Luiz Carlos Machado

Terminei a graduação, vou fazer minha inscrição do mestrado no segundo semestre de 2015 com o professor Diogo Meyer, em um projeto que estou escrevendo dentro de um tema geral em genética de populações humanas.

Meus Exercícios

Linque para a página com os meus exercícios resolvidos aqui

#Ex_Aula_1 F

#Ex_Aula_4 F

#Ex_Aula_5 F

#Ex_Aula_7A_e_7B_juntos F

#Ex_Aula_8 F

#EX_Aula_9 F

Proposta de Trabalho Final

Proposta A

Motivação:

O genoma humano pode ser considerado com um conjunto de sítios variáveis (SNPs), agrupados em genes (definidos funcionalmente ou com base em predições computacionais) e regiões intergênicas. Estamos interessados em investigar os processos que levam a variação genética dentro de um subconjunto de genes, definidos por um critério prévio, difere daquela do restante do genoma. Para isso utilizaremos como ferramenta estatística a razão entre polimorfismos não-sinônimos por sinônimos (Pn/Ps).

Proposta para a função:

Suponha que previamente encontramos um subconjunto gene X, X={X1, X2, X3, Xn'} contendo k SNPs de alto Fst, estatística que calcula a medida de diferenciação entre subpopulações considerando o modo como a diversidade genética se encontra distribuída dentro e entre essas subpopulações. Para realizarmos estudos comparando trechos com diferentes Fst, adotamos como controle outros

fragmentos que não tenham alto Fst (visto que nossa amostra é formada por SNPs com alto Fst) compostos por um conjunto de gene Y, Y={Y1, Y2, Y3, Yn''}. Um dos problemas com qual nos deparamos com a utilização desse controle é que em geral o conjunto Y é muito maior que o conjunto X (ou se seja, há menos genes e SNPs com alto Fst do que no grupo controle).

Para controlar para esse efeito propomos gerar uma coleção de dados de SNPs que nos permita calcular um Pn/Ps controle comparavel ao Pn/Ps empirico.

Como entrada, teremos o conjunto X (primeiro argumento) e o conjunto de Y (segundo argumento) como dois vetores lógicos de tamanhos iguais, além do vetor de caracteres Z, com o DNA referência. Também será necessário mais dois argumentos, formados por dois vetores lógicos, que definirão se os SNPs indicados em X e Y são sinônimos ou não-sinônimos.

A função proposta reamostraria ao acaso uvm subconjunto de genes de Y para obter o mesmo numero de genes observado em X e dentro desses subconjuntos sorteará k SNPs (mesmo número de SNP encontrado no conjunto X), assim poderemos compara-los com uma estatistíca de interesse, no nosso caso Pn/Ps. A função também desempenhará n reamostragens dessas quebras de Y, obtendo o subconjunto Y' , Y' = {Y'1, Y'2, Y'3, Y'n} que serão todos aproximadamente do mesmo tamanho e grau de desequilíbrio de ligação que X, tornando-os bem pareados e portanto fazendo com que cada Y' seja um bom controle para X, imunes a outros efeitos (número de SNPs, forma como eles estão distribuídos no genoma), passiveis de ser encontrados em Y. Neste ponto, a função calculará o Pn/Ps do elementos do subconjunto Y', tendo como saída uma distribuição nula que será comparada ao Pn/Ps do conjunto X, por um teste de p value empírico.

> As propostas são bacanas, mas acho que o melhor seria misturar elas um pouco. A proposta de objetos de entrada da primeira função é meio tosca, pq trabalhar com vetores lógicos que vem do além se vc pode receber as sequencias e calcular tudo isso dentro da função? Melhor receber sequencias alinhadas ou algo do gênero. Acho que a função deve detectar SNPs sinônimos e não sinônimos e depois fazer a reamostragem adequada e o teste. Não é particularmente mais complicado e fica bem mais útil.
>
> —-*Diogo Melo*

Proposta B

Motivação:

Em um futuro não distante, após um jantar o filho levanta a voz e diz “tenho que entregar minha genotipagem pra professora, ela quer medir a variação de Pn/Ps de algum gene para todos os alunos do 7° ano e mostrar todo mundo em algum gráfico, sei lá.” e o pai responde “tudo bem, pedi pra sua mãe passar para seu notebook sua genotipagem que fizemos no ano passado”. Talvez a professora tenha sorte, caso recuse minha proposta A, eu já terei feito a função que possibilita essa comparação.

Proposta para a função:

Calcular os polimorfismos sinônimos e não sinônimos de indivíduos que já tenham realizado sua genotipagem completa. A função terá como entrada um vetor contendo o cDNA referência do gene a ser estudado (primeiro argumento) e um segundo vetor contendo o cDNA da genotipagem do indivíduo. Dentro da função criarei um catálogo em formato de matriz que contenha a informação de qual aminoácido corresponde cada trinca de bases, possibilitando assim definir se uma mutação ocorrida no vetor de cDNA do indivíduo é sinônima ou não sinônima. O cálculo do Pn/Ps se dará pela simples contagens de polimorfismos não-sinônimos e sua divisão pela contagem dos sinônimos, assim teremos como saída o resultado dessa razão para o gene especifico de entrada.

A função também terá como saída um gráfico que localize onde estarão as mutações não-sinônimas (bolinhas vermelhas) e mutações sinônimas (bolinhas verdes) plotadas em uma linha com Y definido e X variando proporcionalmente ao trecho de DNA estudado. Assim a professora poderá mostrar aos alunos a localização de mutações em diferentes indivíduos.

Ps: Como dado de entrada para testar a função (cDNA do menino) vou colocar uma sequência hipotética de cDNA que possibilite me mostrar as variações de Pn/Ps utilizando como cDNA referência uma informação real para algum gene.

####Arquivo para exemplo

Tabela_Teste

Help

```
 reamost                       packege: nenhum                    R
Documentation

Description:
A Função "reamost" constrói de um Data Frame, um conjunto formado por genes
com Fst elevados e reamostra de outras regições genomicas conjuntos com o
mesmo tanho. Calcula o Pn/Ps de todos os SNPs, constrói uma distribuição
nula dos conjuntos reamostrados e aponta onde o Pn/Ps do primeiro conjunto
formado está na distribuição.

Usage:
reamost(data, ...)
#método em Default:

reamost(data, hqnt= 0.99, lqnt=0.6, sample= 100)

Arguments:
data: Um data frame onde cada linha representa um SNP, a coluna 2 tem que
ser uma coluna que informe a qual gene pertence os SNPs, a coluna 3 tem que
conter informação a respeito dos Fst dos SNPs e a coluna 4 se o SNP é
sinônimo ou não-sinônimo.

hqnt: Quantil dos Fst mais elevados, como Default, os 1% maiores valores de
Fst.
```

```
lqnt: Quantil dos Fst menos elevados, como Defautt, os 50% menores valores
de Fst.

sample: Quantas amostras formar na reamostragem, como default 100 amostras
que serão gravadas em um data frame.

Os argumentos que definem a quantidade máxima de SNPs por genes (limclas1,
limclas2, limclas3), são eles:
limclas1: Tem como limite máximo por default o valor de 10
limclas2: tem como limite máximo por default o valor de 19
limclas3: tem como limite máximo por default o valor de 32

Datails
O genoma humano pode ser considerado com um conjunto de sítios variáveis
(SNPs), agrupados em genes (definidos funcionalmente ou com base em
predições computacionais) e regiões intergênicas. com o interesse de
investigar os processos que levam a variação genética dentro de um
subconjunto de genes, definidos por um critério prévio, difere daquela do
restante do genoma. Para isso utilizaremos como ferramenta estatística a
razão entre polimorfismos não-sinônimos por sinônimos (Pn/Ps).
A função reamostraria ao acaso uvm subconjunto de genes de Y para obter o
mesmo numero de genes observado em X e dentro desses subconjuntos sorteará
SNPs dentro de categorias estabelecidas no argumento (para aproximar ao
maximo do numero de SNPs encontrado no conjunto X), assim ela  compara-los
com uma estatistíca de interesse, no caso da função, Pn/Ps. A função também
desempenhará n reamostragens dessas quebras de Y, obtendo o subconjunto Y' ,
Y' = {Y'1, Y'2, Y'3, Y'n} que serão todos aproximadamente do mesmo tamanho e
grau de desequilíbrio de ligação que X, tornando-os bem pareados e portanto
fazendo com que cada Y' seja um bom controle para X, imunes a outros efeitos
(número de SNPs, forma como eles estão distribuídos no genoma), passiveis de
ser encontrados em Y. A função calcula o Pn/Ps do elementos do subconjunto
Y', tendo como saída um histograma, neste terá uma linha vertical indicando
onde da distribuição está o Pn/Ps do conjunto X.

Value:
Retorna um histograma com uma distribuição nula do Pn/Ps calculados a partir
dos conjuntos contruídos aleatoriamente com o mesmo número de genes do
conjunto formado a partir do argumento "hqnt", retorna também no histograma
a linha onde o Pn/Ps dos dados estabelecidos está na distribuição. Retorna
também dentro da lista o Pn/Ps do conjunto x e todos os Pn/Ps dos elementos
do conjunto Y' e a proporção de valores da distribuição de conj_y_linha que
são maiores que o Pm/Ps do conj_x

Exemple:
reamost(data)
reamost(data, sample=1000000, limclas1=20, limclas2= 29, limclas3= 35, hqnt
= 0.80, hqnt= 0.6)

Author:
Luiz Carlos Machado
```

```
References:
The 1000 Genomes Project. (2012). An integrated map of genetic variation
from 1,092 human genomes. Nature, 491(7422), 56–65. doi:10.1038/nature11632
```

Função

```
##Conj_x (high_fst_gene)
new_table <- read.csv("~/laboratório/mestrado/scripts/new_table.csv")
# O "new_table" tem uma quantidade excessiva de NAs que influênciam na
estatística que calcularei mais adiante no Script, assim, resolvi tira-los.
new_table <- new_table[!is.na(new_table$exonic_func) &
!is.na(new_table$gene) & !is.na(new_table$FST_overall),]
new_table <- new_table[1:1000, ]
#newtable é a novatabela sem os NA
#fst_gene, calculei a média dos dos Fst por gene, logo depois com quantil
escolhi os 1% maiores para formar meu objeto conj_x

reamost <- function(data, hqnt= 0.99, lqnt=0.6, sample= 100, limclas1= 10,
limclas2= 19, limclas3= 32) {
colnames(data)[1]="X"
colnames(data)[2]="gene"
colnames(data)[3]="FST_overall"
colnames(data)[4]="exonic_func"
#A ideia inicial é formar com a new_table, 4 classes que vão separar os
genes por intervalos de SNPs:
#formando classe_1df
    classe1_df<- t(as.data.frame(rep(NA,length(new_table))))
    colnames(classe1_df)<-colnames(new_table)
#formando classe_2df
    classe2_df<- t(as.data.frame(rep(NA,length(new_table))))
    colnames(classe2_df)<-colnames(new_table)
#formando classe_3df
    classe3_df<- t(as.data.frame(rep(NA,length(new_table))))
    colnames(classe3_df)<-colnames(new_table)
#formando classe_4df
    classe4_df<- t(as.data.frame(rep(NA,length(new_table))))
    colnames(classe4_df)<-colnames(new_table)
#intervalos do número de linhas por gene.
    BLA<-limclas1
    BLE<-limclas2
    BLI<-limclas3
#Nesse for eu crio um data frame composto por genes que tenham um número de
linhas menor que definido em BLA
new_table$gene <- as.character(new_table$gene)
    for(i in 1: length(unique(new_table$gene))){
if((nrow(new_table[new_table$gene==unique(new_table$gene)[i],])<BLA)==TRUE)
      {classe1_df<-
rbind(new_table[(new_table$gene==unique(new_table$gene)[i]),],classe1_df) }
#Nesse for eu crio um data frame composto por genes que tenham um número de
```

```
linhas menor que definido em BLE
      if(nrow(new_table[new_table$gene==unique(new_table$gene)[i],])>=BLA &
nrow(new_table[new_table$gene==unique(new_table$gene)[i],])<BLE)# BLE =
limente inferior da classe 3
      {classe2_df<-
rbind(new_table[new_table$gene==unique(new_table$gene)[i],],classe2_df) }
#Nesse for eu crio um data frame composto por genes que tenham um número de
linhas menor que definido em BLI
      if(nrow(new_table[new_table$gene==unique(new_table$gene)[i],])>=BLE &
nrow(new_table[new_table$gene==unique(new_table$gene)[i],])<BLI)# BLI =
limente inferior da classe 4
      {classe3_df<-
rbind(new_table[new_table$gene==unique(new_table$gene)[i],],classe3_df) }
#Nesse for eu crio um data frame composto por genes que tenham um número de
linhas maior ou igual que definido em BLI
      if(nrow(new_table[new_table$gene==unique(new_table$gene)[i],])>=BLI)
      {classe4_df<-
rbind(new_table[new_table$gene==unique(new_table$gene)[i],],classe4_df) }
                                        }

#Calcular a média dos fst pra classe_1df
fst_gene_1 <-(tapply(classe1_df$FST_overall, classe1_df$gene, mean))
#Calcular a média dos fst pra classe_2df
fst_gene_2 <-(tapply(classe2_df$FST_overall, classe2_df$gene, mean))
#Calcular a média dos fst pra classe_3df
fst_gene_3 <-(tapply(classe3_df$FST_overall, classe3_df$gene, mean))
#Calcular a média dos fst pra classe_4df
fst_gene_4 <-(tapply(classe4_df$FST_overall, classe4_df$gene, mean))
fst_gene <- c(fst_gene_1, fst_gene_2, fst_gene_3, fst_gene_4)
#Escolhendo os genes com média alta de Fst no conj_x_1
conj_x_1 <- which(fst_gene_1 > quantile(fst_gene_1, hqnt, na.rm=T))
#Escolhendo os genes com média alta de Fst no conj_x_2
conj_x_2 <- which(fst_gene_2 > quantile(fst_gene_2, hqnt, na.rm=T))
#Escolhendo os genes com média alta de Fst no conj_x_3
conj_x_3 <- which(fst_gene_3 > quantile(fst_gene_3, hqnt, na.rm=T))
#Escolhendo os genes com média alta de Fst no conj_x_4
conj_x_4 <- which(fst_gene_4 > quantile(fst_gene_4, hqnt, na.rm=T))
#Conj_x concatenando os conj_x_1, conj_x_2, conj_x_3 e conj_x_4
conj_x <- c(conj_x_1, conj_x_2, conj_x_3, conj_x_4)



#Nos próximos passos vou calcular o Pn/Ps dos SNPs dos genes que pertencem a
conj_x, como conj_x tem as posições em new_table pensei em fazer um for que
com os nomes dos genes de conj_x calculasse o Pn/Ps pelas informações da
coluna exonic_func.
result_1 <- numeric(length(conj_x))
names(result_1) <- names(conj_x)

for(x in names(conj_x)) {
  teste <- table(new_table[new_table$gene == x, "exonic_func"])
```

```
#"nonsynonymous" e "synonymous" são levels da coluna exonic_func de
new_table
  result_1[x] <- teste["nonsynonymous"]/teste["synonymous"]
}
#pnps_conj_x é o pn/ps dos SNPs dos genes que pertencem ao conj_x, nessa
linha, eu tirei os NA e os inf, como pn/ps vem da divisão de simples
contagens dos polimorfismos não-sinônimos com os sinônimos, divisões como
0/3 daria resultados indesejados.
pnps_conj_x <- mean(result_1[result_1 > -Inf & result_1 < Inf], na.rm = T)

##Conj_y
#Aqui contruí o conj_y(1,2,3,4) e para isso repeti o modo como fiz o conj_x,
porém contando os quantil abaixo de 50% dos valores de Fst, esse valor
também é um argumento da função (lqnt) com default de 0.5
#conj_y_1 com a contagem dos quantis abaixo de 0.5 dentro de fst_gene_1
conj_y_1 <-which(fst_gene_1 < length(quantile(fst_gene_1, lqnt, na.rm=T)))
#conj_y_2 com a contagem dos quantis abaixo de 0.5 dentro de fst_gene_2
conj_y_2 <-which(fst_gene_2 < length(quantile(fst_gene_2, lqnt, na.rm=T)))
#conj_y_3 com a contagem dos quantis abaixo de 0.5 dentro de fst_gene_3
conj_y_3 <-which(fst_gene_3 < length(quantile(fst_gene_3, lqnt, na.rm=T)))
#conj_y_4 com a contagem dos quantis abaixo de 0.5 dentro de fst_gene_4
conj_y_4 <-which(fst_gene_4 < length(quantile(fst_gene_4, lqnt, na.rm=T)))
#conj_y <- c(conj_y_1, conj_y_2, conj_y_3, conj_y_4)
#result_2 = numeric(length(conj_y))

##Conj_y_linha [y'1, y'2, y'3, y'n]
#conj_y_linha criado om um for, fazendo um sample de fragmentos do
conj_y(1,2,3,4) garantindo que estejam com o número de SNPs dentro das
classes pré estabelecidas e com o mesmo número de genes do conj_x.
conj_y_linha_1 <- data.frame(NA)
for(i in 1:(sample/4))
{
  conj_y_linha_1[1:length(conj_x_1),i]<-sample(conj_y_1, length(conj_x_1),
replace = T)
}
#conj_y_linha_2
conj_y_linha_2 <- data.frame(NA)
for(i in 1:(sample/4))
{
  conj_y_linha_2[1:length(conj_x_2),i]<-sample(conj_y_2, length(conj_x_2),
replace = T)
}
#conj_y_linha_3
conj_y_linha_3 <- data.frame(NA)
for(i in 1:(sample/4))
{
  conj_y_linha_3[1:length(conj_x_3),i]<-sample(conj_y_3, length(conj_x_3),
replace = T)
}

#conj_y_linha_4
```

```
conj_y_linha_4 <- data.frame(NA)
for(i in 1:(sample/4))
{
  conj_y_linha_4[1:length(conj_x_4),i]<-sample(conj_y_4, length(conj_x_4),
replace = T)
}
#Formação do conj_y_linha que contém as reamostragens do conj_y nas classes
definidas, usei cbind para fazer de conj_y_linha um data frame cujas as
linhas são os SNPs e as colunas os genes reamostrados.
conj_y_linha <- cbind(conj_y_linha_1, conj_y_linha_2, conj_y_linha_3,
conj_y_linha_4)
################################
############# PnPs #############
################################

#com o conj_y_linha formado realizei o for para calcular o pn/ps de todos os
genes dos elementos do conjunto. No primeiro giro do primeiro for eu
descobri os nomes dos genes pertencentes a coluna 1. No segundo for, com os
nomes, voltei ao new_table, e calculei o Pn/Ps dos SNPs dos genes
encontrados pela simples divisão de "nonsynonymous" e "synonymous" da coluna
"exonic_func".
pnps_y_linha <- rep(NA, sample)
for(i in 1:length(conj_y_linha)){
  v2 <- conj_y_linha[ ,i]
  V2 <- names((fst_gene)[v2])
  resulty = numeric(length(v2))
  names(resulty) <- V2
  for(k in V2) {
    teste2 <- table(new_table[new_table$gene == k, "exonic_func"])
    resulty[k] <- teste2["nonsynonymous"]/teste2["synonymous"]
  }
#realizei a mesma estratégia para retirar "inf" que em conj_x
  pnps_y_linha[i] <- mean(resulty[resulty > -Inf & resulty < Inf], na.rm =
T)
}
#histograma com as médias
hist(pnps_y_linha, main='Distribuição de Pn/Ps de Y')
#linha vertical indicando de verde onde está o Pn/Ps da de conj_x
abline(v=pnps_conj_x, col="green", lwd=3)
#Proporção de valores da distribuição de conj_y_linha que são maiores que o
Pm/Ps do conj_x


return(list(sum(pnps_y_linha[pnps_y_linha!="NaN"] >
pnps_conj_x)/length(pnps_y_linha)
, pnps_y_linha, pnps_conj_x))


}
```

###Exemplo para rodar :

Utilizando o arquivo “Tabela Teste”: data← read.csv(“new_tablepeq.csv”, sep=“ ”)

reamost(data)