Anna Flavia Figueredo Benedetti

Mestranda no programa de Endocrinologia da Faculdade de Medicina da USP - FMUSP. Faz pesquisa na área de biologia molecular e bioinformática.

exec

Trabalho Final - Propostas A e B

> Anna Flavia, antes de mais nada, por favor poste suas propostas por extenso no seu Wiki, segundo as instruções da página trabalho_final. Postando num arquivo de word não será possível acompanhar da forma desejada pelos monitores e o professor. Att, — *Gustavo Agudelo 2017/06/02 20:00*

PROPOSTAS PARA O TRABALHO FINAL

Proposta A

Função plot.coverage

Função irá criar um gráfico detalhando a cobertura do sequenciamento por base de uma ou várias amostras sequenciadas por exoma. O eixo X terá os números de pares de bases, enquanto o eixo Y tem a cobertura base a base, e uma linha pelo gráfico indicando os pontos da cobertura. A importância de se ter uma cobertura boa de cada base é a confiança do que se está vendo é real ou não, ou seja, que a presença de variante, indels ou até mesmo a falta de variantes na sequencia esteja realmente no material genético do paciente. Muitos laboratórios padronizam o mínimo de cobertura desejável como 20 reads por base, e pedaços do genoma que não alcançam essa cobertura acabam por ter de ser refeitos, o que gera um grande custo para o laboratório. A ideia desse gráfico, então, é de facilitar a ilustração e, consequentemente, a visualização da quantidade de reads por base em um gene, ou seja, a cobertura do sequenciamento. Será tomado como exemplo os gráficos de cobertura do site do ExAC (http://exac.broadinstitute.org/). Todas as informações necessárias para a criação do gráfico serão retiradas de um arquivo .BED provindo de um .BAM que contenha as informações de cobertura. Este arquivo possui formato de tabela, universal para o tipo de arquivo, e poderá ser lido como um data frame. Cada uma das colunas possui uma informação importante para a estimativa dessa cobertura por base, sendo elas o cromossomo, início do transcrito, final do transcrito, nome do gene, tamanho do fragmento, base do fragmento e cobertura na base. O gene a ser plotado também poderá ser escolhido pelo usuário como argumento da função. A função também terá como opção fazer o gráfico de apenas uma amostra, que no caso só iria plotar as informações presentes no arquivo, dependendo do gene escolhido pela pessoa (que deve estar no arquivo de origem, se não a função retornará erro), ou então fazer o gráfico de cobertura de diversas amostras (que entrarão em um for), podendo serem feitas a média (mean()), mediana (median()) ou variancia (var()) por base, dependendo da escolha do usuário entre cada uma ou todas as três, que serão então plotadas em gráficos diferentes.

Proposta B

Função plot.saturation

Função irá criar um gráfico de saturação de cobertura por quantidade total de reads na amostra, para responder a questão de quantos reads são necessários para se atingir uma média/mediana de 100 vezes de cobertura. O gráfico então terá o eixo X com a quantidade de reads em milhões, e o eixo Y com a média/mediana de 0 a 100. Para isso será necessário como entrada uma tabela com a informação de cobertura com a quantidade de reads desejados para várias amostras, a fim de criar o gráfico. Cada uma das amostras vira um ponto nos gráficos, que serão unidos por uma linha. A ideia do gráfico é ajudar o pesquisador a decidir a quantidade de reads necessários para o sequenciamento chegar a uma cobertura ideal, para que informações importantes como por exemplo variantes pontuais, não sejam perdidas na análise.

> Olá Anna,
>
> Vou fazer um comentário para ambas as propostas porque eu acho que aplica para as duas. Você pensou em funções gráficas, isso é legal porque a visualização dos dados, seja exploratória ou para publicações é um requisito da boa pesquisa científica. Eu tenho várias dificuldades para entender suas propostas, não só por causa do contexto teórico que você utiliza como também porque são confusas. Primeiro, a função que vocês devem criar deve ser (de preferência) genêrica, ou seja que deve poder ser aplicada a diferentes contextos teóricos. Você pode claro usar o contexto da sua pesquisa para ilustrar apenas um dos usos da sua função, não o foco. Por exemplo, ao invés de “Função irá criar um gráfico detalhando a cobertura do sequenciamento por base de uma ou várias amostras sequenciadas por exoma”, é preferível “função irá criar um gráfico xy de duas variáveis quantitativas...”. Veja que seu gráfico deve conseguir lidar com diferentes variáveis do mesmo tipo, não só aquelas que você menciona. Outra coisa, não fica claro quais serão os argumentos e seus valores, nem como estará estruturado seu arquivo de dados. Na proposta B por exemplo não é claro o que você quer dizer por 'amostra', e como você representaria cada uma por um ponto (e.g., a média?). Tenta fazer um pesudo-código definindo os argumentos e seus valores, e detalhando de forma específica o que a função irá retornar. Dá uma pensada nisso para poder te direcionar para alguma das propostas. Se tiver alguma dúvida, só mandar uma mensagem, ok? — *Gustavo Agudelo 2017/06/06 14:27*

Apesar das funções propostas não serem funções genéricas, são função que poderiam ser adaptadas para outros usos, uma vez que se utilizam de uma entrada que possui um formato simples e padronizado, e tem como finalidade fazer medidas estatísticas de variáveis, afim de criar gráficos. Além disso, são funções interessantes para a área de sequenciamentos de larga escala, como aquelas presentes em pacotes do Bioconductor, que possui 1383 pacotes de R para análises e compreensão de dados genômicos. Portanto, são funções que poderiam ser utilizadas por todos aqueles que trabalham com sequenciamento de larga escala, uma vez que a cobertura é muito importante para a análise final e validação das variantes encontradas.

Proposta A

plot.baseCoverage(x, geneID, med = c("mean", "median", "var"))

Estrutura do arquivo de dados (universal, explicado por extenso no FAQ do UCSC), que possui:

# cromossomo	**Ínicio do transcrito**	**Final do transcrito**	**Nome do gene**	**Tamanho do transcrito**	**+/- dependendo da fita**	**# da base dentro do transcrito**	**# de reads (ou seja, a cobertura)**

Sendo que as duas últimas colunas são específicas do OUTPUT do coverageBed com a opção -d, utilizado para verificar a cobertura de regiões de interesse do pesquisador. Também são OUTPUTs deste programa o default e o -hist, que são como seguem:

Default:

# cromossomo	**Ínicio do transcrito**	**Final do transcrito**	**# de reads encontrados nesse transcrito**	**# de bases que tiveram um ou mais reads**	**Tamanho do transcrito**	**Fração de bases que tiveram um ou mais reads**

-hist:

# cromossomo	**Ínicio do transcrito**	**Final do transcrito**	**Gene**	**1**	**+/- dependendo da fita**	**# de reads**	**# de bases**	**nessa cobertura**	**Tamanho do transcrito**	**Fração coberta**

E, no final do arquivo, um histograma resumindo a cobertura de todas as regiões.

Entretanto, para esses dois outros OUTPUTs não seria possível fazer um gráfico xy, e sim um histograma. Por esse motivo, é mais interessante que essas funções fossem desenvolvidas separadamente. Dessa forma, optou-se pelo desenvolvimento da primeira função aqui descrita na proposta A, dado que envolve mais passos do que apenas a criação de um gráfico.

Antes de iniciar a função:

- Ler o arquivo de dados e salvar em um objeto
- OU ler todos os arquivos de uma pasta e salvar em um objeto

Na função:

- SE x == 1:
- gene ← tabela aonde aparece o gene passado como argumento
- xmin ← (primeiro valor da coluna 2) + 5

- xmax ← (ultimo valor da coluna 3) + 5
- cov ← ultima coluna da tabela gene
- eixo.x ← (coluna 2) + (coluna 7)
- Plotar um gráfico com eixo.x e cov, utilizando como limites do eixo x os objetos xmin e xmax (plot(type="l")).
- SE x > 1:
- eixo.x, xmin, xmax, covs ← NULL
- Entrar em um FOR (i in files)
- x ← ler(arquivo)
- gene ← tabela do gene dado como argumento
- eixo.x ← (coluna 2) + (coluna 7)
- xmin ← (primeiro valor da coluna 2) + 5
- xmax ← (ultimo valor da coluna 3) + 5
- covs ← cbind(covs, gene$ncol(gene))
- SE o argumento em med for "mean"
- mea ← apply(covs, 1, mean)
- Plotar gráfico eixo.x e mean
- SE o argumento em med for "median"
- med ← apply(covs, 1, median)
- Plotar gráfico eixo.x e med.
- SE o argumento em med for "var"
- vari ← apply(covs, 1, var)
- Plotar gráfico eixo.x e vari.
- SE não houver argumento, fazer os três.

> Olá Anna,
>
> Dessa vez suas funções ficaram mais claras. Tudo bem você propor uma função que aplique em determinado contexto. Meu comentário anterior ia mais no sentido de usar esse contexto para exemplificar a função, e não usar o contexto como o foco, pois ao final das contas o trabalho tem como objetivo avaliar o entendimento e domínio da linguagem. Vendo os pseudocódigos percebo que você sabe o que sua função faria, e na minha opinião a sua proposta A parece mais desafiadora. Eu aconselho você continuar com a proposta A então. Imagino que o 'x' é o equivalente a um 'data' e que os dados estariam salvos num objeto previamente. Sendo assim, você definiria os valores do argumento 'geneID' por indexação do 'x', né? Não se esqueça de permitir que os gráficos sejam plotados no mesmo dispositivo gráfico quando o usuário solicitar dois ou mais, e veja qual a melhor forma de organizâ-los. Qualquer coisa, é só escrever. Att, — *Gustavo Agudelo 2017/06/09 11:18*

Proposta B

plot.saturation(x)

Fará um gráfico xy de variáveis quantitativas que se encontrem em mais de um arquivo, onde o dado se localize na mesma coluna. O arquivo de dados utilizados pode ser uma tabela simples em que os dados se encontram.

Antes da função: folders ← nomes(pastas em um diretório) para colocar na função

Quando na função:

- Criar objetos nulos
- FOR (i in 1:(ultimo vetor em folders))
- Ler arquivos presentes em folders[i]
- Salvar em diferentes objetos para cada folder
- FOR (arquivos no objeto pasta 1..ultima)
- ler(arquivo) → tabela
- apply(tabela, 2, mean) → mean
- rbind(objeto, mean) → objeto
- plot(medias)

Trabalho Final

Downloads

datatests.zip plot.baseCoverage.R help_page.txt

Página Help

plot.baseCoverage package:unknown R Documentation

Plots coverage graphs

Description:

Plots graphs from output .BEDs from bedTools coverage function.

Usage:

plot.baseCoverage(x, geneID, measure)

Arguments:

x dataset. Either comes in form of a dataframe read from a file, or a list of files from which the information will be read from. geneID name of the gene desired

for the plot. Has to be the same as it appears on file. measure only necessary if there is more than one file in x. Measure desired, can be “mean”, “median” or “variance”/“var”, for mean, median or variance, respectively. If left blank, it will return all three graphs in the same window, each at a time (press enter to see the next one, in that case).

Details:

This function's main goal is to help the representation and visualization of the coverage per base or bald spots in samples that were sequenced using any high throughput technology. It takes the information stored in an output of the bedTools coverage suite, while using the -d option, which comes as a .BED file, and plots graphs with these informations. May be used to plot a single sample's coverage, as well as to calculate the mean, median and variance of several samples, and plot these informations on graphs.

Notes:

Every tick found on X axis represents either the first or the last base of the exon of the gene of choice. In the case of exome sequencing and some gene panels, points and lines will only appear inside these ticks, as only the exons are sequenced. Parts of the gene that had no coverage will appear as blank. The geneID argument must receive the gene name as it is written on the original file, or it will return an error. The measure argument is not used when the x argument has only has one file in it, as it is only has one coverage information, which will be plotted by itself in the graph.

Author(s):

Benedetti, A.

anna.detti@gmail.com

References:

UCSC FAQ for UCSC explanation on .bed files and bedTools coverage tool for a full explanation on the coverage tool of bedTools.

Examples:

#download and unzip files from datatests.zip into a folder of choice #change directory into this folder with setwd()

#for one file x ← read.table(“file1.bed”, sep = “\t”, as.is = T) #read your file plot.baseCoverage(x, geneID = “LHX3”)

#for more than one file x ← dir(path = “.”, pattern = “.bed”) #save several file names in one object plot.baseCoverage(x, geneID = “LHX3”, measure = “mean”)

Código

```
plot.baseCoverage <- function(x, geneID, measure){ ##função e seus
argumentos padrões
  if(missing(geneID)) { ##se não for colocado algo no argumento do geneID
    stop("A gene identification is necessary for this function") ##dar um
aviso que ele é um argumento necessário para que a função rode
  }
  if(class(x) == "data.frame"){ ##testa se o primeiro objeto inserido com um
argumento é um dataframe, que é o objeto criado pelo read.table
    genes.ids <- unique(x$V4) ##cria um objeto com todos os nomes de genes
como aparecem no arquivo
    if (any(geneID == genes.ids)){ ##se o argumento dado for encontrado no
arquivo, continuar
      gene <- x[x$V4 == geneID,] ##salva apenas a tabela do gene de
interesse no objeto, dado como argumento no geneID
      gene.min <- gene[1,2] ##determina a primeira base do gene
      gene.max <- gene[nrow(gene),3] ##determina a ultima base do gene
      gene$exact.base <- gene$V2 + gene$V7 ##cria uma coluna na tabela do
gene correspondente a cada uma das bases que estão na tabela original
      dataf1 <- cbind(gene$exact.base, gene$V8) ##cria um dataframe com a
cobertura em cada base e sua base correspondente
      gene.extension <- seq(gene.min, gene.max) ##cria um vetor com cada
base no gene inteiro (incluindo introns, se não estiverem na tabela
original)
      fragments <- c(unique(gene$V2), unique(gene$V3)) ##cria um objeto com
o começo e o final de todos os fragmentos, ou seja, inicio e fim de todos os
exons
      fragments <- sort(fragments) ##ordenar os fragmentos em ordem numérica
      dataf2 <- NULL ##cria um objeto nulo para:
      dataf2 <- cbind(dataf2, gene.extension) ##cria um dataframe com todas
as bases do gene inteiro. a ideia é que se tiverem bases que não tiveram
leitura, não se tenha um ponto para elas. por isso:
      dataf3 <- merge(dataf2, dataf1, by.x = "gene.extension", by.y = "V1",
all = T) ##junta os dois dataframes anteriores criados em um só pela base,
colocando NA nas bases que não possuem leitura
      par(las = 1, bty = "l") ##especifica para o gráfico o tipo de caixa,
números na horizontal
      plot(dataf3, type = "l", xaxt = "n", ##plota o gráfico final, com uma
linha ligando todos os pontos, sem marcas no eixo X
         ylim = range(0:max(gene$V8)), ##padroniza o eixo Y para que seja
possível se ver quais bases chegam até 100,
         xlab = geneID, ylab = "Coverage", main = "Sample") ##nomeia os
eixos X e Y e o gráfico
      axis(1, at = fragments, labels = F) ##coloca marcas em cada começo e
fim dos fragmentos no eixo X do gráfico
    } else { ##caso o gene dado no argumento não seja encontrado no arquivo
      stop("The geneID was not found in the file! Please make sure it is the
same as appears on your file or it is not misspeled!") ##retornar mensagem
de erro
    }
  } else { ##caso o primeiro if retorne negativo, entrar na opção de vários
arquivos
```

```
    ##criação de objetos que serão utilizados após o for
    coverages <- NULL ##para salvar as coberturas
    gene.extension <- NULL ##para salvar a extensão do gene
    exact.base <- NULL ##para salvar a base exata de onde se tem cobertura
    for (i in x){ ##para cada um dos arquivos cujo nomes estão em x
      table <- read.table(i, header = F, sep = "\t", as.is = T) ##leitura
dos arquivos como dataframes
      genes.ids <- unique(table$V4) ##cria um objeto com todos os nomes de
genes como aparecem no arquivo
     if (any(geneID == genes.ids)){ ##se o argumento dado for encontrado no
arquivo, continuar
        gene <- table[table$V4 == geneID,] ##salva apenas a tabela do gene
de interesse dado no argumento geneID no objeto
        gene.min <- gene[1,2] ##determina a primeira base do gene
        gene.max <- gene[nrow(gene),3] ##determina a ultima base do gene
        exact.base <- gene$V2 + gene$V7 ##cria um vetor que possua cada uma
das bases que possuem cobertura
        fragments <- c(unique(gene$V2), unique(gene$V3)) ##cria um objeto
com todos os inícios e fins dos fragmentos, ou seja, o inicio e o fim dos
exons, que são a parte que terão cobertura no caso do exoma
        fragments <- sort(fragments) ##ordena esse objeto em ordem numérica
        coverages <- cbind(coverages, gene$V8) ##cria dataframe apenas com
as coberturas de todos os arquivos
        gene.extension <- seq(gene.min, gene.max) ##cria um vetor com cada
base no gene inteiro (incluindo introns, se eles não estiverem na tabela
original)
    } else { ##caso o gene dado no argumento não seja encontrado no arquivo
        stop("The geneID was not found in the file! Please make sure it is
the same as appears on your file or it is not misspeled!") ##retornar
mensagem de erro
      }
        }
    dataf2 <- NULL ##cria um objeto nulo para:
    dataf2 <- cbind(dataf2, gene.extension) ##transforma o gene.extension
criado anteriormente em dataframe
    par(las = 1, bty = "l") ##padroniza para todos os gráficos feitos o tipo
de caixa e os números nas horizontais nos eixos
    if(missing(measure)){ ##se não tiver o argumento med, retornar os 3
gráficos
        media <- apply(coverages, 1, mean) ##aplicar a função média
escolhida no argumento
        dataf1a <- cbind(exact.base, media) #juntar as médias com as bases
aonde se encontram em um dataframe
        dataf3a <- merge(dataf2, dataf1a, by.x = "gene.extension", by.y =
"exact.base", all = T) ##juntar os dois dataframes criados em um só pela
base, afim de se ter um dataframe no qual não se tem nada nas bases que não
tiveram cobertura nenhuma
        plot(dataf3a, type = "l", xaxt = "n", ##plota o gráfico com a média
das bases e retira as marcas e números no eixo X
             ylim = range(0:max(media)), ##padroniza o eixo Y
```

```
             xlab = "geneID", ylab = "Coverage", main = "Mean Coverage")
##nomeia o gráfico e seus eixos
        axis(1, at = fragments, labels = F) ##coloca marcas em cada começo e
fim dos fragmentos no eixo X do gráfico
        mediana <- apply(coverages, 1, median) ##aplicar a função média
escolhida no argumento
        dataf1b <- cbind(exact.base, mediana) #juntar as médias com as bases
aonde se encontram em um dataframe
        dataf3b <- merge(dataf2, dataf1b, by.x = "gene.extension", by.y =
"exact.base", all = T) ##juntar os dois dataframes criados em um só pela
base, afim de se ter um dataframe no qual não se tem nada nas bases que não
tiveram cobertura nenhuma
        par(ask=T) ##pergunta antes de plotar os gráficos, pedindo para que
o enter seja apertado
        plot(dataf3b, type = "l", xaxt = "n", ##plota o gráfico com a
mediana e retira os números do eixo X
             ylim = range(0:max(mediana)), ##padroniza o eixo Y
             xlab = geneID, ylab = "Coverage", main = "Median Coverage")
##nomeia o gráfico e seus eixos
        axis(1, at = fragments, labels = F) ##coloca marcas em cada começo e
fim dos fragmentos no eixo X do gráfico
        dataf1c <- cbind(exact.base, coverages) #juntar as médias com as
bases aonde se encontram em um dataframe
        plot(NULL, ##plota o gráfico vazio
             ylim = range(0:max(coverages)), xlim = range(gene.min,
gene.max), ##especifica os limites dos eixos X e Y
             xaxt = "n", ##retira as marcas do eixo X
             xlab = geneID, ylab = "Coverage", main = "Variance") ##nomeia o
gráfico e os dois eixos
        axis(1, at = fragments, labels = F) ##coloca marcas em cada começo e
fim dos fragmentos no eixo X do gráfico
        for(i in 2:ncol(dataf1c)){ ##entra em um for para plotar todos os
pontos de cobertura encontrados no dataframe
          points(dataf1c[,1], dataf1c[,i], pch = 20) #plota cada um dos
pontos no gráfico
        }
        lines(dataf3a, col = "red") ##plota a linha de média de referência
      }
      else if(measure == "mean"){ ##ou, caso o argumento dado em med for
"mean", ou seja, a média
        media <- apply(coverages, 1, mean) ##aplica a função média a cada
linha do dataframe, para determinar a média de cobertura de cada base
        dataf1 <- cbind(exact.base, media) #junta as médias com suas bases
em um dataframe
        dataf3 <- merge(dataf2, dataf1, by.x = "gene.extension", by.y =
"exact.base", all = T) ##juntar os dois dataframes criados em um só pela
base, afim de se ter um dataframe no qual as bases que não tiveram cobertura
são NAs
        plot(dataf3, type = "l", xaxt = "n", ##plota o gráfico com a média
das bases e retira os números do eixo X
           ylim = range(0:max(media)), ##padroniza o eixo Y
```

```
           xlab = geneID, ylab = "Coverage", main = "Mean Coverage")
##nomeia o gráfico e seus eixos
        axis(1, at = fragments, labels = F) ##coloca marcas em cada começo e
fim dos fragmentos no eixo X do gráfico
      }
      else if(measure == "median"){ ##ou, caso o argumento dado em med for
"median", ou seja, mediana
        mediana <- apply(coverages, 1, median) ##aplica a função mediana em
cada linha do dataframe, e guarda em um objeto com apenas essas medianas
        dataf1 <- cbind(exact.base, mediana) #junta as medianas e as bases
correspondentes em um dataframe
        dataf3 <- merge(dataf2, dataf1, by.x = "gene.extension", by.y =
"exact.base", all = T) ##junta os dois dataframes criados em um só pela
base, afim de se ter um dataframe no qual as bases que não tiveram
coberturas apresentam NAs
        plot(dataf3, type = "l", xaxt = "n", ##plota o gráfico com a mediana
e retira os números do eixo X
           ylim = range(0:max(mediana)), ##padroniza o eixo Y
           xlab = geneID, ylab = "Coverage", main = "Median Coverage")
##nomeia o gráfico e seus eixos
        axis(1, at = fragments, labels = F) ##coloca marcas em cada começo e
fim dos fragmentos no eixo X do gráfico
      }
      else if(measure == "variance" || measure == "var"){ ##ou, caso o
argumento dado em med for "var", ou seja, variancia
        dataf1 <- cbind(exact.base, coverages) #juntar as coberturas com as
bases
        media <- apply(coverages, 1, mean) ##aplica a função média a cada
linha do dataframe, para determinar a média de cobertura de cada base
        datafm <- cbind(exact.base, media) #junta as médias com suas bases
em um dataframe
        datafm <- merge(dataf2, datafm, by.x = "gene.extension", by.y =
"exact.base", all = T) ##juntar os dois dataframes criados em um só pela
base, afim de se ter um dataframe no qual as bases que não tiveram cobertura
são NAs
        plot(NULL, ##plota o gráfico vazio
           ylim = range(0:max(coverages)), xlim = range(gene.min, gene.max),
##especifica os limites dos eixos X e Y
           xaxt = "n", ##retira as marcas do eixo X
           xlab = geneID, ylab = "Coverage", main = "Variance") ##nomeia o
gráfico e os dois eixos
        axis(1, at = fragments, labels = F) ##coloca marcas em cada começo e
fim dos fragmentos no eixo X do gráfico
        for(i in 2:ncol(dataf1)){ ##entra em um for para plotar todos os
pontos de cobertura encontrados no dataframe
          points(dataf1[,1], dataf1[,i], pch = 20) #plota cada um dos pontos
no gráfico
        }
        lines(datafm, col = "red") ##plota a linha de média de referência
      }
```

```
  }
}
```