Função sample.suff

Código

```
#Fazer função sample.suff
sample.suff <- function(dados, col.ID, col.pop, col.loci, na.code=NA,
IC.plot=TRUE, gen.div=c("He","Ho","Ar"), nsim=1000){
  # Avaliar se o esforço amostral foi suficiente para inferir a diversidade
genética das populações.
  #
  # Argumentos:
  #    dados: data frame com um indivíduo por linha e um locus por coluna,
com alelos separados por /. Além
  #           disso, data frame deve ter uma coluna com identificação dos
indivíduos e outra com identificação das
  #           populações.
  #   col.ID: número da coluna com identificação dos indivíduos.
  #  col.pop: número da coluna com identificação da população a qual os
indivíduos pertencem.
  # col.loci: intervalo das colunas correspondentes aos loci microssatélite
genotipados. Note que a função foi
  #           elaborada para espécies diplóides. Portanto, cada loco deve
conter dois alelos.
  #  na.code: código utilizado para valores faltantes (Default = NA; neste
caso, valores faltantes no data frame
  #           dados não recebem nenhum valor ou código). Para evitar
conflitos com o R, não utilizar "NA" como
  #           código para dados faltantes!
  #  IC.plot: vetor lógico que indica se intervalos de confiança devem ou
não ser plotados (Default = TRUE).
  #  gen.div: estimativa de diversidade genética a ser calculada, podendo
ser heterozigosidade esperada e observada
  #           (He e Ho, respectivamente) e riqueza alélica (Ar).
  # nsim: número de simulações (Default = 1000).
  #
  # Retorna:
  #  (1) Gráfico com a média da estimativa de diversidade genética simulada
em função do tamanho amostral da
  #      população (podendo ou não incluir o intervalo de confiança das
estimativas).
  #  (2) Objeto 'resultado' (classe list), contendo (i) data frame com
estimativa de diversidade genética observada
  #      para as populações e (ii) data frame com média e intervalos de
confiança superior e inferior da estimativa
  #      de diversidade genética para cada população em cenários simulados
com diferentes números de indivíduos.
  #
  # Passos da função:
  # Passo(1): Verificar argumentos da função
```

```
  # Verificar se  col.ID, col.pop e col.loci são números inteiros maiores
que zero
  if(col.ID != round(col.ID) | col.ID <= 0 | col.pop != round(col.pop) |
col.pop <= 0 | any(col.loci !=
  round(col.loci)) | any(col.loci <= 0 ))
  {
    # Se não forem, parar a função e escrever mensagem de erro
    stop("col.ID, col.pop e col.loci devem ser números inteiros maiores que
zero")
  }
  # Verificar se dados é da classe data.frame, com cada linha correspondendo
a um único indivíduo
  if(class(dados)!="data.frame" & length(unique(dados[,col.ID])) !=
length(dados[,col.ID]))
  {
    # Se não for, parar a função e escrever mensagem de erro
    stop("Objeto de entrada deve ser da classe data.frame com uma linha por
indivíduo")
  }
  # Verificar se há apenas uma coluna por locus
  if(length(unique(colnames(dados[col.loci]))) !=
length(colnames(dados[col.loci])))
  {
    # Se houver mais de uma coluna por locus, primeiramente imprimir no
console um exemplo de data frame de dados.
    # Este exemplo é um subconjunto do dataset disponível no trabalho de
Mori et al. 2015 PLoS ONE 10(2): e0118710.
      print(data.frame(ID=c("AsAJU01","AsAJU02","AsAJU03"),
pop=rep("Braganca-PA",3),
          locus01=c("221/221","223/223","225/219"),
locus02=c("150/150","148/150","148/148"),
          locus03=c("213/213","","227/233")),row.names=F)
    # Depois, parar a função e escrever mensagem de erro, com referência ao
exemplo acima.
    stop("Loci devem ser organizados em uma coluna por locus, com alelos
separados por /. Veja o exemplo acima.")
  }
  # Verificar se IC.plot é um vetor lógico
  if(IC.plot != TRUE & IC.plot != FALSE)
  {
    # Se não for, parar a função e escrever mensagem de erro
    stop("IC.plot deve ser um vetor lógico (TRUE ou FALSE)")
  }
  # Verificar se gen.div é uma das opções de estimativa de diversidade
genética (He, Ho ou Ar)
  if(gen.div != "He" & gen.div != "Ho" & gen.div != "Ar")
  {
    # Se não for, parar a função e escrever mensagem de erro
    stop("gen.div deve ser uma das seguintes opções: He, Ho ou Ar")
  }
```

```
  # Verificar se nsim é um número inteiro maior que zero
  if(nsim != round(nsim) | nsim <= 0)
  {
    # Se não for, parar a função e escrever mensagem de erro
    stop("nsim deve ser um número inteiro maior que zero")
  }else  # Além disso, verificar se nsim é manor que 1000
    if(nsim < 1000)
    {
      # Em caso positivo, imprimir no console uma mensagem de alerta
      warning("Um número baixo de simulações pode afetar a confiabilidade
dos resultados!")
    }
  # Passo (2): Calcular estimativa de diversidade genética observada e
estimada nas populações em cenários simulados
  #            com diferentes números de indivíduos.
  # Carregar pacote adegenet necessário para rodar a função
  require(adegenet)
  # Transformar dados em objeto 'dados1' da classe genind. Para isso os
argumentos col.ID, col.pop e col.loci serão
  # usados. Determinar também a ploidia como sendo igual a 2 (espécies
diplóides) e o tipo de marcador como
  # codominante
  dados1 <- df2genind(dados[,col.loci], sep = "/", ploidy = 2,
ind.names=dados[,col.ID],pop=dados[,col.pop],
                      loc.names = names(dados[,col.loci]),NA.char = na.code,
type="codom")
  # Criar um objeto 'simula.allpop' da classe list onde serão armazenados os
resultados (média e intervalos de
  # confiança superior e inferior) de diversidade genética em cenários
simulados com diferentes números de
  # indivíduos. 'simula.allpop' é uma lista vazia, o que permite a inserção
de quantos objetos forem necessários à
  # lista
  simula.allpop <- list()
  # Criar um objeto 'observado.allpop' com número de NAs correspondente ao
número de populações de 'dados1'. Nesse
  #objeto será armazenado a estimativa de diversidade genética observada
para as populações, ou seja, estimativas
  # feitas considerando todos os indivíduos das populações
  observado.allpop <- rep(NA,nlevels(dados1@pop))
  #
  # Entrar em um ciclo(1º) com contador i de 1 a número de populações de
'dados1'. Ou seja, para cada população,
  # fazer o que se segue
  for(i in 1:nlevels(dados1@pop))
  {
   # Criar um objeto 'simula.media' com número de NAs correspondente ao
número de indivíduos da população. Neste
   # objeto será armazenada a média da diversidade genética em cenários
simulados com diferentes números de
   # indivíduos
```

```
   simula.media <- rep(NA, nrow(dados1[pop=i]@tab))
   # Criar um objeto 'simula.ICsup' com número de NAs correspondente ao
número de indivíduos da população. Neste
   # objeto será armazenada o intervalo de confiança superior da diversidade
genética em cenários simulados com
   # diferentes números de indivíduos
   simula.ICsup <- rep(NA, nrow(dados1[pop=i]@tab))
   # Criar um objeto 'simula.ICinf' com número de NAs correspondente ao
número de indivíduos da população. Neste
   # objeto será armazenada o intervalo de confiança inferior da diversidade
genética em cenários simulados com
   # diferentes números de indivíduos
   simula.ICinf <- rep(NA, nrow(dados1[pop=i]@tab))
   # Estimar a diversidade genética observada para as populações, o que
dependerá do estimador de diversidade
   # genética escolhido no argumento gen.div
   # Se gen.div for He
    if(gen.div == "He")
    {
    # Calcular a heterozigosidade esperada para cada população de 'dados1' e
armazenar na posição i do objeto
    #'observado.allpop'. Note que a heterozigosidade esperada da população é
a média das heterozigosidades esperadas
    # dos loci genotipados na população
    observado.allpop[i] <- mean(summary(dados1[pop=i])$Hexp)
    }
   # Se gen.div for Ho
    if(gen.div == "Ho")
    {
    # Calcular a heterozigosidade observada para cada população de 'dados1'
e armazenar na posição i do objeto
    # 'observado.allpop'. Note que a heterozigosidade observada da população
é a média das heterozigosidades
    # observadas dos loci genotipados na população
    observado.allpop[i] <- mean(summary(dados1[pop=i])$Hobs)
    }
   # Se gen.div for Ar
    if(gen.div == "Ar")
    {
    # Calcular a riqueza alélica para cada população de 'dados1' e armazenar
na posição i do objeto
    # 'observado.allpop'
    observado.allpop[i] <- unname(summary(dados1[pop=i])$pop.n.all)
    }
      # Entrar em um ciclo(2º) com contador j de 1 a número de indivíduos na
população. Ou seja, para cada cenário
      # simulado com um número de indivíduos diferente, fazer o que se segue
      for(j in 1:nrow(dados1[pop=i]@tab))
      {
      # Criar objeto 'simula.nind' com número de NAs correspondente ao
```

```
número de simulações determinadas pelo
      # argumento nsim. Neste objeto será armazenada a estimativa de
diversidade genética para o cenário simulado de
      # j indivíduos
      simula.nind <- rep(NA,nsim)
        # Entrar em um ciclo(3º) com contador k de 1 a número de simulações.
Ou seja, para cada simulação, fazer o
        # que se segue
    for(k in 1:nsim)
        {
        # Criar objeto 'sim' da classe genind, sendo um subconjunto de
'dados1' com uma amostragem de j indivíduos
        # da população i (usar replace = FALSE)
    sim <- dados1[sample(indNames(dados1[pop=i]),j),]
        # Estimar a diversidade genética para o cenário simulado (objeto
'sim'), o que dependerá do estimador de
        # diversidade genética escolhido no argumento gen.div
    # Se gen.div for He
     if(gen.div == "He")
         {
         # Calcular a heterozigosidade esperada para o cenário 'sim' e
armazenar na posição k do objeto
         # 'simula.nind'. Note que a heterozigosidade esperada da população
é a média das heterozigosidades
         # esperadas dos loci genotipados na população
         simula.nind[k] <- mean(summary(sim)$Hexp)
         }
        # Se gen.div for Ho
         if(gen.div == "Ho")
         {
         # Calcular a heterozigosidade observada para o cenário 'sim' e
armazenar na posição k do objeto
         # 'simula.nind'. Note que a heterozigosidade observada da população
é a média das heterozigosidades
         #observadas dos loci genotipados na população
     simula.nind[k] <- mean(summary(sim)$Hobs)
         }
        # Se gen.div for Ar
         if(gen.div == "Ar")
         {
         # Calcular a riqueza alélica para o cenário 'sim' e armazenar na
posição k do objeto 'simula.nind'
     simula.nind[k] <- unname(summary(sim)$pop.n.all)
         }
      }
    # Armazenar a média das estimativas de diversidade genética das nsim
simulações de cada cenário na posição j do
    #objeto 'simula.media'
    simula.media[j] <- mean(simula.nind)
    # Armazenar o intervalo de confiança superior das estimativas de
diversidade genética das nsim simulações de
```

```
    #cada cenário na posição j do objeto 'simula.ICsup'. Note que foi
determinado um intervalo de confiança de 95%
    simula.ICsup[j] <- quantile(simula.nind, probs=0.975)
    # Armazenar o intervalo de confiança inferior das estimativas de
diversidade genética das nsim simulações de
    # cada cenário na posição j do objeto 'simula.ICinf'. Note que foi
determinado um intervalo de confiança de 95%
    simula.ICinf[j] <- quantile(simula.nind, probs=0.025)
    }
  # Para cada população de 'dados1' criar um data frame com cinco colunas:
n.ind (número de indivíduos), pop
  # (população), mean (objeto 'simula.media'), ICsup (objeto 'simula.ICsup')
e ICinf (objeto 'simula.ICinf').
  # Armazenar o data frame na posição i da lista 'simula.allpop'
  simula.allpop[[i]] <- data.frame(n.ind=1:nrow(dados1[pop=i]@tab),
pop=dados1[pop=i]@pop, mean=simula.media,
  ICsup=simula.ICsup, ICinf=simula.ICinf)
  }
  # Criar um objeto gen.div.obs da classe data frame com duas colunas: uma
com as populações de 'dados1' e outra a
  # partir do objeto observado.allpop
  gen.div.obs <- data.frame(levels(dados1@pop),observado.allpop)
  # Renomear colunas do data frame 'gen.div.obs'
  colnames(gen.div.obs) <- c("Pop",gen.div)
  # Juntar todos os data frames armazenados em 'simula.allpop' em um único
data frame 'gen.div.rarefaction'
  gen.div.rarefaction <- do.call(rbind.data.frame, simula.allpop)
  # Passo (3): Construir gráfico com resultados das simulações
  # Abrir um novo dispositivo gráfico
  x11()
  # Determinar o layout da figura a ser criada
  layout(matrix(c(1,2),1,2,byrow=T),widths = c(3,2),heights = 3,TRUE)
  # Determinar parâmetros gráficos a serem usados
  par(tcl=0.3, bty="l", cex=1.2, las=1, mar=c(4,4,1,1))
  # Plotar um gráfico apenas com os eixos e sem legendas. Uma vez que todas
as populações serão plotadas em um mesmo
  #gráfico, usar como limite máximo de x o número máximo de indivíduos entre
todas as populações, e como limite de
  # y, o valor máximo entre as colunas media, ICsup e ICinf do data frame
gen.div.rarefaction
  plot(0,0,xlim=c(1,max(gen.div.rarefaction[,1])),ylim =
c(0,max(gen.div.rarefaction[,c(3:5)])),type = "n",ann=F)
  # Criar paleta de cores a serem usadas para linhas no gráfico, sendo que o
número de cores é correspondente ao
  #número de populações de 'dados1'
  cl <- rainbow(nlevels(dados1@pop),start=1)
  # Criar paleta de cores a serem usadas para áreas sombreadas no gráfico,
sendo que o número de cores é
  # correspondente ao número de populações de 'dados1'. Nessa paleta,
colocar transparência nas cores
```

```
  cl.transp <- rainbow(nlevels(dados1@pop),start=1,alpha = 0.3)
  # Entrar em um ciclo (4º) com contador z de 1 a número de populações de
'dados1'. Ou seja, para cada população,
  # fazer o que se segue
  for (z in 1:nlevels(dados1@pop))
  {
   # Fazer um subset de gen.div.rarefaction para população z
   pop <-
gen.div.rarefaction[gen.div.rarefaction$pop==levels(gen.div.rarefaction$pop)
[z],]
   # Se argumento IC.plot é TRUE
    if(IC.plot == TRUE)
    {
    # Plotar intervalos de confiança das estimativas de diversidade genética
das simulações de cada população como
    # área sombreada no gráfico. Usar uma cor para cada população (utilizar
cores determinadas no objeto
    # 'cl.transp')
    polygon(x=c(1,1,2:(nrow(pop)-1), nrow(pop), nrow(pop),(nrow(pop)-1):2),
y=c(pop$ICinf[1],pop$ICsup[1],pop$ICsup[2:nrow(pop)],pop$ICinf[nrow(pop)],po
p$ICinf[(nrow(pop)-1):2]),
    col=cl.transp[z],border = NA)
    }
   # Plotar linhas correspondentes às médias das estimativas de diversidade
genética das simulações de cada
   #população. Usar uma cor para cada população (utilizar cores determinadas
no objeto 'cl')
   lines(x=pop$n.ind, y=pop$mean,col = cl[z],type = 'l',lty=1,lwd=0.5)
  }
  # Adicionar legenda do eixo x no gráfico
  mtext("Número de indivíduos", side=1, cex=1.4,line=3.0)
  # Adicionar legenda do eixo y no gráfico, de acordo com a estimativa de
diversidade genética escolhida no
  # argumento gen.div
  # Se gen.div for He
  if(gen.div == "He")
  {
   # Adicionar legenda do eixo y como Heterozigosidade esperada
   mtext("Heterozigosidade esperada", side=2, cex=1.4, line=3,las=3)
  }
  # Se gen.div for Ho
  if(gen.div == "Ho")
  {
   # Adicionar legenda do eixo y como Heterozigosidade observada
   mtext("Heterozigosidade observada", side=2, cex=1.4, line=3,las=3)
  }
  # Se gen.div for Ar
  if(gen.div == "Ar")
  {
   # Adicionar legenda do eixo y como Riqueza Alélica
   mtext("Riqueza alélica", side=2, cex=1.4, line=3,las=3)
```

```
  }
  # Adicionar legenda no gráfico para indicar a cor correspondente a cada
população. Para isso, utilizar a outra
  # área da figura definida no layout. Ou seja, começar um novo plot
  plot.new()
  # Definir parâmetro de margem para a legenda
  par(mar=c(1,1,4,4))
  # Se IC.plot for TRUE
  if(IC.plot == TRUE)
  {
   # Plotar na legenda quadrados preenchidos com as cores do objeto
'cl.transp'
   legend("left", legend=c(levels(dados1@pop)),col=cl.transp, lwd=1,
lty=c(rep(0,nlevels(dados1@pop))),
          pch=c(rep(15,nlevels(dados1@pop))), bty='n',text.col = "white")
  }
  # Plotar na legenda linhas com as cores do objeto 'cl' e adicionar texto
com a população correspondente
  legend( "left",legend=c(levels(dados1@pop)),col=cl, lwd=1,
lty=c(rep(151,nlevels(dados1@pop))),
         pch=c(rep(NA,nlevels(dados1@pop))), bty="n")
  # Criar objeto 'resultado' da classe lista que será a saída da função
sample.suff, contendo os objetos
  # 'gen.div.obs' e 'gen.div.rarefaction'
  resultado <- list(gen.div.obs,gen.div.rarefaction)
  # Renomear itens da lista de acordo com a estimativa de diversidade
genética escolhida no argumento gen.div
  names(resultado) <- c(paste0(gen.div,"-observada"),paste0(gen.div,"-
simulada"))
  # Retornar, no fim da função, o objeto 'resultado'
  return(resultado)
}
```

Arquivo

Este documento contém o mesmo código acima, porém armazenado em um arquivo .R:Código-Função sample.suff