taxa padronizada `st_rate`
    1. Atribui à um data.frame uma primeira coluna correspondente à coluna de identificação do objeto `df.cases` e uma segunda coluna com o cálculo da taxa, arredondando para duas casas decimais
        1. Por meio da função `apply`: faz a somatória, por linhas, de todas as colunas dos eventos esperados `expct_event`, exceto a coluna um, que é de identificação.
        2. Multiplica este valor pelo argumento `mult`
        3. Divide pela somatória, por meio da função `apply`, por linhas, de todas as colunas das populações padrão `df.pop.st`, exceto a coluna um.
6. Aplica uma condição:
    1. Se `crude_rt` == TRUE: cria um objeto`st_crude`, da classe data.frame: da taxa não ajustada (bruta), arredondando para duas casas decimais:
        1. A primeira coluna recebe a coluna de identificação do objeto `df.cases`
        2. Por meio da função apply: faz a somatória, por linhas, de todas as colunas dos eventos observados `df.event`, exceto a coluna um, que é de identificação
        3. Multiplica este valor pelo argumento `mult`
        4. Divide pela somatória, por meio da função `apply`, por linhas, de todas as colunas das populações observadas `df.pop`, exceto a coluna um.
        5. Cria um objeto `exit`
            1. Faz a combinação por colunas das taxas padronizadas `st_rate` e com as taxas brutas `st_crude`, esta última sem a coluna de identificação
            2. Renomeia os nomes das colunas para “ID“, “Taxa padronizada”, “Taxa bruta”)
    2. Se não:
        1. Cria um objeto `exit` apenas com as taxas padronizadas `st_rate`
        2. Renomeia os nomes das colunas para “ID” e “Taxa padronizada”
7. Retorna `exit`

SAÍDA:

- Data.frame com o mesmo número de linhas e mesma coluna de identificação do objeto de entrada `df.event`, com uma coluna referente as taxas padronizadas e, se escolhido pelo usuário, com uma coluna referente as taxas brutas.

GENERALIDADE NO USO DA FUNÇÃO:

- O usuário insere a variável de evento desejada. Não há limite para o número de linhas. Desde modo, a função se aplica a:
    - Qualquer agravo ou doença escolhido pelo usuário
    - Qualquer e quantas localidades (ou períodos de tempo) escolhidos
- A função também foi pensada para não limitar o número de colunas. Além disso, possibilita inserir um dataframe com a população padrão da escolha do usuário. Assim:
    - É possível padronizar por idade por quantos estratos de faixa etárias forem necessários
    - Também é possível utilizar este método para ajuste com outras variáveis como, por exemplo, ajuste por sexo [9)]

Comentários Lucas Camacho

Oi Patricia, tudo bem?

Sua proposta está super completa e bem estruturada. Acho que você estudou bem como fazer ela e gostei bastante do tópico sobre generalização da função e como dá pra usar elas em outros contextos.

Enfim, acho legal você tentar entregar ela.

Resposta Patricia

Oi Lucas, muito obrigada! Ficarei com a proposta A.

Te mandei um e-mail com uma dúvida.

Grata! Abraços

Plano B: Função para média móvel

CONTEXTUALIZAÇÃO:

A média móvel simples tem o objetivo de suavizar uma tendência de um conjunto de dados que apresenta flutuação aleatória ao longo do tempo. Seu cálculo é dado por meio de média aritmética, comumente realizada a cada três valores. Assim, esta se inicia na posição um e vai se descolando até o final, ao longo do conjunto de dados, de uma em uma unidade de período de tempo (dias, meses, ano, etc). Exemplo

MINHA FUNÇÃO:

Título: movMean (Moving mean).

Entrada: movMean(`data`, `period`, k = 3)

ARGUMENTOS:

- `data` Um vetor numérico quantitativo
- `period`Um vetor numérico quantitativo correspondente ao período de cada valor em 'data'
- k Um valor correspondente a quantidade de valores em 'data' que serão contabilizados para a média móvel em cada deslocamento

CONTROLES DE FLUXO:

- Os objetos de entrada `data` e `period` são numéricos? Se não, a função para e retorna uma mensagem de erro.
- Os objetos de entrada `data` e `period` possuem o mesmo tamanho? Se não, a função para e retorna uma mensagem de erro.
- O objeto `data` possui NA? Se sim, a função para e retorna uma mensagem de erro.

PSEUDO-CÓDIGO:

1. Cria um objeto vazio para receber as médias móveis
    1. Atribui a um objeto `mm` repetições de NA com tamanho correspondente a variável `data` subtraindo o valor dois, já que a primeira e última posição não serão ocupadas.
2. Cria um objeto `j` com atribuição de k
3. Cria um ciclo `For` com contador `i` indo de 1 até o tamanho da variável `data` menos dois
    1. Atribui ao objeto `mm` na posição `i` a soma de `data` nas posições de `i` a `j`, dividindo por k
    2. Atribui ao objeto `j`, a operação `j + 1`
4. Faz o plot de um gráfico de linhas do objeto `data`e do objeto das médias móveis `mm`
5. Sobrescreve o vetor `mm` acrescentando NA na primeira e última posição, para que o vetor tenho o mesmo tamanho dos demais vetores
6. Cria um objeto `exit` contendo um dataframe com os valores dos períodos `period`, do vetor `data` e das médias móveis `mm`.
7. Retorna `exist`

SAÍDA:

- Gráfico de linhas de comparação dos valores observados e dos valores estimados.
- Data frame com os vetores fornecidos pelos usuários e os valores das médias móveis.

GENERALIDADE NO USO DA FUNÇÃO:

- A função pode calcular a média móvel para qualquer evento de interesse.
- Alterando o valor do argumento k: a função pode realizar médias de diferentes tamanhos de períodos de tempo. Exemplo, com k = 4 e com unidade de medida do argumento `period` em ano, serão calculadas médias móveis quadrienais.
- Ao retornar um dataframe, o usuário pode obter os dados das médias para fazer outros tipos de gráficos de sua escolha.

Comentários Lucas Camacho

Essa função também está bem estruturada e parece um pouco mais simples do que a proposta A.

Patricia, ambas as propostas estão boas e acho que você pode escolher qual das duas você quer entregar como trabalho final.

Qualquer dúvida que você tiver pode me mandar e-mail (lucas.camacho@usp.br) e podemos conversar.

Abraços

1)

de incidência, prevalência ou mortalidade

2)

dentro do link, clique na última linha, em 'view the steps'

3)

0 a 14 anos de idade

4)

15 a 60 anos de idade

5)

60 anos e mais de idade

6)

casos, óbitos, agravos ou outros

7)

etários ou outros

8)

local ou ano

9)

Basta o usuário inserir os objetos nos argumentos correspondentes aos data.frames: `df.event`, `df.pop` e `df.st`, cada um contendo uma coluna de identificação e uma coluna de contagem para cada estrato (sexo masculino e sexo feminino).