CURSO DE INTRODUÇÃO À BIOINFORMÁTICA APLICADA A GENÔMICA

De 28/09 a 09/10 de 2015

ANÁLISE DE EXPRESSÃO DIFERENCIAL EM TRANSCRIPTOMAS

Rogério F. de Souza

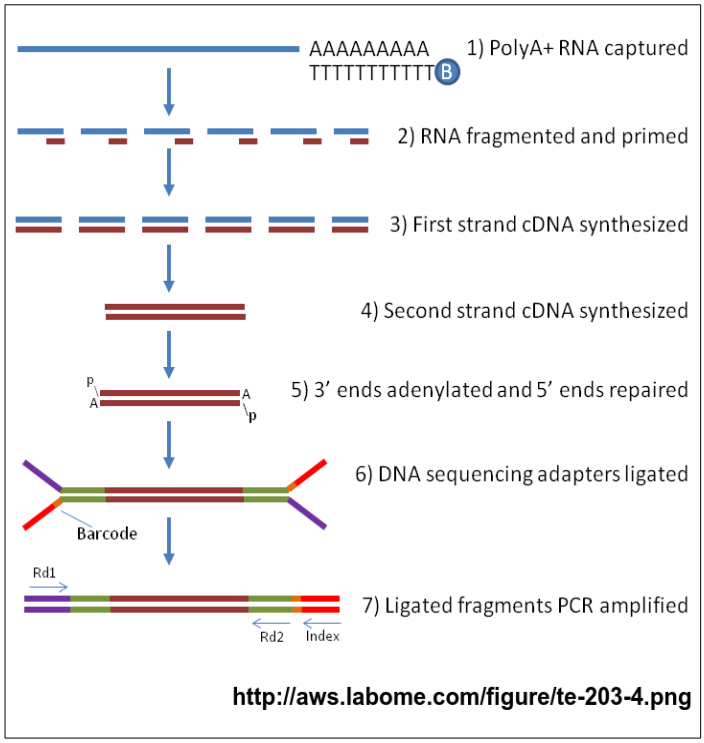
Sumário

1 – Introdução

O que seriam os RNAs-seq?

RNAs-seq é o nome dado a as novas tecnologias de sequenciamento (*Next-generation sequencing*) aplicadas aos transcriptomas, ou seja, às regiões do DNA transcritas em moléculas de RNAs.

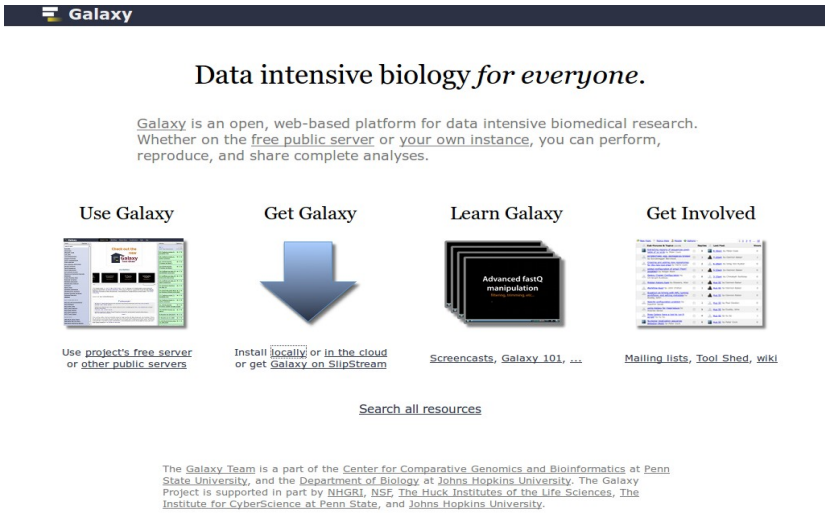
Os métodos de sequenciamento Illumina e Torrent usam uma metodologia de sequenciamento via síntese. Neste processo, as moléculas de RNAs sintetizadas por um determinado organismo ou de um determinado tecido são fragmentadas para então serem transformadas em pequenas moléculas de DNA (cDNA), gerando então uma biblioteca de cDNA. Com essa técnica é possível:



- Fazer uma análise da expressão gênica (dos éxons e das diferentes isoformas dos locos que sofrem processamento alternativo);
- Descobrir novas regiões dos genomas que são transcritas;
- Estudar a expressão diferencial de diferentes locos em diferentes tecidos ou condições ambientais;
- Identificar moléculas de RNAs que participam de processos regulatórios etc.

Alguns aspectos das metodologias de análise dos RNAs-seq

Quando os transcriptomas são submetidos ao sequenciamento do tipo Illumina, são geradas milhões de pequenas sequências de cDNAs. Dependendo do organismo, eucarioto ou procarioto, os arquivos produzidos neste processo costumam ser muito grandes, comumente com mais de 10 ou 20 GB. Assim, o grande desafio para os bioinformatas é processar e manipular tais tipos de arquivos. Existem uma série de programas pagos e gratuitos disponíveis para essas tarefas, bem como vários tutoriais na Internet sobre o seu funcionamento e a forma de utilização. Os arquivos gratuitos permitem um maior controle das diferentes etapas de estudo. Porém, muitas dessas análises precisam ser feitas em terminal GNU/Linux, via linha de comando, o que, por um lado pode ser complicado para iniciantes. Mas, por outro, essa forma de trabalho costuma ser mais rápida e eficiente principalmente quando é necessário lidar com arquivos contendo uma quantidade muito grande de dados. Também existem uma série de plataformas *on line*, como a Galaxy (https://galaxyproject.org/), que reúne uma série de programas que podem ser bastante úteis neste tipo de análise:



Assim, muitas dessas análises podem ser feitas *online*. Porém, neste caso, podem haver limitações para a transferência de arquivos, devido ao tamanho destes e a velocidade da conexão de internet. Além disso, também pode-se encontrar problemas com o espaço disponibilizado por essas plataformas para as análises. Por outro lado, muitos desses programas de análise podem ser baixados para computadores ou servidores do próprio usuário.

Quais são os passos para o estudo dos RNAs-seq?

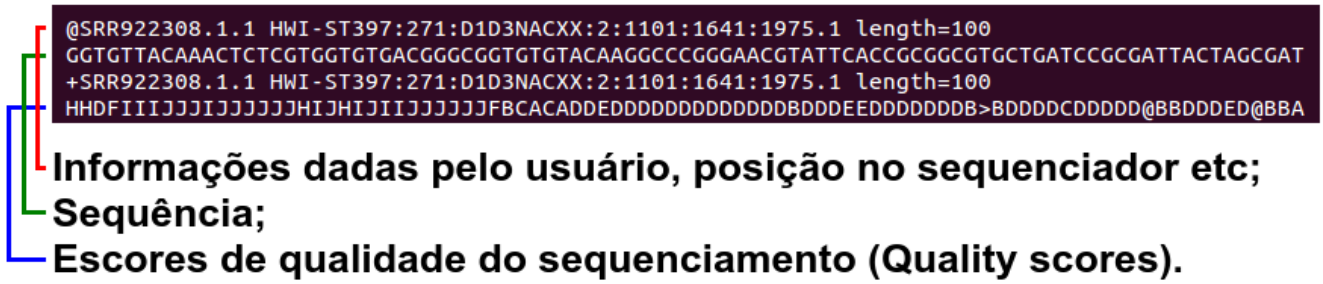
As diferentes tecnologias e protocolos de sequenciamento atualmente disponíveis costumam compartilhar os mesmos passos de pré-processamento e de análise (Dillies *et al.*, 2012), descritos a seguir:

- Os *short reads* (sequências curtas) provenientes do sequenciamento são pré processados, a fim de remover os adaptadores e as sequências com baixa qualidade, sendo em seguida mapeados em um genoma de referência ou a um genoma alinhado;
- O nível de expressão é estimado para cada transcrito (por exemplo, para cada loco);
- Os dados são normalizados;
- Uma análise estatística é usada para identificar os transcritos diferencialmente expressos.

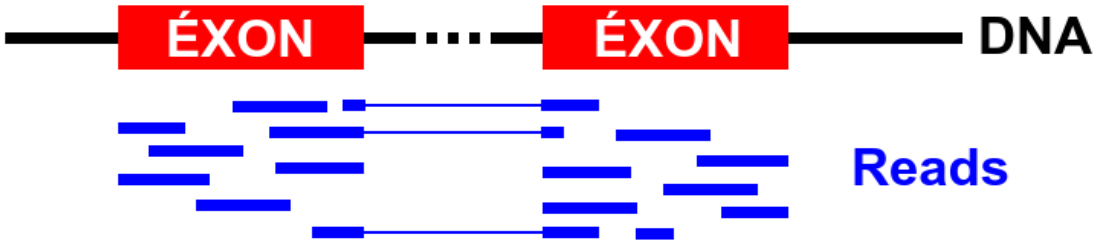
Que tipos de arquivos normalmente são usados nessas análises?

Arquivo do tipo FASTQ

Os arquivos fastq são gravados em formato texto, sendo gerados durante o processo de sequenciamento:



Um arquivo FASTQ representa uma biblioteca das sequências de cDNAs produzidas a partir de uma população de moléculas de RNAs. Lembrando que, pelo método de sequenciamento Illumina, esse arquivo conterá milhões de trechos pequenos, também chamadas de ***reads***, que devem apresentar homologia com as sequências do DNA transcritas em RNAs:



Para cada trecho sequenciado, também são colocadas outras informações, inclusive, sobre a qualidade do sequenciamento. Para cada base sequenciada, esta qualidade é dada por um símbolo em ASCII específico, onde "!" e "~" indicam menor e maior qualidade, respectivamente:

Trecho sequenciado:	**GCCAGTGATGGTACCGATTATATATG**
Qualidade desse sequenciamento:	***!"%&_`ABCDEPQRSTUVWXYZ{|}~***

Os escores de qualidade indicam a probabilidade de que uma determinada base tenha sido corretamente identificada durante o sequenciamento:

Phred Quality Score	Probability of incorrect base call	Base call accuracy
10	1 in 10	90%
20	1 in 100	99%
30	1 in 1000	99,9%
40	1 in 10,000	99,99%
50	1 in 100.000	99,999%
60	1 in 1.000.000	99,9999%

Fonte: https://en.wikipedia.org/wiki/Phred_quality_score

Arquivos do tipo FASTA

Os arquivos no formato FASTA contém um nome de entrada, seguido de uma sequência nucleotídica específica, que pode representar um loco completo, um éxon, um retroelemento etc:

```
>ncRNA_Candidate_11 ncRNA_Candidate_11 undefined product 174448:174960 forward
ttgtgaaggcctttcacaagtataattttatcattttattgggaaaaagtaataaaaaaa
cataactagttaatgttttttttggatattatgttatttaactagatagttattaaagtta
aatgttatagatatactgttattacataataatttttatatgatttgttattttattaga
ttttctaatggattatttctttagctgattttgattacaaagttaaggtattctaatcat
gttttcttgatttataacatgtaaaggttagaataaattcgttggggttagacagacgat
atttgttgtttttatgaaaaaattgtgggtattagatatgtatatttttttcgataactat
ttactctatggtaaaaattctaagctgtttgtttggtgttatttcttaataaaaataagg
tagtcatgtataataactatatttaaaagttgtatctttaggaatgttgttcattaata
aaaagtgtagtcaattgactacacttttttattg
```

Os arquivos FASTA são usados para se tentar descobrir se, na nossa biblioteca de cDNAs, ou seja, nos arquivos FASTQ, determinadas sequências de interesse estão sendo expressas e, caso positivo, com que intensidade.

Arquivos do tipo SAM/BAM

SAM (*Sequence Alignment/Map*) é um formato genérico para a estocagem de grandes quantidades de alinhamentos de nucleotídeos. Eles são arquivos de tabela gravados em formato de texto e que contém as informações sobre os alinhamentos das sequências que foram buscadas na biblioteca de cDNAs, ou seja, em nossos arquivos FASTQ. Por exemplo, neste tipo de arquivo é informado quantos trechos presentes em nossa biblioteca de cDNA foram compatíveis com cada sequência FASTA que estamos buscando. Um arquivo no formato SAM é separado por tabulações e possui um cabeçalho (opcional) e uma seção de alinhamento. A linha do cabeçalho, quando presente, começa com "@", sendo seguida pelas linhas do alinhamento. Cada linha de alinhamento possui 11 campos informacionais obrigatórios (tais como as posições das sequências ou *reads* em relação a sequência buscada) e um número variável de campos contendo outras informações:

```
SRR922313.12748.1     0     gbs0210 262   255   91M     *     0     0     TTAGAAGTTGACCCAGATTTCCGCCGATTCATTGAAACGTGCTGGACTTCTTACACGTGACGCACGTATGGTTGAACGTAAAAAACCAGGTC     HHHHJJJJJIJJJJJJIJJJI$
SRR922313.24260.1     16    gbs0210 248   255   91M     *     0     0     TCTCACGTGCACTTTTAGAAGTTGACCCAGATTTCCGCGATTCATTGAAACGTGCTGGACTTCTTACACGTGACGCACGTATGGTTGAACG     ABADDBDDDC>ACCCCDDCA>$
SRR922313.31452.1     0     gbs0210 241   255   91M     *     0     0     CATGGTATCTCACGTGCACTTTTAGAAGTTGACCCAGATTTCCGCGATTCATTGAAACGTGCTGGACTTCTTACACGTGACGCACGTATGG     HHHHJGGIIJIIIIHIIJJJJJ$
SRR922313.36757.1     0     gbs0210 173   255   91M     *     0     0     AAGGTTCATACGATGTTTTTGTTAACGTTGTAGGTGGTGGATATGCAGGTCAATCAGGTGCGATCCGTCATGGTATCTCACGTGCACTTTT     HHHHHIIJJJJJJJJJJJJJJJJ$
SRR922313.54731.1     0     gbs0210 292   255   91M     *     0     0     TTGAAACGTGCTGGACTTCTTACACGTGACGCACGTATGGTTGAACGTAAAAAACCAGGTCTTAAGAAAGCTCGTAAAGCTAGCCAGTTCT     FFFDHIGHAGCBFHGGGGGCFB$
SRR922313.56365.1     16    gbs0210 313   255   75M12I4M      *     0     0     0       ACACGTGACGCACGTATGGTTGAACGTAAAAAACCAGGTCTTAAGAAAGCTCGTAAAGCTAGCCAGTTCTCAAAACGTTAAGAACCGCTTT     DDDDDDDDDBBCD$
SRR922313.57310.1     0     gbs0210 218   255   91M     *     0     0     CAGGTCAATCAGGTGCGATCCGTCATGGTATCTCACGTGCACTTTTAGAAGTTGACCCAGATTTCCGCGATTCATTGAAACGTGCTGGACT     HHHHHHIJIJJJJJHGIIIIJJ$
SRR922313.114410.1    0     gbs0210 291   255   91M     *     0     0     ATTGAAACGTGCTGGACTTCTTACACGTGACGCACGTATGGTTGAACGTAAAAAACCAGGTCTTAAGAAAGCTCGTAAAGCTAGCCAGTTC     GHHDIIEHIIJJJJJJIJJJJJ$
SRR922313.152678.1    16    gbs0210 206   255   91M     *     0     0     GTGGTGGATATGCAGGTCAATCAGGTGCGATCCGTCATGGTATCTCACGTGCACTTTTAGAAGTTGACCCAGATTTCCGCGATTCATTGAA     BDDDDDDDDDDCDDDDDCDDDD$
SRR922313.155423.1    16    gbs0210 301   255   91M     *     0     0     GCTGGACTTCTTACACGTGACGCACGTATGGTTGAACGTAAAAAACCAGGTCTTAAGAAAGCTCGTAAAGCTAGCCAGTTCTCAAAACGTT     ADDDDCDDDCA<DBBDDDDDD$
SRR922313.179506.1    16    gbs0210 271   255   91M     *     0     0     GACCCAGATTTCCGCGATTCATTGAAACGTGCTGGACTTCTTACACGTGACGCACGTATGGTTGAACGTAAAAAACCAGGTCTTAAGAAAG     CCCA>DCDDDDDBDDDEDDCD$
SRR922313.195279.1    16    gbs0210 247   255   91M     *     0     0     ATCTCACGTGCACTTTTAGAAGTTGACCCAGATTTCCGCGATTCATTGAAACGTGCTGGACTTCTTACACGTGACGCACGTATGGTTGAAC     CDDDDDDDDDCCCDCDDDDDDD$
```

Embora possa ser utilizado por uma série de programas, arquivos neste formato costumam ser maiores, ocupando grande espaço no computador. Além disso, alguns programas trabalham com arquivos contendo sequências alinhadas em outros formatos. Por exemplo, arquivos no formato **BAM** (Binary *Alignment/Map*), pelo fato de serem estocadas em formato binário, além de serem mais compactos, podem ser mais facilmente usado por diferentes programas de análise de sequências. Os arquivos no formato BAM (Binary *Alignment/Map*) contém as mesmas informações dos arquivos SAM (*Sequence Alignment/Map*). Entretanto, eles são comprimidos em formato BGZF (um formato de compressão padrão do gzip), não podendo ser visualizados por editores de texto. Algumas informações mais aprofundadas sobre programas, metodologias etc podem ser conseguidas no seguinte endereço:

http://www.labome.com/method/RNA-seq-Using-Next-Generation-Sequencing.html

2 – Preparando as bibliotecas de cDNA para a análise

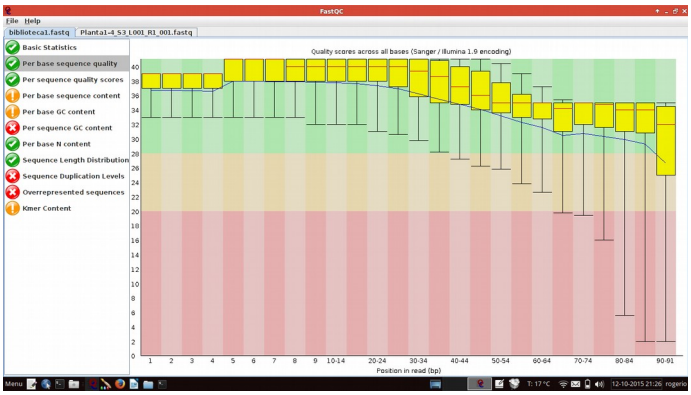
Para a busca por sequências específicas, é preciso que as bibliotecas de cDNA, gravadas em formato FASTQ, sejam limpas. A limpeza e o preparo das bibliotecas de cDNA antes de serem analisadas são necessários porque:

- Durante o processo de sequenciamento são usados adaptadores – pequenas sequências de DNA/RNA que permitem a amplificação dos genomas – que precisam ser retirados antes das análises subsequentes;
- Muitos trechos sequenciados podem apresentar baixa qualidade – ou seja, por algum motivo não foram sequenciados adequadamente – ou têm tamanho muito reduzido – o que pode interferir na montagem correta dos genomas ou na identificação de sequências de interesse.

a) Analisando a qualidade das bibliotecas de cDNA

O programa **FastQC** permite verificar diferentes parâmetros de qualidade das bibliotecas de cDNA. Ele facilita a determinação do tipo de limpeza a que cada biblioteca de cDNA deverá ser submetida, tais como a retirada:

- Das sequências dos adaptadores usados na construção da biblioteca;
- Das caudas poli-A dos RNAs mensageiros;
- De trechos que foram mal sequenciados e que apresentam baixa qualidade;
- Dos trechos sequenciados, mas de tamanho muito reduzido etc.



Observações

- O programa FastQC serve apenas de verificação da qualidade das bibliotecas, não permitindo a sua manipulação e, portanto, a sua limpeza;
- Este programa pode ser instalado diretamente em um terminal:

  ~$ sudo apt-get install fastqc

- Ou então, pode ser baixado do seguinte endereço:

  http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

b) Limpando as bibliotecas de cDNA

O programa **PRINSEQ** permite analisar a qualidade das bibliotecs de cDNA e fazer os cortes e ajustes necessários. Pode-se fazer o *upload* das sequências (compactadas em diferentes formatos) para se realizar a análise *online* no sítio do programa ou instalá-lo em um computador com GNU/linux, utilizando comandos no terminal.

Usando o PRINSEQ na internet

- Escolher o formato do arquivo (por exemplo, em fastq);
- Fazer o *upload* (este pode estar compactado);
- Escolher os parâmetros de limpeza:

- O tamanho mínimo das sequências a serem mantidas;
- Os trechos de cada sequência que serão cortados (ex: para a retirada das regiões adaptadoras);
- A manutenção apenas das sequências com um escore mínimo de qualidade etc;

http://prinseq.sourceforge.net/

Usando o PRINSEQ no computador via terminal

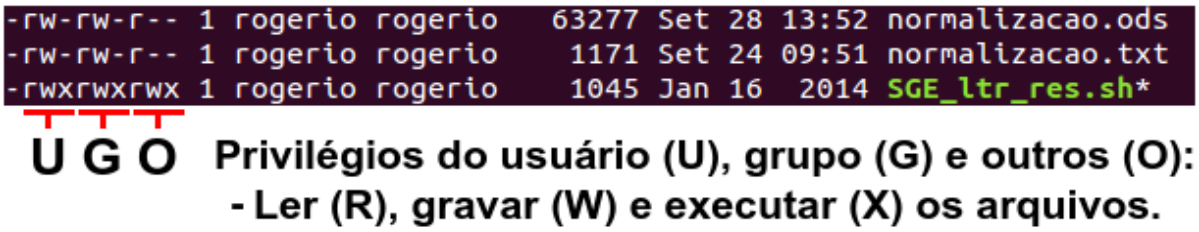
Depois de descompactado, é necessário ir até o diretório onde este foi colocado e mudar a permissão de uso do programa para torná-lo executável (via terminal):

~$ chmod 766 prinseq-lite.pl

~$ chmod 766 prinseq-graphs.pl

Observações

- O comando **chmod** permite mudar os privilégios que o indivíduo, o grupo e outros usuários têm para manipular arquivos e programas:



- As permissões rwx (read, write e execute) para cada um dessas categorias são: (0) permissão negada para qualquer atividade; (1) permissão de execução (para caso do arquivo ser um programa); (2) permissão de gravação do arquivo; (3) permissão de gravação e execução do arquivo (para programas); (4) permissão de leitura do arquivo; (5) permissão de leitura e execução do arquivo (para programas); (6) permissão de leitura e gravação do arquivo e (7) permissão de leitura, gravação e execução do arquivo (para programas). Por exemplo, para permitir que o usuário, o grupo e usuários externos possam utilizar o prinseq, basta digitar no terminal o seguinte comando:

~$ chmod 777 prinseq-lite.pl

- Para permitir que apenas o usuário e o grupo leiam, escrevam e executem esse arquivo, deve-se digitar:

~$ chmod 770 prinseq-lite.pl

Para saber quais comandos utilizar com o PRINSEQ

~$ perl ./prinseq-lite.pl -h

Usando o comando do PRINSEQ para a limpeza da biblioteca

No diretório em que estiver instalado o PRINSEQ, depois de se saber quais tipos de limpeza deverão ser feita:

```
~$ perl prinseq-lite.pl -fastq nome_da_biblioteca.fastq -out_format 3 -min_len 40 -trim_left 9
-trim_qual_right 20
```

Significado de cada opção

- **-fastq →** informa que o arquivo a ser limpo (biblioteca.fastq) está no formato fastq;
- **-out_format 5 →** indica o formato de saída que se deseja: 1 (fasta), 2 (fasta e qual), 3 (fastq), 4 (fastq e fasta), 5 (fastq, fasta e qual);
- **-min_len 40 →** elimina as sequências menores que X nucleotídeos (neste caso X = 40);
- **-trim_left 9 →** corta X (neste caso X = 9) nucleotídeos à esquerda (devido a baixa qualidade média dessa região apontada pelo FastQC ou a presença de adaptadores);
- **-trim_qual_right 20 →** elimina as sequências com escore de qualidade inferior a X (neste caso X = 20).

Pode-se trabalhar com arquivos compactados (gzip)

Neste caso, aciona-se o gzip antes de executar o programa prinseq:

```
~$ gzip -dc nome_da_biblioteca.fastq.gz | perl prinseq-lite.pl -fastq nome_da_biblioteca.fastq
-out_format 3 -min_len 40 -trim_left 9 -trim_qual_right 20
```

Onde:

- **gzip →** programa de compactação;
- **-d →** informa ao gzip que este deve descomprimir o arquivo;
- **-c →** informa ao gzip que este mantenha o arquivo original sem modificações (ou seja, compactado), produzindo um arquivo de saída descompactado;
- **| (pipe) →** permite que um segundo processo se inicie automaticamente após a finalização do primeiro processo.

3 – Preparando as sequências a serem buscadas

Uma biblioteca de cDNA conterá sequências de diferentes tipos de transcritos. Sendo assim, é necessário ter uma estratégia de busca dos segmentos de interesse. No nosso caso, cada sequência que será buscada na biblioteca de cDNA, estará em formato FASTA, em arquivos individuais.

a) Criando arquivos fasta individuais

Diferentes bancos proteômicos armazenam dados de sequências gênicas no formato FASTA que podem ser utilizadas em nossos estudos. Se estas sequências estiverem guardadas em um único arquivo, é preciso criar arquivos FASTA individuais para então iniciarmos a busca em nossa biblioteca de cDNA.

```
>ncRNA_Candidate_1 ncRNA_Candidate_1 undefined product 20567:20900 forward
gcctcctaaaaggtaacggaggcgcccaaaggttccctcagactggttggaaatcagtcg
tagagtgtaaaggtataagggagcttgactgcgagagctacaactcgagcagggacgaaa
gtcgggcttagtgatccggtggttccgtatggaagggccatcgctcaacggataaaagct
accctggggataacaggcttatctcccccaagagttcacatcgacggggaggtttggcac
ctcgatgtcggctcgtcgcatcctggggctgtagtcggtcccaagggttgggctgttcgc
ccattaaagcggcacgcgagctgggttcagaacg
>ncRNA_Candidate_2 ncRNA_Candidate_2 undefined product 26399:26732 forward
gcctcctaaaaggtaacggaggcgcccaaaggttccctcagactggttggaaatcagtcg
tagagtgtaaaggtataagggagcttgactgcgagagctacaactcgagcagggacgaaa
gtcgggcttagtgatccggtggttccgtatggaagggccatcgctcaacggataaaagct
accctggggataacaggcttatctcccccaagagttcacatcgacggggaggtttggcac
ctcgatgtcggctcgtcgcatcctggggctgtagtcggtcccaagggttgggctgttcgc
ccattaaagcggcacgcgagctgggttcagaacg
>ncRNA_Candidate_3 ncRNA_Candidate_3 undefined product 28551:28999 forward
ccaagcatttgcttgggcgcgtagctcaggtggttagagcgcacgcctgataagcgtgag
gtcggtggttcgagtccactcgtgcccattaatattaatttggagaattactcaagaggc
tgaagaggacggtttgctaaatcgttaggtcgggtaactggcgcaagggttcgaatccct
tattctccgtaaatattataattttaagagtaattatactctttttttatttctctttta
gtttcactatggagtaattatattgatagagtgaatagaagaaggttatcgttttctaga
tttgattatggttcatttagaaatagttgatttcaaatataaatgtaacaatgacgtaat
ataatttcgtgaatttttttggtaaaatatttaatgtcttatcgtaaaggagtagtttaat
aaaaatgaaaaaaaagaatattatcagcag
```

Quando temos uma grande quantidade de sequências a serem a serem buscadas, podemos usar

uma ferramenta presentes no Emboss, para criarmos esses arquivos individuais. O Emboss, que funciona em GNU/Linux, fornece uma série de ferramentas que permitem manipular esses e outros tipos de arquivos.

About • Applications • GUIs • Servers • Downloads • Licence • User docs • Developer docs • Administrator docs • Get involved • Support • Meetings • News • Credits

EMBOSS was most recently funded from May 2009 to Dec 2011 by BBSRC grant BBR/G02264X/1

Funded from May 2006 to April 2009 by BBSRC grant BB/D018358/1

Caso o computador não possua o Emboss, se o usuário tiver direitos de administrador (root), basta digitar o seguinte comando:

```
~$ sudo apt-get install emboss
```

No nosso caso, podemos usar o **seqretsplit** para gerar os arquivos FASTA individuais. Este permite dividir o arquivo fasta total em arquivos individuais, preservando os nomes de cada sequência. Para tanto, basta digitar no terminal o seguinte comando:

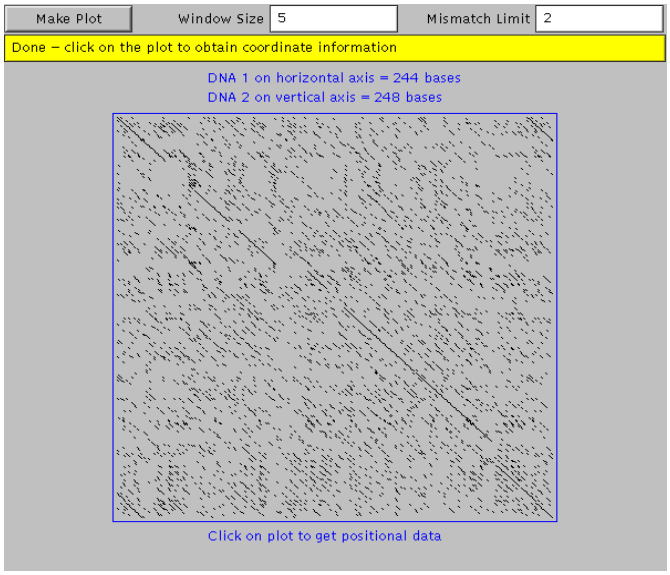
```
~$ seqretsplit nome_do_arquivo.fasta
```

Observação

- Aparecerá a mensagem "*Reads and writes (returns) sequences in individual files output sequence(s)*", mas basta dar enter para conseguir os arquivos individuais.

b) Analisando a integridade de cada sequência a ser buscada

Quando estamos produzindo as nossas próprias sequências FASTA, a partir do sequenciamento ou da análise de genomas específicos, é preciso verificar a disposição e a integridade destas. Por exemplo, no caso do estudo de retroelementos, precisamos analisar as regiões LTRs flanqueadoras, para tentar identificar se existem outros LTRs inseridos na sequência a ser analisada (*nested* LTRs) etc. Além disso, se ambos os LTRs estão intactos, mantendo uma sequência nucleotídica similar, e se não existirem muitas repetições ou mutações na região interna do LTR-RT, isso é um indicativo de que este está ativo. A manutenção da semelhança entre ambos os LTRs ocorre porque a metade final do primeiro (5') serve de molde para a produção da metade inicial do segundo (3') e vice-versa. O programa **Dotter** pode ser usado para verificar a integridade das sequências a serem buscadas em nossa biblioteca de cDNA.



O gráfico do Dotter mostra na diagonal central a sequência analisada e paralelo a ela as sequências duplicadas, assim, é possível definir a porção inicial e final do LTR-RT (o início tem uma

sequência 5´-TG-3´ e o final uma sequência 5´-CA-3´ em cada um dos 2 LTRs). Também é possível analisar a integridade da parte interna do LTR-RT, sendo que as linhas paralelas indicarão a integridade dos genes, a presença de mutações (como *snp*) etc. Quanto mais intacta a ORF (*Open Read Frame*), maior o indício de atividade do LTR-RT.

Exemplos de alguns comandos do Dotter a serem feitos no terminal linux

~$ dotter -l dot nome_do_arquivo.fasta nome_do_arquivo.fasta

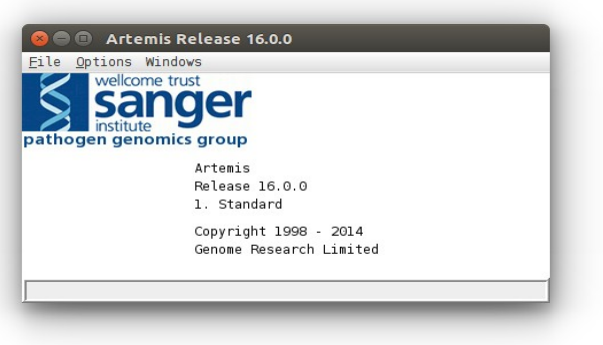
~$ dotter -b dot nome_do_arquivo.fasta nome_do_arquivo.fasta

Observação

- Deve-se repetir o nome do arquivo duas vezes quando se quiser comparar as suas sequências internas. No caso dos retroelementos, isso é relevante devido a presença de sequências repetidas, como as LTRs.

c) Caracterizando melhor cada sequência a ser buscada

Uma análise mais profunda das sequências FASTA pode ser feita, a fim de identificar, por exemplo, a presença de códons de término dentro de uma ORF ou, então para anotar os diferentes componentes dessas regiões, como promotores, introns, exons etc. Neste caso, podemos utilizar o programa Artemis:



Este programa pode ser baixado do seguinte endereço:

https://www.sanger.ac.uk/resources/software/artemis/

Para abrir a interface gráfica do Artemis via terminal, basta ir ao diretório onde este foi descompactado, mudar a permissão de uso do arquivo **arc** (uma programação em Shell que inicia o Artemis) para torná-lo executável:

~$ chmod 777 art

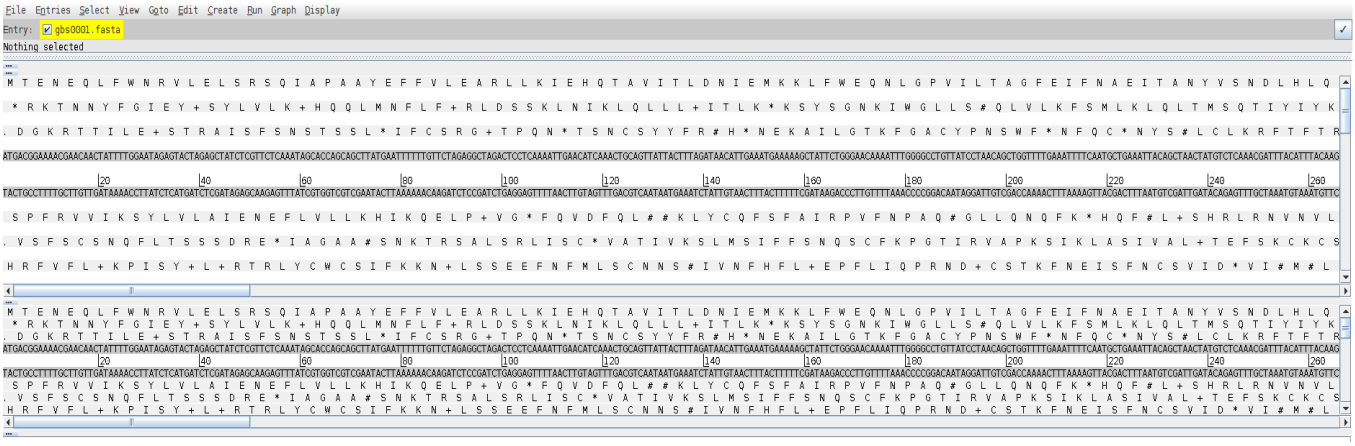
Em seguida, tecla-se:

~$ ./art &

Onde:

- **./ →** representa um comando que indica que arc é uma programação em shell que deverá ser iniciada;
- **& →** permite que o terminal fique livre para realizar outras atividades enquanto o programa acionado (no caso, o Artemis) estiver funcionando.

Este programa pode ser utilizado para demarcar regiões específicas em nossas sequências, como as regiões promotoras, íntrons, éxons etc. Automaticamente, ele também apresenta as três possíveis matrizes de leitura do trecho analisado, os seus respectivos aminoácidos, bem como os códons sem sentido (de término). Assim, no caso da nossa sequência de interesse representar uma ORF (*Open Read Frame* – ou seja, um trecho que costuma ser transcrito e que pode ser traduzido em peptídeo), podemos tirar conclusões importantes sobre a mesma. Por exemplo, a matriz de leitura correta da mesma deverá ser aquela que tenha apenas códons de término na sua parte final. Por outro lado, a presença de vários códons de término dentro de uma ORF nas diferentes matrizes de leitura, é um indicativo de que esta não seja funcional.



4 – Fazendo o alinhamento das sequências

Depois que todos os arquivos estão preparados para a análise, é possível iniciar a busca pelas sequências de interesse em nossa biblioteca de cDNA. Existem diferentes programas que podem fazer esse alinhamento, sendo um dos mais conhecido o BLAST (*The Basic Local Alignment Search Tool*). Entretanto, cada programa costuma se utilizar de algorítimos específicos, e pode servir a diferentes interesses de busca. No endereço abaixo são listados uma série de softwares que permitem fazer o alinhamento de sequências:

https://en.wikipedia.org/wiki/List_of_sequence_alignment_software

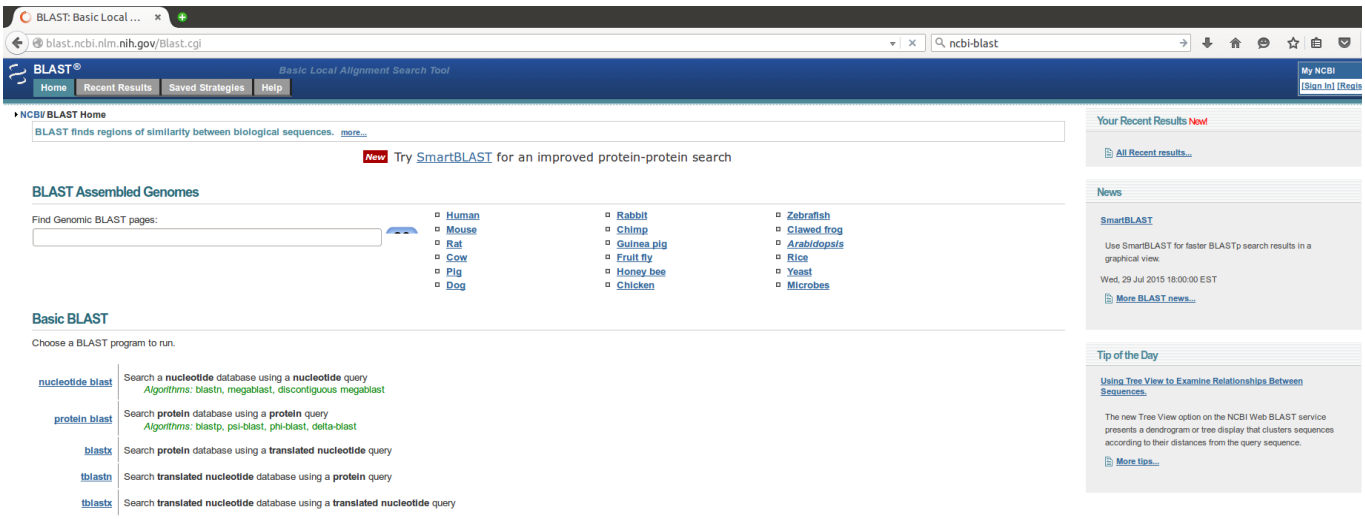
a) Alinhando sequências pelo Blast

BLAST significa Ferramenta de Busca por Alinhamento Local (*Basic Local Alignment Search Tool*) . Esta ferramenta é mantida pelo NCBI (*National Center for Biotechnology Information*), um banco público norte americano de biologia molecular que reúne sequências de proteínas e de nucleotídeos de diferentes espécies, bem como ferramentas que permitem o seu estudo, disponível no seguinte endereço:

http://www.ncbi.nlm.nih.gov/

O BLAST é um algoritmo que serve para compararmos sequências biológicas, como trechos de aminoácidos ou nucleotídeos de diferentes espécies. Embora menos eficiente que outros programas de alinhamento (como o Bowtie), o uso do Blast é importante para termos uma ideia da qualidade dos nossos dados, sendo que os seus resultados poderão ser usados para comparação com aqueles obtidos por outros programas. Na internet, o Blast está disponível no seguinte endereço:

www.ncbi.nlm.nih.gov/BLAST/



Porém, no nosso caso, é possível baixar e instalar no computador ou servidor essa ferramento de alinhamento, a partir do seguinte comando:

~$ sudo apt-get install ncbi-blast+

Dessa forma, poderemos buscar por sequências de interesse em nossa biblioteca de cDNA. Lembrando que isto dever ser feito somente depois de analisarmos a qualidade da nossa biblioteca (com o FastQC) e de eliminarmos dela as sequências curtas ou de má qualidade (com o PRINSEQ). Para tanto, é preciso realizar os seguintes passos:

1 – Converter a nossa biblioteca do formato fastq para fasta, pois este é o formato padrão usado pelo BLAST:

~$awk 'BEGIN{P=1}{if(P==1||P==2){gsub(/^[@]/,">");print}; if(P==4)P=0; P++}' nome_da_biblioteca.fastq > nome_da_biblioteca.fasta

Observação

- Mais informações sobre o programa awk serão dadas logo abaixo.

2 – Criar os arquivos DB (database) de referência do BLAST, a partir da biblioteca em fasta:

~$ makeblastdb -in nome_da_biblioteca.fasta -dbtype nucl

Onde:

- **in →** indica que o arquivo a seguir ser modificado;
- **-dbtype →** indica o tipo de arquivo da biblioteca fasta (proteína ou nucleotídeo – o padrão é proteína, por isso precisa colocar nucl).

Com isso, serão gerados os arquivos índices (com as extensões .nhr, .nin e .nsq) que serão utilizados pelo BLAST para fazer o alinhamento com as nossas sequências de interesse.

3 – Em seguida, é possível fazer a busca da sequência de interesse em nossa biblioteca:

~$ blastn -query nome_do_arquivo.fasta -db nome_da_biblioteca.fasta -out nome_do_arquivo.fasta.res -evalue 10e-20 -outfmt 6

Onde:

- **-query →** informa ao BLAST a sequência que será buscada na biblioteca;
- **-db →** informa ao BLAST a biblioteca onde será feita a busca;
- **-out →** informa ao BLAST o nome do arquivo de saída onde serão gravados os resultados;
- **-evalue 10e-20 →** determina os alinhamentos que serão mantidos no arquivo de saída, considerando um valor mínimo de probabilidade (quanto menor o evalue mais significativo é o escore e o alinhamento – ou seja, menor é a probabilidade de que o alinhamento tenha ocorrido apenas por acaso), sendo o default = 10;
- **-outfmt →** formato em que o arquivo de saída será gravado [Opções: 0 = pairwise, 1 = query-anchored showing identities, 2 = query-anchored no identities, 3 = flat query-anchored, show identities, 4 = flat query-anchored, no identities, 5 = XML Blast output, 6 = tabular, 7 = tabular with comment lines, 8 = Text ASN.1, 9 = Binary ASN.1, 10 = Comma-separated values, 11 = BLAST archive format (ASN.1)].

Como resultado teremos um arquivo no formato tabular contendo as informações sobre o alinhamento da nossa sequência fasta específica com as sequências presentes na nossa biblioteca:

gbs0001	SRR922308.22970895.1	100.00	91	0	0	738	828	1	91	2e-39	169
gbs0001	SRR922308.22905848.1	100.00	91	0	0	1045	1135	1	91	2e-39	169
gbs0001	SRR922308.22893067.1	100.00	91	0	0	70	160	1	91	2e-39	169
gbs0001	SRR922308.21109612.1	100.00	91	0	0	962	1052	1	91	2e-39	169
gbs0001	SRR922308.20753907.1	100.00	91	0	0	836	926	1	91	2e-39	169
gbs0001	SRR922308.20480445.1	100.00	91	0	0	963	1053	91	1	2e-39	169
gbs0001	SRR922308.19361126.1	100.00	91	0	0	698	788	1	91	2e-39	169

.....

Onde, cada uma dessas colunas representam, respectivamente:

- Query id, subject id, % identity, alignment length, mismatches, gap opens, q. start, q. end, s. start, s. end, evalue e bit score.

Observações

- No momento de se fazer o BLAST, pode-se usar a opção **-outfmt 7** para obter essa mesma tabela com as informações sobre cada campo;
- Entretanto, para quem deseja manipular essas informações via terminal, como veremos a seguir, é conveniente que o arquivo a ser trabalhado seja uniforme, ou seja, contenha apenas colunas separadas por um espaçador regular (tabulações, ponto e vírgula etc);
- Lembrando que é possível obter essas informações um dos seguinte comando:

~$ blastn -help ou blastx -help ou blastp - help

Extraindo informações da nossa tabela de resultados com o programa awk

O programa awk permite extrair informações de arquivos com padrões regulares, como tabelas, e reorganizar essas informações para geração de novas tabelas ou arquivos. Combinado com outros comandos (grep, cat etc) é possível extrair rapidamente uma série de informações importantes de um arquivo de resultados com milhares de informações. A seguir são dados alguns exemplos de comandos que podem ser usados para extrair informações do nosso arquivo resultados no formato de tabelas:

~$ cat nome_do_arquivo.fasta.res | awk '{print $1}'

- **Significado:** pegue o arquivo especificado e exiba a primeira coluna.

~$ awk -F';' '{print $2,$4}' nome_do_arquivo.fasta.res

- **Significado:** imprima as colunas 2 (sequência que alinhou com o query) e 4 (comprimento do alinhamento) do arquivo especificado, cujo separador de campos é ';'.

~$ awk '$3 >= “95” {print $3, $4}' nome_do_arquivo.fasta.res

- **Significado:** pegue na coluna 3 (% de identidade entre as sequências) do arquivo especificado os indivíduos com valores iguais ou superiores a 95 (semelhança genética) e exiba essas mesmas colunas.

~$ cat nome_do_arquivo.fasta.res | awk '$4 >= “80” {print $3, $4}'

- **Significado:** no arquivo especificado, se o resultado da coluna 4 (comprimento do alinhamento) for igual ou superior a 80, imprima as colunas 3 (% de identidade entre as sequências) e 4 (comprimento do alinhamento).

~$ awk '{if($3 >= 95 && $4 >= 80) print $2, $3}' nome_do_arquivo.fasta.res

- **Significado:** se no arquivo especificado a coluna 3 (% de identidade entre as sequências) for igual ou maior que 95 e a coluna 4 (comprimento do alinhamento) for igual ou maior que 80, imprima ambas.

~$ awk '{if($3 >= 95 && $4 >= 80) print $2, $3} END {print “Total =” NR}' nome_do_arquivo.fasta.res

- **Significado:** obtém o mesmo resultado anterior e, no final (comando END), exiba o total de registros (NR).

~$ awk '/100.00/ {print $3, $4} nome_do_arquivo.fasta.res

- **Significado:** procure pelo número 100.00 e exiba as respectivas colunas 3 e 4.

~$ awk '$3 ~/100/ {print $3, $4} nome_do_arquivo.fasta.res

- **Significado:** procure pelo número 100 apenas na coluna 3 e exiba as respectivas colunas 3 e 4.

~$ awk '$3 ~/100/ , /95/ {print $3, $4} nome_do_arquivo.fasta.res

- **Significado:** procure pelos valores entre 100 e 95 apenas na coluna 3 (semelhança genética) e exiba as respectivas colunas 3 e 4.

Alguns significados dos comandos utilizados

- **awk -f →** especifica o nome do arquivo que possui o conjunto de padrões a ser usado;
- **$1, $2 etc →** se referem a coluna 1 coluna 2 etc do arquivo a ser analisado;
- **$NF →** se refere sempre à última coluna do arquivo a ser analisado;
- **-F →** define quem é o separador de campos (o padrão é o espaço, mas pode ser tab (/t), ponto e

vírgula etc). Outra forma de informar isso:

- **BEGIN{RS = "/" ou "/t"}** → no primeiro caso o separador dos campos é a barra invertida e no segundo, tabulação;
- **BEGIN e END** → padrões especiais que indicam, respectivamente, o que o awk deve fazer antes de iniciar e depois de terminar a análise.

Operadores relacionais usados no awk

- **==** → igual a;
- **!=** → não igual a;
- **>** → maior que;
- **>=** → maior ou igual a;
- **<** → menor que;
- **<=** → menor ou igual a.

Operadores lógicos usados no awk

- **&&** → e (isso e isso);
- **||** → ou (isso ou aquilo);
- **!** → não (diferente disso).

Redicionadores de Entrada/Saída usados no awk

- **>** → redireciona os resultados para a saída;
- **>>** → redireciona para o final de um arquivo;
- **<** → redireciona para a entrada de um arquivo.

Criando um arquivo awk executável

Um script pode ser usado para analisar diferentes arquivos que tenham uma mesma estrutura. Neste caso:

1 – Abrimos um editor de texto simples (por exemplo, o gedit), que não crie formatações complexas nos arquivos e digitamos as seguintes linhas:

```
awk
'{
if($3 >= 95 && $4 >= 80)
print $2, $3}
{sum+=$4}
END {print "Sequence number =" NR}
END {print "Total lenght of fragments =" sum}'
arquivo.res
```

2 – Salvamos o arquivo colocando a extensão awk, (ex: "comando1.awk").

3 – No terminal, mudamos a permissão desse arquivo para que este se torne executável:

```
~$ chmod 777 comando1.awk
```

4 – Agora podemos extrair os dados automaticamente dos diferentes arquivos de resultados, para os diferentes arquivos fasta alinhados com o BLAST:

```
~$ cat arquivo1_fasta.res | ./comando1.awk
~$ cat arquivo2_fasta.res | ./comando1.awk
~$ cat arquivo3_fasta.res | ./comando1.awk
...
```

Com esse comando podemos descobrir rapidamente em nosso blast o total de sequências que apresentaram identidade igual ou superior a 95% com a nossa sequência modelo (*query*) e que têm tamanho mínimo de 80 nucleotídeos.

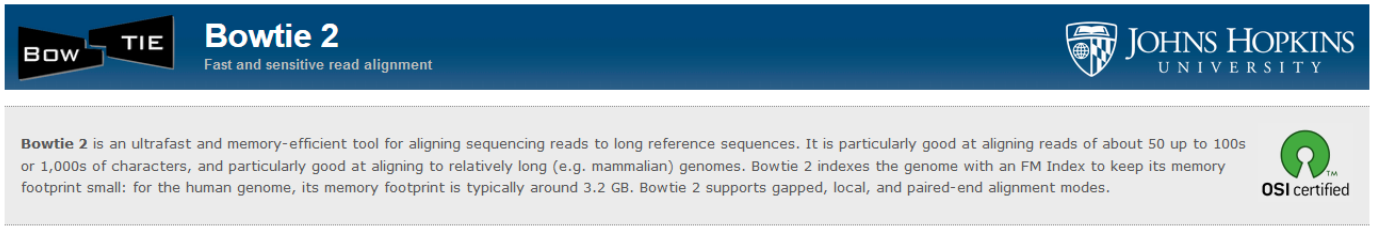
Porém, também é importante comparar o total de sequências obtidas com o nosso arquivo

original – o nosso material sequenciado (arquivo em formato fastq original) – pois cada sequenciamento de cDNA de cada tecido poderá ter tamanhos diferentes (profundidade diferente de sequenciamento). Neste caso, usa-se o programa infoseq do Emboss junto com o awk:

~$ infoseq -only -length -nohead [nome_da_biblioteca.fastq/a] | awk '{sum += $1}END{print sum}'

b) Alinhando sequências com o Bowtie2

O programa Bowtie2 alinha sequências curtas de DNA (*reads*) de grandes genomas, de uma maneira rápida e consumindo pouca memória devido a forma como este indexa os dados. Ele também forma a base de outras ferramentas de análise, como o TopHat. Além disso, devido ao seu algoritmo, ele é uma alternativa melhor que o Blast para esse tipo de análise.



O Bowtie2 pode ser instalado via terminal:

~$ sudo apt-get install bowtie2

Passos necessários

I – Indexando um genoma de referência com o bowtie2-build

Para cada uma das sequências (em formato fasta) que será buscada em nossa biblioteca de cDNA é preciso gerar um grupo de arquivos índices. Isso é conseguido a partir do seguinte comando:

~$ bowtie2-build nome_do_arquivo.fasta nome_do_arquivo.fasta.index

Com isso são criados 6 arquivos índices que serão utilizados pelo Bowtie2 para o alinhamento das sequências:

hf952106_20.fasta	300 bytes	plain text document
hf952106_20.fasta.index.1.bt2	4.2 MB	unknown
hf952106_20.fasta.index.2.bt2	76 bytes	unknown
hf952106_20.fasta.index.3.bt2	17 bytes	unknown
hf952106_20.fasta.index.4.bt2	71 bytes	unknown
hf952106_20.fasta.index.rev.1.bt2	4.2 MB	unknown
hf952106_20.fasta.index.rev.2.bt2	76 bytes	unknown

Depois que o Bowtie2 termina de fazer os alinhamentos, esses arquivos *bt2 podem ser deletados.

II – Alinhando as sequências

Depois que os arquivos índices são gerados, pode-se iniciar o alinhamento de cada uma das sequências em nossa biblioteca de cDNA. Para tanto, basta digitar no diretório em que estão os arquivos fasta, fastq e index, o seguinte comando:

~$ bowtie2 -x nome_do_arquivo.fasta.index -U nome_da_biblioteca.fastq -S nome_do_arquivo.sam -p 2 -a --no-unal --sensitive

Observações

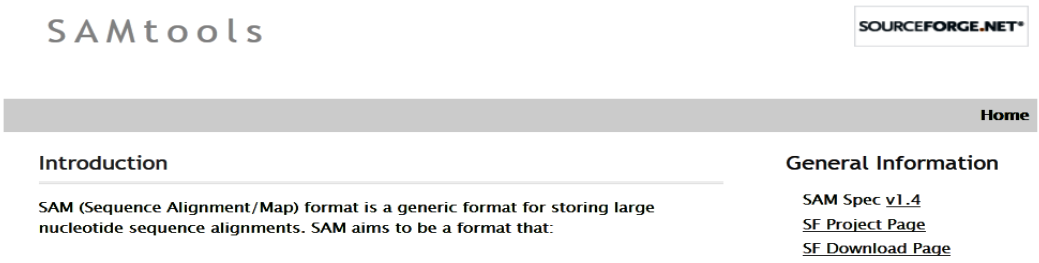
- **Nome do arquivo sam:** convém colocar o nome da sequência buscada e da biblioteca (ex: fasta1.biblioteca5.fastq.sam);
- **–x →** indica o arquivo índice (obtido a partir de cada arquivo fasta isolado e indexado) a ser utilizado;

- **-U →** indica o arquivo (ou arquivos, se separados por vírgula) contendo as sequências não pareadas que serão alinhadas (no caso, a nossa biblioteca de cDNA);
- **-S →** indica o arquivo de saída produzido pelo Bowtie2 e que será gravado no formato SAM (deve-se escolher o mesmo nome dado aos arquivos fasta e fastq buscados quando se trabalha com vários arquivos fasta e são analisadas várias bibliotecas);
- **-a →** indica ao Bowtie2 que este deve buscar pelos alinhamentos válidos sem limitar o número de alinhamentos a serem procurados (demora um pouco mais, mas é mais eficiente);
- **-p →** indica o número de núcleos do servidor/computador que serão utilizados pelo programa para o processamento dos dados (quanto mais núcleos, maior a velocidade de processamento);
- **--sensitive →** indica ao programa que este deve procurar por alinhamentos que envolvam todos os nucleotídeos da sequência buscada e não apenas por alinhamentos parciais (alinhamento "*untrimmed*" ou "*unclipped*");
- **--no-unal →** indica ao programa que este deve eliminar as sequências (*reads*) que não alinharam (o que diminui o tamanho do arquivo de saída), gravando-as em um arquivo (se for especificado → **-- no-unal nome_do_arquivo**);
- **--un →** separa os *reads* que não se parearam e os grava em um arquivo (se for especificado → **--un nome_do_arquivo**).

c) Convertendo arquivos SAM para BAM

Ao utilizarmos o programa Bowtie2, é gerado um arquivo em formato SAM contendo informações sobre todas as sequências que se alinharam na biblioteca de cDNA. Entretanto, como vimos anteriormente, além de ocuparem um grande espaço, eles podem não ser o formato padrão para uma série de programas. Por isso, é comum termos que convertê-los para o formato BAM. O programa SAMtools fornece uma série de sub-rotinas que permitem manipular e converter programas gravados no formato SAM, incluindo o sorteio, união, indexação etc. Para instalar essa ferramenta, basta digitar no terminal:

```
~$ sudo apt-get install samtools
```



Em seguida, basta acessar o terminal e digitar o seguinte comando:

```
~$ samtools view -Sb nome_do_arquivo.sam -o nome_do_arquivo.bam
```

Onde:

- **-S →** informa que arquivo de entrada está em formato SAM;
- **-b →** indica que o arquivo de saída deverá ser gravado no formato BAM;
- **-o →** nome do arquivo de saída;
- **-t →** deve ser usado caso a linha de cabeçalho (@SQ) esteja ausente.

I – Criando um arquivo BAM sorteado

Alguns programas de análise, como o Artemis, exigem que o arquivo no formato BAM seja sorteado, o que garante um acesso mais eficiente à essas sequências. Um arquivo BAM sorteado, além de ser mais compacto é mais conveniente para a descoberta de variação nas sequências estudadas. Para produzi-lo, basta acessar o terminal e digitar o seguinte comando:

```
~$ samtools sort nome_do_arquivo.bam nome_do_arquivo.bam.sorted
```

II – Criando um arquivo BAM indexado

Alguns programas de análise, como o Artemis, também exigem um arquivo BAM indexado para poder acessá-lo de uma maneira mais eficiente. Isso é feito a partir do arquivo bam sorteado, usando-se o seguinte comando:

```
~$ samtools index nome_do_arquivo.sorted.bam
```

III – Convertendo um arquivo BAM em SAM

Como os arquivos no formato BAM são mais compactados que aqueles em formato SAM, é comum manter no computador apenas os primeiros. Mas, às vezes, pode ser necessário retornar um arquivo BAM ao formato SAM para que possamos extrair alguma informação, usando, por exemplo, um editor de texto (uma vez que isso não pode ser feito com os arquivos no formato BAM). Neste caso, pode-se utilizar o seguinte comando:

```
~$ samtools view -h nome_do_arquivo.bam -o nome_do_arquivo.sam
```

Onde:

- **-h →** inclui cabeçalho no arquivo de saída.
- **-o →** nome do arquivo de saída.

d) Visualizando o alinhamento da sequência alvo

É possível visualizar, com o programa Artemis, como as sequências buscadas alinharam em um trecho da nossa biblioteca de cDNA. Para tanto, são necessários os seguintes arquivos:

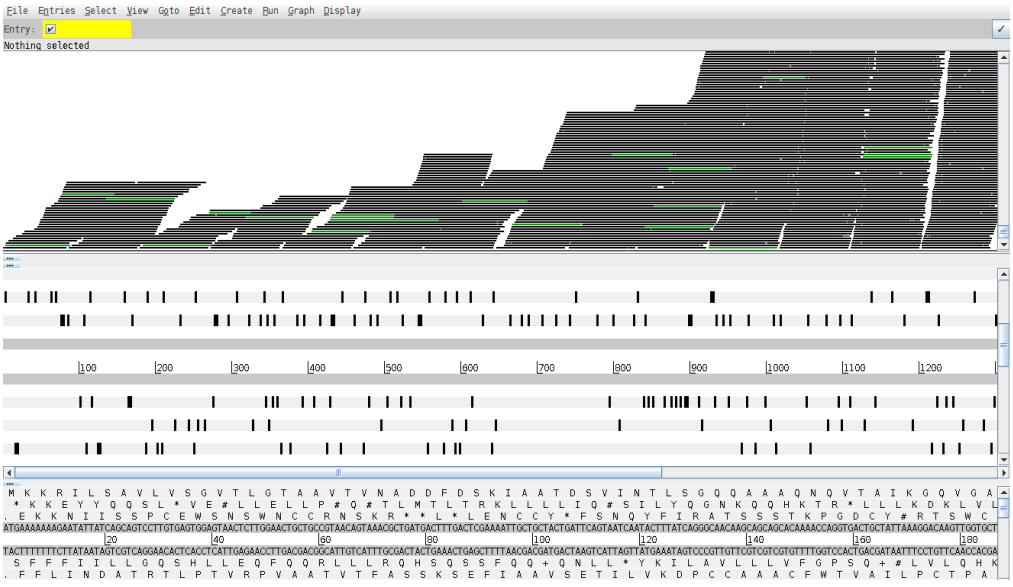
- arquivo.fasta
- arquivo.sorted.bam
- arquivo.sorted.bam.bai

Tendo os arquivos em mãos, basta abrir o programa Artemis e seguir os passos abaixo:

```
~$ ./art &
```

- **Passo 1:** File → Open (ou control O) → arquivo.fasta
- **Passo 2:** File → Read BAM /VCF → arquivo.sorted.bam

Com isso é gerada uma imagem mostrando a posição em que cada *read* pareou:



Observações

- Os *reads* alinhados e com a mesma sequência nucleotídica da biblioteca pesquisada são representados com coloração escura;
- Os reads alinhados e que apresentam sequência inversa à da biblioteca são apresentados na coloração verde;

- Este último caso é bastante comum quando se pesquisa por ncRNA (RNAs não expressos em peptídeos), tendo em vista que muitos deles são reguladores da expressão gênica.

5 – Usando programação Shell

É possível criar uma programação em Shell para fazer boa parte dessas análises, bem como para extrair as informações de interesse. Por exemplo, pode-se fazer as buscas de todas as sequências fasta na biblioteca fastq usando o programa Bowtie2, combinar subrotinas do Emboss (como o **infoseq**) com a linguagem de programação AWK e com outros módulos do Shell, obter normalizações etc. Com isso, boa parte da busca por sequências nas bibliotecas de cDNA podem ser automatizadas. Abaixo é apresentado uma programação em Shell desenvolvida pelo Dr. Romain Guyot para tal tipo de estudo:

```
#!/bin/sh
##############################################
#           Fazendo a indexação e a busca das sequências                 #
#            de interesse (*.fasta) em nosso transcriptoma               #
#          (*.fastq) usando o programa Bowtie2 e usando o                #
#   programa Samstools para converter os arquivos sam em bam      #
##############################################

# Colocando a data de análise nos arquivos gerados:
        DATE=`date '+%d_%m_%Y'`
        for j in *.fasta
        do

# Indexando os arquivos fasta:
        bowtie2-build ${j} ${j}.index

# Mapeando as sequências (fasta) na biblioteca com o Bowtie2:

        bowtie2 -x ${j}.index -U nome_do_arquivo.fastq -S ${j}.nome_do_arquivo.sam -a --no-unal
        --sensitive --met-file README.${j}.nome_do_arquivo.res

# Opções:
# --no-unal: elimina as sequencias (reads) que não alinharam, tornando o arquivo menor.
# –a: busca pelos alinhamentos validos sem limitar o numero de alinhamentos a serem procurados.

# Convertendo o arquivo SAM em BAM:
        samtools view -Sb ${j}.nome_do_arquivo.sam -o ${j}. nome_do_arquivo.bam

# Sorteando e indexando o arquivo BAM para ser usado no programa Artemis:
        samtools sort ${j}.nome_do_arquivo.bam ${j}.nome_do_arquivo.bam.sorted
        samtools index ${j}.nome_do_arquivo.sorted.bam

##############################################
#           Coletando os dados e gravando uma tabela com                 #
#                        os resultados de interesse                      #
##############################################

# Obtendo o tamanho da sequencia fasta que foi buscada:
        SIZE=`infoseq -only -length -nohead ${j} | sed 's/ //g'`
        echo ${SIZE:-PROBLEM}

# O arquivo acima entrará na variável FILE
# indexando a variável FILE

# Calculando o número de sequências encontradas na biblioteca:
        NB=`awk 'END {print $2}' README.{j}.nome_do_arquivo.res`

# Calculando o número de sequencias alinhadas nos arquivos SAM:
        sed -i '/^@/d' ${j}.nome_do_arquivo.sam
        READS=`grep -c '^' ${j}.nome_do_arquivo.sam`
```

```
# Agrupando os dados e fazendo as normalizações:

        N1=$( echo "scale=11; ${READS} / ${NB} * 1000000" | bc )
        N2=$( echo "scale=11; ${N1} / ${SIZE} * 1000" | bc )
        echo  "${j};${SIZE:-PROBLEM};nome_do_arquivo.fastq;${NB:-PROBLEM};${READS:-PROBLEM};
        ${N1:-PROBLEM};${N2:-PROBLEM}"  >> nome_do_arquivo.fastq.RESULTS_all_db.${DATE}
#
        done
```

Alguns comandos a serem usados em linha de comando para se rodar ou se fazer alterações em nosso programa em Shell:

~$ chmod 770 nome_do_programa.sh

- **Significado:** muda a permissão do arquivo e indica que este é um programa que pode ser lido, aberto e executado pelo usuário e pelo grupo.

~$ ./nome_do_programa.sh

- **Significado:** para iniciar o programa pelo terminal.

~$ ./nome_do_programa.sh &

- **Significado:** colocando o & no final do comando fará o processo correr em segundo plano, liberando o terminal para outras atividades.

~$ sed -i "s/nome_a_ser_trocado/novo_nome/g" nome_do_programa.sh

- **Significado:** usando esse comando, não é preciso abrir o arquivo de programa para trocar os nomes das bibliotecas e dos arquivos correspondentes (sam, bam, res etc), o que facilita sobremaneira a análise quando se tem um grande número de bibliotecas a serem estudadas e um grande número de trocas dentro dos arquivos.

6 – Normalização dos reads quando não se tem repetições para as diferentes bibliotecas

Um dos objetivos do sequenciamento de RNAs (RNA-Seq) é tentar descobrir como os genes se expressam diferencialmente em diferentes condições ou tecidos. Acontece que, o número de *reads* mapeados para cada tipo de RNA é linearmente correlacionado com a sua abundância dentro da célula:

número de reads → nível de expressão do gene

Assim, o sequenciamento de RNA (RNA-Seq) oferece uma aproximação quantitativa da abundância dos transcritos-alvo na forma de *counts* (contagens). Entretanto, essas contagens precisam ser normalizadas para a remoção de viéses técnicos decorrentes dos passos necessários para a realização do RNA-seq, particularmente aquelas referentes ao comprimento das diferentes moléculas de RNAs e a profundidade do sequenciamento de diferentes bibliotecas ou de amostras:

- Por exemplo, é esperado que sequenciamentos mais profundos resultem em um maior número de *counts*, provocando um viés se formos comparar transcritos de diferentes sequenciamentos com diferentes profundidades;
- Similarmente, sequências maiores de RNAs são mais prováveis de serem sequenciadas, resultando em uma maior quantidade de *counts*, provocando um viés nas comparações entre sequências com diferentes tamanhos.

Portanto, é preciso equacionar os dados para se ter uma medida mais correta da taxa de expressão dos RNAs nas diferentes bibliotecas. Por esse motivo é feita a normalização dos *counts*. Existem várias metodologias utilizadas na normalização das sequências detectadas nas bibliotecas de cDNA e, cada uma delas pode ser mais ou menos adequada, dependendo do tipo de dados que dispomos. Além disso, por se tratar de uma metodologia recente, as vantagens e as desvantagens de cada uma delas ainda estão sendo avaliadas pela comunidade científica, não existindo ainda uma metodologia definitiva. Na verdade, estudos recentes apontam que os resultados da normalização e da análise da expressão diferencial pode variar bastante de acordo com a metodologia que usamos. Principalmente para os casos onde os experimentos são planejados sem repetições. Na verdade, atualmente é totalmente recomendável que os experimentos sejam feitos com repetições e as análises

realizadas com metodologias mais eficientes, como veremos mais à frente.

Exemplificando

Imagine que foram realizados dois sequenciamentos, sendo que, no RNA-seq1 foram mapeados 6 milhões de reads e no RNA-Seq2, 8 milhões de reads, sendo obtidos os seguintes dados:

Loco	Tamanho do loco	Nº de reads para o loco (RNA-Seq 1)	Nº de reads para o loco (RNA-Seq 2)
A	600	12	19
B	1100	24	28
C	1400	11	16

I – A normalização via RPKM

RPKM significa Reads por Kilobase por Milhão de Reads Mapeados (*Reads Per Kilobase per Million mapped reads*). Este leva em conta o número de reads, o tamanho dos locos e o número total de reads obtidos.

RPKM = C/LN

Onde:

- C: Número de reads mapeados para uma sequência (ex. transcrito, exon, etc).
- L: Comprimento da sequência (em kb).
- N: Número total de reads mapeados (em milhões).

Os valores ajustados via RPKM para os locos analisados nos RNA-seq1 e 2 serão:

Amostra 1 - foram mapeados 6 milhões de reads:

- Loco A: C = 12 reads, L = 0,6 e N = 6 → RPKM = 12/(0,6x6) = 3,33
- Loco B: C = 24 reads, L = 1,1 e N = 6 → RPKM = 24/(1,1x6) = 3,64
- Loco C: C = 11 reads, L = 1,4 e N = 6 → RPKM = 11/(1,4x6) = 1,31

Amostra 2 - foram mapeados 8 milhões de reads:

- Loco A: C = 19 reads, L = 0,6 e N = 8 → RPKM = 19/(0,6x8) = 3,96
- Loco B: C = 28 reads, L = 1,1 e N = 8 → RPKM = 28/(1,1x8) = 3,18
- Loco C: C = 16 reads, L = 1,4 e N = 8 → RPKM = 16/(1,4x8) = 1,43

II – Normalização via TPM

TPM significa Transcritos por milhão (*Transcripts per million*), e foi proposto por Wagner *et al.*, 2012 (http://link.springer.com/article/10.1007%2Fs12064-012-0162-3 e https://www.biostars.org/p/133488/), como uma alternativa de normalização mais consistente que o RPKM:

TPM = (Nx1000000)/(TxL)

Onde:

- N: Número de reads mapeados para uma sequência (ex. transcrito, exon, etc)
- L: Comprimento da sequência (em Kb)
- T: Soma de todas as taxas de transcrição (Soma de N/L) = Comprimento médio dos reads

Retornando ao nosso exemplo:

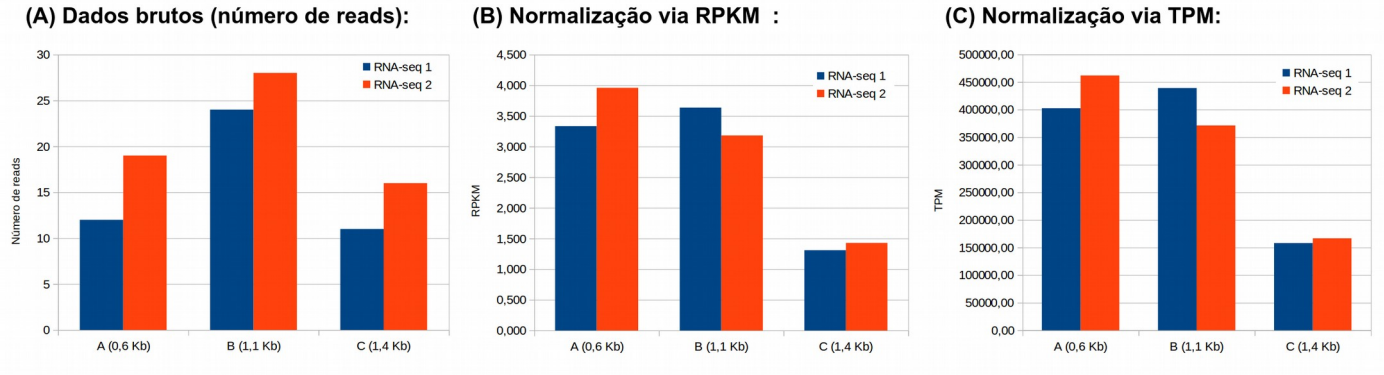
Amostra 1:

Loco	N	L	N/L	T [Σ(N/L)]	TPM (Nx1000000)/(TxL)
A	12	0,6	20,000	49,675	402614,38
B	24	1,1	21,818	49,675	439215,69
C	11	1,4	7,857	49,675	158169,93
		T =	**49,675**		

Amostra 2:

Loco	N	L	N/L	T [Σ(N/L)]	TPM (Nx1000000)/(TxL)
A	19	0,6	31,667	68,550	461951,37
B	28	1,1	25,455	68,550	371329,33
C	16	1,4	11,429	68,550	166719,29
		T =	**68,550**		

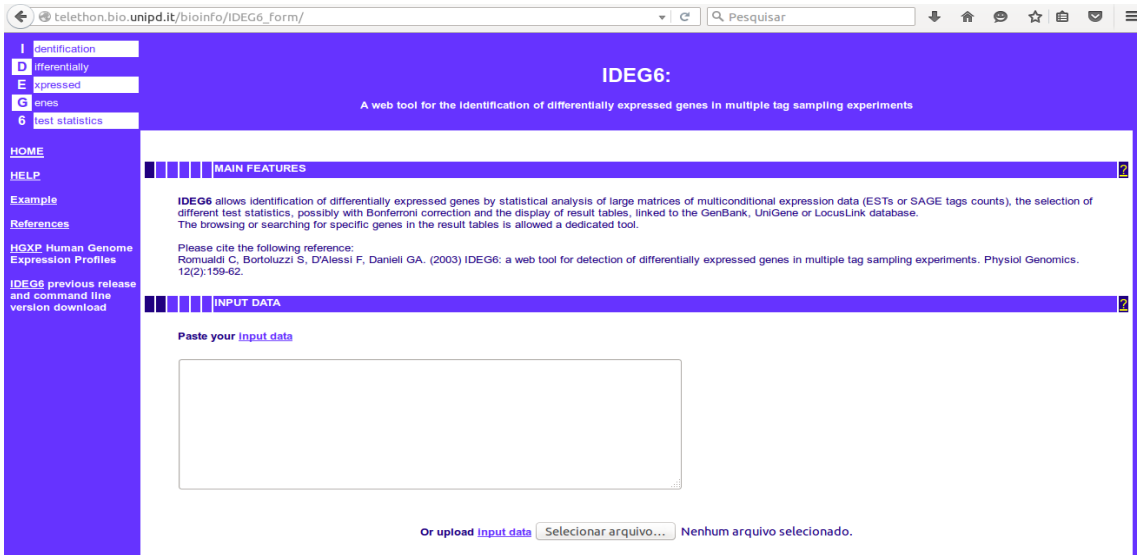
Agora, podemos comparar os resultados das nossas normalizações com os nossos dados brutos:



Pode-se observar com as normalizações que as diferenças encontradas entre os RNA-seq1 e 2, que tinham profundidades diferentes de sequenciamento, são minimizadas. Além disso, ocorre uma inversão para o loco B em termos de expressão entre o RNA-seq 1 e 2.

7 – Análise da expressão diferencial quando se tem repetições para as diferentes bibliotecas

Neste caso, podemos utilizar programa IDEG6 (Identification Differentially Expressed Genes test statistics), disponível no site http://telethon.bio.unipd.it/bioinfo/IDEG6_form/).



A montagem do arquivo para a normalização e o teste de expressão diferencial é feita da seguinte forma:

UNIQID Description	32000000	19581904	30541254	27345728	21296792
Loco 1	229	171	252	27	41
Loco 2	483	297	289	41	21
Loco 3	5	4	6	2	40
Loco 4	151	112	282	29	63
Loco 5	17	13	48	1	5
Loco 6	164	107	246	20	68
Loco 7	956	526	1441	65	94

Loco 8	114	93	131	18	37
Loco 9	271	131	228	24	48
Loco 10	351	280	325	360	29 0

...

Onde:

- A primeira coluna contém os locos buscados em nossa biblioteca;
- As colunas subsequentes contém os números de reads de cada biblioteca estudada;
- Na primeira linha deve ser colocado o tamanho (o número total de reads) de cada biblioteca;
- Nas linhas subsequentes são colocados os números de reads (number of aligned/mapped reads) para cada loco;
- Genes que não apresentaram nenhum read em nehuma das bibliotecas precisam ser retirados.

No nosso exemplo, utilizamos a análise de Audic and Claverie (1997), que desenvolveram um teste estatístico especificamente adaptado para a análise da expressão diferencial de genes (tags). Neste caso:

- Dado p(x), que é a probabilidade de observarmos um número x de tags em uma biblioteca genômica A;
- Eles obtiveram a probabilidade de observarmos y número de tags na biblioteca B, tendo em vista que x tags foram observados em A;
    - Considerando-se que NA e NB são o número total de tags para as bibliotecas A e B, respectivamente;
    - E sob a hipótese de expressão igual desse mesmo gene nas condições A e B (ou seja, de mesmos valores em ambas as condições).

Nesta análise, quanto menor a probabilidade p(y|x), mais diferencialmente expresso será o loco analisado entre ambas as bibliotecas analisadas. Lembrando que estas comparações serão sempre duas a duas. Então, se tenho os resultados da expressão as bibliotecas A, B e C, serão obtidas as comparações para cada loco: (A x B), (A x C) e (B x C). Também trabalharemos com a correção de Bonferroni, que pode ser usada caso o mesmo teste estatístico seja aplicado repetidamente usando o mesmo conjunto de dados. Neste caso, o nível alfa (a probabilidade de se cometer o erro estatístico tipo I) será ajustado de acordo com os resultados da nossa análise. Ou seja, a nossa linha de corte para definirmos os resultados significativos dos não significativos não será necessariamente um valor fixo de alfa (como o 5% comumente usado, por exemplo).

Os resultados produzidos pelo IDEG6 para os genes acima são dados na figura abaixo. Observe que como o resultado da correção de Bonferroni foi de 0,005, todos as comparações cuja probabilidade foi igual ou maior que esse valor foram consideradas não significativas (apenas as expressões significativas entre cada par de comparação são marcadas em amarelo):

Results of running IDEG6

n° of GENES: 10
n° of TISSUES/LIBRARIES: 5
The following statistical tests were applyed:
Pairwise Audic & Claverie test

Selected significance threshold (Bonferroni corr.):
0.0005

UNIQID	Description	Lib1	Lib2	Lib3	Lib4	Lib5	Lib1(norm)	Lib2(norm)	Lib3(norm)	Lib4(norm)	Lib5(norm)	AC 1 2	AC 1 3	AC 1 4	AC 1 5	AC 2 3	AC 2 4	AC 2 5	AC 3 4	AC 3 5	AC 4 5
Loco	1	229	171	252	27	41	0.1	0.1	0.1	0.0	0.0	0.003671	0.005513	0.000000	0.000000	0.013109	0.000000	0.000000	0.000000	0.000000	0.001327
Loco	2	483	297	289	41	21	0.2	0.2	0.1	0.0	0.0	0.018204	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.016868
Loco	3	5	4	6	2	40	0.0	0.0	0.0	0.0	0.0	0.149182	0.112425	0.109593	0.000000	0.097812	0.064438	0.000000	0.071108	0.000000	0.000000
Loco	4	151	112	282	29	63	0.0	0.1	0.1	0.0	0.0	0.009507	0.000000	0.000000	0.000212	0.000001	0.000000	0.000003	0.000000	0.000000	0.000001
Loco	5	17	13	48	1	5	0.0	0.0	0.0	0.0	0.0	0.075458	0.000012	0.000123	0.027584	0.000596	0.000040	0.011015	0.000000	0.000000	0.030506
Loco	6	164	107	246	20	68	0.1	0.1	0.1	0.0	0.0	0.026916	0.000001	0.000000	0.000115	0.000051	0.000000	0.000062	0.000000	0.000000	0.000000
Loco	7	956	526	1441	65	94	0.3	0.3	0.5	0.0	0.0	0.001936	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000020
Loco	8	114	93	131	18	37	0.0	0.0	0.0	0.0	0.0	0.004273	0.009108	0.000000	0.000015	0.015946	0.000000	0.000000	0.000000	0.000000	0.000136
Loco	9	271	131	228	24	48	0.1	0.1	0.1	0.0	0.0	0.002121	0.006829	0.000000	0.000000	0.010303	0.000000	0.000000	0.000000	0.000000	0.000027
Loco	10	311	298	315	310	299	0.1	0.2	0.1	0.1	0.1	0.000000	0.012372	0.002747	0.000001	0.000000	0.000019	0.009631	0.008380	0.000014	0.000565

8 – Normalização dos reads quando se tem repetições para as diferentes bibliotecas

A situação ideal é que façamos 3 ou mais sequenciamentos independentes para cada tratamento e para cada grupo controle. Algo que está se tornando cada vez mais viável, com o barateamento e rapidez do processo de sequenciamento graças as tecnologias de sequenciamento nova geração (NGS –

Next Generation Sequencing). Quando temos tais repetições, podemos usar programas como o EdgeR e o DESeq2 para fazer a normalização e as análises estatísticas.

É possível usar o DESeq para analisar experimentos sem repetição, mas os resultados costumam não ser tão precisos. Com o uso de repetições, minimizamos os erros amostrais do nosso experimento, o que aumenta a chance de descobrirmos realmente quais locos influenciam o fenômeno que estamos estudando. O programa DESeq2 se utiliza te teste exato baseado em distribuição negativa binomial para realizar a análise de expressão diferencial. Como vimos anteriormente, faz-se a normalização:

- Quando trabalhamos com sequências de diferentes tamanhos;
- Para que as sequências totais obtidas de diferentes condições (profundidades de sequenciamento) possam ser comparadas.

Existem diferentes tutoriais disponíveis na internet para se fazer a normalização com o DESeq2 com o pipeline a ser seguido, também chamados de **vignettes**. Vamos utilizar apenas um deles. Este programa roda em R, e deve ser instalado a partir do site Bioconductor, sendo necessária a instalação de vários programas diferentes:

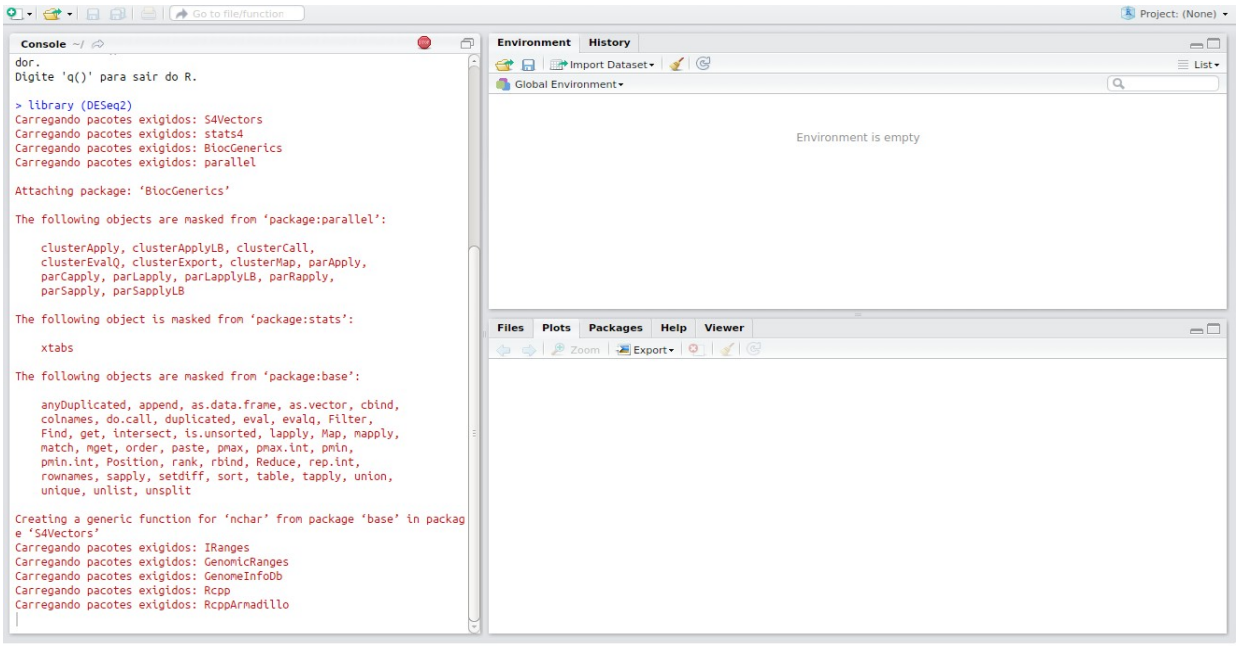
https://www.bioconductor.org/

O programa R está disponível para diferentes sistemas operacionais e o programa Rstudio possui uma interface gráfica que facilita o uso deste programa. Existem versões para o Windows e Linux desses programas:

Site do R: https://www.r-project.org/

Site do RStudio: https://www.rstudio.com/

Lembrando que em ambos os casos, as análises ocorrerão a partir da digitação de comandos. Abaixo é apresentada a interface gráfica do programa RStudio para GNU/Linux:



A seguir são listados alguns dos programas necessários para se rodar o DESeq2:

BIOCONDUCTOR:	**CRAN:**		
source("https://bioconductor.org/biocLite.R") biocLite("Biobase") biocLite("S4Vectors") biocLite("IRanges") biocLite("GenomicRanges") biocLite("GenomeInfoDb") biocLite("BiocParallel") biocLite("genefilter") (exige instalação do gfortran pela central de programas) biocLite("annotate") biocLite("biogenerics") biocLite("geneplotter") biocLite("ggprot2") (instalado a partir de download do arquivo) biocLite("AnnotationDbi") biocLite("DESeq") biocLite("DESeq2")	DBI RSQLite xtable Rcpp reshape2 digest stringr stringi magrittr RcppArmadillo gtable BH lambda.r futile.options futile.logger	liblapack (exige instalação pela central de programas) lattice snow locfit RColorBrewer dichromat colorspace munsell xml2 (precisa instalar xml2-dev -> sudo apt-get install libxml2-dev) labeling scales	proto MASS Formula latticeExtra cluster rpart nnet acepack Hmisc plyr survival XML foreign gridExtra

Lembrando que a instalação de alguns desses programas pode depender da instalação prévia dos outros. Depois de instalado precisamos preparar os dados para a análise. O tutorial aqui apresentado foi modificado do seguinte endereço:
https://harshinamdar.wordpress.com/2014/11/11/quick-tutorial-on-deseq2/

Neste caso, são necessários 2 arquivos:

- Arquivo_de_dados.tbl
- Arquivo_de_informação_do_experimento.tbl

Exemplo do formato do arquivo de dados:

gene	Controle1	Controle2	Tratamento1	Tratamentok2
loco1	29	10	660	963
loco2	123	33	1295	1671
loco3	9	4	145	250	504

...

Exemplo do formato do arquivo de informação do delineamento experimental:

samples	condition
Controle1	control
Controle2	control
Tratamento1	treatment
Tratamento2	treatment

Exemplo de um vignette para se analisar esses dados pelo DESeq2:

```
#############################################################
# Fazendo as análises no DESeq2
#############################################################

# Abrindo o programa:
library("DESeq2")

# Informando o arquivo de dados:
countData <- read.table("strepto_data.tbl",header=TRUE,row.names=1)
countData
```

```
# Aparecerá a seguinte informação:
#gene Controle1      Controle2      Tratamento1   Tratamentok2
#loco129      10      660     963
#loco2123     33      1295    1671
#loco39       4       145     250     504
#...

# Informando o arquivo com o modelo experimental:
colData <- read.table("strepto_info.tbl",header=TRUE,row.names=1)
colData

# Aparecerá a seguinte informação:
#samples condition
#Controle1    control
#Controle2    control
#Tratamento1          treatment
#Tratamento2          treatment

# Criando a matriz de dados do DESeq (DESeqDataSet):
dds <- DESeqDataSetFromMatrix(countData = countData,colData = colData,design = ~ condition)
dds

# Aparecerá a seguinte informação:
# class: DESeqDataSet
# dim: 2176 5
# exptData(0):
# assays(1): counts
# rownames(2176): ncRNA_1 ncRNA_2 ... gbs2133 gbs2134
# rowRanges metadata column names(0):
# colnames(5): broth1 broth2 milk1 milk2 milk3
# colData names(1): condition
 # Fazendo a análise de expressão diferencial padrão do DESeq:
dds <- DESeq(dds)

# Obtendo os genes diferencialmente expressos:
res <- results(dds)
res

# Aparecerá a seguinte informação:
# log2 fold change (MAP): condition treatment vs control
# Wald test p-value: condition treatment vs control
# DataFrame with 2176 rows and 6 columns
#           baseMean log2FoldChange     lfcSE       stat       pvalue         padj
#          <numeric>      <numeric> <numeric>   <numeric>    <numeric>    <numeric>
# loco10 3.145789e+06     -8.6242939 0.5344340 -16.1372471  1.396316e-58  1.242721e-56
# loco23 3.145789e+06     -8.6242939 0.5344340 -16.1372471  1.396316e-58  1.242721e-56
# loco100 8.369083e+04     -8.7677753 0.3341674 -26.2376724 9.882579e-152 6.450030e-149
# ...             ...           ...       ...        ...          ...          ...

# Para obter informações sobre quais variáveis e testes foram usados e quantos genes foram
significativamente
# up (LFC > 0) e down (LFC < 0)regulados:
summary (res)

# Aparecerá a seguinte informação:
# out of 2176 with nonzero total read count
# adjusted p-value < 0.1
# LFC > 0 (up)     : 452, 21%
# LFC < 0 (down)   : 497, 23%
# outliers [1]     : 0, 0%
# low counts [2]   : 218, 10%
```

```
# (mean count < 8)
# [1] see 'cooksCutoff' argument of ?results
# [2] see 'independentFiltering' argument of ?results

# Gravando esses resultados em um arquivo:
write.csv(res, file="a_strepto_dif_expressao_total.csv")

# Organizando os resultados de acordo com os valores ajustados de p (padj):
resOrdered <- res[order(res$padj),]

# Gravando esses resultados em um arquivo:
write.csv(as.data.frame(resOrdered), file="b_strepto_dif_expressao_total_ordenados.csv")

# Listando apenas os genes que apresentaram um limite/limiar significativo de expressão (o padrão é
padj < 0,1):
resSig <- subset(resOrdered, padj < 0.1)
resSig

# Gravando esses resultados em um arquivo:
write.csv(resSig, file="c_strepto_dif_expressao_padj01.csv")


# Listando apenas os genes que apresentaram um limite/limiar significativo de expressão (padj < 0,1)
e
# uma mudança na taxa de expressão de maior que 2 (fold-change)|>2:
resSig2 <- resOrdered[!is.na(resOrdered$padj) & resOrdered$padj<0.10 &
abs(resOrdered$log2FoldChange)>=1,]
resSig2

# Gravando esses resultados em um arquivo:
write.csv(resSig2, file="d_strepto_dif_expressao_padj01_logchange1.csv")


############################################################
# Obtendo os dados normalizados para a construção do heatmap

# Visualizando os dados normalizados:
counts(dds, normalized=TRUE)

# Criando um arquivo calc desses dados normalizados:
write.csv(counts(dds, normalized=TRUE), file="e_strepto_deseq_normalizado.csv")

# Criando um arquivo txt desses dados normalizados:
write.table(counts(dds, normalized=TRUE), file="e_strepto_deseq_normalizado.txt", sep=";",quote=F)

# Obtendo algumas informações sobre o significado de cada coluna dada nos resultados:
mcols(res,use.names=TRUE)
# Aparecerá a seguinte informação:
# DataFrame with 6 rows and 2 columns
#                       type                                  description
#                <character>                                  <character>
# baseMean       intermediate            mean of normalized counts for all samples
# log2FoldChange      results log2 fold change (MAP): condition treatment vs control
# lfcSE               results        standard error: condition treatment vs control
# stat                results        Wald statistic: condition treatment vs control
# pvalue              results      Wald test p-value: condition treatment vs control
# padj                results                             BH adjusted p-values

# Visualizando graficamente os genes diferencialmente expressos (em vermelho):
plotMA(dds)

# Para plotar em um gráfico as mudanças com log2 vezes em relação a média normalizada.
```

```
# Os pontos em vermelho representam os genes cujo valor do ajuste de p é menor que 0,1.
# Os pontos que saem fora da janela são plotados como triângulos vermelhos.
plotMA(res, main="DESeq2", ylim=c(-2,2))
```

---

9 – Produzindo um heatmap para os nossos resultados

Depois de definirmos quais genes se expressaram diferencialmente, podemos criar um heatmap para visualizar melhor esses resultados. Neste caso, fazemos a normalização dos nossos genes, ou por linha de comando ou em uma planilha tipo Excell. No exemplo abaixo, são apresentados os dados brutos e as normalizações RPKM e TPM para um pequeno conjunto de genes:

Loco	Loco size	Tissue 1 Library size	Number of reads	RPKM	TPM	Tissue 2 Library size	Number of reads	RPKM	TPM
A	11588	51581904	40	0,0669	4377,8012	30541254	18	0,0509	1591,9756
B	10320	51581904	288	0,5410	35392,9958	30541254	145	0,4600	14399,9407
C	10765	51581904	403	0,7258	47478,3422	30541254	307	0,9338	29227,8413
D	10206	51581904	467	0,8871	58031,7745	30541254	199	0,6384	19983,4244
E	10416	51581904	196	0,3648	23864,9009	30541254	91	0,2861	8953,9120
F	9591	51581904	581	1,1744	76827,5132	30541254	362	1,2358	38682,7266
G	10190	51581904	45	0,0856	5600,7071	30541254	10	0,0321	1005,7689
H	9382	51581904	367	0,7584	49610,6789	30541254	198	0,6910	21629,2849
I	9648	51581904	325	0,6531	42721,9041	30541254	136	0,4615	14446,8782
J	9562	51581904	208	0,4217	27587,9309	30541254	92	0,3150	9860,7849
K	16231	51581904	6044	7,2191	472262,7723	30541254	12622	25,4622	796994,4501
L	14132	51581904	1741	2,3883	156242,6789	30541254	596	1,3809	43223,0123

Continuação...

Loco	Loco size	Tissue 3 Library size	Number of reads	RPKM	TPM	Tissue 4 Library size	Number of reads	RPKM	TPM
A	11588	27345728	4	0,0126	6408,5980	21296792	0	0,0000	0,0000
B	10320	27345728	59	0,2091	106141,1625	21296792	15	0,0682	33140,2509
C	10765	27345728	51	0,1732	87956,4449	21296792	11	0,0480	23298,2275
D	10206	27345728	84	0,3010	152804,1846	21296792	14	0,0644	31276,3959
E	10416	27345728	50	0,1755	89121,1042	21296792	8	0,0361	17511,8991
F	9591	27345728	5	0,0191	9678,7136	21296792	19	0,0930	45168,3202
G	10190	27345728	4	0,0144	7287,8149	21296792	12	0,0553	26850,4329
H	9382	27345728	3	0,0117	5936,5940	21296792	20	0,1001	48604,7593
I	9648	27345728	34	0,1289	65426,4186	21296792	14	0,0681	33085,2919
J	9562	27345728	14	0,0535	27182,5892	21296792	6	0,0295	14306,9395
K	16231	27345728	321	0,7232	367173,4586	21296792	445	1,2874	625113,6225
L	14132	27345728	57	0,1475	74882,9168	21296792	63	0,2093	101643,8603

Podemos então gravar arquivos individuais em formato txt contendo os dados normalizados que serão apresentados na forma de um heatmap:

```
Tissue1;Tissue2;Tissue3;Tissue4
A;0.0669197232;0.0508600934;0.0126229835;0
B;0.541022618;0.4600461918;0.2090657182;0.0682491697
C;0.7257610271;0.9337647529;0.1732473707;0.047980466
D;0.8870823692;0.6384261207;0.3009776399;0.0644107391
E;0.3648024373;0.2860576432;0.1755413942;0.0360640774
F;1.1743968361;1.2358273851;0.0190641139;0.0930198255
G;0.0856132455;0.0321320883;0.0143547726;0.0552958926
H;0.7583562438;0.6910077159;0.0116932797;0.1000968424
I;0.6530534037;0.461545742;0.1288700902;0.068135987
J;0.4217132308;0.3150302248;0.0535414103;0.0294637704
```

```
K;7.2190792435;25.462206338;0.7232197291;1.2873615801
L2.3883489157;1.3808794485;0.1474965076;0.2093257864
```

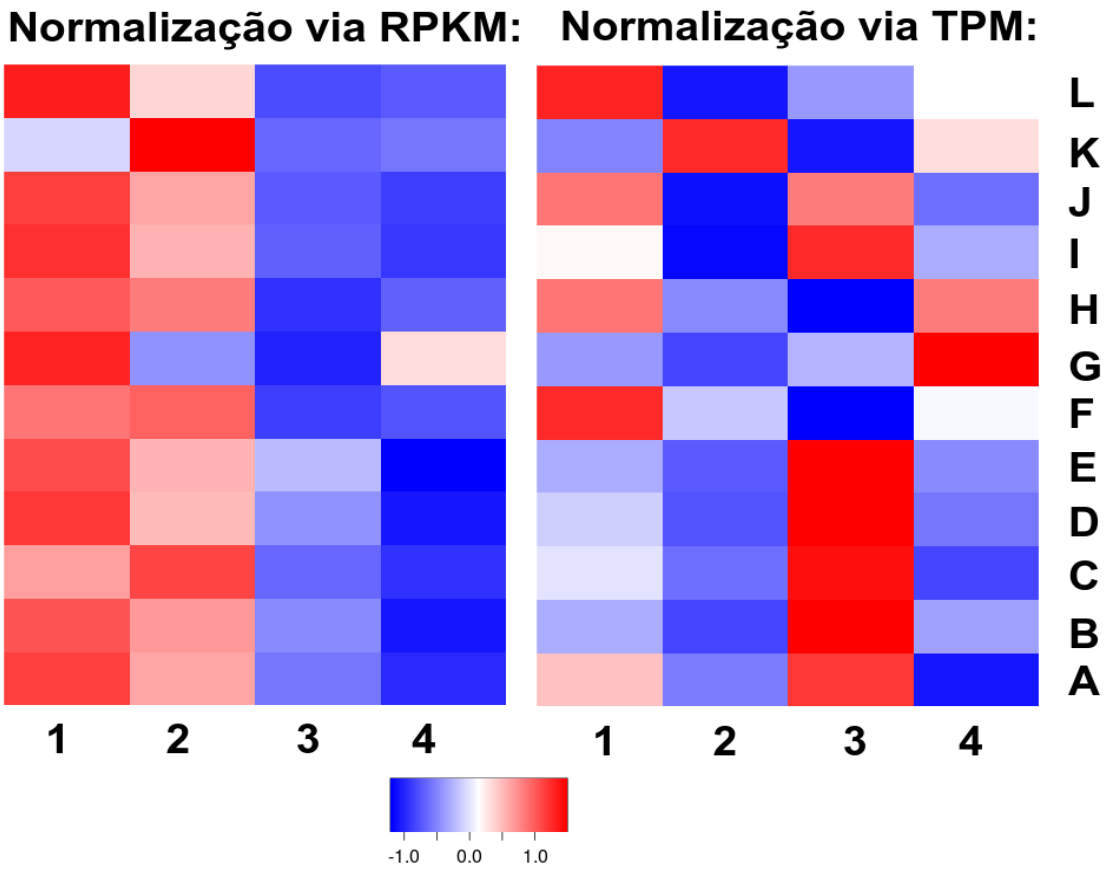
Neste caso:

- A primeira linha conterá as informações das bibliotecas analisadas;
- A primeira coluna contém os nomes dos locos estudados;
- Lembrando que os dados deverão ser separados por ponto e vírgula e as vírgulas dos números (ex: 103141,28) precisam ser trocadas por ponto (ex: 103141.28).

Abaixo é apresentado um script para se fazer um heatmap:

```
# Obs: trocar as vírgulas por ponto e TAB por ponto e vírgula!
# Verificar se o gplots, o heatmap3 e o RColorBrewer estão instalados (se não, ver script abaixo)


# Definindo o diretório de trabalho:
setwd("~/Documentos/02_bioinformatica/01_analises/02_estrepto/04_analises_st/04_heatmap_st")

#############################################################
### A) Lendo os dados para transformação em formato de matriz
#############################################################

normalized <- read.table("02_dados_reads.txt", header=T, sep=";", row.names=1)

head(normalized)

#############################################################
### B) Transformando os dados em uma matriz
#############################################################

normalized_matrix <- data.matrix(normalized)

head(normalized_matrix)

#############################################################
### C) Criando o heatmap
#############################################################

library(gplots)
library(heatmap3)

# Opção 1:
heatmap3(normalized_matrix, col=bluered(75), Rowv=NA, Colv=NA, balanceColor = F,
cexRow=0.8,cexCol=1.1, margins=c(5,15))

# Opção 2:
heatmap3(normalized_matrix, col=bluered(75), Rowv=NA, Colv=NA, balanceColor = F,
cexRow=0.8,cexCol=1.1, margins=c(5,15))
```

Os heatmaps abaixo foram produzidos para os mesmos locos (A a L) e as mesmas condições (1 a 4), considerados diferencialmente expressos (http://telethon.bio.unipd.it/bioinfo/IDEG6_form/) e normalizados via RPKM e TPM:



Podemos observar que os resultados, em termos de maior (vermelho) ou menor (azul) grau de expressão nas diferentes condições são muito diferentes. Portanto, reforça-se a necessidade de se trabalhar com repetições e com ferramentas mais adequadas para este tipo de análise, pois os resultados parecem não ser confiáveis.