

AULA RNAseq

Primeira parte: MONTAGEM E ANÁLISE DE EXPRESSÃO

1º Passo: atenção - login hj será bioinfo2 (agostinhocarrara) e não bioinfo (taxiscarrara)

Ao entrar na bioinfo, é possível ver entre outras pastas:

- **anaconda3:** pasta contendo todo o ambiente dos programas que serão utilizados nesta aula;
- **data:** pasta com todos os arquivos de entradas necessários para realizarmos a montagem dos transcritos.

Vamos criar então o nosso local de trabalho com o comando mkdir:

```
mkdir <seunome>
```

2º Passo:

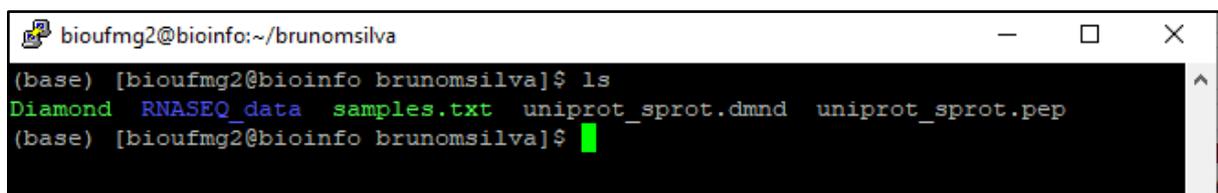
Vamos copiar todos os arquivos necessários para podermos realizar esta prática. Para isso, vamos para o seu local de trabalho:

```
cd <seunome>
```

Agora vamos copiar os arquivos da pasta **data**, a um diretório acima do seu:

```
cp -r ../data/* .
```

Muito importante o ponto final depois do espaço depois do asterisco! Dá um **ls**



```
bioufmg2@bioinfo:~/brunomsilva
(base) [bioufmg2@bioinfo brunomsilva]$ ls
Diamond RNASEQ_data samples.txt uniprot_sprot.dmnd uniprot_sprot.pep
(base) [bioufmg2@bioinfo brunomsilva]$
```

Dentro de sua pasta é possível encontrar os seguintes arquivos:

- **rnaseq_data:** pasta - com todas as *reads* que serão utilizadas no Trinity!
- **uniprot_sprot.pep:** arquivo - contendo os fastas de proteínas do banco de dados Uniprot;
- **uniprot_sprot.dmnd:** arquivo – database já formatada para o Diamond usar, a partir dos fastas do Uniprot acima;
- **diamond:** alinhador – tipo blastp que iremos utilizar para anotar os transcritos montados;
- **samples.txt:** arquivo que descreve as amostras e as condições utilizadas pelo experimento necessário para as comparações de expressão;
- **filtrar.sh:** um scriptzinho para poder pegar as sequencias dos genes, lá no final.

3º Passo:

Nesta aula utilizaremos dados de RNA-Seq correspondentes ao fungo *Schizosaccharomyces pombe*, contendo reads de 76 pares de base, paired-end, correspondentes a duas amostras: Sp_log (crescimento logarítmico) e Sp_plat (fase de platô). Dentro da pasta RNASEQ_data temos essas reads, e é possível encontrar os arquivos com final "left.fq" e "right.fq" no formato FASTQ, saída ao estilo do sequenciador Illumina:

Para poder realizar a montagem dos transcritos (*leva 16 min*), vamos utilizar todas as reads para montar **um único arquivo fasta contendo todos os transcritos montados**. Isto é feito para futuramente utilizarmos este arquivo para realizar a análise de genes diferenciais. Iremos rodar o Trinity com o seguinte comando (sopie e cole, depois explicamos):

```
Trinity --seqType fq --SS_lib_type RF --left
rnaseq_data/Sp_log.left.fq.gz,rnaseq_data/Sp_plat.left.fq.gz --right
rnaseq_data/Sp_log.right.fq.gz,rnaseq_data/Sp_plat.right.fq.gz --CPU 1 --
max_memory 1G --output trinity_saida/
```

Detalhes:

- **--left** (*rnaseq_data/Sp_log.left.fq.gz,rnaseq_data/Sp_plat.left.fq.gz*): indicamos o caminho de todas as reads no sentido forward, tanto log quanto plat;
- **--right** (*rnaseq_data/Sp_log.right.fq.gz,rnaseq_data/Sp_plat.right.fq.gz*): indicamos o caminho de todas as reads no sentido reverse;
- Não se esqueça de que estes arquivos devem ser separados por vírgulas, e não espaçamentos.
- **--SS_lib_type (RF)**: seleciona o tipo da sua biblioteca, se é **paired-end** (reverse+forward) ou single-end;
- **--CPU (1)**: quantidade de threads que iremos utilizar;
- **--max_memory (1G)**: quantidade máxima de memória que o Trinity irá utilizar;
- **--output (trinity_saida/)**: local no qual o Trinity irá salvar todos os dados relacionados à montagem dos transcritos. A barra no fim diz que vai ser um diretório!
- **Importante** que esta pasta esteja criada nomeada como **trinity_saida** para o resto funcionar.

Após completar a montagem, será criada a pasta chamada **trinity_saida**, tal como o seu output foi descrito. Dentro desta pasta é possível encontrar todos os dados relacionados à montagem dos transcritos.

```
bioufmg2@bioinfo:~/tutorial
(base) [bioufmg2@bioinfo tutorial]$ ls
trinity_saida
(base) [bioufmg2@bioinfo tutorial]$ ls trinity_saida/
both.fa                partitioned_reads.files.list
both.fa.ok             partitioned_reads.files.list.ok
both.fa.read_count    read_partitions
chrysalis              recursive_trinity.cmds
inchworm.K25.L25.fa   recursive_trinity.cmds.completed
inchworm.K25.L25.fa.finished recursive_trinity.cmds.ok
inchworm.kmer_count   right.fa.ok
jellyfish.kmers.fa    Trinity.fasta
jellyfish.kmers.fa.histo Trinity.timing
left.fa.ok
(base) [bioufmg2@bioinfo tutorial]$
```

4º Passo:

Vamos agora verificar o arquivo de montagem dos transcritos com os comandos:

cd trinity_saida ou **pwd** pois acho que vc já está na pasta!

less Trinity.fasta

```
bioufmg2@bioinfo:~/tutorial/trinity_saida
(base) [bioufmg2@bioinfo tutorial]$ cd trinity_saida/
(base) [bioufmg2@bioinfo trinity_saida]$ head Trinity.fasta
>TRINITY_DN94_c0_g1_i1 len=656 path=[634:0-655] [-1, 634, -2]
GTTGCGAAAGGTTTTTCGTTGGTGGCTTCTTCGTGCGTTCGCTTCTGGTTTGACCAACTTA
GCTTTGCCCGTCTACAAAGGAGAGTTGTTTTCATCCACCGAACCAATGGGAAACTGCCAGTA
TTTATCTTCAGCCATGGATTGGTTGGCTCGAGAAATGTGTATTCTTCGTTATGTGGTACA
ATCGCTTCTATGGTATCGTCTGCTTGGCCATGGAGCATAGAGATAAAGTCCGCCATCATA
TCTACAGTGCGTGATCCATTACATCCTGAAGAACCCCGTACGTTGTTTCAGTATCGCGAG
ATAAGCGACTTTTATGCAGACGCTACGGTTGTGCTTCAGAATGAACGACTTTTATTTTCA
CAGCAGGAAATCCAAATAGCCCTCCAGATGATTTCGAAATATCAATGACCTTGGAAGTCCG
GACGAAAAGTACCCTTTCTTTGCTCTGTGGACTCTTCTTTTATAAATCTGTTTTCCAA
TCCATGAAGGGTAATTTGAATACCGCTCAAGGAGAATTGATTGTTGCTGGTCATTCTTTT
(base) [bioufmg2@bioinfo trinity_saida]$
```

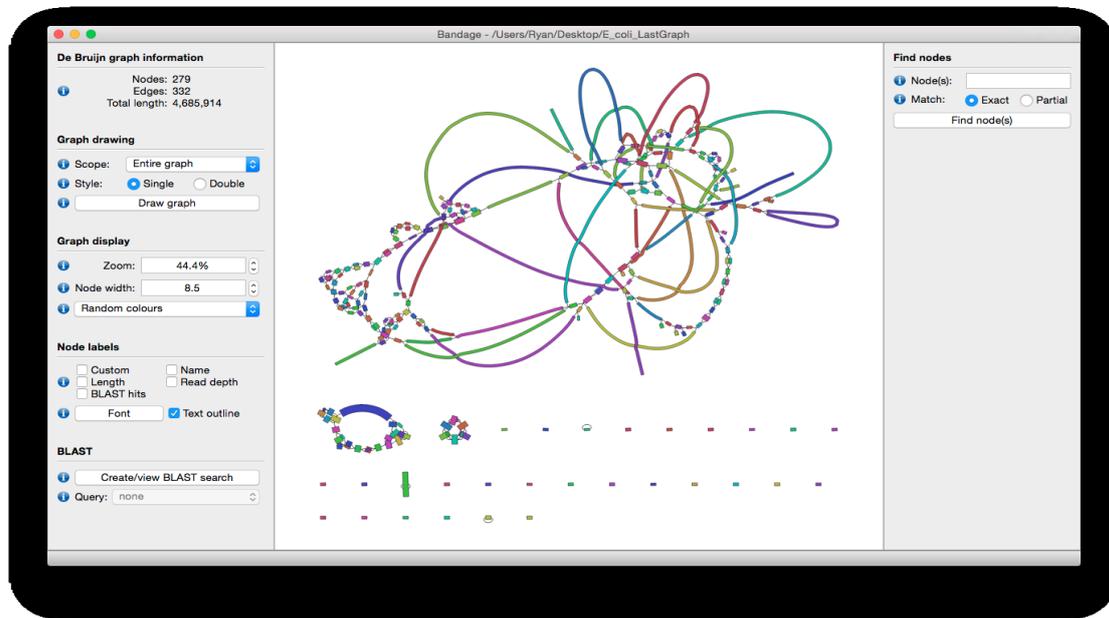
Informações importantes:

- O identificador ">" é utilizado em todo arquivo fasta, para ser um cabeçalho para a sequência;
- **TRINITY_DN94**, como no exemplo, é o nome do transcrito montado;
- **_g1**: indica que é o gene 1 montado;
- **_i1**: indica que é a isoforma 1 do gene 1 montado (ele pode ou não montar mais que uma isoforma do mRNA).

CURIOSIDADE:

Como os transcritos são processados em forma de gráficos de De Bruijn, nós podemos visualizar utilizando o programa Bandage (<https://rrwick.github.io/Bandage/>).

Vídeo explicativo caso queira tentar em casa em:
<https://www.youtube.com/watch?v=VuRN28XyFcl>



5º Passo:

Vamos analisar agora os dados relacionados a esta montagem utilizando um script do Trinity chamado TrinityStats.pl. Para isso, vamos para a pasta do trinity_saida:

```
cd trinity_saida
```

Agora digite o comando para executar o script:

```
TrinityStats.pl Trinity.fasta
```

Resultado:

```
#####  
## Counts of transcripts, etc.  
#####  
Total trinity 'genes': 396  
Total trinity transcripts: 400 <<< logo 4 genes teriam 2 isoformas cada  
Percent GC: 39.16
```

```
#####  
Stats based on ALL transcript contigs:  
#####
```

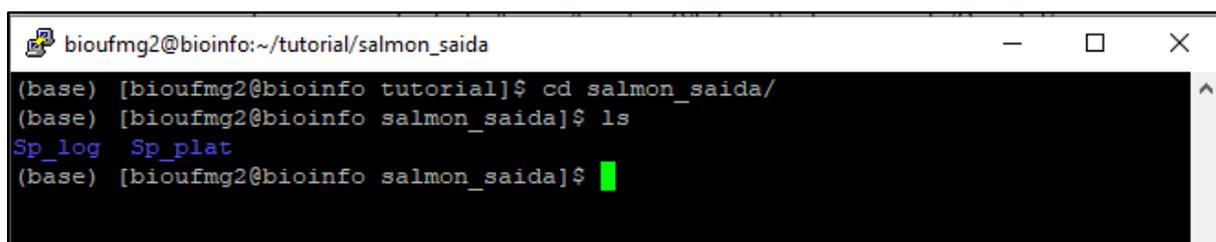
```
/home/bioufmg2/anaconda3/pkgs/trinityrnseq-  
v2.12.0/util/align_and_estimate_abundance.pl --seqType fq --left  
rnseq_data/Sp_log.left.fq.gz --right rnseq_data/Sp_log.right.fq.gz --transcripts  
trinity_saida/Trinity.fasta --est_method salmon --trinity_mode --prep_reference --  
output_dir salmon_saida/sp_log/
```

```
/home/bioufmg2/anaconda3/pkgs/trinityrnseq-  
v2.12.0/util/align_and_estimate_abundance.pl --seqType fq --left  
rnseq_data/Sp_plat.left.fq.gz --right rnseq_data/Sp_plat.right.fq.gz --transcripts  
trinity_saida/Trinity.fasta --est_method salmon --trinity_mode --prep_reference --  
output_dir salmon_saida/sp_plat/
```

Detalhes (foi rápido né?):

- **align_and_estimate_abundance.pl**: script do pacote Trinity que irá realizar todo o pipeline;
- **--seqType** (fq): indicamos qual o formato dos arquivos das reads;
- **--left** (rnseq_data/Sp_log.left.fq.gz): indicamos o caminho de todas as reads no sentido forward;
- **--right** (rnseq_data/Sp_log.right.fq.gz): indicamos o caminho de todas as reads no sentido reverse;
- **--transcripts** (trinity_saida/Trinity.fasta): o caminho do arquivo contendo todos os transcritos montados pelo Trinity;
- **--est_method** (salmon): indicamos qual o método/software de estimação iremos utilizar;
- **--trinity_mode**: irá gerar um arquivo que irá mapear todas as isoformas em seus respectivos genes;
- **--prep_reference**: irá criar índices no arquivo dos transcritos para agilizar os processos;
- **--output_dir** (salmon_saida/sp_log ou sp_plat): indicamos qual o local dos arquivos de saída.

Ao final do processo, irá ser criada uma pasta chamada salmon_saida com duas pastas: Sp_log e Sp_plat, contendo todos os resultados obtidos pelo Salmon:



```
bioufmg2@bioinfo:~/tutorial/salmon_saida  
(base) [bioufmg2@bioinfo tutorial]$ cd salmon_saida/  
(base) [bioufmg2@bioinfo salmon_saida]$ ls  
Sp_log  Sp_plat  
(base) [bioufmg2@bioinfo salmon_saida]$
```

7º Passo:

Vamos para a pasta contendo os resultados de uma das condições para podermos analisar os arquivos de saída gerados pelo Salmon:

cd salmon_saida/sp_log (ou cd salmon_saida seguido de cd sp_log)

Dentro da pasta podemos encontrar 2 arquivos importantes, o **quant.sf** e o **quant.sf.genes**

```
bioufmg2@bioinfo:~/tutorial/salmon_saida/Sp_log
(base) [bioufmg2@bioinfo Sp_log]$ ls
aux_info          lib_format_counts.json  logs          quant.sf.genes
cmd_info.json    libParams              quant.sf
```

- quant.sf: quantificação da abundância dos transcritos;
- quant.sf.genes: quantificação de abundância a nível de genes (sumariza aqueles 4).

Estes arquivos são separados por tabulações e possuem colunas das quais podemos identificar o **nome** do transcrito/gene, o tamanho, e os valores de TPM e de suas reads (Transcritos Por Milhão, uma *pormilhãonagem* ao invés de porcentagem):

Name	Length	EffectiveLength	TPM	NumReads
TRINITY_DN10_c0_g1_i1	334	135.084	0	0
TRINITY_DN11_c0_g1_i1	319	120.926	6158.51	13
TRINITY_DN12_c0_g1_i1	244	57.9931	0	0
TRINITY_DN17_c0_g1_i1	229	47.8216	2395.84	2
TRINITY_DN18_c0_g1_i1	633	432.567	662.169	5
TRINITY_DN18_c1_g1_i1	289	93.8233	3663.47	6
TRINITY_DN19_c0_g1_i1	283	88.6594	646.141	1
TRINITY_DN21_c0_g1_i1	242	56.5791	1012.5	1

Utilizando o comando:

less quant.sf.genes

Obtemos o seguinte resultado (não se preocupe se o seu resultado estiver diferente da imagem, o importante são as colunas e seus valores):

```
bioufmg2@bioinfo:~/tutorial/salmon_saida/Sp_log
(base) [bioufmg2@bioinfo Sp_log]$ head quant.sf.genes
Name      Length  EffectiveLength  TPM      NumReads
TRINITY_DN100_c1_g1  341.00  90.59   948.01  5.96
TRINITY_DN135_c0_g1  206.00  15.90   4350.03 4.80
TRINITY_DN232_c0_g1  681.00  389.87  647.42  17.52
TRINITY_DN173_c0_g1  4337.00 4068.87 1688.42 476.88
TRINITY_DN180_c0_g2  989.00  720.87  266.99  13.36
TRINITY_DN79_c1_g1   833.00  564.87  1106.02 43.37
TRINITY_DN116_c0_g1  354.00  99.22   2022.78 13.93
TRINITY_DN185_c0_g1  1476.00 1207.87 1276.18 107.00
TRINITY_DN19_c0_g1   1134.00 865.87  133.10  8.00
(base) [bioufmg2@bioinfo Sp_log]$
```

O gene DN173 está bem expresso aí e o DN135 tem o menor TPM.

8º Passo:

Agora vamos comparar os genes e transcritos entre as amostras log e plat utilizando um script perl do pacote Trinity chamado: [abundance_estimates_to_matrix.pl](#). Primeiramente vamos para a nossa pasta de trabalho subindo dois diretórios (dê um pwd depois):

```
cd ../.. (ou cd .. duas vezes, atenção, pwd tem que mostrar a pasta com seu nome)
```

Este script irá calcular tanto a nível de genes quanto de transcritos a abundância com o seguinte comando:

```
-----
/home/bioufmg2/anaconda3/pkgs/trinityrnaseq-
v2.12.0/util/abundance\_estimates\_to\_matrix.pl --est_method salmon --out_prefix
salmon_saida/salmon salmon_saida/sp_log/quant.sf salmon_saida/sp_plat/quant.sf --
gene_trans_map trinity_saida/Trinity.fasta.gene_trans_map --
name_sample_by_basedir
-----
```

Detalhes:

- **--est_method** (salmon): indicamos qual o software de estimação foi utilizado para o cálculo de abundância;
- **--out_prefix** (salmon_saida/salmon): indicamos a pasta e o nome dos arquivos de saída;
- **--gene_trans_map** (trinity_saida/Trinity.fasta.gene_trans_map): um índice contendo cada transcrito associado com as suas isoformas gerado no passo 5;
- **--name_sample_by_basedir**: utilizado para poder nomear cada condição de acordo com o nome da pasta que as contém, ou seja, com a condição experimental.

Agora na pasta **salmon_saida** podemos ver que apareceram novos arquivos:

```
cd salmon_saida e ls
```

```
bioufmg2@bioinfo:~/tutorial
(base) [bioufmg2@bioinfo tutorial]$ ls salmon_saida/
salmon.gene.counts.matrix
salmon.gene.TMM.EXPR.matrix
salmon.gene.TPM.not_cross_norm
salmon.gene.TPM.not_cross_norm.runTMM.R
salmon.gene.TPM.not_cross_norm.TMM_info.txt
salmon.isoform.counts.matrix
salmon.isoform.TMM.EXPR.matrix
salmon.isoform.TPM.not_cross_norm
salmon.isoform.TPM.not_cross_norm.runTMM.R
salmon.isoform.TPM.not_cross_norm.TMM_info.txt
Sp_log
Sp_plat
(base) [bioufmg2@bioinfo tutorial]$
```

Arquivos importantes:

- Aqueles que possuem isoform são os resultados a nível dos transcritos e suas isoformas e aqueles que possuem gene são a nível de genes (agrupa isoformas);
- Arquivos terminados em **.counts.matrix** possuem a estimativa da contagem dos fragmentos de RNA-Seq;
- **.TPM.not_cross_norm**: matriz contendo os valores de expressão TPM que não estão normalizadas por amostra com genes muito expressos;
- **.TMM.EXPR.matrix**: matriz contendo os valores de expressão TMM normalizados.

Vamos para a salmon_saida:

cd salmon_saida (se vc ainda não estiver nela, olhe seu cursor ou dê pwd)

E digitar:

less salmon.gene.counts.matrix

```
bioufmg2@bioinfo:~/tutorial/salmon_saida
(base) [bioufmg2@bioinfo salmon_saida]$ head salmon.gene.counts.matrix
Sp_log Sp_plat
TRINITY_DN0_c0_g1      43.29  37.10
TRINITY_DN100_c0_g1   379.69 112.06
TRINITY_DN100_c1_g1    34.64  47.66
TRINITY_DN100_c2_g1   197.58  0.00
TRINITY_DN101_c0_g1    31.33  18.58
TRINITY_DN101_c1_g1   181.64 103.98
TRINITY_DN102_c0_g1    25.82  64.88
TRINITY_DN104_c0_g1   122.90 205.85
TRINITY_DN105_c0_g1    39.21 128.21
(base) [bioufmg2@bioinfo salmon_saida]$
```

Podemos observar que as colunas dos valores foram nomeadas de acordo com a pasta em que elas estavam, facilitando o trabalho; nos próximos passos, para identificar qual dado é de qual amostra é importante. Chegaremos lá!

Também fique a vontade para ver os outros arquivos e perceber que todos eles seguem o mesmo padrão.

9º Passo:

Partiu agora descobrir os genes que estão diferencialmente expressos e montar uns gráficos para melhor visualização destes dados que criamos. Vamos utilizar o programa **edgeR** para este cálculo e outro script do Trinity chamado **run_DE_analysis.pl**.

Vamos agora sair da pasta do salmon_saida e vamos para a nossa pasta de trabalho:

```
cd ..
```

Certifique-se que está na sua pasta de trabalho para evitar problemas. Caso esteja perdido, utilize o comando `pwd` para ver o local do seu terminal.

Bora rodar:

```
/home/bioufmg2/anaconda3/pkgs/trinityrnaseq-  
v2.12.0/Analysis/DifferentialExpression/run_DE_analysis.pl --matrix  
salmon_saida/salmon.gene.counts.matrix --method edgeR --output edgeR_saida/ --  
samples_file samples.txt --dispersion 0.1
```

Detalhes:

- **--matrix** (salmon_saida/salmon.gene.counts.matrix): indicamos de qual matriz contém os dados de abundâncias dos transcritos de genes. Neste caso, iremos calcular somente os genes diferenciados e não suas isoformas.
- **--method** (edgeR): indicamos qual programa usar;
- **--samples_file** (samples.txt): um arquivo que descreve as replicatas e condições dos seus experimentos. Já já iremos vê-lo;
- **--dispersion** (0.1): parâmetro utilizado na função binomial negativa do edgeR para estimar a contagem dos genes.

O arquivo `samples.txt` é muito importante nessa etapa uma vez que ele indica qual a condição e suas replicatas para o programa. Ele possui somente duas colunas, com suas linhas separadas por tabulações, e os nomes destas colunas devem estar de acordo com o nomes do passo 7, dentro do arquivo que possui o final **.counts.matrix**. Exemplo:

#condition	#samples
condA	repA1
condA	repA2
condB	repB1
condB	repB2

```
bioufmg2@bioinfo:~/tutorial
(base) [bioufmg2@bioinfo tutorial]$ head ../data/samples.txt
Log_Phase      Sp_log
Plat_Phase     Sp_plat

(base) [bioufmg2@bioinfo tutorial]$
```

10º Passo:

Com os cálculos do edgeR realizados, vamos agora ver os arquivos de saída. Vamos para a pasta do edgeR_saida:

cd edgeR_saida

```
bioufmg2@bioinfo:~/tutorial
(base) [bioufmg2@bioinfo tutorial]$ ls edgeR_saida/
salmon.gene.counts.matrix.Log_Phase_vs_Plac_Phase.edgeR.count_matrix
salmon.gene.counts.matrix.Log_Phase_vs_Plac_Phase.edgeR.DE_results
salmon.gene.counts.matrix.Log_Phase_vs_Plac_Phase.edgeR.DE_results.MA_n_Volcano.pdf
salmon.gene.counts.matrix.Log_Phase_vs_Plac_Phase.Log_Phase_vs.Plac_Phase.EdgeR.Rscript
(base) [bioufmg2@bioinfo tutorial]$
```

Arquivos importantes:

- O nome do arquivo segue uma lógica: {prefix}.**sampleA_vs_sampleB**.{method};
- **.DE.results**: arquivo contendo os resultados das análises, incluindo o **fold change** e a significância estatística (**FDR**);
- **.MA_n_Volcano.pdf**: nossos gráficos juntos, sendo um de MA e um Volcano!

Para ficar facilitar a visualização no navegador, digite um comando para simplificar o nome dele e escrever ele na sua pasta (um diretório acima) assim:

```
cp
salmon.gene.counts.matrix.Log_Phase_vs_Plac_Phase.edgeR.DE_results.MA_n_Volcano.pdf ../MA_Volcano.pdf
```

Abra agora uma nova página no seu navegador no link:

<http://bioinfo.icb.ufmg.br/bioufmg/look/>

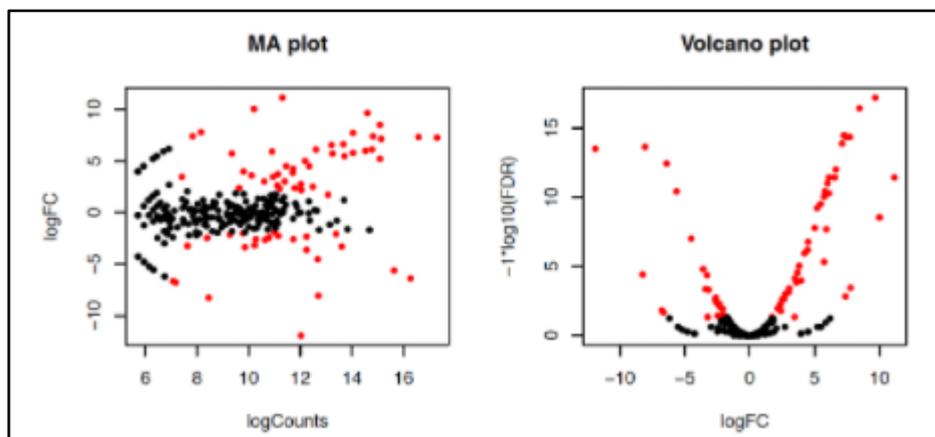
Será necessário **um login (bioufmg) e senha (carrarataxis)**. Ao acessar, entre na pasta **look** e depois na pasta com o seu nome.

← → ↻ ⚠ Não seguro | bioinfo.icb.ufmg.br/bioufmg/

Index of /bioufmg

Name	Last modified	Size	Description
 Parent Directory		-	
 exemplo/	2018-04-28 11:32	-	
 look/	2021-03-16 11:13	-	
 olhe/	2021-03-11 21:15	-	

Ao abrir o arquivo PDF (**MA_Volcano.pdf**) podemos verificar que as contagens com um valor de **FDR < 0.5** (os bãos de olhar) estão coloridas de vermelho. FOld change expressa o número de dobradas ou caídas pela metade: *dobrar é 1, 2 é quadruplicar...*



11° Passo:

Quantos genes estão diferencialmente expressos? Vamos descobrir com o seguinte comando (dentro da pasta **edgeR_saida!** **pwd** se estiver perdido):

```
cat salmon.gene.counts.matrix.Log_Phase_vs_Plac_Phase.edgeR.DE_results | awk '{
if ($7 <= 0.05) print;}' | wc -l
```

Este comando irá quantificar quantos genes possuem um valor de FDR < 0.05 (coluna 7). Sinta-se livre para brincar com este valor.

12° Passo:

Vamos agora pegar todas as sequências de todos os genes que possuem o **FDR <= 0.05** utilizando o script **filtrar.sh**. Para isso, vamos voltar para a nossa pasta de trabalho:

```
cd ../
```

Agora só rodar:

```
sh filtrar.sh
```

Após dar um **ls**, é possível ver que foram criados 2 arquivos:

DE_genes.txt: contém o nome de cada gene diferencialmente expresso;

DE_genes.fasta: contém o cabeçalho e a sequência do gene (o fasta);

2° PARTE: ANOTAÇÃO E EXPLORAÇÃO DOS GENES DIFERENCIALMENTE EXPRESSOS COM BIOLOGIA DE SISTEMAS

1° Passo:

Iremos agora extrair esses genes e gerar os famosos gráficos de **heatmap**. Para isso utilizaremos o **analyze_diff_expr.pl** do Trinity, mas precisamos estar dentro da pasta **edgeR_saida**:

```
cd edgeR_saida/
```

```
-----  
/home/bioufmg2/anaconda3/opt/trinity-  
2.1.1/Analysis/DifferentialExpression/analyze_diff_expr.pl -P 1e-3 -C 2 --matrix  
../salmon_saida/salmon.gene.TMM.EXPR.matrix --samples ../samples.txt  
-----
```

Detalhes:

- **-P (1e-3)**: define o p-valor a ser utilizado para ser considerado um gene diferencialmente expresso;
- **-C (2)**: define o valor do fold change (**quadruplica ou cai abaixo de 1/4**);
- **--matrix** (../salmon_saida/salmon.gene.TMM.EXPR.matrix): indicamos de qual matriz contém os dados de abundâncias dos transcritos de genes gerados nos passos anteriores contendo os valores de TMM entre as amostras;
- **--samples** (../samples.txt): aquele mesmo arquivo que descreve as replicatas e condições do seus experimentos.

Dando um **ls** nós vemos que arquivos novos que apareceram:

Arquivos importantes:

- **.{sampleA}-UP.subset**: features que estão reguladas positivamente na amostra A;
- **.{sampleB}-UP.subset**: features que estão reguladas positivamente na amostra B;
- **diffExpr.{pvalor}_{valorFC}.matrix.log2.dat**: todas as features encontradas que estão diferencialmente expressas em todas as comparações entre as amostras;

Vamos agora brincar com estes arquivos.

2º Passo:

Vamos conhecer os genes super (UP) regulados em Plat_Phase sendo o p-valor menor que 0,001 e mais que o dobro de valor. Repare na coluna 4 (Fold Change)

less salmon.gene.counts.matrix.Log_Phase_vs_Plat_Phase.edgeR.DE_results.P1e-3_C2.Plat_Phase-UP.subset

```
bioufmg2@bioinfo:~/tutorial/edgeR_saida
(base) [bioufmg2@bioinfo edgeR_saida]$ head salmon.gene.counts.matrix.Log_Phase_vs_Plat_Phase.edgeR.DE_results.P1e-3_C2.Log_Phase-UP.subset
id      sampleA sampleB logFC  logCPM  FValue  FDR      Sp_log  Sp_plat
TRINITY_DN231_c0_g1  Log_Phase  Plat_Phase  -9.34314191033395      16.5890207144163      5.7732652361922e-20      2.10146954597396e-17      580.450 188015.104
TRINITY_DN212_c0_g1  Log_Phase  Plat_Phase  -9.495074280824      13.9080324431247      5.97262704247614e-19      1.08701812173066e-16      41.457  28240.327
TRINITY_DN232_c0_g1  Log_Phase  Plat_Phase  -7.57704742296257      15.5067165483438      1.51573337690246e-17      1.83908983064165e-15      464.796 88524.117
TRINITY_DN187_c0_g1  Log_Phase  Plat_Phase  -7.32901752126435      17.4231676037577      2.14723758248106e-17      1.95398620005777e-15      2116.915 334269.891
TRINITY_DN208_c0_g1  Log_Phase  Plat_Phase  -7.70661592621119      13.4049282020839      5.26192524761001e-16      3.54689187184754e-14      107.972 20519.442
TRINITY_DN189_c0_g1  Log_Phase  Plat_Phase  -7.40496628650093      13.9361129538057      5.84652506348496e-16      3.54689187184754e-14      170.943 29684.255
TRINITY_DN77_c0_g2   Log_Phase  Plat_Phase  -7.07256138620175      14.1302531707243      1.41218490791238e-15      7.3433615211444e-14      251.852 33920.141
TRINITY_DN189_c0_g2  Log_Phase  Plat_Phase  -6.12108529873924      14.0071965979643      3.30842234208808e-13      1.50533216565007e-11      458.195 30920.280
TRINITY_DN99_c0_g1  Log_Phase  Plat_Phase  -6.79208033425515      12.1954801318546      7.70905993349752e-13      3.11788646199233e-11      77.586  8744.188
(base) [bioufmg2@bioinfo edgeR_saida]$
```

Fique a vontade para conhecer os genes *upregulated* da outra amostra!

3º Passo:

Volte à sua pasta com **cd ..** e **cd ..** vamos analisar os fastas de **DE_genes.fasta**

Vamos utilizar o **Diamond** (tipo um blastx acelerado programaticamente) para poder identificar cada gene para podermos conhecê-los. Para isso, certifique-se que esteja na sua pasta de trabalho. Caso esteja perdido: **pwd**

Para rodar o Diamond, rode o seguinte comando:

Diamond blastx -d ../data/uniprot_sprot.dmnd -q DE_genes.fasta -v --outfmt 6 -e 1e-10 --max-target-seqs 1 --outfmt 6 -o blastx.outfmt6 -b4

Detalhes:

- Estamos rodando o blastx do Diamond;
- **-d** (../data/uniprot_sprot.dmnd): indicamos o caminho do banco de dados que será utilizado para blastar as sequências. Neste caso, estamos utilizando o SwissProt-Uniprot;
- **-q** (DE_genes.fasta): indicamos a **query**, que neste caso é o arquivo que contém a sequência dos genes que possuem um valor de FDR <= 0.05;
- **-v**: modo verbose, para poder mostrar lettrinhas no terminal;
- **-e** (1e-10): valor de e-value;
- **--max-target-seqs** (1): para poder anotar **somente o best-hit** de cada transcrito;
- **--outfmt** (6): define o modelo do qual será o arquivo de saída, que neste caso, é igual ao formato de saída do programa BLAST;
- **-o** (blastx.outfmt6): arquivo de output.

- **-b4**: argumento para poder não limitar a memória no uso do Diamond.

4° Passo:

Vamos agora conhecer os nossos transcritos a partir do arquivo de saída **blastx.outfmt6**. Se dermos um **less** neste arquivo podemos perceber que é um arquivo texto tabulado:

```

bioufmg2@bioinfo:~/tutorial
(base) [bioufmg2@bioinfo tutorial]$ head blastx.outfmt6
TRINITY_DN94_c0_g1_i1    PLG7_SCHPO    100.0    218    0    1    654    79    296    8.1e-125    447.6
TRINITY_DN94_c1_g1_i1    PLG7_SCHPO    100.0    140    0    0    3    422    299    438    3.2e-76    286.2
TRINITY_DN94_c2_g1_i1    PLG7_SCHPO    100.0    85    0    0    107    361    1    85    1.9e-46    186.4
TRINITY_DN99_c0_g1_i1    DAK2_SCHPO    100.0    590    0    0    3    1772    2    591    0.0e+00    1158.3
TRINITY_DN99_c0_g2_i1    DAK2_SCHPO    100.0    447    0    0    433    1773    145    591    8.8e-255    880.9
TRINITY_DN8_c0_g1_i1    YG06_SCHPO    100.0    427    0    0    2    1282    40    466    9.3e-242    837.0
TRINITY_DN16_c0_g1_i1    WTF7_SCHPO    100.0    201    0    0    3    605    18    218    1.6e-93    343.6
TRINITY_DN0_c0_g1_i1    ATP23_SCHPO    100.0    161    0    0    3    485    6    166    9.7e-91    334.0
TRINITY_DN11_c0_g1_i1    RMI1_SCHPO    99.3    142    1    0    1    426    91    232    5.5e-74    278.1
TRINITY_DN11_c1_g1_i1    RMI1_SCHPO    100.0    70    0    0    91    300    1    70    2.2e-32    139.4
(base) [bioufmg2@bioinfo tutorial]$

```

Nesse caso temos as seguintes colunas, como no BLAST:

- Coluna 1: Nome da query;
- **Coluna 2**: Nome da referência encontrada pelo transcrito (**hit**);
- Coluna 3: porcentagem de identidade;
- Coluna 4: tamanho do alinhamento;
- Coluna 5: número de mismatches;
- Coluna 6: número de gaps;
- Coluna 7; posição de início na sequência da query;
- Coluna 8: posição final na sequência da query;
- Coluna 9: posição de início na sequência da referência;
- Coluna 10: posição final na sequência da referência;
- Coluna 11: **e-value**;
- Coluna 12: bit score.

5° Passo:

Vamos agora identificar quais genes essas proteínas pertencem utilizando a plataforma online do Uniprot na aba de Get Gene Name. Primeiramente vamos printar somente a segunda coluna do nosso blast, referente aos nomes das proteínas:

awk '{print \$2}' blastx.outfmt6

```
bioufmg2@bioinfo:~/brunomsilva
URG1_SCHPO
LAS1_SCHPO
ZHF1_SCHPO
RL401_SCHPO
YAY8_SCHPO
YKN2_SCHPO
HSP71_SCHPO
HSP72_SCHPO
HSP71_SCHPO
HSP71_SCHPO
YAAB_SCHPO
RM01_SCHPO
SSDH2_SCHPO
RS30A_SCHPO
YDD3_SCHPO
HSP90_SCHPO
ZYM1_SCHPO
SSDH2_SCHPO
HSP31_SCHPO
YAAB_SCHPO
HSP31_SCHPO
INV1_SCHPO
(base) [bioufmg2@bioinfo brunomsilva]$
```

Selecione todas as linhas contendo estes nomes:

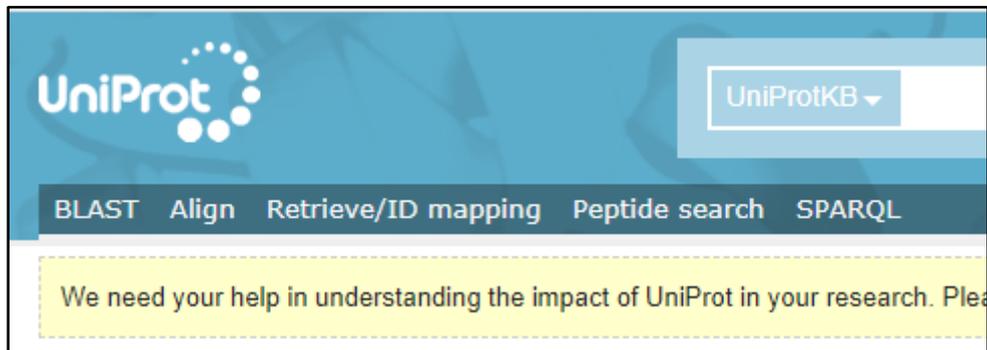
```
bioufmg2@bioinfo:~/brunomsilva
URG1_SCHPO
LAS1_SCHPO
ZHF1_SCHPO
RL401_SCHPO
YAY8_SCHPO
YKN2_SCHPO
HSP71_SCHPO
HSP72_SCHPO
HSP71_SCHPO
HSP71_SCHPO
YAAB_SCHPO
RM01_SCHPO
SSDH2_SCHPO
RS30A_SCHPO
YDD3_SCHPO
HSP90_SCHPO
ZYM1_SCHPO
SSDH2_SCHPO
HSP31_SCHPO
YAAB_SCHPO
HSP31_SCHPO
INV1_SCHPO
(base) [bioufmg2@bioinfo brunomsilva]$
```

Lembrando que o PuTTY copia automaticamente ao selecionar alguma coisa no terminal.

6º Passo:

Acesse agora, numa nova aba do seu navegador, o site uniprot.org.

Logo no cabeçalho do site vamos clicar em **ID mapping**:



Cole o texto que você copiou do PuTTY (**ctrl + v**) na região do **Enter one of more IDs**; selecione, na parte de Select Options, To **Gene Name**. Por fim, clique em **Submit**.

Retrieve/ID mapping

How to use Retrieve/ID mapping tool

Enter or upload a list of identifiers to do one of the following:

- Retrieve the corresponding UniProt entries to download them or work with them on
- Convert identifiers which are of a different type to UniProt identifiers or vice versa a

1. Provide your identifiers

```
YDD3_SCHPO
HSP90_SCHPO
ZYM1_SCHPO
SSDH2_SCHPO
HSP31_SCHPO
YAAB_SCHPO
HSP31_SCHPO
INV1_SCHPO
```

OR upload your own file: Nenhum arquivo selecionado

Run in a new window.

2. Select options

From: To:

Irá abrir então uma página contendo todos os gene names dos seus identificadores:

7º Passo:

Retorne à página anterior do seu browser para fazermos uma nova busca utilizando os mesmos identificadores. Desta vez, vamos selecionar To **KEGG** dentro do grupo de **Genome Annotation Databases**:

2. Select options

From To

Agora irá abrir uma página contendo os identificadores KEGG que dão acesso diretamente a esse banco de dados:

Results

67 out of 71 identifiers from UniProtKB AC/ID were successfully mapped to 72 KEGG IDs.
[Click here to download unmapped identifier\(s\)](#)

From	To
F16P_SCHPO	spo:SPBC1198.14c
RRS1_SCHPO	spo:SPBC29A3.16
NOL10_SCHPO	spo:SPCC330.09
YER5_SCHPO	spo:SPAC2F3.05c
CARA_SCHPO	spo:SPBC56F2.09c
SAHH_SCHPO	spo:SPBC8D2.18c
HS104_SCHPO	spo:SPBC16D10.08c
COX8_SCHPO	spo:SPAC24C9.16c
YDPH_SCHPO	spo:SPAC29A4.17c
YDYE_SCHPO	spo:SPAC11E3.14
GLD1_SCHPO	spo:SPAC13F5.03c
YAKC_SCHPO	spo:SPAC1F7.12
YG06_SCHPO	spo:SPBC1604.06c
YK62_SCHPO	spo:SPAC607.02c
MU122_SCHPO	spo:SPCC1682.15
FIO1_SCHPO	spo:SPAC1F7.08
YBN4_SCHPO	spo:SPBC8E4.04
TOR1_SCHPO	spo:SPBC30D10.10c
CUT7_SCHPO	spo:SPAC25G10.07c
RL24A_SCHPO	spo:SPAC6G9.09c
YGU4_SCHPO	spo:SPBC3B9.04
YOF7_SCHPO	spo:SPBP4H10.07
YA7D_SCHPO	spo:SPAC24H6.13
MOK11_SCHPO	spo:SPAC1527.01
DIS3_SCHPO	spo:SPBC26H8.10

14º Passo:

Vamos explorar alguma dessas vias. Clique na primeira opção de KEGG com o botão direito e vá em **Abrir em uma nova guia**. Explore as proteínas: SPBC56F2.09c e SPBC8D2.18c.

Results

67 out of 71 identifiers from UniProtKB AC/ID were successfully mapped to 72 KEGG IDs.
[Click here to download unmapped identifier\(s\)](#)

[Download](#)

From	To
F16P_SCHPO	spo:SPBC1198.14c
RRS1_SCHPO	spo:SPBC1198.14c
NOL10_SCHPO	spo:SPCC1198.14c
YER5_SCHPO	spo:SPAC1198.14c
CARA_SCHPO	spo:SPBC1198.14c
SAHH_SCHPO	spo:SPBC1198.14c
HS104_SCHPO	spo:SPBC1198.14c
COX8_SCHPO	spo:SPAC1198.14c
YDPH_SCHPO	spo:SPAC29A4.17c

Na aba que irá abrir, será do banco de dados do KEGG:



Schizosaccharomyces pombe (fission yeast): SPBC1198.14c

Help

Entry	SPBC1198.14c CDS T00076
Gene name	fbp1
Definition	(RefSeq) fructose-1,6-bisphosphatase Fbp1
KO	K03841 fructose-1,6-bisphosphatase I [EC:3.1.3.11]
Organism	spo Schizosaccharomyces pombe (fission yeast)
Pathway	<ul style="list-style-type: none"> spo00010 Glycolysis / Gluconeogenesis spo00030 Pentose phosphate pathway spo00051 Fructose and mannose metabolism spo00680 Methane metabolism spo01100 Metabolic pathways spo01110 Biosynthesis of secondary metabolites spo01200 Carbon metabolism
Brite	KEGG Orthology (KO) [BR:spo00001] 09100 Metabolism 09101 Carbohydrate metabolism 00010 Glycolysis / Gluconeogenesis SPBC1198.14c (fbp1) 00030 Pentose phosphate pathway SPBC1198.14c (fbp1) 00051 Fructose and mannose metabolism SPBC1198.14c (fbp1) 09102 Energy metabolism 00680 Methane metabolism SPBC1198.14c (fbp1) 09180 Brite Hierarchies 09183 Protein families: signaling and cellular processes 04147 Exosome [BR:spo04147] SPBC1198.14c (fbp1) Enzymes [BR:spo01000] 3. Hydrolases 3.1 Acting on ester bonds 3.1.3 Phosphoric-monoester hydrolases 3.1.3.11 fructose-bisphosphatase SPBC1198.14c (fbp1) Exosome [BR:spo04147] Exosomal proteins Exosomal proteins of other body fluids (saliva and urine) SPBC1198.14c (fbp1)

All links

- Ontology (3)
- KEGG BRITE (3)
- Pathway (11)
- KEGG PATHWAY (7)
- KEGG MODULE (4)
- Chemical substance (6)
- KEGG COMPOUND (6)
- Chemical reaction (3)
- KEGG ENZYME (1)
- KEGG REACTION (2)
- Genome (1)
- KEGG GENOME (1)
- Gene (11)
- KEGG ORTHOLOGY (1)
- RefGene (3)
- NCBI-PROTEINID (1)
- NCBI-Gene (1)
- RIKEN BRC-DNA (3)
- OC (1)
- POMBASE (1)
- Protein sequence (3)
- UniProt (1)
- SWISS-PROT (1)
- RefSeq(pep) (1)
- DNA sequence (1)
- RefSeq(nuc) (1)
- Protein domain (3)
- Pfam (3)
- All databases (42)

[Download RDF](#)

[BRITE hierarchy](#)

8º Passo:

Uma outra maneira de explorar genes diferencialmente expressos é a biologia de sistemas, Entre no site do String: . <https://string-db.org/> e clique em SEARCH

Opte na barra lateral por Multiple proteins

Indique o organismo Schizosaccharomyces pombe

Aceite a conferência da 91 entradas e veja a rede de interações PPI

Melhor trocar em Settings para High confidence 0.7

Peça para mostrar não mais que 10 interações com a base de dados e UPDATE

Veja em Analysis as vias mais enriquecidas

Teste o clustering com MCL e inflation 2, selecione o maior cluster

Volte em Settings e deixe só as arestas de dados experimentias

Estes são os parâmetros a explorar no STRING

