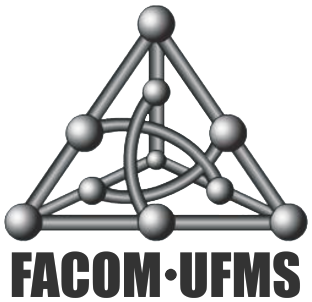
Uma ferramenta para integração de dados de expressão diferencial e interação funcional de proteínas

Dissertação de Mestrado

Rodolpho Gheleri de Oliveira Lima

Orientação: Prof. Dr. Nalvo Franco de Almeida Jr.

Área de concentração: Bioinformática



Faculdade de Computação
Universidade Federal de Mato Grosso do Sul

Campo Grande, Setembro de 2016

Agradecimentos

Agradeço ao meu orientador, Prof. Dr. Nalvo Franco de Almeida Junior por ter escolhido me orientar, por ter me ensinado um pouco sobre bioinformática, paciência e motivação para a realização deste trabalho.

À Christiane Nishibe, Tainá Raiol Alencar, Leonardo Rippel Salgado, Marcelo de Macedo Brigido e Guilherme Pimentel Telles pela ajuda, ideias, motivação e dados que tornaram possível a execução desse trabalho.

À todos os professores e funcionários da FACOM que participaram e contribuíram para a minha formação ao longo desses anos.

À todos os amigos e familiares que incentivaram e caminharam juntos desde minha inserção no mundo acadêmico.

Ao CNPq pelo apoio financeiro.

Resumo

Este trabalho descreve o desenvolvimento de duas ferramentas computacionais: um pipeline para análise de expressão diferencial de genes e outra para a visualização gráfica de resultados de expressão diferencial, chamada `DEPICTViz`, que agrega ao resultado da análise de expressão diferencial informações de interação funcional de proteínas. O pipeline desenvolvido realiza o mapeamento de *reads* no genoma de referência, a ordenação dos *reads* mapeados, a construção e união de transcritos e a análise de expressão diferencial. A ferramenta `DEPICTViz` foi desenvolvida para a visualização de experimentos de expressão diferencial de genes e interação de proteínas. `DEPICTViz` foi testada com sucesso em vários estudos de caso.

Palavras-chave: Genômica, transcritômica, expressão diferencial de genes, interação de proteínas, RNA-Seq.

Abstract

This work describes the development of two computational tools: a pipeline for differential gene expression analysis and another one, called `DEPICTViz`, for graphically visualizing differential gene expression analysis and protein interactions. The developed pipeline executes the read mapping into the reference genome, the sorting of the mapped reads, the building and the union of transcripts and finally the differential gene expression analysis. `DEPICTViz` was developed to visualize differential gene expression analysis and protein interactions. `DEPICTViz` has been successfully tested on several case studies.

Keywords: Genomics, transcriptomics, differential gene expression, protein interactions, RNA-Seq.

Lista de Tabelas

2.1 Diferença na contagem de amostras 24

3.1 Sistema de pontuação utilizado pelo `segemehl`. 33

4.1 Campos que devem ser preenchidos no arquivo de entrada 56
4.2 Exemplo de arquivo de entrada válida para a ferramenta. 58

5.1 Região utilizada para criar o conjunto dos *clusters* HOX. 79
5.2 Total de genes presentes em cada um dos experimentos. 91

Lista de Figuras

2.1 Representação do RNA e do DNA 7
2.2 Representação do dogma central da biologia molecular 8
2.3 Esquema de replicação do DNA 9
2.4 Representação esquemática da iniciação da transcrição 10
2.5 Elongação da transcrição 11
2.6 Etapa do processamento do pré-RNA 12
2.7 Processamento do pré-mRNA: adição do cap 5' e cauda poli-A. 12
2.8 Processamento do pré-mRNA: (RNA *splicing*) 13
2.9 *Splicing* alternativo 14
2.10 Caminho da transcrição do mRNA até sua tradução em uma proteína 16
2.11 Visão geral de um *pipeline* para análise de expressão diferencial . . . 21

3.1 *Array* de sufixo aprimorado 33
3.2 Mapeamentos no genoma completo e em trechos de interesse 35
3.3 Ilustração do *pipeline* até a construção dos transcritos 37
3.4 Gráfico *Smear* gerado utilizando o mapeamento pelo `Bowtie`. 39
3.5 Gráfico *Smear* gerado utilizando o mapeamento feito pelo `segemehl`. 40
3.6 Ilustração do restante do *pipeline* proposto 41

4.1 Mensagens de erro 59
4.2 Janela de seleção de organismos. 60
4.3 Ferramenta completamente carregada com informações referente ao experimento descrito na Seção 5.3 61
4.4 Seletor de filtro permite escolher o campo em que se deseja realizar a busca. 62
4.5 Campo de texto para inserir a *string* de busca 62
4.6 Seleção de *strings* de busca durante a busca múltipla 63
4.7 Janela aberta após o usuário buscar identificadores Swiss-Prot e clicar no botão '*Multi Search*' 63
4.8 Informações de um ponto ao passar o *mouse* sobre ele 64

4.9 Aplicação de *zoom in* 65
4.10 Janela aberta ao clicar em um ponto na área do gráfico 65
4.11 Arestas em cor azul que mostram a interação do gene selecionado com os demais, segundo o STRING. 66
4.12 Arestas de diferentes cores e seus significados 67
4.13 Janela aberta para a busca de um Swiss-Prot na base de dados UniProt 67
4.14 Janela aberta para a busca de um EC *number* na base de dados do ExPASy. 68
4.15 Janela aberta para a busca de um EC *number* na base de dados do KEGG 68
4.16 Janela aberta para buscar por um GO 69

5.1 Distribuição dos genes HOX 76
5.2 Níveis de expressão de genes HOX para diferentes tipos de câncer . . 77
5.3 Gráfico *Smear* representando um conjunto grande de genes 79
5.4 Gráfico gerado pelo `cummeRbund` 82
5.5 Gráfico gerado pelo `cummeRbund` que mostra que o gene HOXD-AS1 é mais expresso em amostras Ctrl do que as de HD. 83
5.6 Dendrograma que mostra o relacionamento de todas amostras utilizadas neste experimento. 84
5.7 Gráfico MDS que mostra a similaridade de todas amostras utilizadas neste experimento. 85
5.8 Visualização de miRNAs e genes HOX na `DEPICTViz` obtidos do experimento HD executado pelo *pipeline*. 85
5.9 Visualização de genes HOX e miRNAs diferencialmente expressos para amostras HD em relação as de controle. 86
5.10 Trancritos considerados diferencialmente expressos 88
5.11 Informações usadas nos experimentos de *Pseudomonas corrugata* . . . 89
5.12 Experimento `WT vs DR` 90
5.13 Experimento `WT vs RfiA` 90
5.14 Experimento `WT vs DR` contendo todos os genes do experimento . . . 91
5.15 Experimento `WT vs RfiA` contendo todos os genes do experimento . . 91

Sumário

1 Introdução **1**

2 Contextualização **5**
2.1 Genômica 5
2.2 Transcritômica 19
2.3 Análise de expressão diferencial 20
2.4 Análise funcional 25

3 Pipeline de expressão diferencial **31**
3.1 Mapeamento 32
3.2 Construção dos Transcritos 35
3.3 União dos transcritos de todas as amostras analisadas 38
3.4 Expressão diferencial 38
3.5 Análise dos resultados 41
3.6 Construção do Pipeline 42

4 DEPICTViz **49**
4.1 Funcionalidades 50
4.2 Requisitos 54
4.3 Representação da saída 59
4.4 Implementação 69

5 Experimentos **75**
5.1 Genes HOX e câncer 75
5.2 Genes HOX e Doença de Huntington 80
5.3 *Brachiaria decunbens* 86
5.4 *Pseudomonas corrugata* 88

6 Conclusão **93**

A Pipeline User's Guide **107**

B DEPICTViz User's Guide **117**

Capítulo 1

Introdução

Os avanços de áreas como a de Biologia Molecular, em particular dos métodos de sequenciamento de DNA, tornaram possível a identificação da exata sequência de bases que formam essas moléculas, possibilitando o sequenciamento do genoma completo de bactérias, animais, plantas e vírus, além do sequenciamento de RNAs, produzindo importantes avanços nas áreas da genômica, metagenômica e transcritômica [85].

O genoma de um organismo é formado por longas moléculas de DNA (ácido desoxirribonucleico) e funcionalmente é dividido em genes [78]. Cada gene é uma sequência linear de DNA que codifica uma molécula composta por outro tipo de ácido, o RNA (ácido ribonucleico). O RNA é formado após a etapa de transcrição do DNA e contém a mesma informação da sequência molde de DNA.

Existem diversos tipos de RNAs transcritos a partir do DNA. O principal tipo, o RNA mensageiro (mRNA) desempenha um papel vital na síntese de proteínas, mas há também RNAs que não codificam proteínas, chamados de RNAs não codificadores (ncRNAs), que por si só desempenham o seu papel funcional, sem que haja a necessidade da síntese de proteína a partir dele [116].

Em qualquer caso, o estudo dos transcritos de um organismo, de um tecido, ou de uma célula em determinada situação é importante. A esse conjunto de transcritos denomina-se transcritoma. E esse estudo é denominado trancritômica.

O transcritoma constantemente altera-se, tanto em termos qualitativos (o que está ou não sendo transcrito) quanto em termos quantitativos (com que intensidade determinados genes estão sendo transcritos) [25]. Dessa forma, a análise do transcritoma estuda o nível de expressão dos genes de maneira qualitativa, identificando os genes que são expressos e os que não são expressos em um determinado momento, e de forma quantitativa, medindo a variação de expressão dos diferentes genes em diferentes situações. Esse estudo é conhecido como Análise de Expressão Diferencial.

Por meio da comparação dos transcritomas de diferentes tipos de células, ou de células expostas a diferentes condições, é possível entender a constituição de um tipo específico de célula, como essa célula normalmente funciona e como mudanças no nível de expressão gênica podem refletir ou contribuir para mudanças fenotípicas. Além disso, é possível ter uma visão completa dos genes que estão ativos em uma célula específica ou ainda caracterizar mRNAs diferentes oriundos de um gene específico, pois alguns genes codificam múltiplos mRNAs [68].

A análise de expressão diferencial dos genes tem se tornado um importante instrumento de diagnóstico em certas áreas da Medicina. O exame do perfil de expressão gênica de um tumor, por exemplo, pode ajudar no diagnóstico do tipo de neoplasia, na determinação da probabilidade de metástase e na estratégia de tratamento mais eficaz [25]. Esse tipo de análise tornou-se muito popular a partir do uso de sequenciadores de alto desempenho para esse fim.

Outro aspecto importante envolvendo a expressão de genes, em particular aqueles que codificam proteínas, é que dificilmente as proteínas agem isoladamente no exercício das suas funções [115]. Mais de 80% das proteínas não operam sozinhas, mas sim em complexos [21]. As proteínas envolvidas em um mesmo processo celular são repetidamente encontradas interagindo entre si. O estudo dessas interações também é importante para inferir a função da proteína dentro da célula. A funcionalidade de proteínas não identificadas podem ser previstas sob a evidência da sua interação com outra proteína, cuja função já é conhecida [111].

O estudo desse tipo de interação tem acelerado a modelagem das vias funcionais para exemplificar os mecanismos moleculares dos processos celulares [117] e identificado alvos para medicamentos [84].

Este trabalho se insere na área de Bioinformática, exatamente na junção desses dois aspectos: análise de expressão diferencial e interação de proteínas. O trabalho consistiu no desenvolvimento de duas ferramentas computacionais: um *pipeline* para análise de expressão diferencial e outra de visualização de resultados de expressão diferencial, chamada DEPICTViz, que agrega informações de interação funcional de proteínas, *Gene Ontology* (GO), grupo de genes ortólogos (COG), classificação de enzimas (EC *number*) e vias metabólicas relacionadas.

O *pipeline* desenvolvido realiza o mapeamento das sequências no genoma de referência utilizando o `segemehl` [58, 59], ordenação dos *reads* mapeados com o `SAMtools` [69], construção e união de transcritos e análise de expressão diferencial utilizando o pacote `Cufflinks` [107, 108].

A outra ferramenta resultante deste trabalho é chamada de `DEPICTViz`, desenvolvida para a visualização de experimentos de expressão diferencial de genes e interação de proteínas. A ferramenta *web* surgiu da necessidade de se visualizar pequenos conjuntos de genes de interesse em experimentos de expressão diferencial de RNA-Seq. Sua principal característica é a apresentação de um gráfico MA (*MA plot*), similar aos gráficos *Smear Plot*, presentes nos pacotes `EdgeR` [91] e `DESeq2` [73].

O principal diferencial da `DEPICTViz` é o fato de ser uma ferramenta interativa. Além disso, `DEPICTViz` é de fácil instalação, e também de fácil utilização. A entrada da ferramenta consiste de uma tabela onde os campos são separados por tabulação, contendo todas as informações necessárias para a construção do gráfico e suas funcionalidades. Como resultado, a saída da ferramenta é apresentada em uma janela no navegador *web* do usuário, podendo este hospedá-la em um servidor ou mantê-la localmente.

Avaliações, tanto do *pipeline* de análise de expressão diferencial, quanto da `DEPICTViz`, foram feitas em quatro estudos de caso contendo diferentes tipos de dados. Dos quatro estudos de caso, os dois primeiros, relacionados aos chamados genes HOX, mostram a utilização do *pipeline* para análise de expressão diferencial. Os demais estudos de caso apresentam diferentes usos da ferramenta DEPICTViz. A partir desses estudos de caso, as publicações [70, 71, 72, 94] foram geradas e uma publicação descrevendo e disponibilizando o DEPICTViz está em preparação. Ambas as ferramentas estão disponíveis para download no endereço `https://git.facom.ufms.br/bioinfo/`.

O texto segue organizado da seguinte forma. O Capítulo 2 descreve conceitos básicos. No Capítulo 3 o *pipeline* de expressão diferencial é descrito. DEPICTViz e seus detalhes são descritos no Capítulo 4. Os resultados obtidos para alguns estudos de caso são apresentados no Capítulo 5. No Capítulo 6, considerações finais são feitas. Finalmente, os Apêncides A e B descrevem os manuais de instalação e utilização do *pipeline* de expressão diferencial e da ferramenta DEPICTViz, respectivamente.

Capítulo 2

Contextualização

Este capítulo tem como objetivo abordar os conceitos necessários para o entendimento do trabalho, assim como contextualizá-los no âmbito das ferramentas apresentadas nos capítulos 3 e 4. O nível de detalhamento na descrição de cada conceito depende do seu grau de importância no contexto das ferramentas.

2.1 Genômica

Apesar da similaridade em nível fenotípico entre os seres vivos, os organismos mantêm diferenças fundamentais em nível celular, podendo ser classificados em dois grupos: os procariotos e eucariotos. Os organismos procariotos são unicelulares e de organização simples, não possuem núcleo e seu material genético é armazenado na região chamada de nucleóide, não existindo membrana que separa essa região do restante da célula. Fazem parte desse grupo as bactérias e as arqueas. Já os organismos eucariotos são mais complexos, com células possuindo núcleo e membrana nuclear que armazenam o material genético. Pertencem a esse grupo as plantas pluricelulares, animais, fungos e alguns organismos unicelulares, como leveduras e algas verdes.

Independentemente da classificação de um organismo, toda a sua informação genética, necessária para a sobrevivência da espécie e também para seu funcionamento, é mantida em seu genoma, composto por longas sequências de ácido desoxirribonucleico

(DNA), que fornecem toda a informação genética utilizada no desenvolvimento, comportamento e reprodução do organismo [90]. A genômica é o estudo das principais características dessas moléculas, como elas estão organizadas e em que porções elas carregam de fato informação genética capaz de regular funcionalmente a célula.

Fisicamente, o genoma pode estar dividido em várias moléculas de fita dupla de DNA e chamadas de replicons, pelo fato de possuírem regiões ou sítios de replicação. O sítio de replicação determina o início da replicação da molécula. Os cromossomos são os replicons maiores da célula e são responsáveis pelo armazenamento da maior parte da informação genética do organismo [63].

Alguns organismos, especialmente procariotos, possuem também replicons menores, chamados de plasmídeos, que também são moléculas de fita dupla de DNA que carregam informação genética. Em bactérias, por exemplo, plasmídeos costumam conter genes responsáveis pela resistência a antibióticos. Quando uma célula se replica, todos os plasmídeos também são copiados, mas os plasmídeos podem se replicar independentemente [60].

Funcionalmente, o genoma está dividido em genes [78]. Cada gene é uma sequência linear de DNA que codifica uma molécula de RNA funcional, ou uma única unidade de transcrição [83].

Existem dois tipos de ácidos nucleicos: o ácido desoxirribonucleico (DNA) e o ácido ribonucleico (RNA), mostrados na Figura 2.1. O RNA é formado após a etapa de transcrição do DNA e contém a mesma informação da sequência molde de DNA, sendo então traduzido durante a etapa de tradução, resultando na síntese de proteínas. Ambos ácidos nucleicos são macromoléculas formadas a partir de unidades menores e mais simples chamadas de nucleotídeos.

Cada nucleotídeo é composto por um açúcar (pentose), um grupo fosfato e uma base nitrogenada. A molécula de açúcar é formada por cinco átomos de carbono (numerados de 1' até 5'). O grupo fosfato conecta o carbono 3' da molécula de açúcar de um nucleotídeo ao carbono 5' da molécula de açúcar do próximo nucleotídeo, formando uma cadeia ou fita. Essa fita possui uma orientação que inicia na extremidade 5' e acaba na extremidade 3'.

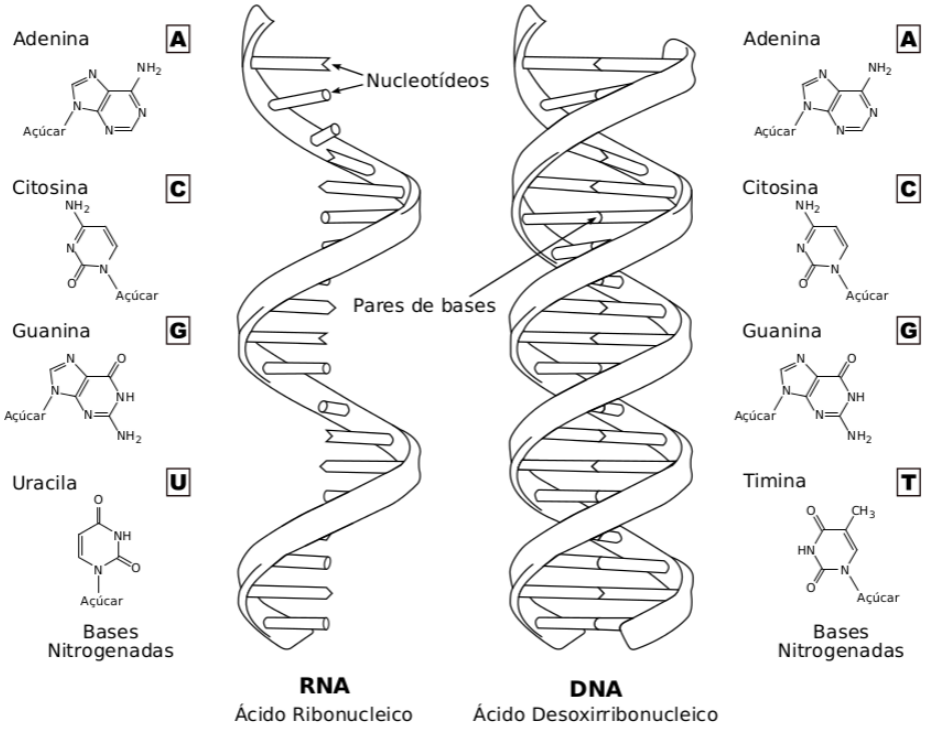


Figura 2.1: Representação do RNA, DNA e das estruturas químicas das bases nitrogenadas. Figura adaptada de [28].

A base nitrogenada pode ser: adenina (`A`), citosina (`C`), guanina (`C`), timina (`T`) ou uracila (`U`). Adenina e guanina pertencem ao grupo das purinas, enquanto citosina, timina e uracila pertencem ao grupo das pirimidinas [79].

Quando o DNA forma sua dupla hélice, um nucleotídeo do grupo das purinas é ligado com um nucleotídeo do grupo das pirimidinas na outra fita. Elas são ligadas por pontes de hidrogênios formando pares de bases, sendo que adenina é pareada com timina (`A-T`) e citosina com guanina (`C-G`). Os pares `A-T` e `C-G` são denominados pares de bases complementares, ou simplesmente pares de bases.

Algumas diferenças entre os dois tipos de ácido nucleico residem no tipo de açúcar e na composição de bases nitrogenadas da molécula. No DNA, a pentose é sempre a desoxirribose e, no RNA, a ribose. Adenina, citosina e guanina estão presentes tanto no DNA quanto no RNA, mas a timina ocorre apenas no DNA, enquanto que a uracila está presente apenas no RNA. Entretanto, a principal diferença entre DNA e RNA é o fato das moléculas de RNA serem compostas por uma única fita e existirem diferentes tipos de RNAs, classificados de acordo com suas localizações e suas funções na célula.

Uma célula contém diferentes tipos de RNAs, onde os mais importantes e que estão ligados diretamente com a síntese de proteínas são o RNA mensageiro (mRNA), RNA transportador (tRNA) e RNA ribossomal (rRNA) [79].

A descoberta de novas classes de RNA levou a uma classificação geral dos RNAs em codificadores e não codificadores. RNAs codificadores são exclusivamente os mRNAs que contém a informação genética para a síntese de proteínas. Os demais RNAs são denominados não codificadores (ncRNAs) e incluem o rRNA e tRNA, além de outros como *small nuclear* RNA (snRNA), *small nucleolar* RNA (snoRNA), *micro* RNA (miRNA) e *small interfering RNA* (siRNA) [27].

Dogma central da biologia molecular

O Dogma Central da Biologia Molecular (Figura 2.2) define como ocorre o fluxo da informação genética do DNA, passando pelo ácido ribonucleico até a proteína. Atualmente, esses passos são bem conhecidos e detalhados. O DNA pode se replicar, dando origens a novas moléculas de DNA ou pode ser transcrito em uma molécula de RNA, que contém a mesma informação da sequência molde de DNA. Logo, o RNA transcrito pode ser um mRNA que será traduzido em proteínas ou, pode ser transcrito em um ncRNA que por si só possui uma função definida na célula [41].

Figura 2.2: Representação do dogma central da biologia molecular [78].

Os conceitos apresentados nos itens adiantes foram extraídos de [20, 27, 28, 79].

Replicação

Na célula, a replicação do DNA tem início em locais específicos, como já foi dito. De uma cadeia original de DNA formam-se duas. A replicação do DNA é o processo de auto-duplicação do material genético, mantendo o padrão de herança ao longo das gerações.

Para que o processo de replicação se inicie, é necessária a atuação de uma enzima, a helicase. A enzima liga-se à cadeia de DNA e desliza sobre esta, quebrando as ligações de hidrogênio, ficando então as duas cadeias de DNA separadas. Em seguida liga-se às cadeias de DNA a enzima RNA primase que sintetiza um *primer*, que consiste numa sequência de bases de RNA que iniciam a síntese, visto que a DNA-polimerase não é capaz de atuar pela ausência de nucleotídeos livres. Após a síntese do *primer*, a DNA-polimerase da continuidade ao processo que ocorre no sentido da extremidade 5' para a extremidade 3' da nova cadeia. Uma das fitas, chamada *leading strand*, é alongada continuamente no sentido 5' $\rightarrow$ 3' enquanto a replicação ocorre. Já a outra fita, denominada *lagging strand*, é primeiro sintetizada como uma série de fragmentos denominados fragmentos de Okazaki. Em seguida, uma enzima, a DNA-ligase, junta os fragmentos de Okazaki em uma única fita de DNA. Durante todo o processo de replicação atuam outras enzimas entre elas as SSB (*Single Strand Binding Proteins*) e as topoisomerases que têm como função evitar o enrolamento da cadeia durante a síntese. A Figura 2.3 mostra o processo de replicação.

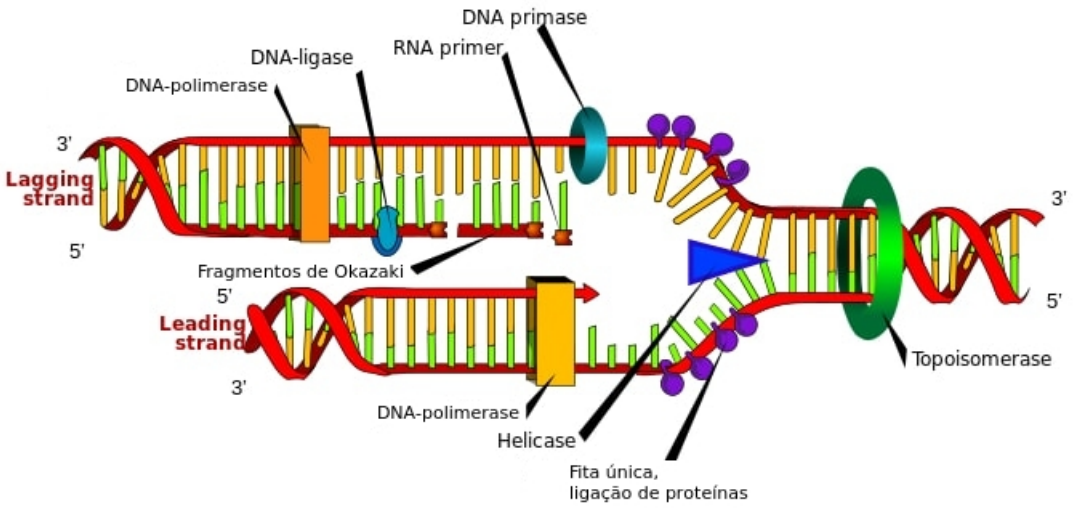


Figura 2.3: Esquema de replicação do DNA. Figura adaptada de `https://en.wikipedia.org/wiki/DNA_replication`

Transcrição

A transcrição é o processo de formação de uma fita de RNA, que carrega a informação genética do DNA que no caso dos mRNA, será utilizado na síntese de proteínas da célula.

A enzima responsável pela transcrição é a RNA-polimerase. A transcrição pode ser dividida em três estágios: iniciação, elongação e terminação. A sequência de DNA onde a RNA-polimerase se liga e inicia a transcrição é conhecida como promotor. O promotor inclui o ponto de início da transcrição e tipicamente inicia-se algumas dezenas de nucleotídeos acima desse ponto de início. Além de determinar a posição em que a transcrição começa, o promotor determina qual das fitas de DNA será usada como molde.

Certas seções do promotor são especialmente importantes para a ligação da RNA-polimerase, chamadas de `TATA` *boxes*. Em procariotos, a própria RNA-polimerase reconhece e se liga ao promotor. Já em eucariotos, um conjunto de proteínas, chamadas fatores de transcrição, regulam a ligação da RNA-polimerase e o início da transcrição. Dessa forma, somente após os fatores de transcrição estarem ligados ao promotor é que a RNA-polimerase se liga. O conjunto completo dos fatores de transcrição e a RNA-polimerase ligada ao promotor é denominada complexo de iniciação da transcrição, conforme ilustrado na Figura 2.4.

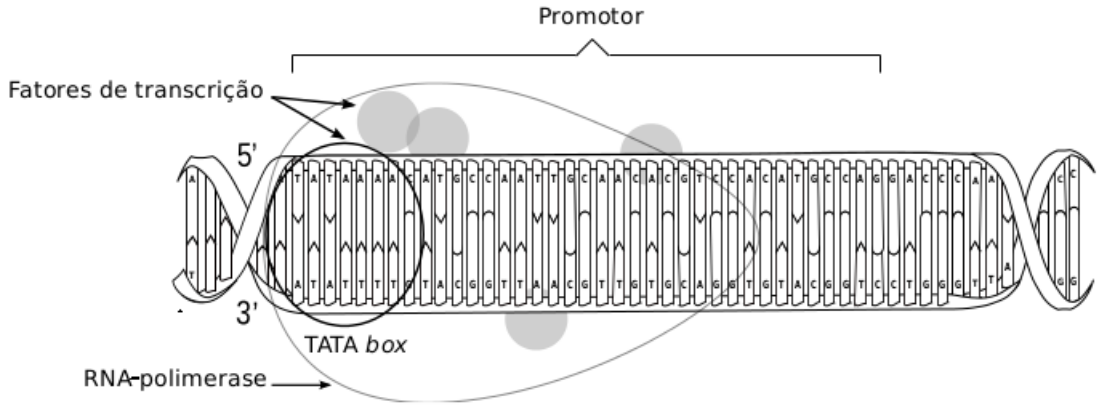


Figura 2.4: Representação esquemática da iniciação da transcrição. Nos eucariotos, os fatores de transcrição se ligam ao promotor para que a RNA-polimerase reconheça e se ligue ao `TATA` *boxes*, formando o complexo de iniciação da transcrição. Figura adaptada de [28].

Depois que a polimerase está devidamente fixada ao promotor do DNA, ela catalisa a formação de ligações fosfodiéster entre os nucleótidos que formam a cadeia simples do RNA, permitindo que ribonucleosídeos trifosfato se liguem a esta por complementariedade de bases. Conforme a enzima percorre a fita molde no sentindo 5' $\rightarrow$ 3', são formadas ligações fosfodiéster entre os ribonucleotídeos por meio da hidrólise do trifosfato pelas RNA polimerases, liberando pirofosfato. Paralelamente, o pa-

reamento das bases nitrogenadas é desfeito, de forma que a molécula de RNA em formação é estendida lateralmente a região transcrita.

Após iniciada a transcrição, a RNA-polimerase move-se ao longo do DNA, desenrolando a dupla hélice e expondo cerca de 10-20 bases do DNA de cada vez para o pareamento com os nucleotídeos do RNA em formação (Figura 2.5). Enzimas adicionam nucleotídeos à extremidade 3' da molécula de RNA, alongando-a. O DNA já transcrito volta a se enrolar, recompondo sua estrutura em dupla hélice.

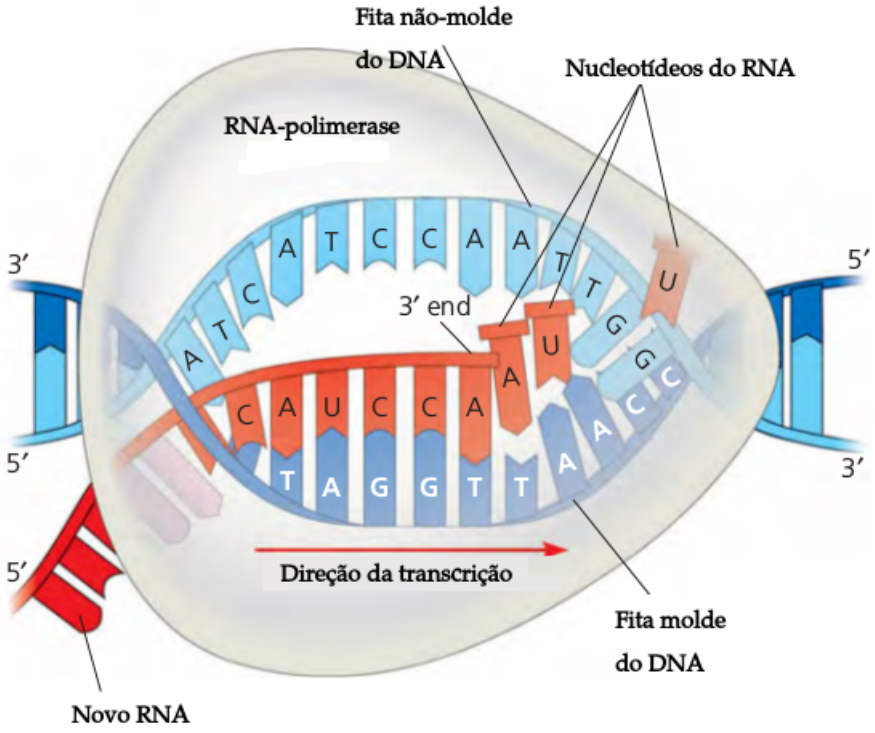


Figura 2.5: Representação da elongação da transcrição. Depois de estar devidamente ligada ao promotor, belicases e topoisomerases desenrolam a dupla hélice do DNA e a RNA-polimerase inicia a síntese do RNA. O DNA já transcrito recompõe sua dupla hélice. Figura adaptada de [28].

A transcrição continua até que a RNA-polimerase transcreva uma sequência de DNA conhecida como terminador. Em seguida, o mRNA é liberado e a RNA-polimerase dissocia-se do DNA.

Nos procariotos, o mRNA transcrito é imediatamente traduzido em uma sequência de aminoácidos de acordo com seu código genético. Ao contrário dos eucariotos,

onde o mRNA precursor, também conhecido como pré-mRNA, precisa ser processado antes de ser traduzido (Figura 2.6).

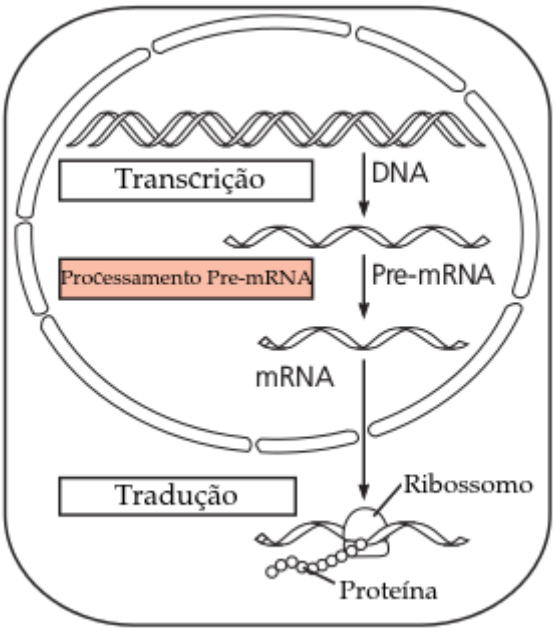


Figura 2.6: Etapa do processamento do pré-RNA é uma etapa intermediária que é feita entre a transcrição e a tradução. Figura adaptada de [28].

A primeira etapa do processamento do pré-mRNA é o *capping* que ocorre na extremidade 5' do pré-mRNA, onde é adicionada uma forma modificada de guanina denominada cap. Em seguida, ocorre a poliadenilação na extremidade 3' do pré-mRNA, onde uma enzima adiciona uma cauda poli-A formada de 50-250 nucleotídeos de adenina. Essas duas modificações (Figura 2.7) protegem o mRNA de ser degradado e ajudam na fixação do ribossomo para a tradução.

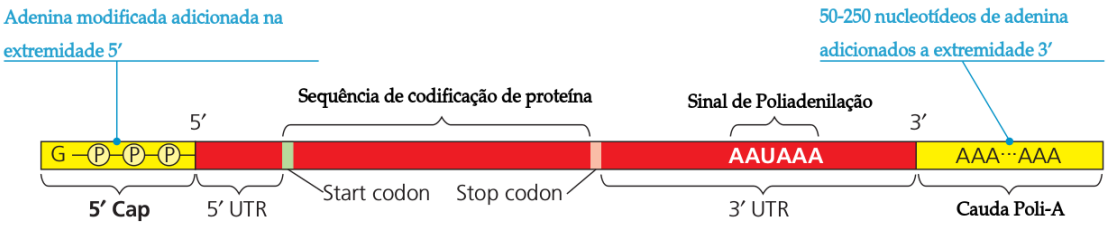


Figura 2.7: Processamento do pré-mRNA: Enzimas modificam as extremidades (adição do cap 5' e cauda poli-A) da molécula de pré-RNA em eucariotos. As extremidades modificadas ajudam a proteger o mRNA da degradação e facilita a ligação dos ribossomos. Figura adaptada de [28].

Outra etapa importante no processamento do pré-mRNA é um processo conhecido como processamento do RNA em eucariotos. Durante esse processo, as regiões não codificadoras, chamadas de íntrons, são removidas e as regiões codificadoras, chamadas de éxons, são unidas (Figura 2.8). Uma importante consequência da presença de íntrons em genes é que um único gene pode codificar mais de um tipo de proteína. Muitos genes são conhecidos por codificar duas ou mais proteínas, dependendo de quais *éxons* são usados durante o processamento do pré-mRNA; este processo é conhecido como *alternative splicing* (Figura 2.9). Cada *alternative splicing* gera o que é chamado de isoformas de RNA.

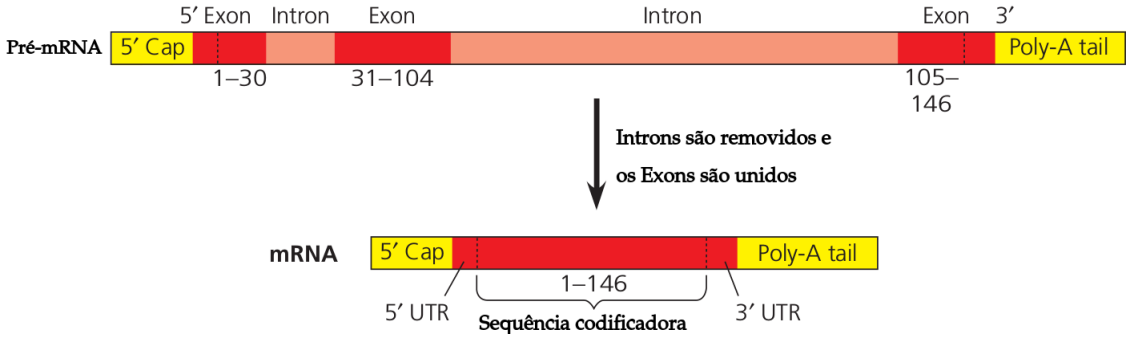


Figura 2.8: Processamento do pré-mRNA: (RNA *splicing*). A molécula de RNA mostrada codifica a *β-globina*, um dos polipeptídeos da hemoglobina. Seu pré-mRNA possui três éxons. Durante o processamento do pré-mRNA, os íntrons são removidos e os éxons são unidos. Em muitos genes, os íntrons são muito maiores que os éxons. Figura adaptada de [28].

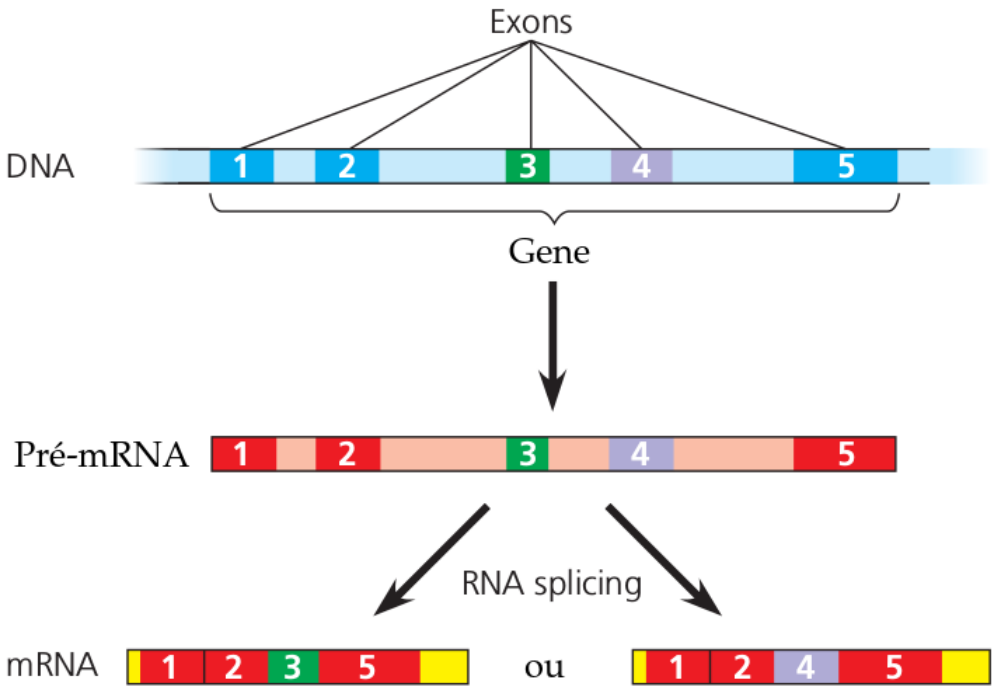


Figura 2.9: *Splicing* alternativo. Um gene pode ser processado em mais de uma maneira, gerando diferentes moléculas de mRNA. Note que os mRNA gerados possuem duas sequência de éxons diferentes, que serão traduzidos em proteínas diferentes. Figura adaptada de [28].

Tradução

A tradução é o processo pelo qual uma molécula de mRNA maduro, resultante após o processamento do pré-mRNA, é utilizada como um molde para a síntese de uma proteína. As proteínas são macromoléculas constituídas por uma ou mais cadeias de aminoácidos, participam em praticamente todos os processos celulares e possuem diversas funções nos organismos.

A tradução é realizada pelos ribossomos, que são grandes complexos macromoleculares de várias moléculas de rRNA e de proteína. Os ribossomos são responsáveis pela decodificação do código genético no mRNA, convertendo-o na sequência de aminoácidos de proteínas [89].

O código genético do mRNA é lido três nucleótidos por vez, em unidades chamadas códons, através de interações do mRNA com moléculas de RNA especializadas chamadas RNA de transferência (tRNA). Cada tRNA tem três bases desemparelhadas,

conhecidos como anticódon, que são complementares aos códons. O tRNA também é covalentemente ligado ao aminoácido especificado pelo seu anticódon. Quando o tRNA se liga ao seu códon complementar de uma cadeia de mRNA, o ribossomo liga sua carga de aminoácidos para a cadeia polipeptídica em construção. Quando a síntese da proteína é concluída, a proteína é liberada do ribossomo. Durante e após a sua síntese, a nova proteína deve-se enovelar para ativar sua estrutura tridimensional para que ela possa cumprir a sua função celular [110].

Uma única molécula de mRNA pode ser traduzida várias vezes e, assim, produzir muitas proteínas idênticas em função da sua meia-vida na célula, isto é, o tempo médio que permanece no interior da célula antes de ser degradada.

Todo o caminho percorrido desde a transcrição até a tradução de proteína pode ser vista na Figura 2.10.

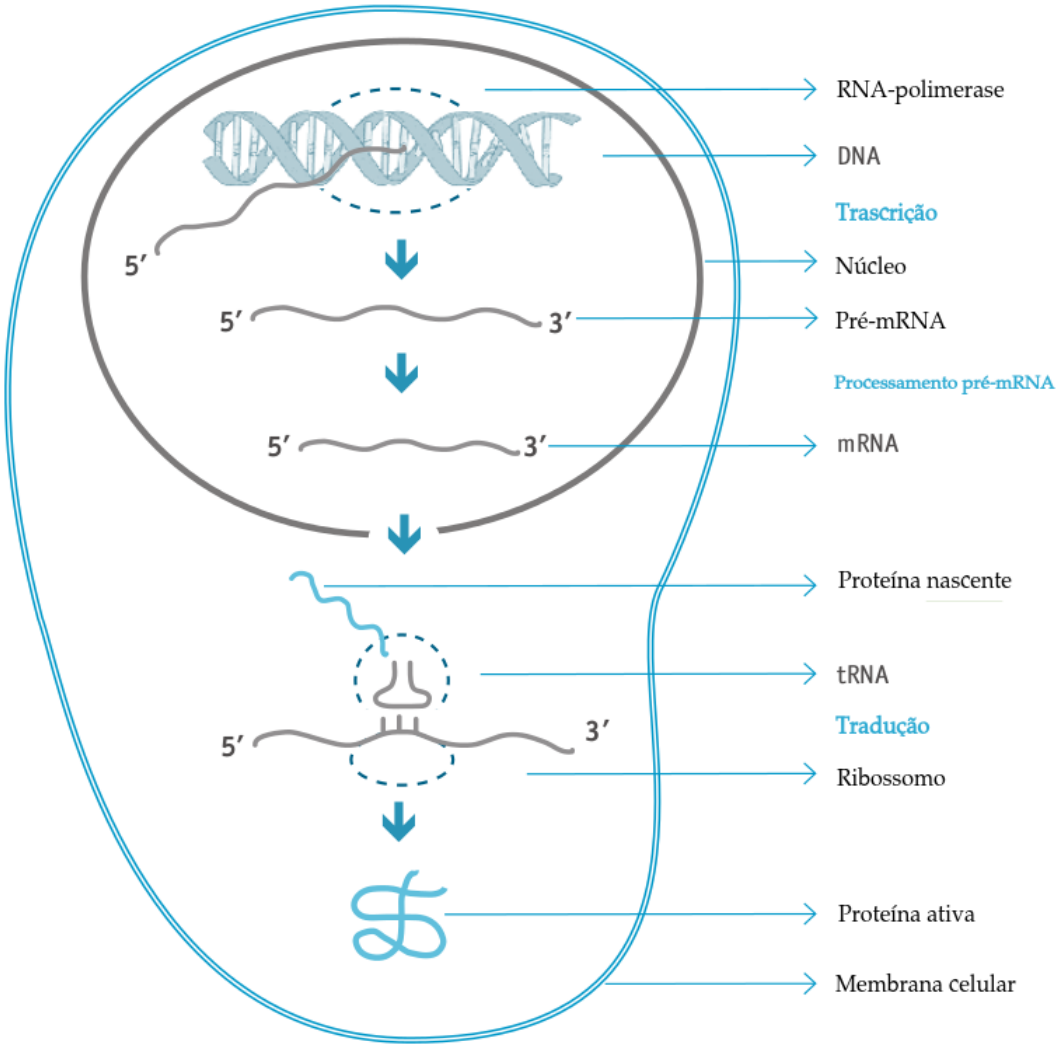


Figura 2.10: Fluxograma que mostra o caminho da transcrição do mRNA até sua tradução em uma proteína. Figura adaptada de [79].

Montagem e mapeamento

Para se obter a sequência de cada replicon de um genoma, usa-se o que é conhecido como sequenciamento, que consiste em uma técnica de biologia molecular que basicamente determina a sequência de nucleotídeos em uma determinada região do replicon.

A despeito do atual avanço nas técnicas conhecidas , o sequenciamento de genomas ainda é feito de maneira aleatória e gera sequências relativamente curtas, o que

resulta na necessidade de um processo de montagem dos fragmentos para obter a sequência do DNA. O processo de montagem consiste na sobreposição dessas sequências ou fragmentos, conhecidos como *reads*, com o objetivo de formar trechos contínuos de DNA, denominados *contigs*.

Apesar da grande quantidade e qualidade dos *reads* produzidos pelas diferentes plataformas de sequenciamento, é preciso considerar os seguintes aspectos: os *reads* são pequenos e são gerados milhões deles. Além disso, há erros de sequenciamento e há regiões repetitivas nos genomas. Todas essas características dificultam o processo de montagem e requerem o desenvolvimento de algoritmos e ferramentas computacionais mais eficientes para analisar esse grande volume de dados.

Se existir algum organismo muito semelhante ao genoma estudado, este pode ser usado como referência. Dessa maneira os *reads* podem ser mapeados nesse organismo. Neste caso, a montagem é denominada montagem por referência, mapeamento ou alinhamento, ou ainda montagem guiada. O genoma de referência deve ser muito próximo filogeneticamente do genoma estudado.

Na ausência de um genoma de referência, a estratégia utilizada é conhecida como montagem *de novo*. Nesse caso, a montagem não toma qualquer informação acerca do genoma estudado e monta os *contigs* somente com base na sobreposição dos fragmentos.

Os algoritmos e técnicas conhecidos para a montagem e para o mapeamento estão fora do escopo deste trabalho. No entanto, na Seção 3.1 descrevemos em detalhes um software que faz mapeamento, uma vez que este é um passo do pipeline descrito naquela seção.

Anotação

A anotação do genoma é o processo de atribuir informação biológica às suas sequências [99]. O processo é composto de três etapas principais: anotação em nível de DNA, anotação em nível de proteína e anotação em nível de processos biológicos [99].

A anotação em nível de nucleotídeos consiste na detecção dos genes codificadores e não codificadores de proteínas, além das regiões repetitivas e marcadores genéticos conhecidos.

Os genomas de bactérias e de arqueas possuem uma alta densidade de genes e pequenas regiões intergênicas adicionais. Normalmente, possuem cerca de um gene a cada 1000 nucleotídeos [78]. Essa característica dos procariotos tornam a identificação dos genes uma tarefa mais simples e consiste em encontrar longas ORFs (*Open Reading Frames*). Uma ORF é uma subsequência de DNA cujo comprimento é múltiplo de 3, começa com um *start* códon (`ATG` por exemplo) e termina imediatamente antes de um *stop* códon (`TAA`, `TAG`, `TGA`). Programas como GLIMMER [95] e GenMark [74] eficientemente encontram genes em bactérias [85].

Em contraste com os genomas procariotos, os genomas eucariotos contêm uma porcentagem pequena de genes e enormes quantidades de regiões não codificadoras. Este material não codificante inclui regiões repetitivas e íntrons que são removidos de transcritos de RNAs. Por isso, identificar genes codificadores de proteínas em genomas eucariotos é muito mais complexo do que em procariotos [85].

A anotação em nível de proteínas cataloga as proteínas encontradas, nomeando-as e identificando suas prováveis funções. Apesar de parecer um processo simples, na prática isso não é verdade. Durante a evolução, as proteínas podem ser duplicadas uma ou várias vezes, e suas cópias são diferentes, formando uma família de proteínas relacionadas, conhecidas como parálogas. Entretanto, nem sempre proteínas da mesma família possuem funções similares [99].

Uma forma comum de fazer a anotação de proteínas é procurar por similaridade, usando ferramentas como BLASTP [15], em diferentes bancos de dados de proteínas. Uma estratégia complementar é buscar por domínios protéicos em banco de dados como o PFAM [46]. Isso é possível porque partimos do pressuposto que proteínas muito similares têm estruturas similares e, portanto, tendem a desempenhar a mesma função [98].

A anotação em nível de processos identifica de quais vias metabólicas e processos os diferentes genes/proteínas participam. Essa etapa depende da existência de um banco de dados com a capacidade de relacionar genes que foram anotados por diferentes grupos de pesquisa. Essa base de dados é o *Gene Ontology* (GO), criada

em 1991 [52]. O GO é um vocabulário padrão para descrever as funções de genes. Ele está dividido em três partes: função molecular, processo biológico e componente celular. A parte de função molecular descreve as tarefas realizadas por produtos gênicos individuais. Processos biológicos são utilizados para objetivos biológicos mais amplos, como meiose ou crescimento e manutenção celular, por exemplo. Componentes celulares descrevem os genes em termos da estrutura subcelular na qual estão localizadas, como as organelas, bem como o complexo macromolecular a que pertencem, como os ribossomos, por exemplo [78].

Genômica comparativa

A partir da anotação de um genoma, é possível a execução de diversas análises, que são muito importantes na determinação de funcionalidades do organismo estudado.

A genômica comparativa é um campo de pesquisa biológica em que as características genômicas de diferentes organismos são comparados [104, 114]. As características genômicas podem incluir a sequência de DNA, genes, sequências regulatórias, e outros marcos estruturais genômicos [114]. Neste ramo da genômica, partes inteiras ou grandes do genoma resultantes de projetos de genoma são comparados para estudar semelhanças biológicas básicas e diferenças, bem como as relações evolutivas entre os organismos [87, 93, 104, 56].

2.2 Transcritômica

Com o surgimento, há cerca de 10 anos, das novas plataformas de sequenciamento, também chamadas de tecnologias de sequenciamento de nova geração (NGS), tornou-se possível gerar milhões de pares de base em um tempo muito curto e a um custo muito baixo. Essas tecnologias vêm se desenvolvendo muito rapidamente. Com essas técnicas, tornou-se possível o estudo do transcritoma de um organismo de forma bastante eficiente.

O transcritoma é o conjunto completo de transcritos de uma célula, e suas quantidades, para uma determinada condição ou um estágio em que ela se encontra. É o conjunto de todos os RNAs que estão sendo expressos em um determinado instante,

sejam RNAs que, após traduzidos, se transformam em proteínas, ou sejam, RNAs que codificam proteínas (mRNAs), ou RNAs que por si só têm sua função definida, sem que sejam traduzidos. Esses últimos são conhecidos por RNAs não-codificadores ou não-codificantes (ncRNAs).

Entender o transcritoma é essencial para a descoberta de funcionalidades de células e tecidos e também para a determinação do desenvolvimento de doenças. A transcritômica é a área da Biologia Molecular que consiste em catalogar todos os transcritos de uma célula, incluindo mRNAs e ncRNAs [112].

Uma das técnicas mais utilizadas para a transcritômica é conhecida como RNA-Seq, que permite a análise do transcritoma inteiro de um organismo de maneira muito eficiente. Em particular, RNA-Seq permite a determinação do nível de expressão de RNAs quando a célula encontra-se em determinada condição. Dessa forma é possível, por exemplo, quantificar diferentes níveis de expressão de cada RNA (seja ele mRNA ou ncRNA) em células saudáveis e com câncer. Essa técnica é conhecida como análise de expressão diferencial, e tem sido muito utilizada para a determinação de genes responsáveis pelo desenvolvimento de doenças.

2.3 Análise de expressão diferencial

Uma das contribuições deste trabalho é um pipeline para a análise de expressão diferencial, descrito no Capítulo 3. Por isso, nesta seção, esse tópico será abordado.

Um gene (seja ele resultante de mRNA ou ncRNA) é dito diferencialmente expresso se, dados dois conjuntos de amostras, é mais (*up-regulated*) ou menos (*down-regulated*) expresso, se comparado ao mesmo gene no outro conjunto. Essa análise leva ao desenvolvimento de várias estratégias para melhor entender essas diferenças.

Apesar de existirem diferentes *pipelines* para análise de expressão diferencial [29, 82], incluindo o proposto neste trabalho, eles podem ser resumidos em cinco passos, mostrados na Figura 2.11. A partir do resultado do sequenciamento, que basicamente consiste em um conjunto (em geral) de milhões de pequenas sequências ou pequenos fragmentos, a análise de expressão diferencial precisa de um pré-processamento, que é a verificação e limpeza dos *reads*. Após esse pré-processamento, eles são mapeados

em um genoma de referência. Em seguida, com o conjunto de *reads* mapeados e a anotação do genoma de referência utilizado, é feita a contagem de *reads* mapeados em cada gene. Por fim, métodos estatísticos são utilizados na análise de expressão diferencial. Essas etapas são agora descritas.

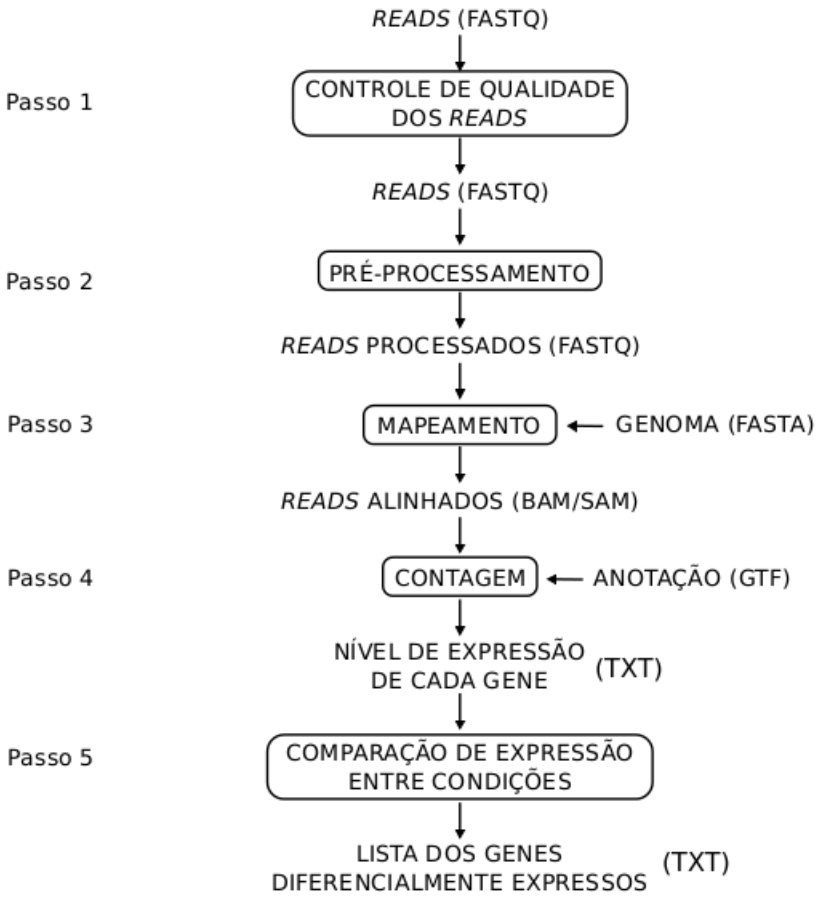


Figura 2.11: Visão geral de um *pipeline* para análise de expressão diferencial. Inicialmente os dados são avaliados e bases com baixa qualidade, adaptadores e contaminações são removidos, além disso, sequências menores que um determinado tamanho são removidas. As sequências restantes são mapeadas e o seu resultado é utilizado para estimar a abundância de cada gene. Por fim, os resultados são avaliados e os genes diferencialmente expressos são identificados. Entre parênteses estão os formatos de arquivos produzidos em cada etapa [78].

Controle de qualidade e pré-processamento

Os dados produzidos pelos sequenciadores apresentam ruídos, como sequências com baixa qualidade, contaminações e adaptadores. Nessa primeira etapa, é verificada

a qualidade de cada *read* na busca de ruídos e eles são removidos. O passo inicial de limpeza reduz o conjunto de *reads*, economizando recursos computacionais e aumentando a confiabilidade dos dados [78]. FastX-Toolkit [65], Cutadapt [75] e FastQC [18] são exemplos de programas disponíveis para a tarefa de controle e filtragem de *reads*.

Mapeamento de transcritos

Após toda a fase inicial de controle de qualidade e pré-processamento, os *reads* devem, após serem agrupadas com base em sobreposições, reconstruir o transcrito de onde foram sequenciados. O mapeamento consiste em mapear os *reads* no chamado genoma de referência, permitindo assim a determinação dos transcritos. O genoma de referência deve ser muito próximo filogeneticamente do genoma estudado. O mapeamento feito na ausência de um genoma de referência é chamado de montagem *de novo* e não toma qualquer informação acerca do genoma estudado, sendo considerada uma tarefa computacional muito custosa.

O mapeamento dos *reads* fornece várias informações sobre a amostra sequenciada, como polimorfismo e evidências de novos genes e ncRNAs, além de ser usado para quantificar o nível de expressão de genes e transcritos, uma vez que o número de *reads* produzidos por um transcrito é proporcional à sua abundância na célula [106].

Alguns exemplos de mapeadores utilizados atualmente são: `segemehl`, `Bowtie` [66], `Bowtie2` [67], `Tophat` [109], `Tophat2` [64] e `STAR` [30].

Contagem

Depois dos *reads* terem sido mapeados no genoma de referência, juntamente com a anotação genômica, é possível quantificar a expressão gênica. A maneira mais comum de estimar a expressão é realizar a contagem de *reads* que foram mapeados em cada gene, transcrito ou éxon. Programas como `HTSeq` [17], `BEDTools` [88] e `Cufflinks` são utilizados para essa função.

Teoricamente, contar o número de *reads* mapeados forneceria uma forma direta de estimar a abundância de um transcrito porém, na prática, outros fatores devem ser

considerados, pois o número de *reads* gerados por transcrito depende da cobertura do sequenciamento e do tamanho do transcrito. Logo, cada *software* utiliza diferentes estratégias para a contagem de *reads*.

Expressão diferencial

Após feita a contagem dos *reads* das amostras, a fase para identificar genes diferencialmente expressos envolve a normalização de dados, a escolha de um modelo estatístico e o teste para expressão diferencial. Esses passos são implementados por diferentes programas, dentre os quais podemos citar: `DESeq` [16], `DESeq2`, `EdgeR` e `Cuffdiff` [107].

Os métodos de normalização tentam controlar as diferenças das amostras, e assim facilitar a precisão das comparações entre as diferentes condições estudadas. Existem diversos métodos desenvolvidos para a normalização de dados, entretanto não há uma unanimidade sobre qual a melhor escolha. Entre os métodos disponíveis estão *total count* (TC), RPKM [77] e FPKM [107].

O método TC divide a contagem de cada gene pelo número total de *reads* mapeados na amostra, e multiplica o resultado pela contagem total média de todas as amostras do conjunto de dados [38]. Já o conceito de RPKM (*Reads Per Kilobase per Million mapped reads*) normaliza o número de *reads* mapeados em um dado transcrito, visto que transcritos maiores tendem a gerar mais *reads* que transcritos menores. Além disso, em amostras diferentes, a diferença entre o montante total de *reads* usados pode influenciar na contagem. O cálculo de RPKM é dado pela fórmula

$$\text{RPKM} = \frac{C}{\frac{L}{10^3} \times \frac{N}{10^6}} = \frac{10^9 \times C}{L \times N},$$

onde C é o número de *reads* mapeados no transcrito, L é o tamanho do transcrito e N é o número total de *reads* mapeados. As divisões por 10^6 e 10^3 são feitas proporcionalmente aos altos valores de N e L, respectivamente. A divisão por L é uma normalização que leva em conta os diferentes tamanhos dos transcritos, enquanto que a divisão por N leva em conta o número total de *reads* mapeados.

De modo análogo, o FPKM (*Fragments Per Kilobase of transcript per Million fragments mapped*) lida com dados *paired-end*. O FPKM é uma unidade de expressão

que considera cada fragmento mapeado e não cada *read*. O fragmento é definido como um par de *reads*.

Outro ponto que deve ser considerado diz respeito à contagem total de uma amostra, que pode ser altamente dependente de poucos genes altamente expressos (Tabela 2.1) [78]. Isso faz com que a expressão dos demais genes seja subestimada. Para lidar com esse problema, a estratégia é utilizar fatores de escala como *Trimmed Means of M values* (TMM) [92] ou *upper-quartile* [35].

Tabela 2.1: Diferença na contagem das amostras. Exemplo de contagem onde o número de *reads* nas duas amostras são iguais, mas com diferentes distribuições.

	Amostra A	**Amostra B**
gene 1	100	80
gene 2	100	80
⋮	⋮	⋮
gene 99	100	80
gene 100	20	2000
	$d_A = 9920$	$d_A = 9920$

O TMM é calculado removendo uma determinada porcentagem dos genes considerados os mais extremos entre as amostras comparadas. Este fator de normalização é então usado para corrigir as diferenças no tamanho das amostras. Já o *upper-quartile* é semelhante ao principio de *total count*, onde a contagem total são substituídos pelo quartil superior de contagens diferentes de 0 no cálculo dos fatores de normalização.

A comparação do nível de expressão dos genes em duas diferentes condições invariavelmente possuirá diferentes contagens. Essa diferença pode ser devida à verdadeira diferença entre as condições ou apenas a ruídos do experimento. A combinação de modelos de distribuição e testes estatísticos permitem distinguir entre essas duas possibilidades [78].

2.4 Análise funcional

A análise funcional de um organismo passa pelo estudo de seu ciclo de vida e por como os mecanismos e processos celulares se comportam frente a diferentes situações. Todos esses mecanismos são executados, em última instância, pelas proteínas produzidas pela célula e principalmente como elas interagem.

Nesta seção alguns conceitos, banco de dados e ferramentas relacionadas à análise funcional são brevemente descritos. Como a ferramenta DEPICTViz, descrita no Capítulo 4, relaciona dados de análise de expressão diferencial com dados de interação entre proteínas, o que listamos a seguir são softwares ou banco de dados que envolvem essas informações, particularmente aqueles que foram sugeridos nos projetos onde DEPICTViz foi utilizado. Esses projetos estão descritos no Capítulo 5.

COG

As relações entre genes de diferentes genomas são naturalmente representadas como um sistema de famílias homólogas que incluem ambos os ortólogos e parálogos. Ortólogos são genes de diferentes espécies que evoluíram a partir de um gene ancestral comum por especiação. Já os parálogos, são genes originados por duplicação dentro de um genoma [47]. Normalmente, ortólogos possuem a mesma função no curso da evolução, ao passo que parálogos evoluem novas funções, mesmo que estejam relacionadas à original. Logo, a identificação de ortólogos é fundamental para a previsão confiável de funções de genes em genomas recém-sequenciados [101]. É igualmente importante para a análise filogenética, pois as árvores filogenéticas interpretáveis geralmente podem ser construídas apenas dentro de conjuntos de ortólogos [48].

Dada as várias relações que podem existir entre ortólogos, foi criado o conceito de *Clusters of Orthologous Groups* (COGs). Cada COG consiste de genes ortólogos individuais ou grupos ortólogos de parálogos de três ou mais linhagens filogenéticas. Em outras palavras, quaisquer duas proteínas diferentes a partir de linhagens que pertencem ao mesmo COG são ortólogos. Cada COG é assumido ter evoluído a partir de um gene ancestral através de uma série de eventos de especiação e duplicação [101].

A aplicação mais direta para os COGs é a predição de funções de proteínas individuais ou grupos de proteínas, incluindo as encontradas em novos genomas recentemente sequenciados [100]. Os COGs são classificado em 26 categorias funcionais [42] representadas por letras, onde cada COG pode ser associado a uma ou mais categorias. Cada identificador COG tem o formato COGX, onde X representa uma sequência de números inteiros. Dentre as bases de dados que utilizam COG, podemos citar: NCBI [50], EggNOG [36] e UniProt [19].

GO

Similarmente ao COG, o crescente volume de dados e a ampla difusão de base de dados biológicos trouxe a necessidade de se fazer o melhor proveito de toda essa informação, onde diferentes tipos de informação de diferentes fontes precisam ser integradas de maneira a serem úteis para os biólogos. Um grande esforço realizado para tal foram o desenvolvimento e uso de padrões estruturados de anotação como ontologias. As ontologias fornecem os conceitos de domínios de conhecimento e facilita a comunicação de pesquisadores e a interoperabilidade entre base de dados [52]. O projeto *Gene Ontology* (GO) tem como objetivo desenvolver e utilizar ontologias para apoiar anotação biologicamente significativa de genes e seus produtos de uma grande variedade de organismos [102].

O GO é composto por três ontologias [33]: (i) a ontologia componente celular, que pode ser usada para descrever os produtos de genes de acordo com a sua localização celular ou como parte de um complexo de proteínas, (ii) a ontologia função molecular, que inclui termos descrevendo atividades enzimáticas e de ligação e (iii) a ontologia processo biológico, que contém termos que descrevem qualquer série de eventos em uma célula ou organismo realizado por um ou mais conjuntos ordenados de funções moleculares.

A atribuição de um termo (identificador) GO para um produto de gene é baseada no conhecimento de qualquer produto de gene que foi adquirido com a experimentação direta, como relatado na literatura, ou por meio de similaridade de sequência para outro produto de gene que se tem sido descrito experimentalmente. O GO é frequentemente utilizado na análise de conjuntos de *high-throughput datasets*, particularmente para a análise de enriquecimento onde um conjunto de genes de interesse

são analisados através dos termos GO [44] e para a identificação de módulos funcionais dentro das redes proteína-proteína [33]. Cada identificador GO possui uma representação semelhante ao COG, cujo formato é GOX, sendo X uma sequência de números inteiros. Dentre as várias base de dados que utilizam GO, destaca-se: AmiGO [45], EggNOG e UniProt.

UniProtKB/Swiss-Prot

O *UniProt Knowledgebase* (UniProtKB) é um banco de dados manualmente curado, sendo um ponto de acesso central para informações de proteínas e integrado com referências cruzadas de múltiplas fontes (COG e GO incluídos). Uma de suas seções, conhecida como UniProtKB/Swiss-Prot, é responsável por armazenar anotações manuais que foram extraídas da literatura e de análises computacionais após a avaliação de um especialista. A anotação consiste em descrever informações de uma proteína como: funções, informações enzimáticas, domínios e locais biologicamente relevantes, estrutura, interações e doenças associadas ou deficiências [103].

Sequências de um mesmo gene ou de uma mesma espécie são unidas em uma mesma entrada na base de dados, a fim de manter o mínimo de redundância possível. Diferenças entre sequências são identificadas e documentadas (*splicing* alternativo, variação natural, etc) [1]. Cada identificador do Swiss-Prot pode ser representada como X_Y, onde X é um mnemônico de caracteres alfanuméricos que representam o nome da proteína e Y é um código de, no máximo, cinco caracteres alfanuméricos representando a fonte biológica da proteína [113]. Além do UniProtKB, outras bases usam identificadores do Swiss-Prot, destacando-se: OMA Browser [96], SWISS-MODEL [55] e STRING [37].

EC

O *Enzyme Commission number* (EC *number*) é um sistema de classificação numérica para as enzimas, com base nas reações químicas que elas catalisam. Como é um sistema de nomenclatura para enzimas, cada EC *number* é associado a um nome recomendado para a respectiva enzima. Estritamente falando, EC *numbers* não especificam enzimas, mas sim as reações catalisadas por enzimas. Se enzimas dife-

rentes (por exemplo, a partir de diferentes organismos) catalisam a mesma reação, logo recebem o mesmo EC *number* [14].

Um EC *number* é representado pelas letras 'EC' seguido por quatro dígitos separados por pontos (EC 1.1.1.1). Esses números representam uma classificação progressivamente mais detalhada da enzima. O primeiro, segundo, terceiro e quarto número representam, respectivamente, a classe, subclasse, subsubclasse e serial. A classe representa o tipo de reação química realizada pela enzima (Oxirredutases, Transferases, Hidrolases, Liases, Isomerases e Ligases), a subclasse especifica o tipo da reação enzimática, a subsubclasse especifica os substratos necessários e o serial é usado para identificar enzimas individuais dentro da subsubclasse [76]. Quando não é possível fazer uma completa caracterização enzimática em um dos níveis, o EC *number* é considerado incompleto [32] e os níveis são preenchidos com o carácter '-' (EC 1.1.1.-, EC 1.1.-.- e EC 1.-.-.-). Dentre as diversas bases de dados que utilizam EC *numbers* estão: ExPASy [49], BRENDA [97], ExplorEnz [34], MetaCyc [43] e KEGG [39].

KEGG

O KEGG (Kyoto *Encyclopedia of Genes and Genomes*) é uma base de dados de recursos que integra informações genômicas, químicas e sistemáticas funcionais para interpretação de sequências de um genoma e *high-throughput data*. O principal objetivo do KEGG tem sido estabelecer ligações entre conjuntos de genes do genoma e funções de alto nível da célula e do organismo. Com o grande volume de dados biológicos que estão sendo gerados atualmente, o KEGG se tornou uma das bases de dados biológicas mais utilizadas em todo o mundo [39].

A anotação genômica no KEGG é feita através da associação de grupos ortólogos à funções moleculares de genes e proteínas armazenadas em uma base de dados chamada KEGG *Orthology* (KO), permitindo assim a extensão de evidência experimental de um organismo específico para outros organismos. A anotação de genes individuais é feita atribuindo identificadores para as entradas da base KO, chamado K *numbers*. Um K *number* tem a forma KX, onde X representa uma sequência de números inteiros que identifica a entrada. A representação das funções de alto nível associadas podem ser visualizadas através do KEGG *Pathway* [39].

Uma via metabólica (*pathway*) é uma série de ações entre moléculas em uma célula que leva a um certo produto ou mudança na célula. Dessa forma, um *pathway* pode desencadear a síntese de novas moléculas, tais como um ácido graxo ou proteína. *Pathways* também podem ativar ou inibir genes ou estimular uma célula a se mover. Os *pathways* biológicos mais comuns estão envolvidos no metabolismo, regulação de expressão de genes e transmissão de sinais. [3].

UCSC Genome Browser

O *University of California Santa Cruz* (UCSC) *Genome Browser* é um conjunto de ferramentas disponíveis publicamente para visualização e análise de dados biológicos de uma grande variedade de espécies de vertebrados e invertebrados, fornecidos pela Universidade da Califórnia. Há ainda a possibilidade do usuário fazer o *upload* e visualizar seu próprio conjunto de dados.

O UCSC *Genome Browser* provê uma plataforma web interativa para a visualização de grandes conjuntos de dados genômicos, conhecidos como *tracks*, que incluem predições de genes, níveis de expressão e elementos regulatórios. A interface do usuário oferece uma ampla gama de personalização, tanto na exibição das *tracks* disponíveis, bem como da própria janela do *browser*. A janela de cada *track* é dinâmica, onde as *tracks* podem ser adicionadas, removidas ou realocadas. O *Genome Browser* também incluí várias opções para a navegação entre diferentes regiões do genoma, permitindo *zoom* em diferentes níveis do genoma, desde uma única base até um cromossomo inteiro. Após configurar o *Genome Browser* para seu conjunto de dados, o usuário pode salvar sua visualização em uma sessão, para uma análise posterior ou compartilhar com outras pessoas [40].

Interação funcional de proteínas

Interações proteína-proteína (PPIs - *Protein-protein interactions*) controlam uma vasta gama de processos biológicos, incluindo as interações célula-a-célula, controle metabólico e desenvolvimental [26].

Dificilmente as proteínas agem isoladamente no exercício das suas funções *in vivo* [115]. Foi revelado que mais de 80% das proteínas não operam sozinhas, mas sim em complexos [21]. Além disso, proteínas envolvidas em um mesmo processo celular são repetidamente encontradas interagindo entre si. O estudo de PPIs também é importante para inferir a função da proteína dentro da célula. A funcionalidade de proteínas não identificadas podem ser previstas sobre a evidência da sua interação com outra proteína, cuja função já é conhecida [111].

Dentre várias funções que PPIs desempenham estão [86]: modificar propriedades cinéticas de enzimas, ativação ou supressão de proteínas e papel regulador em qualquer nível acima ou abaixo.

O estudo detalhado dessas interações tem acelerado a modelagem das vias funcionais para exemplificar os mecanismos moleculares dos processos celulares [117] e identificado alvos para medicamentos [84]. Entre as bases de dados existentes sobre interação funcional de proteínas, podemos citar o STRING [37].

Capítulo 3

Pipeline para análise de expressão diferencial

Este capítulo descreve o desenvolvimento de um pipeline para análise de expressão diferencial. No contexto deste trabalho, entende-se por *pipeline* uma sequência ordenada de tarefas tais que a execução de uma tarefa tem como entrada a saída da tarefa anterior, na sequência.

A abordagem adotada para o desenvolvimento do pipeline pressupõe a seguinte sequência de passos:

- mapeamento das sequências no genoma de referência;
- construção dos transcritos;
- união dos transcritos;
- análise de expressão diferencial; e
- visualização dos resultados.

Um ponto importante a se destacar nessa abordagem é que estamos considerando que os *reads* já passaram por quaisquer processos de filtragem e limpeza. Esta etapa, portanto, fica fora do escopo deste trabalho.

3.1 Mapeamento

O conjunto de *reads* gerado pelo sequenciamento devem, após serem agrupados com base em sobreposições, reconstruir o transcrito de onde foram sequenciados. O processo de montagem *de novo* dessas sequências é uma tarefa computacional muito custosa. Uma abordagem menos custosa, e que é utilizada neste trabalho, consiste em mapear as sequências no chamado genoma de referência, permitindo assim a determinação dos transcritos.

Primeiramente, foi preciso decidir qual o *software* a ser usado para fazer mapeamento dos *reads*. O mapeamento dos *reads* no genoma de referência é uma das mais importantes tarefas computacionais de todo o processo, uma vez que geralmente eles podem ser mapeados em várias posições do genoma. Para essa etapa, foi decidido utilizar o *software* `segemehl` que é um mapeador que lida eficientemente com *mismatches*, inserções e remoções em *reads* de tamanhos pequenos e médios.

Para chegar a essa decisão, foi realizado um experimento de análise de expressão diferencial descrito na seção 5.1. Pela análise deste experimento realizado e pelo `segemehl` lidar eficientemente com *mismatches*, inserções e remoções, foi o mapeador escolhido.

`segemehl`

A estratégia empregada pelo `segemehl` é baseada no uso de *arrays* de sufixo aprimorados (Figura 3.1), que procuram encontrar o melhor alinhamento local dos *reads* e o genoma de referência utilizando um simples sistema de pontuação apresentado na Tabela 3.1. Isso é feito determinando, para cada sufixo do *read*, o maior prefixo que ocorre no genoma de referência. Isto forma um *matching backbone*, onde um número limitado de ramificações são derivadas por *mismatches*, inserções e remoções [59].

i	suf	lcp	$S_{\text{suf}[i]}$
0	0	0	*acttcttcggc$*
1	7	0	*cggc$*
2	4	1	*cttcggc$*
3	1	4	*cttcttcggc$*
4	10	1	*c$*
5	9	0	*gc$*
6	8	1	*ggc$*
7	6	0	*tcggc$*
8	3	2	*tcttcggc$*
9	5	1	*ttcggc$*
10	2	3	*ttcttcggc$*
11	11	0	$

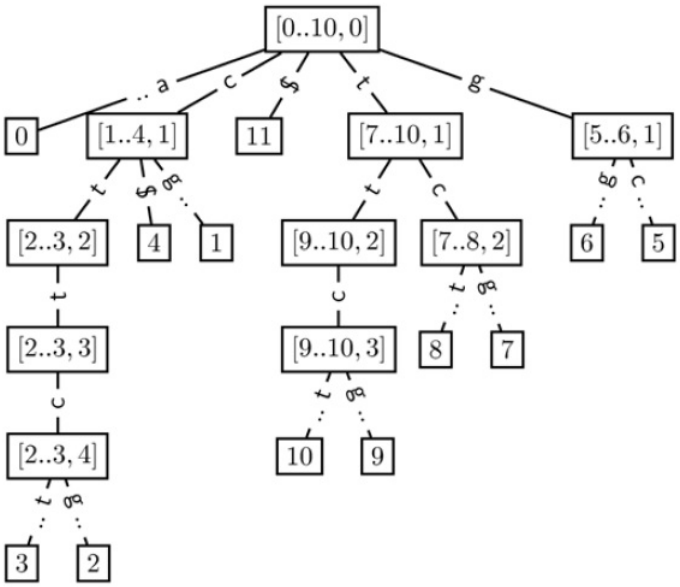


Figura 3.1: *Array* de sufixo aprimorado (esquerda) para a sequência S = attcttcggc e sua árvore de intervalo de sufixos (direita). O *array* 'suf' representa os sufixos de S ordenados lexicograficamente, 'lcp' representa o tamanho do maior prefixo comum de $S_{suf[i]}$ e $S_{suf[i-1]}$. A partir do *array* de sufixo aprimorado é possível gerar a árvore de intervalos de sufixo, onde um intervalo de sufixo é denotado por *[l..r,h]*, sendo que todos os sufixos do intervalo *[l+1..r]* possuem prefixo comum máximo menor ou igual a h [59].

Tabela 3.1: Sistema de pontuação utilizado pelo `segemehl`.

Match	+1
Mismatch	-1
Inserção	-1
Remoção	-1

Antes de fazer o mapeamento, é necessário construir os índices para o genoma de referência apenas uma vez, de modo que a estrutura fique salva em disco. A construção é feita em tempo linear a partir do seguinte comando:

```
./segemehl.x -x file.idx -d file.fa,
```

onde o `segemehl` é executado com a opção `-x`, que indica que o índice será gerado e armazenado em disco. `file.idx` é o nome do arquivo do índice que será criado, `-d` é a

opção que indica que o próximo parâmetro será o arquivo contendo a referência para a qual se deseja criar o índice e, por fim, o arquivo `file.fa` contendo a referência em formato `fasta`. Para se ter uma ideia, o *array* de sufixo aprimorado ocupa aproximadamente 4GB de espaço em disco para o cromossomo 1 humano e consome cerca de 6 GB de RAM.

Após concluída a construção do índice, o mapeamento de um conjunto de *reads* é feito a partir do comando

```
./segemehl.x -i file.idx -d file.fa -q myreads.fa > mymap.sam,
```

onde o `segemehl` é executado com a opção `-i`, que indica que o próximo parâmetro a ser passado será o índice que foi criado na etapa anterior. `file.idx` é o caminho para o índice, `-d` indica que o próximo parâmetro a ser passado será a referência usada para criar o índice, `file.fa` é a referência do índice, `-q` indica que o próximo parâmetro a ser passado será o conjunto de *reads* que se deseja mapear, `myreads.fa` é o conjunto de *reads* e `> mymap.sam` indica que a saída do `segemehl` deve ser redirecionada para o arquivo `mymap.sam` de formato `SAM`. Todas as informações adicionais necessárias podem ser encontradas em [57].

Modo de mapeamento

Após o mapeador ter sido escolhido, foi preciso decidir como o mapeamento deve ser feito, se utilizando o genoma humano completo como referência ou apenas os locais onde os genes de interesse se encontram, na tentativa de diminuir o tempo e espaço necessário para o mapeamento.

Após a realização do experimento descrito na Seção 5.1, constatou-se que a melhor decisão é a de utilizar o genoma completo, pois ao mapear os *reads* apenas em trechos onde os genes de interesse estão localizados, o mapeador tende a mapeá-los naquelas posições de alguma forma. Ao mapear contra o genoma completo, o mapeador encontrará a melhor posição possível para cada *read*, sendo mais preciso. A Figura 3.2 ilustra os dois casos, onde G é o mapeamento feito no genoma completo, H apenas nas regiões onde são encontrados os genes de interesse e a região destacada é região de interesse.

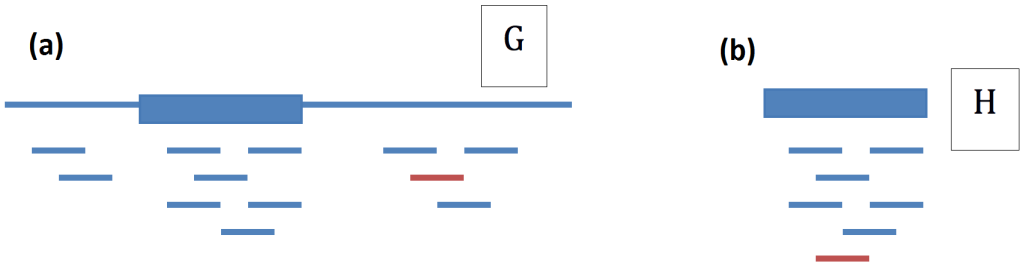


Figura 3.2: Mapeamentos no genoma completo e em trechos de interesse: (a) Mapeamento feito no genoma completo, onde o *read* destacado em vermelho é mapeado fora da região de interesse e em (b) o mesmo é forçado a ser mapeado na região.

3.2 Construção dos Transcritos

Neste ponto do processo tem-se o resultado do mapeamento de cada uma das amostras contra o genoma de referência. A construção dos transcritos é feita pelo `Cufflinks` [107, 108]. Vale notar aqui que `Cufflinks` é o nome dado tanto para o pacote todo, que executa os passos de construção dos transcritos, união dos transcritos e análise de expressão diferencial, como também para o *software* que faz a construção dos transcritos propriamente dita.

Uma adaptação à saída do `segemehl` precisou ser feita para que o `Cufflinks` pudesse tomá-la como entrada. Basicamente, essa adaptação consiste na ordenação pela posição de cada *read* mapeado no genoma de referência. Isso é feito usando o *software* `SAMtools`. Mais detalhes podem ser vistos na Seção 5.2.

O `Cufflinks` constrói transcritos individuais de *reads* de RNA-seq que foram mapeados no genoma. Como métrica de contagem de *reads* mapeados, o programa utiliza o conceito de RPKM (*Reads per kilobase per million mapped reads*) para a normalização do número de *reads* mapeados em um dado transcrito, descrito na Seção 2.2.

Outro passo importante na determinação dos transcritos tem a ver com um fenômeno muito comum em eucariotos, chamado de *splicing* alternativo e descrito na Seção 2.1, onde apenas alguns éxons de um transcrito são efetivamente usados na transcrição.

Quando um gene sofre *splicing* alternativo e produz múltiplas isoformas na mesma amostra, muitos *reads* serão mapeados em éxons compartilhados, o que complica o processo de contagem dos *r*eads para cada transcrito. Assim, para calcular com precisão o nível de expressão para cada transcrito, um procedimento simples de contagem não é suficiente. Para resolver esse problema, o pacote `Cufflinks` implementa um modelo estatístico linear que observa os *reads* com maior verossimilhança (método conhecido como *maximum likelihood*) [106].

A estimativa por máxima verossimilhança é um método para estimar os parâmetros de um modelo estatístico. Assim, dado um conjunto de dados e um modelo estatístico, o método de máxima verossimilhança estima os valores dos diferentes parâmetros do modelo estatístico de maneira a maximizar a probabilidade dos dados observados, isto é, busca parâmetros que maximizem a função de verossimilhança.

A construção dos transcritos é um processo oneroso e, após concluído, será armazenado em arquivos as informações refente as contagens de genes, isoformas e a construção dos transcritos em si.

O *pipeline* até este ponto pode ser visto na Figura 3.3, que deve ser executado para cada amostra.

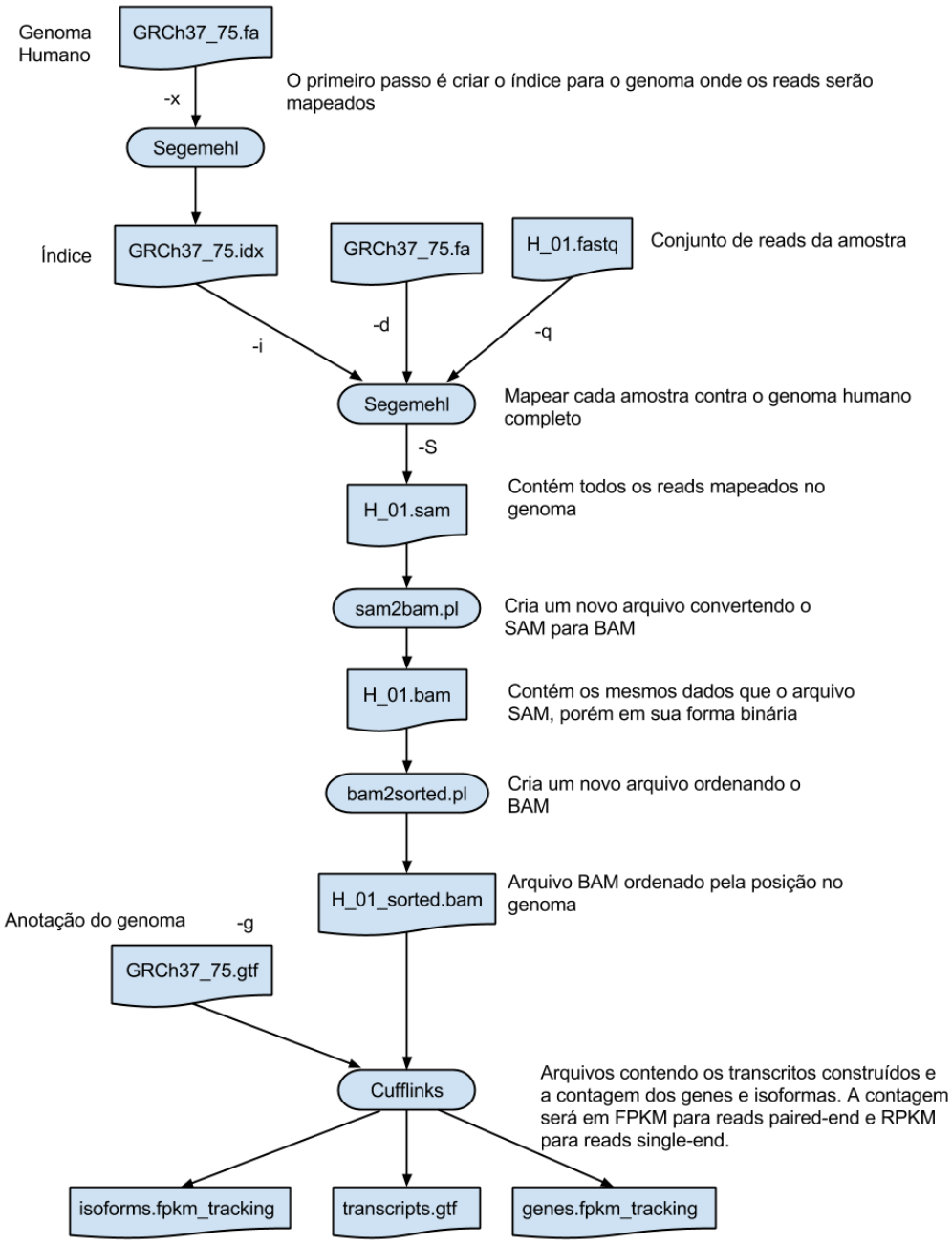


Figura 3.3: Ilustração do *pipeline* até a construção dos transcritos. O exemplo usado nesta Figura é referente ao experimento descrito na Seção 5.2.

3.3 União dos transcritos de todas as amostras analisadas

Após ter executado o `Cufflinks` para todas as amostras como descrito na seção anterior, haverá um arquivo `transcripts.gtf` para cada amostra. O próximo passo a ser executado é unir todos os transcritos construídos em um único arquivo, executado pelo `Cuffmerge`. Essa união irá prover uma base uniforme para o cálculo de expressão de genes e transcritos em cada condição, que será feita pelo `Cuffdiff` e descrita na próxima seção. Ainda, se houver a anotação do genoma disponível, como é o caso deste trabalho, é possível passá-lo ao `Cuffmerge` a fim de unir isoformas novas e conhecidas e maximizar a qualidade geral do arquivo gerado. O principal objetivo do `Cuffmerge` é construir um arquivo GTF adequado para o `Cuffdiff`.

3.4 Expressão diferencial

Ao unir todos os transcritos construídos em um único arquivo GTF, é possível então calcular os níveis de expressão diferencial entre duas ou mais amostras e testar as mudanças significativas observadas entre elas utilizando o `Cuffdiff`. O modelo estatístico usado para avaliar as diferenças assume que o número de *reads* produzido por cada transcrito é proporcional à sua abundância. Com várias replicatas de uma mesma condição, o `Cuffdiff` analisa como a contagem de *reads* varia para cada gene entre as replicatas e utiliza essa estimativa de variância para calcular as diferenças de expressão significantes [106].

O resultado gerado é distribuído em vários arquivos contendo os resultados da análise da expressão diferencial das amostras. Os níveis de expressão de genes e transcritos são reportados em arquivos de simples tabulação, capazes de serem visualizados em qualquer editor de texto disponível. Estes arquivos contêm informações disponíveis como *fold change*, nome de genes e transcritos e localização no genoma. A medida *fold change* é dada pela divisão dos níveis de expressão nas duas condições, digamos A e B. O objetivo é mostrar como uma expressão é mais abundante do que a outra. Geralmente usa-se *log-fold-change* (base 2) para deixar os valores simétricos e de visualização mais simples.

As Figuras 3.4 e 3.5 mostram os gráficos *Smear* resultantes da análise feita no experimento descrito na Seção 5.1, utilizando `Bowtie` e `segemehl`, respectivamente. O gráfico *Smear* mostra o nível de expressão gênica entre conjuntos, onde quanto maior o valor no eixo horizontal, maior é o nível de expressão de um determinado gene; e quanto maior o valor no eixo vertical, maior é a diferença de expressão de determinado gene para o conjunto de dados *treated* em relação ao conjunto *untreated*. Os genes diferencialmente expressos são coloridos em vermelho, os não diferencialmente expressos são coloridos em preto e os coloridos em alaranjado representam genes cuja contagem de *reads* mapeados foi igual a zero para todas as amostras de uma das condições.

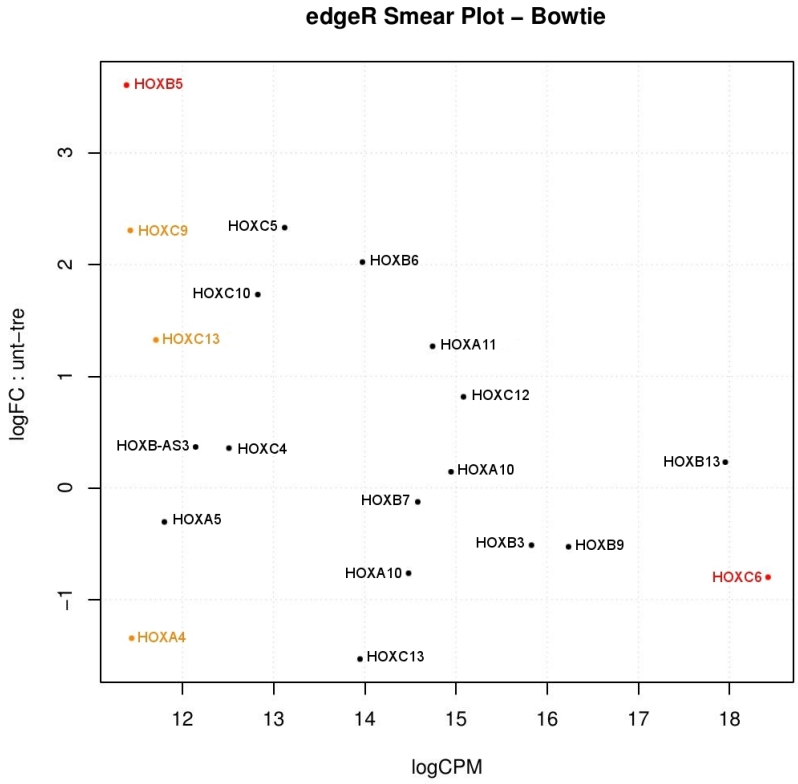


Figura 3.4: Gráfico *Smear* gerado utilizando o mapeamento pelo `Bowtie`.

Há ainda resultados adicionais que são gerados, como a identificação de genes que são diferencialmente *spliced* ou diferencialmente regulados, onde o `Cuffdiff` agrupa isoformas de genes que possuem o mesmo TSS (*Transcription Start Size*) e procura por diferenças relativas entre genes que possuem múltiplos TSS [106].

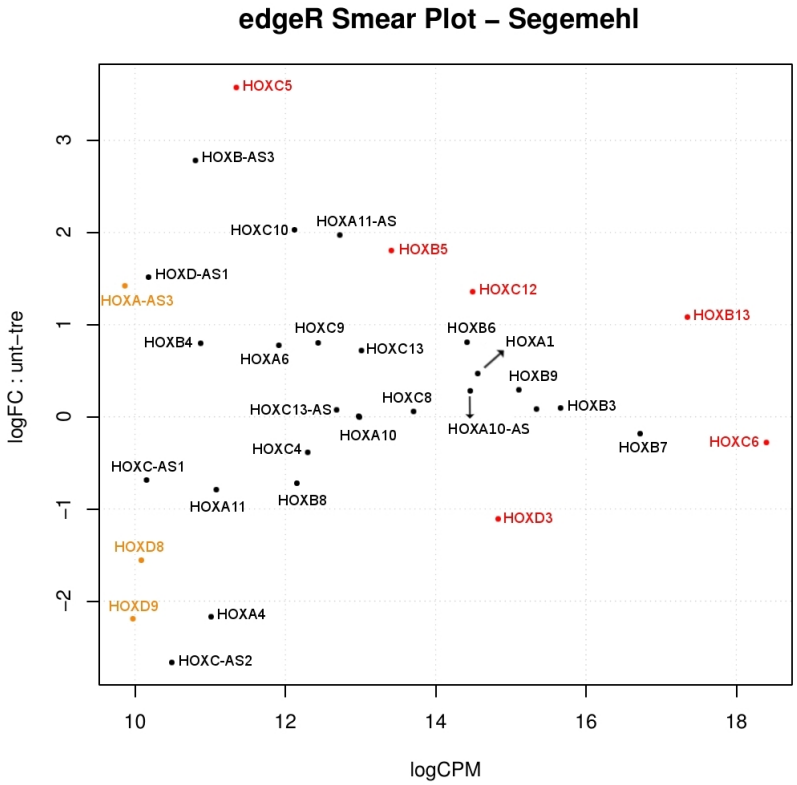


Figura 3.5: Gráfico *Smear* gerado utilizando o mapeamento feito pelo `segemehl`.

Da mesma forma como o `Cufflinks`, a execução do `Cuffdiff` é um passo oneroso e, ao fim deste, a saída gerada consiste em arquivos de quatro extensões diferentes, sendo elas: `diff`, `FPKM_tracking`, `count_tracking` e `read_group_tracking`. Os arquivos com extensão `diff` são resultados de expressão diferencial entre as amostras para transcritos *spliced*, transcritos primários, genes e sequencias codificadoras, onde a última coluna do arquivo informa se o item é diferencialmente expresso ou não. Os arquivos com extensão `FPKM_tracking` mostra os cálculos de RPKM para cada transcrito *spliced*, transcrito primário e gene em cada amostra. Os arquivos com extensão `count_tracking` mostra o número de fragmentos que originou cada transcrito *spliced*, transcrito primário e gene em cada amostra. Por fim, os arquivos com extensão `read_group_tracking` calcula a expressão e contagem de fragmentos para cada transcrito *spliced*, transcrito primário e gene em cada replicata.

3.5 Análise dos resultados

O `cummeRbund` [53], ferramenta projetada para simplificar a visualização de dados de RNA-seq, é um pacote da linguagem `R` que tem como objetivo gerenciar, visualizar e integrar todos os dados produzidos pela análise do `Cuffdiff`, simplificando tarefas comuns de exploração de dados, plotagem de gráficos e análises de expressão diferencial [106]. Há vários gráficos prontos para publicação que podem ser gerados, tais como: MAplot, volcano, *box plot*, similaridade e dendrograma.

O restante do *pipeline* proposto pode ser visto na Figura 3.6.

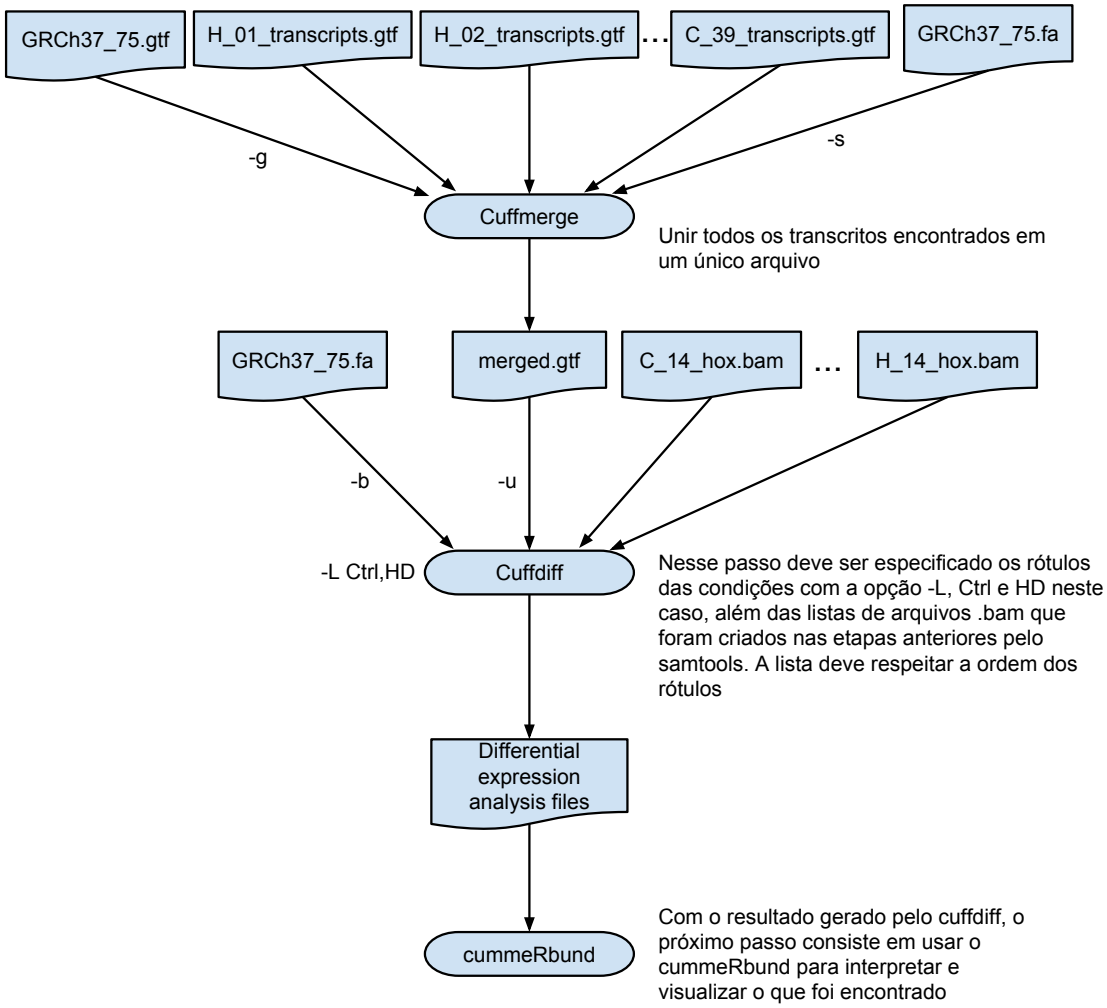


Figura 3.6: Ilustração do restante do *pipeline* proposto, usando dados do experimento descrito na Seção 5.2.

Outra possível análise a ser feita após a construção e determinação dos transcritos tem a ver com a possibilidade de se encontrar novas *features*, não incluídas na anotação dada como entrada. Isso poderá ser feito pela busca em bases de dados previamente definidas pelo usuário no arquivo de configuração do *pipeline*. A ideia é então buscar nessas bases, respostas na identificação dessas novas *features* encontradas para, por exemplo, poder classificar todos os tipos de ncRNAs e mRNAs *up/down-regulated* não identificados na anotação dada.

3.6 Construção do Pipeline

Nesta seção será abordado o desenvolvimento do *pipeline* proposto propriamente dito, realizando todas as etapas descritas até aqui.

Funcionalidades

O *pipeline* desenvolvido realiza o mapeamento das sequências no genoma de referência utilizando o `segemehl`, ordenação dos *reads* mapeados com o `SAMtools`, construção e união de transcritos e análise de expressão diferencial utilizando o pacote `Cufflinks`; e visualização dos resultados utilizando o `cummeRbund` e DEPICTViz, abordado no Capítulo 4.

As funcionalidades implementadas podem ser executadas separadamente ou em conjunto, bastando pequenas modificações em um arquivo de configuração. Os passos de configuração do *pipeline* são descritos nos itens abaixo.

Requisitos do Sistema

Os *scripts* que compõe o *pipeline* foram desenvolvidos em linguagem `Perl` e `R` e são executados através de um terminal, utilizando ambiente Unix. Além das linguagens `Perl` e `R`, o usuário do *pipeline* deve possuir disponíveis e localizados no *PATH* de seu sistema o caminho para o `segemehl`, `SAMtools`, `Cufflinks`, `Cuffmerge`, `Cuffdiff` e

o pacote `cummeRbund` instalado na linguagem `R`. As versões dos programas utilizados durante o desenvolvimento e teste do *pipeline* foram:

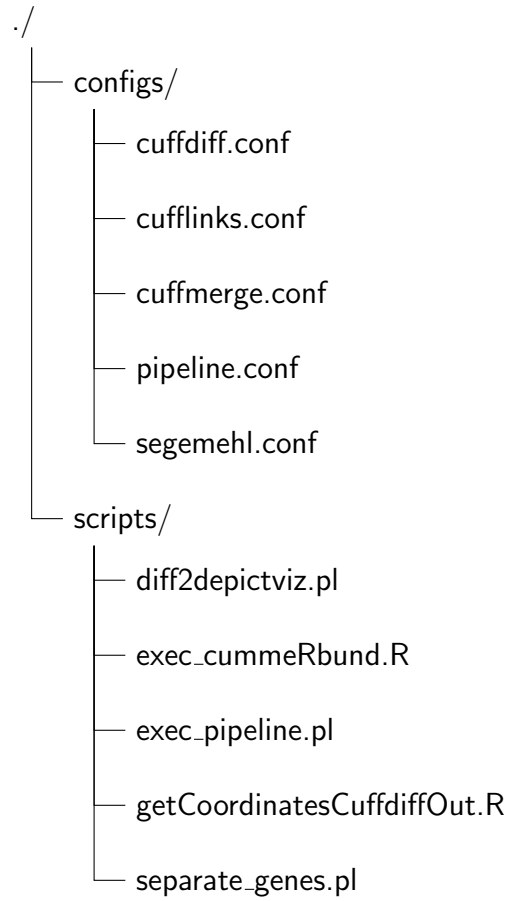
- `Linux`
- `R` (v. 3.3)
- `Perl` (v. 5.18.2)
- `segemehl` (v. 0.2.0-418)
- `SAMtools` (v. 0.1.19)
- `Cufflinks` (v. 2.2.1)
- `Cuffmerge` (v. 2.2.1)
- `Cuffdiff` (v. 2.2.1)
- `cummeRbund` (v. 2.6.1)

Instalação e Configuração

O pacote do *pipeline* para *download* encontra-se disponível na página do projeto em `http://git.facom.ufms.br/bioinfo/pipeline_segemehl_cufflinks`. Sua instalação é feita de forma simples, bastando extrair o conteúdo do pacote para uma pasta de sua preferência e fornecer permissões de leitura, escrita e execução. As permissões permitem ao *pipeline* criar e manipular diretórios e programas.

O *pipeline* é dividido em dois diretórios, chamados de `scripts/` e `configs/`. O diretório `scripts/` contém todos os *scripts* que são executados pelo usuário do *pipeline*, enquanto o diretório `configs/` contém arquivos de configurações que são lidos pelos *scripts* e possuem informações e parâmetros necessários para a execução de cada etapa do *pipeline*, ficando a cargo do usuário editá-los conforme sua necessidade.

O pacote do *pipeline* desenvolvido possui a seguinte árvore de diretórios.



A execução do *pipeline* é feita pelo *script* `exec_pipeline.pl`, porém sua configuração deve ser feita em primeiro lugar pelo arquivo `pipeline.conf`. Neste arquivo deverão ser fornecidas informações importantes como o caminho dos *reads*, genoma e anotação, local onde se encontram os arquivos de configuração dos demais programas, local onde serão escrito os resultados e quais etapas do *pipeline* serão executadas.

Os conjuntos de *reads* que serão mapeados e estarão no diretório informado pelo usuário no arquivo `pipeline.conf`, deverão ter um nome específico. O nome de cada arquivo utiliza o nome da condição, número da replicata e o tipo do read (apenas para *reads paired-end*) separados por "_".

- *reads paired-end*:
  - `condição_replicata_R1.fasta`
  - `condição_replicata_R2.fasta`

- *reads single-end*:
  - `condição_replicata.fasta`

A replicata deve ser consecutivamente ordenada de 1 a n, onde n é o número de replicatas. É importante manter os nomes dos arquivos arquivos porque esta informação é utilizada para gerar as saídas durante a execução do *pipeline*.

Após a configuração do *pipeline*, deve-se agora configurar os arquivos relativos à cada programa que serão executados, sendo eles: `segemehl.conf`, `cufflinks.conf`, `cuffmerge.conf` e `cuffdiff.conf`. Cada um desses arquivos deverão conter os parâmetros que serão passados para os programas. Para todos os arquivos de configuração, caso um parâmetro espera receber um valor, estes devem ser separados por espaço ou tabulação. Parâmetros que esperam receber caminhos (*reads*, genoma, anotação) devem omitir essas informações, visto que já se encontram no arquivo de configuração do *pipeline*. Não há necessidade de configurar os arquivos dos programas que não serão executados. Durante a execução do *script* `exec_pipeline.pl`, serão feitas as leituras dos arquivos de configurações, tanto do *pipeline* quanto dos programas, e cada etapa será executada sequencialmente.

Concluídas todas as configurações descritas até aqui, a execução do pipeline se dá pelo comando

```
$ ./exec_pipeline.pl ../configs/pipeline.conf > log.txt,
```

executando o *script* `exec_pipeline.pl` e passando o arquivo `pipeline.conf` como parâmetro. Os demais arquivos de configuração dos programas são lidos automaticamente, visto que o caminho para esses arquivos já deverão estar corretamente atribuídos no arquivo `pipeline.conf`, juntamente com o local onde deverão ser armazenados o resultado.

Após a análise de expressão diferencial feita pelo `Cuffdiff` na última etapa disponível no *pipeline*, podemos executar o `cummeRbund` para então gerarmos alguns gráficos que podem nos ajudar na visualização do resultado. Tal funcionalidade pode ser executada através do *script* `exec_cummeRbund.R`, que permite executar o `cummeRbund`, gerando gráficos [53] de MA *plot*, Densidade, Volcano, Dendrograma e

MDS. O *script* deve ser executado dentro do diretório onde se encontra o resultado do `Cuffdiff`. Sua execução é dada pelo comando

```
$ Rscript exec_cummeRbund.R Condição1 Condição2,
```

onde deve ser passado como parâmetro o nome das condições, exatamente iguais aos que foram atribuídos ao *Label* (parâmetro -L) para a execução do `Cuffdiff`.

Há ainda três outros *scripts* presentes no pacote, que tem como função adaptar a saída do *pipeline* aqui proposto para a ferramenta *web* `DEPICTViz` desenvolvida e descrita no Capítulo 4. São os *scripts* `getCoordinatesCuffdiffOut.R`, `diff2depictviz.pl` e `separate_genes.pl`.

O *script* `getCoordinatesCuffdiffOut.R` tem como objetivo extrair as informações necessárias da saída do `Cuffdiff` para gerar um *MA plot* de todos os genes presentes no experimento, utilizando para isso o `cummeRbund`. A execução do *script* se dá por

```
$ Rscript getCoordinatesCuffdiffOut.R Condição1 Condição2,
```

onde os parâmetros são, novamente, o nome das condições usadas no *Label* na execução do `Cuffdiff`. Em sua saída é gerada uma tabela em formato `csv` chamada de `coordenadas.csv` contendo as coordenadas de todos os genes genes presentes. Porém, algumas entradas dessa tabelas contém valores que não são reconhecidos pela ferramenta `DEPICTViz`, como valores de mais ou menos infinito (inf e -inf) que são atribuídos quando a contagem do gene é zero para alguma das condições, influenciando o cálculo de LogFC e contagem média. Da mesma forma, a tabela contém colunas não reconhecidas pela ferramenta e seu *header* também não se encontra no padrão correto. Logo, o *script* `diff2depictviz.pl` é utilizado para resolver tais problemas e criar uma tabela contendo as informações compatíveis e pronta para ser usada como entrada para o `DEPICTViz`. O comando para execução do *script* é:

```
$ ./diff2depictviz.pl coordenadas.csv,
```

tendo como parâmetro a tabela de saída do *script* `getCoordinatesCuffdiffOut.R` e cria uma outra tabela chamada de `in_depictviz.csv`. Como geralmente há um

conjunto muito grande de genes presente na tabela, caso o usuário queira limitar esse conjunto é possível utilizar o último *script* desenvolvido nesse pacote, chamado de `separate_genes.pl`. Com ele é possível criar um subconjunto de genes de acordo com uma lista de nomes passados por parâmetro, ou ainda de genes com *p-value* menor que dado valor. A medida *p-value* é abordada na Seção 4.1. A execução do *script* é feito pelo comando

```
$ ./separate_genes.pl -i arquivo_entrada -o arquivo_saida -de 0.05
                    gene_name1 gene_name2...,
```

onde são passados os arquivos de entrada (`in_depictviz.csv`), saída, opcionalmente um valor de corte para o *p-value* e a lista de genes que serão procurados. Por exemplo, caso o usuário queira fazer um subconjunto com todos os genes HOX presentes, basta executar o comando abaixo:

```
$ ./separate_genes.pl -i in_depictviz.csv -o subset_depictviz.csv HOX
```

Como a tabela de saída se trata de um subconjunto do arquivo `in_depictviz.csv`, os valores dos campos e seu *header* já estão corretos e podem ser utilizados sem problemas. Os padrões utilizados no arquivo de entrada da `DEPICTViz` são descritos na Seção 4.2.

Mais detalhes acerca da instalação e da utilização do pipeline podem ser vistos no Apêndice A.

Capítulo 4

DEPICTViz - Differential Expression and Protein InteraCTions Visualization Tool

Uma ferramenta desenvolvida neste trabalho e que consiste na sua principal contribuição é a ferramenta *web* `DEPICTViz`, desenvolvida para a visualização de experimentos de expressão diferencial de genes e interação de proteínas. A ferramenta surgiu da necessidade de se visualizar pequenos conjuntos de genes de interesse em experimentos de expressão diferencial de RNA-Seq. Sua principal característica é a apresentação de um gráfico MA (*MA plot*), similar aos gráficos *Smear Plot* e *MA plot*, presentes nos pacotes `EdgeR` e `DESeq2`, respectivamente. O diferencial da `DEPICTViz` é o fato de ela ser interativa.

O gráfico MA é um gráfico cartesiano onde o eixo `y` representa os níveis de expressão dos genes em Log_2FC, como descrito na Seção 3.4. O eixo `x`, por sua vez, representa a contagem média de cada gene. No gráfico, cada ponto representa um gene diferente. As informações acessíveis a partir de um gene são descritas na Seção 4.1.

A entrada da ferramenta consiste de uma tabela onde os campos são separados por tabulação, contendo todas as informações necessárias para a construção do gráfico e suas funcionalidades. Como resultado, a ferramenta apresenta uma janela no navegador *web* do usuário, podendo este hospedá-la em um servidor ou mantê-la localmente. Demais informações são descritas em detalhes na Seção 4.2.

`DEPICTViz` foi utilizada em vários experimentos, apresentados no Capítulo 5. A seguir, descrevemos as funcionalidades da ferramenta.

4.1 Funcionalidades

Pelo fato da `DEPICTViz` ser interativa, foi possível adicionar várias funcionalidades para visualizar e cruzar informações que podem ser relevantes para uma análise mais detalhada. Dentre todas as funcionalidades que serão descritas nesta seção, as mais básicas são o nome do gene, as coordenadas e a cor que cada ponto no gráfico terá, visto que não é possível construir o gráfico sem elas.

Primeiramente, as coordenadas por si só já dizem muita informação no gráfico, onde quanto maior for o valor em que um gene se encontra no eixo `x`, maior é o nível de expressão do gene; quanto maior (ou menor) o valor no eixo `y`, maior é a diferença de expressão do gene entre as condições do experimento.

Da mesma forma que as coordenadas, a cor de cada gene traz consigo uma informação muito importante, que diz respeito ao seu nível de expressão. Um gene pode ser colorido por três cores: preto, vermelho ou laranja. A cor preta representa genes que foram classificados como não diferencialmente expressos, a cor vermelha representa os que foram classificados como diferencialmente expressos e a cor laranja representa os genes cuja contagem de *reads* mapeados foi igual a zero em uma das condições.

As informações para as coordenadas e cores podem ser obtidas através da expressão diferencial de programas como `EdgeR` e `DESeq2`, além de também serem extraídas no passo de visualização do *pipeline* desenvolvido nesse trabalho. Para o eixo `y`, ambos programas e o *pipeline* utilizam níveis de expressão dado em Log_2FC. Já para o eixo `x`, a contagem utilizada pelo `EdgeR` é dada em Log_2CPM (*Counts Per Million*), onde

$$\text{CPM} = \frac{\text{número de reads mapeados}}{\text{tamanho do experimento}} \times 10^6.$$

O DESeq2 utiliza a contagem dada em *baseMean* que é a média da contagem normalizada dos *reads* mapeados entre todas as amostras. Por sua vez, o *pipeline* mostra a contagem em Log_{10}FPKM.

Para determinar a classificação e cor de um gene, seja ele diferencialmente expresso ou não, DEPICTViz utiliza o *p-value* como medida. *P-value* é a probabilidade de que o resultado observado não tenha co-relação com o que se está realmente testando, determinando a significância de seus resultados [23]. Em outras palavras, dizemos que há significância estatística quando o *p-value* é menor que o nível de significância adotado, geralmente 0.05.

Por padrão, na DEPICTViz o gene é classificado como diferencialmente expresso e recebe a cor vermelha no gráfico para valores de *p-value* menores que 0.05 . Analogamente, valores maiores ou iguais a 0.05 são considerados não diferencialmente expressos e recebem a cor preta. Um gene recebe a cor alaranjado quando o *p-value* tem o valor NA, indicando que a contagem foi igual a zero para todas as amostras de uma das condições. Tal valor de *p-value* pode ocorrer em análises de expressão diferencial realizadas por DESeq2.

DEPICTViz foi desenvolvida utilizando como base os experimentos de expressão diferencial realizados pelos *softwares* EdgeR e DESeq2, utilizando então seus resultados para a captura dos valores necessários para as coordenadas de plotagem (Log_2FC x LogCPM ou *BaseMean*), classificação e coloração (*p-value*). Entretanto, informações oriundas de outros *softwares* que exprimam o mesmo significado também são válidas.

Diferente das coordenadas de plotagem e classificação, que são fundamentais para a construção do gráfico, os demais itens a seguir consistem em funcionalidades que enriquecem a análise e visualização, porém não são obrigatórios para a construção do gráfico. Todas as funcionalidades a seguir empregam conceitos descritos no Capítulo 2, onde sua necessidade surgiu dos experimentos descritos no Capítulo 5. Detalhes de implementação e resultados são descritos na Seção 4.3.

STRING

Durante o desenvolvimento da ferramenta, a primeira funcionalidade inserida e e implementada foi ligar os genes presentes no gráfico ao STRING, cruzando informações referentes a interação de genes e proteínas.

A interação da `DEPICTViz` com o STRING ocorre de duas formas. A primeira é a busca do nome do gene no STRING, que leva o usuário a página do STRING referente ao gene que foi buscado. A segunda forma é para um dado gene selecionado, buscar no STRING a informação referente a todos os genes que possuem interação conhecida com aquele que foi selecionado. Se dentre todos os genes presentes no gráfico houver pelo menos um gene que possua interação conhecida, segundo o STRING, estes são ligados através de arestas representando a interação no gráfico.

COG

Ao depararmos com os resultados presentes no experimento descrito na Seção 5.3, percebemos que havia uma grande quantidade de informações que poderíamos de alguma forma incluir na ferramenta para enriquecer a análise e visualização.

A primeira foi o uso do COG. Há duas funcionalidades relacionadas ao COG que foram desenvolvidas. A primeira é o uso de identificadores COG, que possibilita levar o usuário às páginas do NCBI e EggNOG para consulta de função e informações mais detalhadas. A segunda funcionalidade utiliza a informação de categoria dos COGs, onde para um gene selecionado que possua uma categoria COG atribuída, busca-se dentre todos os genes quais possuem exatamente a mesma função ou possuam pelo menos uma função em comum, e os liga através de arestas. Além dos identificadores, há um campo para se depositar a descrição da função de cada COG.

KEGG

Da mesma forma ao COG, é possível mostrar informações relacionadas ao KEGG *Orthology* (KO) e identificadores de genes no KEGG. Seu uso se dá redirecionando o usuário para as páginas do KEGG onde então são mostradas mais informações de cada entrada.

Swiss-Prot

Entre os resultados obtidos, uma das informações mais pertinentes são os identificadores do Swiss-Prot, devido ao fato de concentrarem informações referentes a outras fontes como COG e GO. Dentre as ações possíveis na `DEPICTViz` está a busca por proteínas no STRING, UniProt, OMA Browser e SWISS-MODEL. Através deste último é possível ver a estrutura 3D da proteína, caso seja conhecida.

GO

Assim como o COG, foram adicionadas identificadores GOs e suas descrições para representar genes e seus produtos, aumentando sua anotação. Para cada uma das três ontologias que compõe o GO, pode haver uma lista de identificadores e descrições para cada gene no gráfico. Através dos identificadores é possível aceder ao NCBI, AmiGO, EggNOG e UniProt.

EC Number

Uma das últimas funcionalidades implementadas até então nesse trabalho foi a inclusão de EC *numbers* na ferramenta. Cada gene no gráfico pode possuir uma lista de EC *numbers* associados que representam reações catalisadas por suas enzimas. Além da ferramenta permitir redirecionar o usuário para bases de dados específicas de enzimas como ExPASy, ExplorEnz, BRENDA e MetaCyc, há a opção do usuário acessar a seção de enzimas do KEGG e seu *pathway* biológico.

Genome Browser

A última funcionalidade implementada é o acesso ao UCSC *Genome Browser*. Essa funcionalidade permite visualizar o gene escolhido em uma *track* no *genome browser*, onde a partir de então é possível adicionar ou cruzar mais informações dentre a gama de opções oferecidas pelo *genome browser*, criando assim uma *custom track* que pode ser salva para análise posterior ou compartilhada.

4.2 Requisitos

Esta seção tem como objetivo descrever os passos, configurações e requisitos necessários para a instalação correta da ferramenta.

Requisitos do Sistema

`DEPICTViz` é uma ferramenta *web* que foi desenvolvida para ser capaz de ser executada em navegadores *web* modernos e utilizando um *web server* em qualquer sistema operacional, embora tenha sido especialmente desenvolvida para executar em plataformas Linux. `DEPICTViz` foi testada com sucesso nas plataformas a seguir, embora deva funcionar normalmente para outras plataformas não descritas aqui:

- Debian 8.2
- Ubuntu 14.04.3 LTS
- Linux Mint 17.2+

Os navegadores *web* Google Chrome (v. 47.0.2526.80) e Firefox (v. 42.0), juntamente com o *web server* Apache2 (v. 2.4.7), foram testados na ferramenta.

Instalação

Atualmente, o *download* da `DEPICTViz` pode ser feito através da página do projeto em `http://projetos.facom.ufms.br/bioinfo/depictviz`. A instalação da ferramenta é simples e rápida, não exigindo grande conhecimento por parte do usuário.

Após completar o *download* da ferramenta e descompactá-la, basta o usuário mover a pasta extraída para um local dentro da pasta raiz de seu servidor. Se o usuário usa o Apache2 no Linux, por exemplo, comumente o diretório raiz é `/var/www/html/`, logo basta mover a pasta para este diretório ou subdiretório dentro do mesmo.

Arquivos de Entrada e Configuração

Após a instalação da ferramenta, o próximo passo é fornecer o arquivo de entrada para a ferramenta. O arquivo de entrada deve estar presente na pasta `input/` no diretório corrente onde a `DEPICTViz` se encontra. Esse arquivo de entrada consiste de uma tabela onde os campos são separados por tabulação (*tab-separated value*) e com um conjunto de *headers* bem definidos identificando os campos presentes. A Tabela 4.1 mostra os campos que podem constar no arquivo de entrada da ferramenta, onde o *header* de cada campo pode variar entre letras maiúsculas e minúsculas (*no case-sensitive*) e a obrigatoriedade de cada um para a construção do gráfico.

Tabela 4.1: Descrição dos campos que devem ser preenchidos no arquivo de entrada, além de sua obrigatoriedade para a construção do gráfico.

Campo	**Descrição**	**Obrigatório**
ID	Identificação de um gene (ID ou nome).	Sim
X	Valor da coordenada do eixo x. Representa a contagem média do gene no experimento.	Sim
Y	Valor da coordenada do eixo y. Representa o Log_2FC entre as condições.	Sim
Pvalue	Valor do *p-value*. Genes com *p-value* < 0.05 são classificados como dif. expressos.	Sim
COG_ID	Identificador COG para um gene.	Não
COG_Description	Descrição da função do COG.	Não
COG_Category	Categoria que cada COG pertence.	Não
Swiss_ID	Identificador Swiss-Prot associado ao gene.	Não
KO_ID	Identificador KEGG *Orthology*.	Não
KEGG_Gene	Identificador do gene no KEGG. Na ausência deste é então usado o campo ID.	Não
EC	EC *number* associado ao gene.	Não
GO_Process	Identificador GO para a ontologia processo biológico.	Não
GO_P_Description	Descrição da função desempenhada pelo GO da ontologia processo biológico.	Não
GO_Function	Identificador GO para a ontologia função molecular.	Não
GO_F_Description	Descrição da função desempenhada pelo GO da ontologia função molecular.	Não
GO_Component	Identificador GO para a ontologia componente celular.	Não
GO_C_Description	Descrição da função desempenhada pelo GO da ontologia componente celular.	Não
Start	*Loci* da posição inicial conhecida no genoma.	Não
End	*Loci* da posição final conhecida no genoma.	Não
Strand	Orientação da fita.	Não
Gene_Product	Produto do gene.	Não
Contig	Contig do qual o gene pertence.	Não

A ordem dos campos não interfere no funcionamento da ferramenta. Caso um campo considerado obrigatório não esteja presente, a ferramenta não irá construir o gráfico e um erro será mostrado indicando o campo que está faltando. Para representar a ausência de informação quando um campo é usado, deve-se usar '--' em vez de deixá-lo vazio.

Os campos de GOs e descrições, identificadores COG e descrições, EC *numbers*, Swiss-Prot, KO e KEGG gene aceitam uma lista de valores para cada entrada, onde estes devem ser separados obrigatoriamente por vírgula. Após a análise dos resultados obtidos no Capitulo 5, percebeu-se que os identificadores GOs não possuíam padrões nos experimentos. Logo, para o 'GO:0007018' por exemplo, as informações podiam estar em três formas distintas: GO:0007018, 0007018 e 7018. Portanto, todas as três formas descritas são aceitas pela ferramenta a fim de evitar problemas de entrada. A Tabela 4.2 mostra um exemplo de uma entrada válida. Outros exemplos de entrada prontos estão disponíveis dentro do diretório `input/` da ferramenta para consulta.

Com o arquivo de entrada pronto e localizado no diretório correto, o próximo passo a ser tomado é editar um rápido arquivo de configuração para terminar a configuração e deixar a ferramenta pronta para a execução.

O arquivo de configuração, chamado de `init.js`, está localizado dentro do diretório `data/js/` na pasta corrente da ferramenta. Neste arquivo pode ser editado informações como: nome do arquivo de entrada que foi depositado no diretório `input/`, limiar do *p-value* utilizado para classificar genes diferencialmente expressos (0.05 por padrão), nome das condições que representam os valores positivos e negativos do eixo y, subtítulo do gráfico e sua descrição.

Tabela 4.2: Exemplo de arquivo de entrada válida para a ferramenta.

ID	X	Y	Pvalue	COG_ID	COG_Description	Swiss_ID	KO_ID	KEGG_Gene	EC	GO_Process	GO_P_Description
Gene1	1.46674	0.95696	0.02270435	COG5262	Histone H2A	--	K01869	vvi:100248927	6.1.1.4	GO:0006412	translation
Gene2	6.49216	6.92388	1.3301E-077	--	--	P05533	--	--	--	GO:0008152	metabolic process
Gene3	1.65539	1.15335	0.00880259	COG5023	Tubulin	O22380	K07375	osa:4335091	1.1.1.1, 1.4.4.2	GO:0042254	ribosome biogenesis
Gene4	2.86755	1.07487	1.06440E-033	COG2730	Endoglucanase	Q84ZL0	K12821	gmx:547756	4.1.1.31	GO:0001510, GO:0006412	RNA methylation, translation
Gene5	2.52314	1.56209	0.4587269	COG0282	Acetate kinase	Q9SJT7	--	--	--	--	--
Gene6	0.26988	3.78942	0.0689871	COG3869	Arginine kinase	--	--	--	--	--	--
Gene7	4.63244	0.98745	1.0258E-09	COG5048	FOG: Zn-finger	Q2V3L3	K09191	bdi:100838357	3.6.3.14	GO:0008152, GO:0006810, GO:0009060	metabolic process, transport, aerobic respiration
Gene8	1.36987	3.22698	9.475E-33	COG2224	Isocitrate lyase	P49296	K01637	ota:OstapMp36	6.3.1.2	GO:0006351	transcription

4.3 Representação da saída

Concluída a instalação e configuração da ferramenta, já podemos então executá-la através do navegador *web*. Para executá-la, basta o usuário acessar o caminho escolhido, abordado na Seção 4.2, partindo da raiz do servidor. Primeiramente, caso exista algum erro referente ao arquivo de entrada, seja por falta de um campo obrigatório para a construção do gráfico, pela existência de um campo não definido, campos duplicados, campo com *header* incorreto ou até mesmo por incoerência dos valores (*strings* onde deveriam haver apenas números), uma mensagem de erro será exibida para o usuário (Figura 4.1) informando o tipo de erro. Após a correção, basta atualizar novamente a página. Caso o erro persista, tente limpar o *cache* de seu navegador.

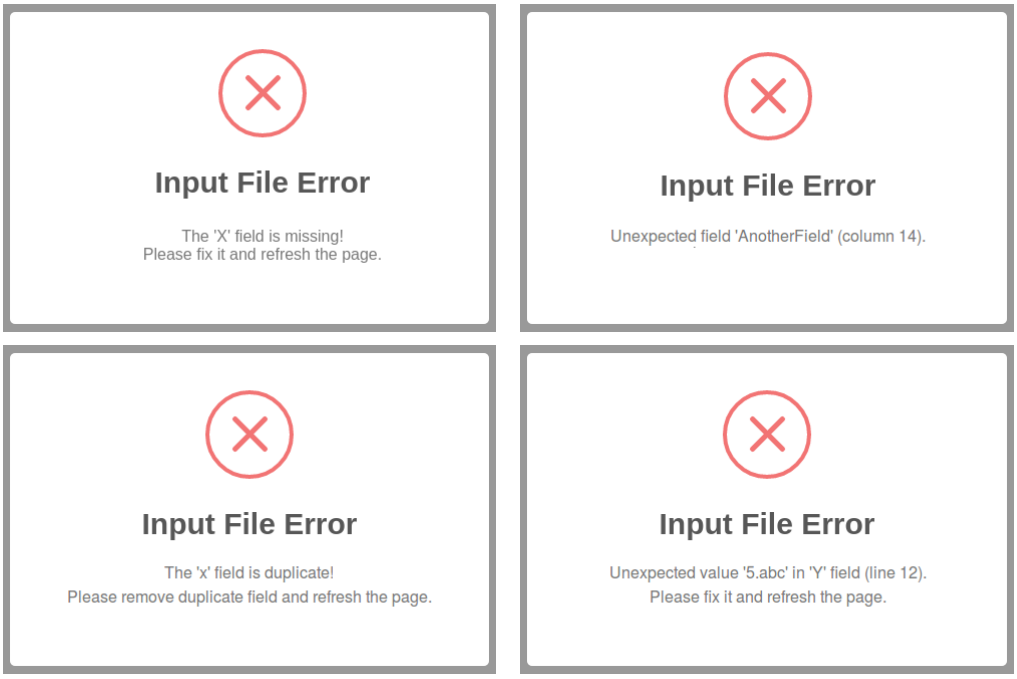


Figura 4.1: Diferentes mensagens de erros que podem ser disparadas devido a erros no arquivo de entrada.

Validando o arquivo de entrada com sucesso, o usuário recebe em uma janela a opção de selecionar o organismo que melhor representa os genes presentes no arquivo de entrada (Figura 4.2). Dentre os experimentos que foram realizados, apenas o genoma do *Homo sapiens* e *Mus musculus* foram usados. Portanto, para organismos diferen-

tes há uma terceira opção chamda de '*Other*'. A seleção de organismos serve para a ferramenta fazer a correta relação entre os dados e as buscas e redirecionamentos para outros bancos, permitindo assim uma busca mais precisa.

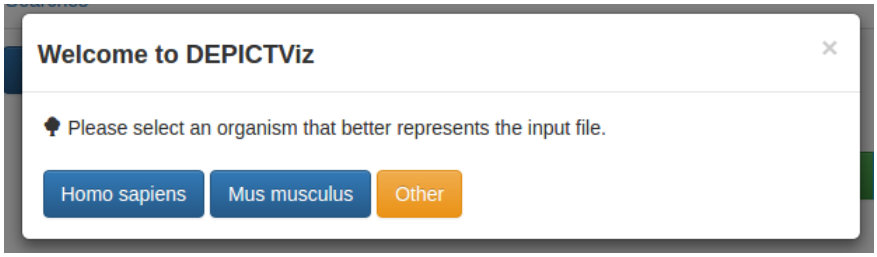


Figura 4.2: Janela de seleção de organismos.

Tendo-se selecionado o organismo que melhor representa o conjunto de dados de entrada, a ferramenta é completamente carregada. Caso o usuário queira selecionar novamente o organismo, o botão '*Reselect organism*' abaixo do gráfico foi adicionado para tal propósito. Podemos separar a ferramenta em três áreas: busca, gráfico e descrição de ajuda. A Figura 4.3 mostra o gráfico e as outras áreas.

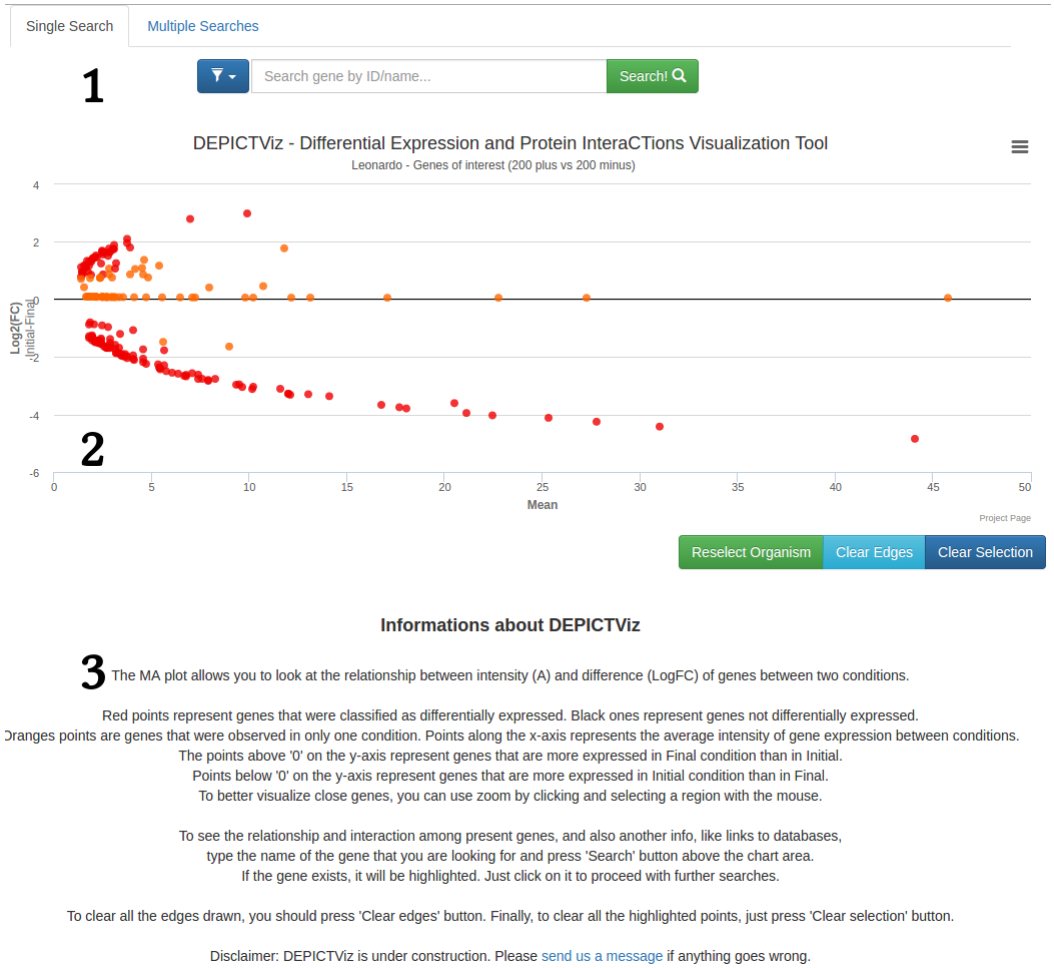


Figura 4.3: Ferramenta completamente carregada com informações referente ao experimento descrito na Seção 5.3. É composta pelas áreas de busca (1), gráfico (2) e descrição (3).

Sistema de busca

A área de busca se encontra acima do gráfico, composta por um filtro seletor, um campo de texto para a busca e um botão para executar a busca. Ao clicar sobre o filtro, um seletor é exibido dando ao usuário a possibilidade de escolher sobre qual campo deseja realizar a busca (Figura 4.4). Só é possível selecionar um campo que esteja presente no arquivo de entrada. Por padrão a busca é feita pelo nome ou ID do gene.

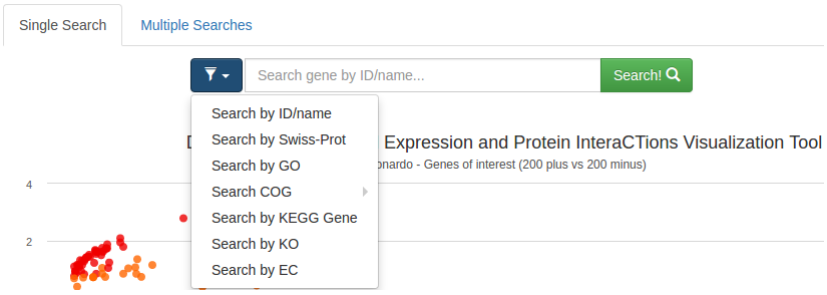


Figura 4.4: Seletor de filtro permite escolher o campo em que se deseja realizar a busca.

A *string* de busca deve ser inserida no campo texto que está localizado ao lado do filtro. À medida que a *string* é digitada, são exibidos sugestões de busca (*autocomplete*) para o usuário (Figura 4.5).

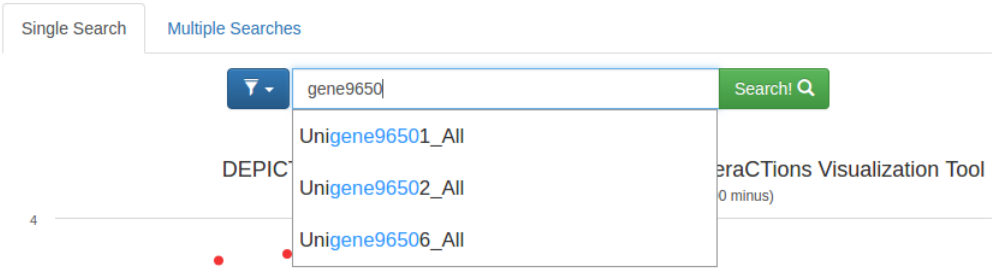


Figura 4.5: Campo de texto para inserir a *string* de busca. O campo de texto possui *Autocomplete* que possibilita mostrar sugestões para o usuário.

Ao executar a busca, o gene que possuir a informação buscada será selecionado na área do gráfico abaixo. Caso contrário, uma mensagem é exibida no canto superior direito da tela informando que não foram encontrados resultados.

Assim como a busca unitária (*Single Search*), há ainda a possibilidade de se fazer buscas múltiplas (*Multiple Searches*). O tipo de busca pode ser alternado através da seleção das abas acima do botão de seleção de filtro. A busca múltipla funciona de uma maneira ligeiramente diferente da busca unitária. Ao inserir as *strings* de busca, o usuário deve clicar em alguma sugestão dentre as que lhe foram dadas ou pressionar a tecla *Enter* para buscar a *string* ou *substring* desejada. À medida que as *strings* são buscadas no gráfico e selecionados os genes onde estas se encontram, também são armazenadas em um campo de *tags* abaixo do campo de texto. Dessa forma, é possível remover uma ou mais *strings* sem que haja a necessidade de refazer

todas as buscas, bastando clicar no ícone "fechar" de cada *tag* (x). A Figura 4.6 mostra a seleção das *strings* de busca e seu armazenamento em *tags*.

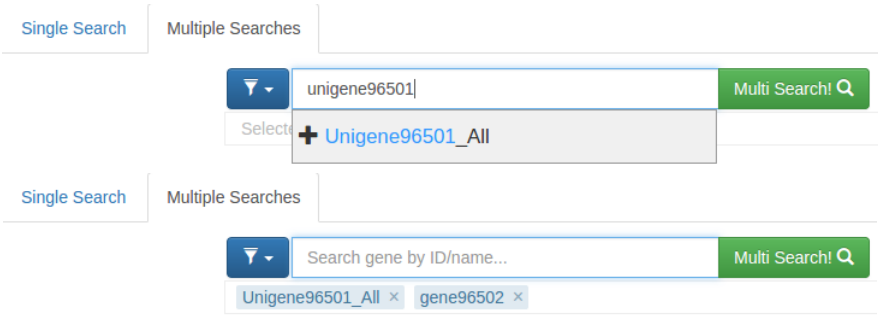


Figura 4.6: Seleção de *strings* de busca durante a busca múltipla. As *strings* são armazenadas em um campo de *tags* abaixo do campo de texto.

O botão '*Multi Search*' visto na Figura 4.6 possui uma ação diferente do seu correspondente da busca unitária. Quando clicado, é exibido uma janela para o usuário que oferece a opção de redirecioná-lo para o STRING, onde são mostrados as interações de genes e proteínas pertencentes aos pontos que foram selecionados na busca múltipla (Figura 4.7). Essa ação é válida somente para buscas de nomes/ID de genes e pelo Swiss-Prot. Portanto, é necessário buscar por estes através do filtro.

Figura 4.7: Janela aberta após o usuário fazer buscas múltiplas de identificadores Swiss-Prot e clicar no botão '*Multi Search*'. A ação ao clicar no botão '*Multi Search STRING*' da nova janela o leva ao STRING, a fim de mostrar a interação de genes e proteínas relacionados aos pontos que foram selecionados na busca.

Gráfico

A área mais importante da ferramenta é a área do gráfico, devido ao fato desta mostrar todas as informações possíveis. Pelo fato do gráfico ser interativo, podemos interagir com qualquer ponto presente nele. Ao realizar uma busca que procura um

ponto ou clicando sobre o mesmo, o ponto é selecionado e permanece destacado dos demais. Para limpar essa seleção, o usuário pode recorrer ao botão '*Clear Selection*' logo abaixo do gráfico.

Ao passar o *mouse* sobre um ponto, pode-se visualizar uma caixa de legenda que mostra de maneira rápida as informações presentes naquele ponto (Figura 4.8).

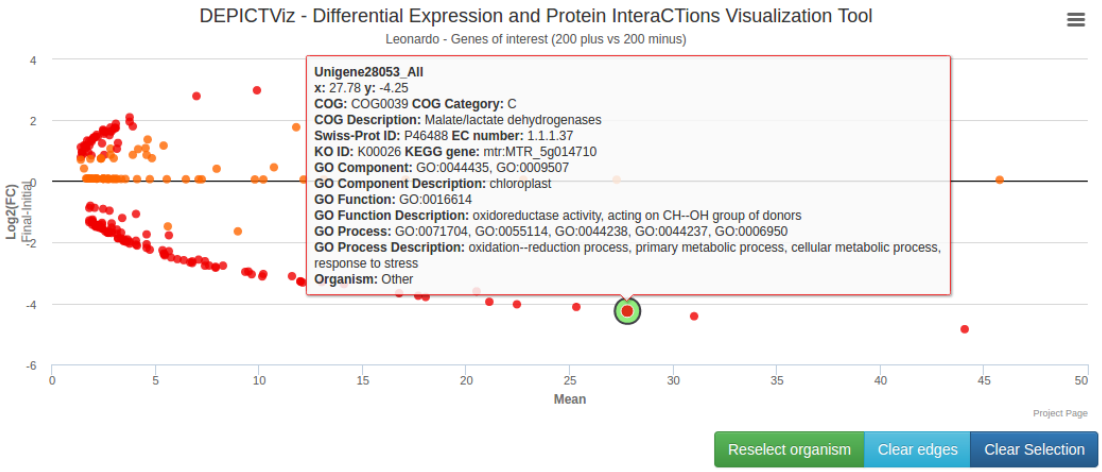


Figura 4.8: Legenda que mostra as informações de um ponto ao passar o *mouse* sobre ele. Figura retirada do experimento da Seção 5.3.

Caso o o número de pontos no gráfico seja grande ou eles estejam muitos próximos entre si, é possível aplicar *zoom in* em uma área desejada. Para isso, o usuário deve, dentro da área do gráfico, clicar com o *mouse* e arrastá-lo para criar a área de seleção desejada. Através desses passos, o *zoom in* será aplicado na área selecionada. Para voltar ao *zoom* original, há um botão no canto superior direito da área do gráfico chamado '*Reset zoom*' (Figura 4.9).

Ao clicar em um ponto no gráfico, uma janela é aberta mostrando todas as informações presentes na legenda, além de botões para executar as ações descritas na Seção 4.1 (Figura 4.10). A disponibilidade de cada opção depende exclusivamente do arquivo de entrada fornecido. A primeira opção presente, 'COG *Search*', tem como objetivo redirecionar o usuário para as páginas do NCBI e EggNOG utilizando um identificador COG, caso este esteja presente no gene selecionado.

Figura 4.9: Aplicação de *zoom in* (1) pela seleção de área com o *mouse* e *zoom out* (2) ao clicar no botão '*Reset zoom*' no gráfico. Figura retirada do experimento da Seção 5.3.

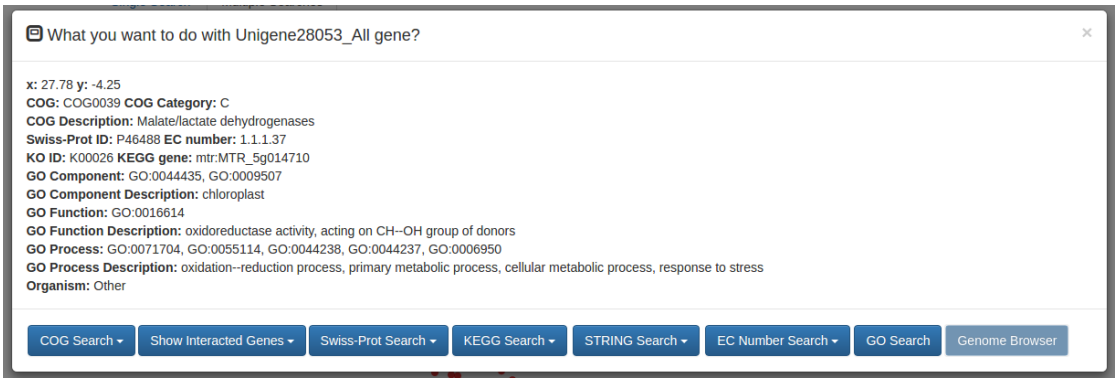


Figura 4.10: Janela aberta ao clicar em um ponto na área do gráfico. Através dessa janela é possível realizar diversas ações com os dados presentes. Figura retirada do experimento da Seção 5.3.

A segunda opção, '*Show Interacted Genes*', mostra as interações do gene selecionado com os demais presentes. Os genes que possuem interação são ligados com arestas entre si. Há dois tipos possíveis de interações que foram adicionadas até o momento da escrita deste trabalho: Interação entre genes segundo informações enviadas pelo STRING e genes que possuem a mesma categoria de COG. A interação entre genes segundo o STRING é mostrada através de arestas de cor azul entre os pontos (Figura 4.11). Já a interação de genes que compartilham as mesmas categorias COG são feitas por arestas de cor roxa, para genes que possuem exatamente o mesmo conjunto de categorias COG, e arestas de cor verde para genes que possuem pelo menos uma categoria em comum (Figura 4.12). Para limpar as arestas que foram adicionadas, o usuário pode usar o botão '*Clear edges*' abaixo do gráfico.

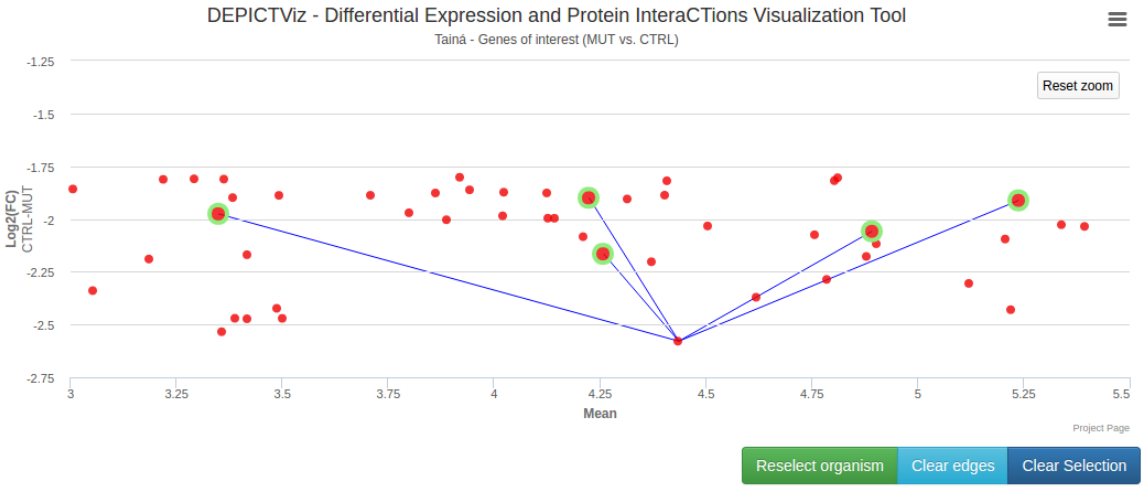


Figura 4.11: Arestas em cor azul que mostram a interação do gene selecionado com os demais, segundo o STRING.

A opção '*Swiss-Prot Search*' permite fazer uma busca nas bases de dados online de proteínas: UniProt, OMA Browser e SWISS-MODEL; utilizando identificadores Swiss-Prot que o gene possuir. Da mesma forma, '*KEGG Search*' e '*STRING Search*' possibilitam uma busca, respectivamente, no KEGG, através de identificadores KO e KEGG Gene ou gene ID, e STRING, por identificadores Swiss-Prot e gene ID. Como é possível haver mais de um identificador Swiss-Prot por entrada, ao usuário clicar em um gene que possua uma lista de identificadores, é apresentado uma janela onde o usuário escolha qual identificador gostaria de selecionar para fazer a busca (Figura 4.13).

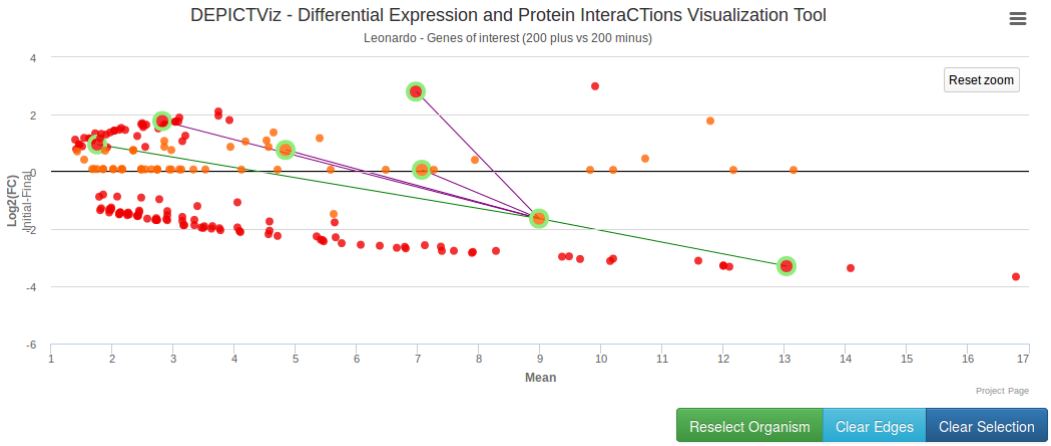


Figura 4.12: Arestas em cor roxa que mostram os genes que possuem exatamente a mesma categoria COG do gene selecionado, e arestas em cor verde para genes que compartilham pelo menos uma categoria COG com o gene selecionado. Figura retirada do experimento da Seção 5.3.

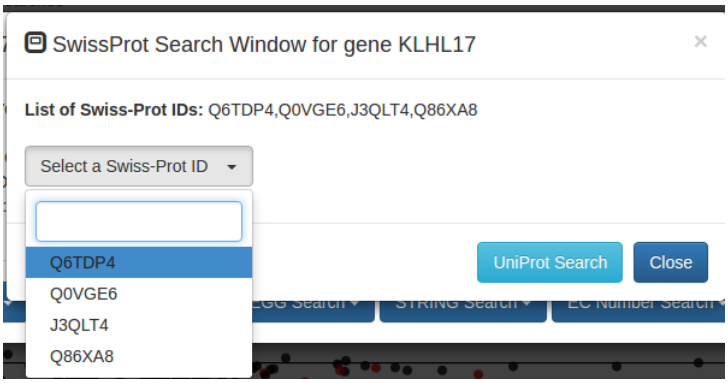


Figura 4.13: Janela aberta para a busca de um Swiss-Prot na base de dados UniProt. Figura retirada do experimento da Seção 5.2.

Uma das últimas opções que foram desenvolvidas e a que traz consigo mais funcionalidades é a '*EC Number Search*'. Através dela é possível fazer buscas em diversas base de dados especializadas em enzimas, sendo eles: ExPASy, ExplorEnz, BRENDA Enzymes, MetaCyc e KEGG. A Figura 4.14 mostra a janela que é aberta para a escolha de um EC *number* dentre a lista de *EC numbers* que cada gene pode possuir. Ao buscar um EC *number* no KEGG, é possível também selecionar um *pathway* biológico para este, caso exista (Figura 4.15).

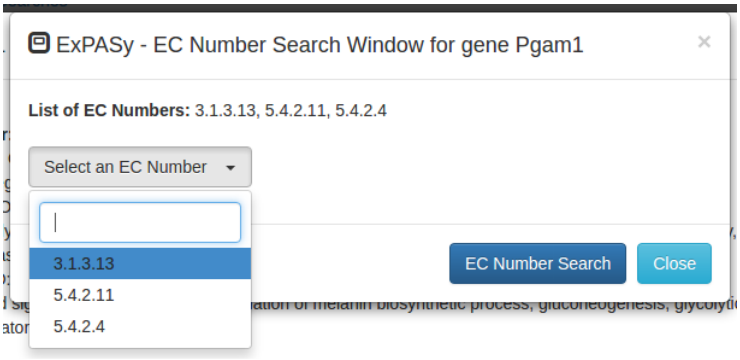


Figura 4.14: Janela aberta para a busca de um EC *number* na base de dados do ExPASy.

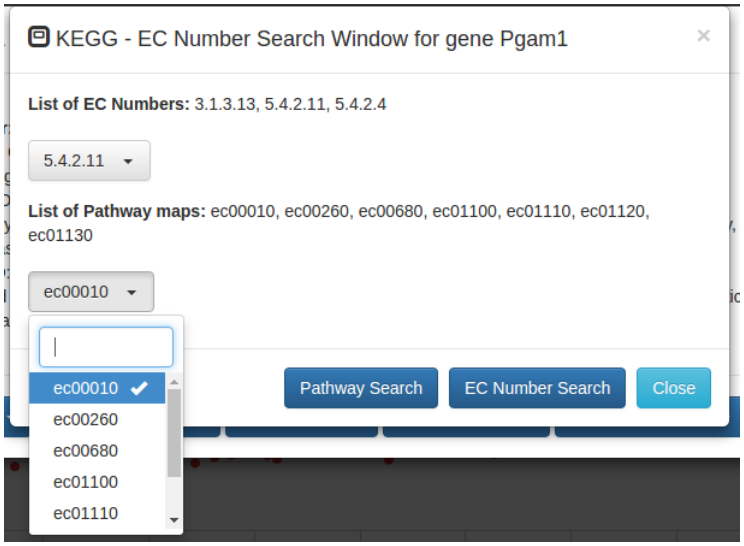


Figura 4.15: Janela aberta para a busca de um EC *number* na base de dados do KEGG. Além da busca de informações de enzimas, o usuário pode fazer uma busca de *pathways* biológicos que estão relacionados ao EC *number* selecionado.

Similar ao '*EC Number Search*', o '*GO Search*' possibilita buscar por um GO nas bases de dados: NCBI, AmiGO, EggNOG e UniProt. Da mesma forma ao busca por um EC *number*, uma nova janela é aberta onde o usuário escolhe um GO dentre a lista de GOs que pode haver atribuída ao gene (Figura 4.16).

Por fim, a última opção que se tem disponível é a '*Genome Browser*'. Para ela estar disponível basta o usuário ter selecionado um organismo conhecido (*Homo sapiens* ou *Mus musculus*). Através dela o usuário é redirecionado para o UCSC *Genome Browser*, onde este pode analisar o gene com um maior nível de detalhes e criar sua própria *custom track*.

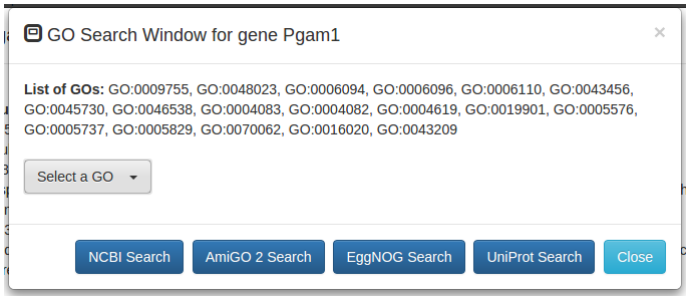


Figura 4.16: Janela aberta para buscar por um GO. Após a escolha de um dos GOs listados, o usuário fica a cargo de escolher em qual base de dados deseja fazer a busca.

Descrição

A última área da ferramenta compõe a descrição ou informação que pode ser editada pelo usuário. O texto pode ser alterado através do arquivo de configuração já mencionado anteriormente. Por padrão, o texto atribuído a descrição descreve brevemente o gráfico em geral (coordenadas, coloração dos pontos, etc).

4.4 Implementação

O desenvolvimento da ferramenta foi totalmente realizado utilizando a linguagem `JavaScript`, `HTML` e diversas APIs que proveem as funções realizadas pela mesma. O pacote da ferramenta já possui todas as APIs e dependências integradas para a sua correta execução. As APIs e suas versões (entre parenteses) utilizadas no desenvolvimento da ferramenta foram:

- `Highcharts` [6] (v. 4.1.9)
- `Bootstrap` [4] (v. 3.3.5)
- `Bootstrap-select` [5] (v. 1.6.4)
- `jQuery` [7] (v. 2.1.4)
- `jQuery-Autocomplete` [8] (v. 1.2.24)
- `jQuery-tagEditor` [10] (v. 1.0.19)

- `jQuery Caret` [9] (v. 1.3.3)
- `Nprogress` [11] (v. 0.2.0)
- `Toastr` [13] (v. 2.1.2)
- `SweetAlert` [12] (v. 1.0.0)

Como a linguagem `JavaScript` é executada no navegador *web* do usuário, todo o processo se torna transparente para quem está o executando, não havendo a necessidade de instalar ou executar algum outro *software* ou dependência. O `jQuery` é uma biblioteca para `JavaScript` que foi utilizada para facilitar o desenvolvimento e melhor manipulação das APIs listadas acima.

A construção do gráfico foi realizado utilizado a API `Highcharts`. É através dela que os dados lidos do arquivo de entrada são convertidos em pontos, são coloridos e plotados no gráfico final. Dessa forma, todas as informações relativas a cada ponto ficam armazenadas em uma estrutura de dados bem definida pela API, tornando o acesso, manipulação e interação de cada ponto muito fácil.

Os dados do arquivo de entrada são lidos linha a linha, onde para cada linha é feita uma verificação para localizar possíveis erros. Se um erro é identificado, o *script* aborta a execução e exibe uma mensagem para o usuário informando o tipo de erro através da API `SweetAlert`, como já mostrado na Figura 4.1. Do contrário, a execução continua e cada linha do arquivo de entrada que corresponde a um gene se transforma em um ponto no gráfico. Caso existam duas entradas com o mesmo ID e coordenadas, a ferramenta irá unir as informações de ambas as entradas em um mesmo ponto, respeitando o valor do *p-value* da primeira ocorrência.

O `Bootstrap` é um *framework open-source* para a criação de *websites* e aplicações *web*. Ele contém inúmeros *templates* e *design* para botões, formulários, navegações e outros componentes, sendo muito utilizado para desenvolver *websites* dinâmicos como neste trabalho. Para a `DEPICTViz`, foi utilizado na criação de todos os botões e janelas desenvolvidos. Para campos de seleção, como os de seleção de EC *number* e GO mostrados nas Figuras 4.14 e 4.16, foi utilizado o `Bootstrap-select` que modifica o seletor presente no `Bootstrap` e adiciona novas funcionalidades.

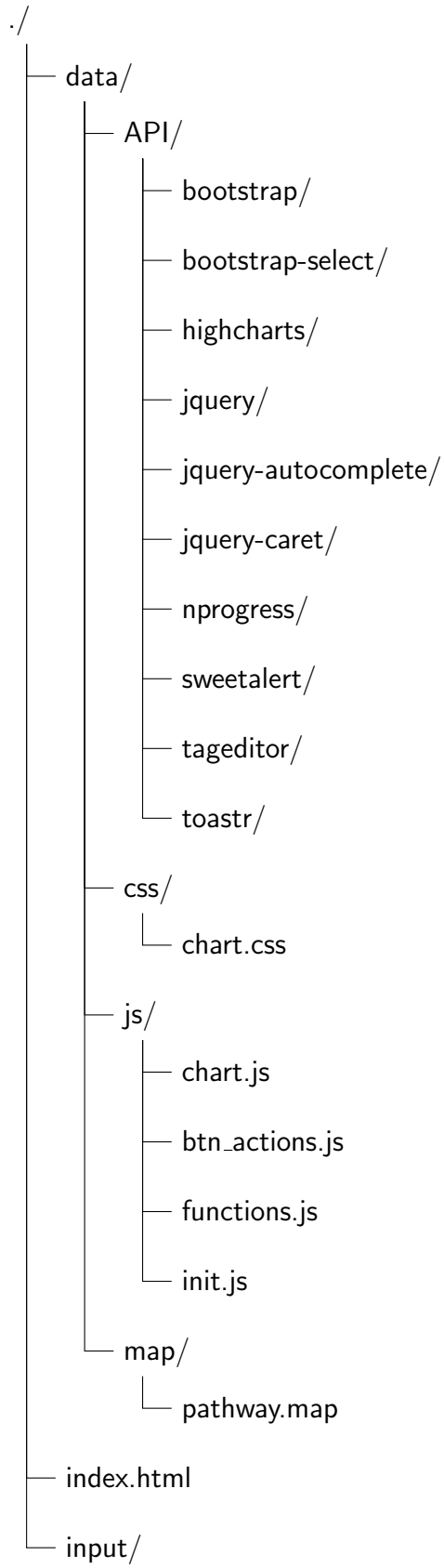
Quando o usuário realiza uma busca, à medida que a *string* vai sendo inserida no campo de texto, sugestões são oferecidas para o usuário. Essas sugestões são

informações na qual a *string* que está sendo digitada é *substring*. Para implementar essa funcionalidade foram utilizados as APIs `jQuery-Autocomplete` e `jQuery Caret`, permitindo ainda um alto grau de customização como a adição de caracteres e símbolos junto às sugestões (símbolo de adição para sugestões de busca múltipla). As *strings* que são inseridas durante a busca múltipla são armazenadas em um campo de *tags*. Este campo foi adicionado através da API `jQuery-tagEditor`, permitindo a adição, remoção e alteração de *tags* durante a execução da busca múltipla.

Durante a navegação do usuário pela ferramenta, informações ou avisos podem ser emitidos informando sobre algum evento ou má operação. Essas mensagens são exibidas em uma caixa dinâmica que surge no campo superior direito da tela, através do uso do `Toastr`. A partir desse *feedback* o usuário poderá melhor interagir com a ferramenta. A última API utilizada foi `Nprogress`. Esta última cria uma pequena barra de progresso localizada acima das abas de busca, mostrada durante ações que exigem um maior tempo de resposta, como a requisição feita junto ao STRING para identificar interações entre genes no gráfico.

O ambiente no qual a ferramenta foi desenvolvida utilizou em um sistema operacional Linux Mint 17.2, editor ATOM [2] e servidor Apache2 local. Após cada atualização no desenvolvimento, o projeto é atualizado na página do GitLab do projeto.

O pacote da ferramenta encontrado na página do projeto possui a seguinte árvore de diretórios abaixo.


./
data/
API/
bootstrap/
bootstrap-select/
highcharts/
jquery/
jquery-autocomplete/
jquery-caret/
nprogress/
sweetalert/
tageditor/
toastr/
css/
chart.css
js/
chart.js
btn_actions.js
functions.js
init.js
map/
pathway.map
index.html
input/

O arquivo `index.html` contém toda a estrutura `HTML` utilizada na ferramenta, incluindo a janela do gráfico, modals e botões. Ao executar a ferramenta no servidor, é o primeiro arquivo a ser buscado e executado pelo servidor. Todas as APIs utilizadas se encontram dentro do diretório `data/API/` e não devem ser modificadas.

O maior esforço para o desenvolvimento dessa ferramenta, assim como o `index.html`, está localizado nos diretórios `data/css/` e `data/js/`. O diretório `data/css/` contém um único arquivo chamado `chart.css` que aplica estilos e define a apresentação de botões e modals definidos em `index.html`. O diretório `data/js/` contém todos os arquivos `JavaScript` que foram desenvolvidos, incluindo o arquivo de configuração da ferramenta. O arquivo `chart.js` inicializa o gráfico e faz a leitura do arquivo de entrada, transformando as informações em pontos no gráfico. O arquivo `btn_-actions.js` contém todos os eventos e ações que são realizados pelos botões da ferramenta; e `functions.js` contém todas as funções auxiliares que são utilizadas pelos outros dois arquivos. O arquivo `init.js` corresponde ao arquivo de configuração que já foi abordado anteriormente na Seção 4.2.

Por fim, o diretório `data/map/` possui o arquivo `pathway.map` que contém uma lista mapeando EC *numbers* com seus respectivos *pathways*. Essa lista foi extraída diretamente do KEGG e pode ser encontrada em `http://rest.kegg.jp/link/pathway/ec`. Através dessa lista foi possível verificar se existe *pathway* disponível para um *EC number* escolhido pelo usuário e levá-lo ao KEGG, como mostrado na Figura 4.15.

Instruções de como instalar e utilizar a DEPICTViz estão descritos no Apêndice B.

Capítulo 5

Experimentos

Neste capítulo são descritos os experimentos realizados com as ferramentas propostas neste trabalho, a partir de vários estudos de caso contendo diferentes tipos de dados. Dos quatro estudos de caso apresentados aqui, apenas os dois primeiros, relacionado aos chamados genes HOX, mostram a utilização do pipeline para análise de expressão diferencial, descrito no Capítulo 3. Os demais estudos de caso apresentam diferentes usos da ferramenta `DEPICTViz`, descrita no Capítulo 4.

5.1 Genes HOX e câncer

Os genes HOX formam a primeira família de genes envolvidos no desenvolvimento, que foram reconhecidos como sendo compartilhados entre vários animais diferentes [51]. Esses genes pertencem à família dos genes homeobox, que são um segmento de genes reguladores, que partilham 120 pares de bases de sequência de DNA em comum, codificando proteínas (chamadas homeodominio) de 61 aminoácidos que podem se ligar ao DNA e interagir com genes alvos, promovendo sua ativação ou repressão.

Essas proteínas agem como fatores de transcrição durante o desenvolvimento embrionário e tem como principal papel a formação do eixo ântero-posterior do embrião, ajudando no estabelecimento da forma básica do corpo. Pequenas alterações em ge-

nes controladores tão poderosos, ou alterações nos genes que são ligados por eles, podem representar uma enorme fonte de mudanças evolutivas [22].

No genoma humano existe um total de 39 genes HOX, que são organizados em *clusters* (entre 9 e 11 genes cada) em uma ordem específica, associada ao local e ordem onde são expressos. São localizados em quatro cromossomos e *clusters* diferentes:

- Cromossomo 2 – *Cluster* HOX D,
- Cromossomo 7 – *Cluster* HOX A,
- Cromossomo 12 – *Cluster* HOX C e
- Cromossomo 17 — *Cluster* HOX B.

A Figura 5.1 ilustra os *clusters* HOX encontrados na mosca da fruta (*Drosophila melanogaster*) e no humano.

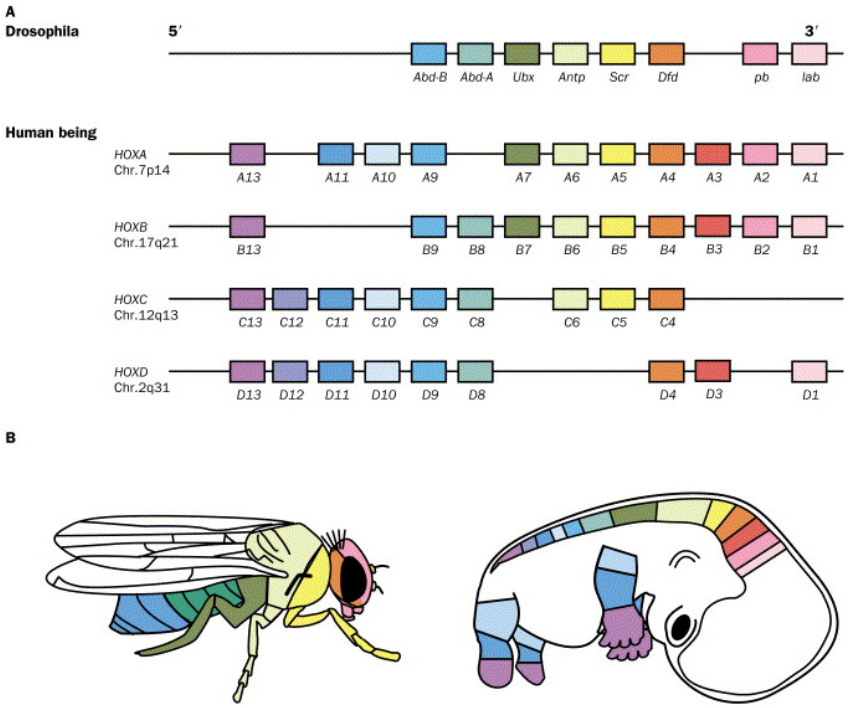


Figura 5.1: Distribuição dos genes HOX: (A) A mosca possui apenas 8 genes HOX, localizados em um único *cluster*, enquanto no humano (B) possui 39 genes HOX, separados em 4 *clusters* e cada *cluster* localizado em um cromossomo distinto [54].

Pesquisas atuais têm mostrado que genes HOX estão envolvidos no desenvolvimento de vários tipos de câncer [22]. Esses resultados têm mostrado que alguns genes HOX apresentam um maior nível de expressão (*up-regulated*) e um menor nível de expressão (*down-regulated*) na promoção da carcinogênese.

Câncer é a segunda maior causa de morte em países desenvolvidos e sua incidência no resto do mundo tem aumentado. De acordo com a Organização Mundial da Saúde (OMS), cerca de $8,2$ milhões de pessoas morreram de câncer em 2012 e a estimativa para 2030 é que este número chegue a quase 17 milhões [81].

O entendimento dos mecanismos de regulação que definem um comportamento diferente entre instâncias de pacientes saudáveis e com câncer, por exemplo, é um passo crucial para o desenvolvimento de novas terapias e tratamentos [24]. A Figura 5.2 mostra os genes HOX que são mais ou menos expressos em diferentes tipos de câncer.

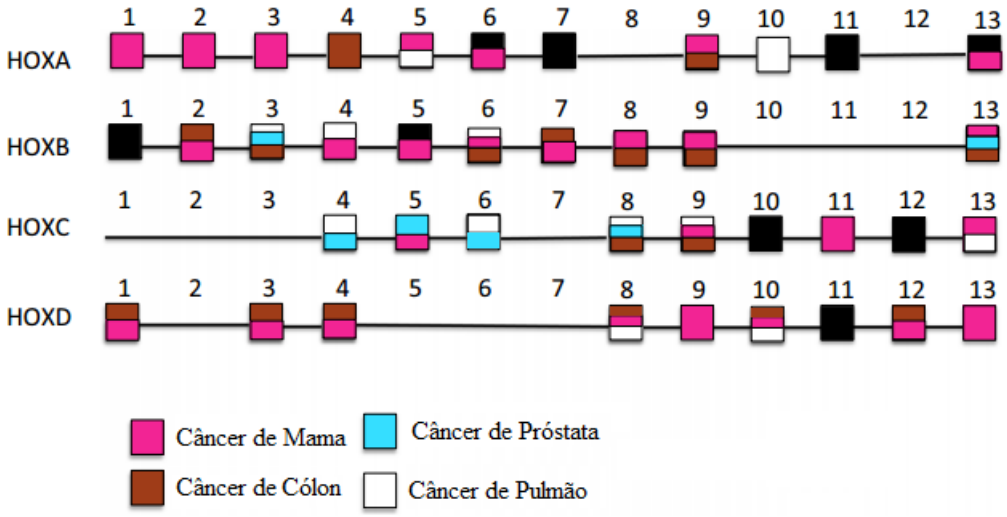


Figura 5.2: Níveis de expressão de genes HOX para diferentes tipos de câncer [22]. A cor de cada gene indica para qual tipo de câncer ele é diferencialmente expresso. Genes na cor preta indicam genes que não estão relacionados aos tipos de câncer apresentados na figura.

Nesta seção apresentamos um experimento que serviu para tomadas de decisões referentes a alguns passos do *pipeline* proposto e descrito no Capítulo 3, além da própria execução do mesmo para um conjunto de testes. Para o experimento aqui descrito utilizamos a versão `GRCh37.p13` do genoma humano.

Este experimento teve como objetivo a análise de expressão diferencial de genes HOX. Através dele foi possível determinar o mapeador a ser utilizado pelo *pipeline*

proposto e, como o mapeamento deveria ser feito (genoma completo ou apenas locais que os genes HOX se encontram), ambos abordados na seção 3.1.

Para o experimento foram utilizados dois conjuntos de dados com amostras de tecidos com câncer de próstata [82], onde um conjunto representa pacientes que receberam tratamentos a base de hormônios (*treated*) e o outro representando pacientes que não receberam tratamento (*untreated*). Dados os dois conjuntos, foi feito o mapeamento no genoma humano desses conjuntos de *reads* utilizando os mapeadores `segemehl` e `Bowtie`, a fim de comparar os resultados obtidos. Após o mapeamento, os *reads* mapeados foram contados pelo *software* `HTSeq` e utilizado o `EdgeR` para fazer a análise da expressão diferencial e gerar gráficos que possam detalhar a análise.

Ao analisarmos ambos os gráficos das Figuras 3.4 e 3.5, percebemos facilmente que o número de genes apresentados no gráfico do `segemehl` é maior que o do `Bowtie`, ou seja, o número de *reads* mapeados pelo `segemehl` é maior em comparação ao `Bowtie`.

Como o conjunto dos genes HOX é pequeno, é possível fazer uma análise a olho nu para cada gene no gráfico. A Figura 5.3 mostra o mesmo gráfico para um conjunto de dados muito maior, onde seria impossível a olho nu analisar cada gene.

Após constatar que o `segemehl` se mostrou melhor que o `Bowtie` para este experimento, foi realizada uma nova rodada para o experimento com o `segemehl` apenas, mas dessa vez fazendo o mapeamento dos *reads* contra o genoma humano completo e somente nos locais onde os genes HOX se encontram em seus respectivos cromossomos (na tentativa de reduzir o tempo e espaço necessário para o mapeamento). Para delimitar o local dos genes HOX, foi criado um subconjunto do genoma humano que contém apenas as regiões dos *clusters* HOX. A Tabela 5.1 mostra as posições utilizadas para criar tal subconjunto, de acordo com as informações presentes na anotação do genoma utilizado, onde a posição inicial é a primeira posição após o último gene localizado antes do *cluster* HOX e a posição final é a posição anterior ao próximo gene que ocorre após o *cluster* HOX.

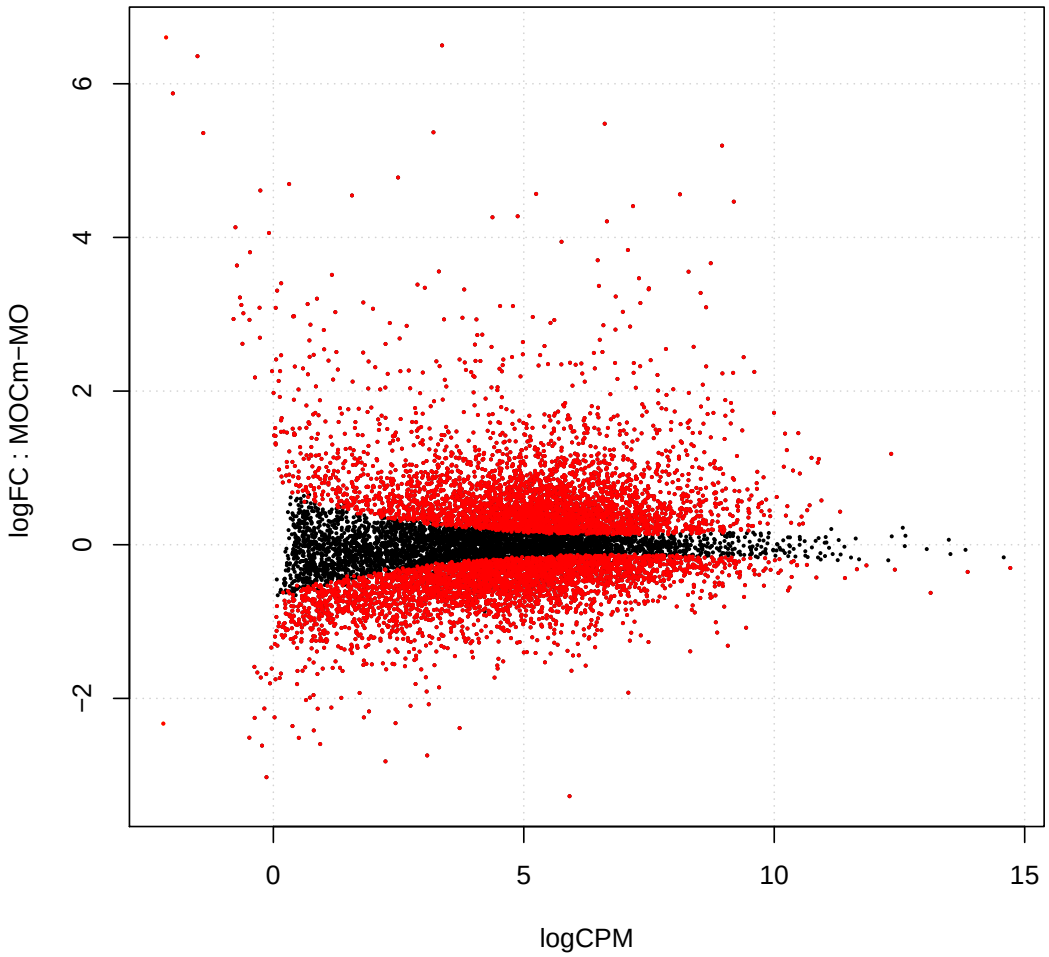


Figura 5.3: Gráfico *Smear* representando um conjunto grande de genes, onde os genes em vermelho acima do 0 no eixo vertical representa genes mais expressos em um certo conjunto A em relação a B e vice-versa [78].

Tabela 5.1: Região utilizada para criar o conjunto dos *clusters* HOX.

Cromossomo	**Pos Inicial**	**Pos Final**
2	176.944.835	177.065.63
7	27.088.248	27.238.193
12	54.210.264	54.452.037
17	46.556.561	46.810.452

Ao comparar os resultados obtidos, notou-se que a taxa de verdadeiros-positivos é muito baixa quando o mapeamento é feito apenas nos locais onde se encontram os genes HOX. A fórmula a seguir mostra a taxa de verdadeiros-positivos, onde TP é o número de verdadeiros-positivos (número de *reads* mapeados corretamente na região de interesse) e FP é o número de falsos-positivos (*reads* mapeados erroneamente na região de interesse).

$$\frac{TP}{TP + FP} = 0,6\%.$$

Portanto, como o número de verdadeiro-positivos é muito pequeno, quando comparado com o total mapeado na região de interesse, o experimento mostrou que não se deve mapear em regiões específicas de interesse, mas sim deve-se mapear contra o genoma humano completo.

5.2 Genes HOX e Doença de Huntington

A Doença de Huntington (*Huntington Disease*, HD) é uma doença neurológica, hereditária e degenerativa, cujos sintomas incluem o aparecimento de movimentos involuntários, demência e mudanças de personalidade. É geneticamente causada pela mutação no gene huntingtin (HTT), especificamente pelo aumento do número de repetições do codon `CAG`. Micro RNAs (miRNAs) têm sido relacionados na literatura a doenças desse tipo, em particular com HD [61].

Para esse experimento, tomamos o conjunto de *reads* de RNA-seq disponível em [61] como sendo nosso *dataset* para o mapeamento. Nesse trabalho, os autores procuram investigar se há expressão alterada de miRNAs em pacientes humanos com HD e qual o papel desses miRNAs no nível de expressão dos seus genes alvos.

Temos dois conjuntos distintos de amostras: um conjunto com 12 amostras de pacientes que sofrem de doença de Huntington e outro conjunto com 9 amostras de pacientes sadios que não possuem a doença (chamados de controle – Ctrl), totalizando 21 amostras diferentes.

Primeiramente, foi feita a construção do índice do genoma humano e, em seguida, o mapeamento dos *reads* de cada amostra utilizando o `segemehl` contra o genoma humano completo (versão `GRCh37.p13`), resultando em um arquivo SAM para cada amostra que servirá de entrada para o `Cufflinks`. Porém, o `Cufflinks` requer que os *reads* estejam ordenados pela sua posição de referência. Como o mapeamento do `segemehl` não estava ordenado, a ordenação foi feita utilizando o *software* `SAMtools`.

Para ordenar, antes tivemos que converter o arquivo de formato SAM para sua forma binária, conhecida como formato BAM. Para diminuir o tempo e automatizar essa tarefa, foi então criado um *script* chamado `sam2bam.pl` escrito em linguagem `perl` que faz a conversão utilizando o `SAMtools` e cria um novo arquivo em formato BAM. Agora já com os arquivos BAM de cada amostra mapeada, podemos fazer o ordenação. Para esta tarefa, também foi criado um *script* em `perl` chamado `bam2sorted.pl` que irá criar um arquivo BAM já ordenado.

Após a ordenação ter sido concluída para o resultado do mapeamento de cada amostra, o próximo passo executado foi a criação dos transcritos, onde o `Cufflinks` foi executado passando como entrada a anotação do genoma (GTF) com a opção `-g` e, o arquivo BAM ordenado para cada uma das amostras. O resultado gerado foram os arquivos contendo a contagem dos genes, isoformas e os transcritos construídos.

O *pipeline* até a construção dos transcritos já foi mostrado na seção 3.2, onde ele deve ser executado separadamente para cada uma das amostras.

O próximo passo foi a união dos transcritos de todas as amostras com o `Cuffmerge`. O `Cuffmerge` foi executado passando como entrada a anotação do genoma com a opção `-g`, o próprio genoma humano em formato FASTA com a opção `-s` e um arquivo texto contendo o caminho para cada transcrito construído pelo `Cufflinks`, um por linha. O resultado gerado é a criação de uma pasta no diretório atual com um único arquivo chamado `merged.gtf`, contendo a união de todos os transcritos das amostras.

Com a união de todos os transcritos em um único arquivo, passamos ao passo de expressão diferencial. O `Cuffdiff` foi executado tomando como entrada o arquivo FASTA do genoma humano com a opção `-b`, o arquivo `merged.gtf` contendo a união dos transcritos com a opção `-u`, o rótulo de ambas as condições das amostras utilizadas, que são `Ctrl` e `HD`, separados por vírgula com a opção `-L` e por fim, as listas

dos arquivos BAM ordenados de cada amostra separado por vírgula, respeitando a ordem que foram informados os rótulos. Logo, como o primeiro rótulo foi `Ctrl`, a primeira lista que foi passada são amostras de controle. A separação entre as listas é feita apenas por um espaço. O resultado gerado por ele são diversos arquivos contendo informações de expressão diferencial para transcritos, isoformas e contagem de RPKM, `reads` que originou cada transcrito e isoformas. O restante do *pipeline* já foi mostrado na seção 3.5.

Com o resultado da expressão diferencial feito pelo `Cuffdiff`, utilizamos o software `cummeRbund` para fazer a análise dos resultados obtidos e geração de alguns gráficos. Dentre os resultados, estão os gráficos mostrados nas Figuras 5.4 e 5.5. O primeiro mostra que os genes HOX-D3, HOX-D4 e o miRNA MIR-10B são mais expressos em amostras HD do que as de Ctrl. O segundo mostra que o gene HOXD-AS1 é mais expresso em amostras Ctrl do que as de HD. Há ainda o dendrograma (Figura 5.6) e um gráfico MDS (Figura 5.7) que mostra o relacionamento e similaridade, respectivamente, de todas as amostras utilizadas no experimento. Através desses gráficos é possível verificar que há amostras de Ctrl muito similares as de HD, o que não é algo bom visto que isso pode influenciar na análise de expressão diferencial.

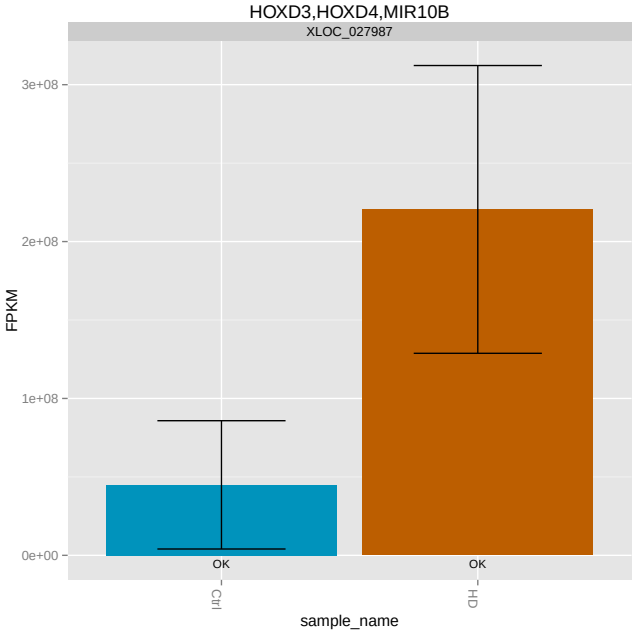


Figura 5.4: Gráfico gerado pelo `cummeRbund` que mostra que os genes HOX-D3, HOX-D4 e o miRNA MIR-10B são mais expressos em amostras HD do que as de Ctrl.

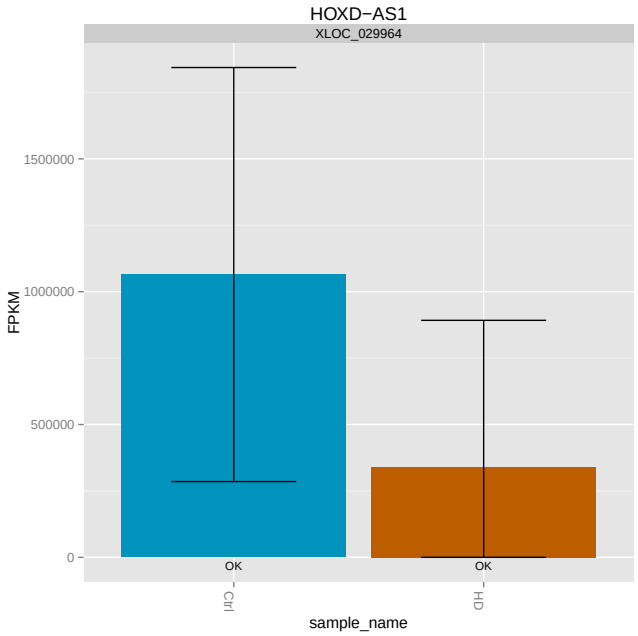


Figura 5.5: Gráfico gerado pelo `cummeRbund` que mostra que o gene HOXD-AS1 é mais expresso em amostras Ctrl do que as de HD.

Apesar de não termos encontrado todos os genes HOX descritos como diferencialmente expressos em [61], todos os que encontramos, listados no parágrafo anterior, foram confirmados.

Pipeline e DEPICTViz

Após o desenvolvimento da `DEPICTViz`, executamos novamente o *pipeline* visando integrar as duas ferramentas. Desta vez utilizamos uma versão mais recente do genoma humano (`GRCh38.p15`) e os resultados do *pipeline* após a execução do `Cuffdiff` foram submetidos aos *scripts* `getCoordinatesCuffdiffOut.R` e `diff2depictviz.pl` discutidos na Seção 3.6 e presentes no pacote do *pipeline*. Através dos *scripts* utilizados foi possível converter a saída do `Cuffdiff` para um formato compatível com a `DEPICTViz`. Como o conjunto de entrada para a `DEPICTViz` carecia de informação, enriquecemos a informação presente após cruzarmos os genes presentes com informações obtidas em `BioMart` (`http://www.ensembl.org/biomart/`) e `UniProt` (`http://www.uniprot.org/uniprot/`) para os mesmos genes presentes em *Homo*

sapiens. Logo, adicionamos informações de Swiss-Prot, EC *number*, KO, KEGG gene e GOs.

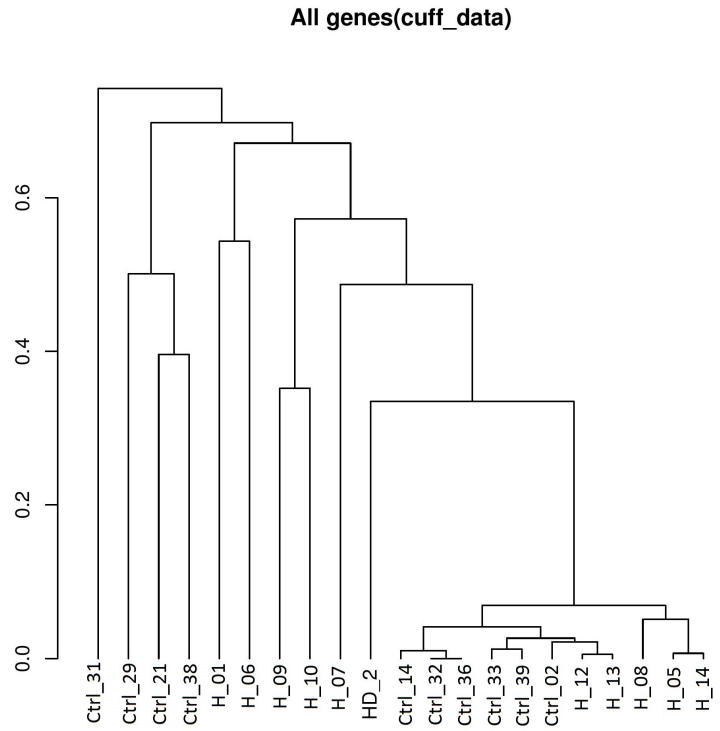


Figura 5.6: Dendrograma que mostra o relacionamento de todas amostras utilizadas neste experimento.

Como o objetivo do experimento era investigar miRNAs e genes HOX, filtramos a tabela de entrada apenas (Figura 5.8) para mostrar esse subconjunto (560 miRNAs e genes HOX) e também pela impossibilidade de mostrar todos os genes do experimento. Embora tenha sido utilizado uma anotação adequada do genoma durante as etapas do *pipeline*, muitos genes não foram reconhecidos devido à ausência de informações específicas na anotação, provavelmente miRNAs e ncRNAs. Logo, utilizados apenas miRNAs e HOX reconhecidos pela anotação.

Percebemos também que todos os miRNAs considerados diferencialmente expressos ($p\text{-}value < 0.05$) pelo *pipeline* estão sendo mais expressos em amostras HD em relação a de controle (Figura 5.9).

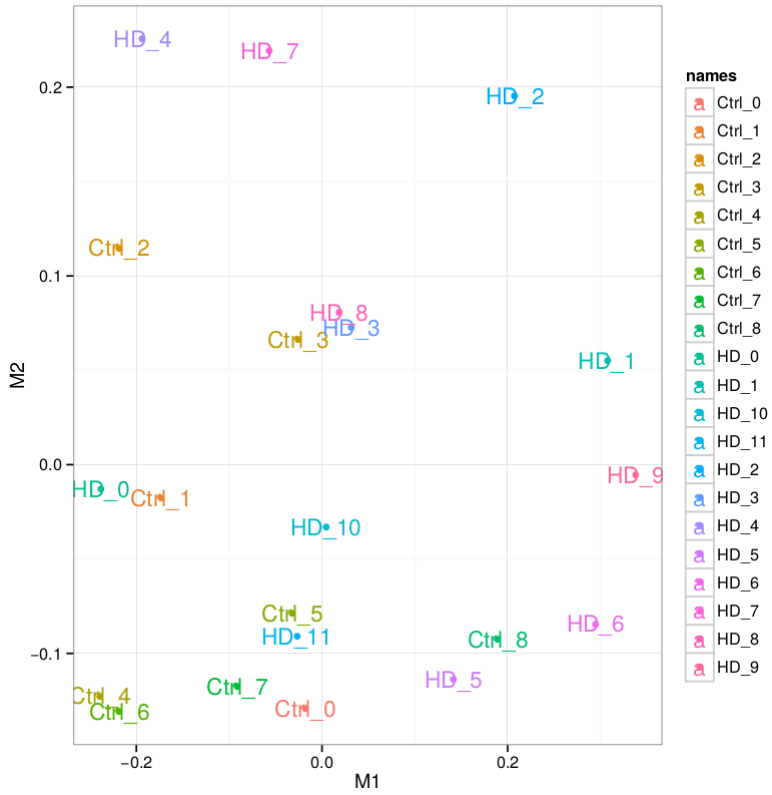


Figura 5.7: Gráfico MDS que mostra a similaridade de todas amostras utilizadas neste experimento.

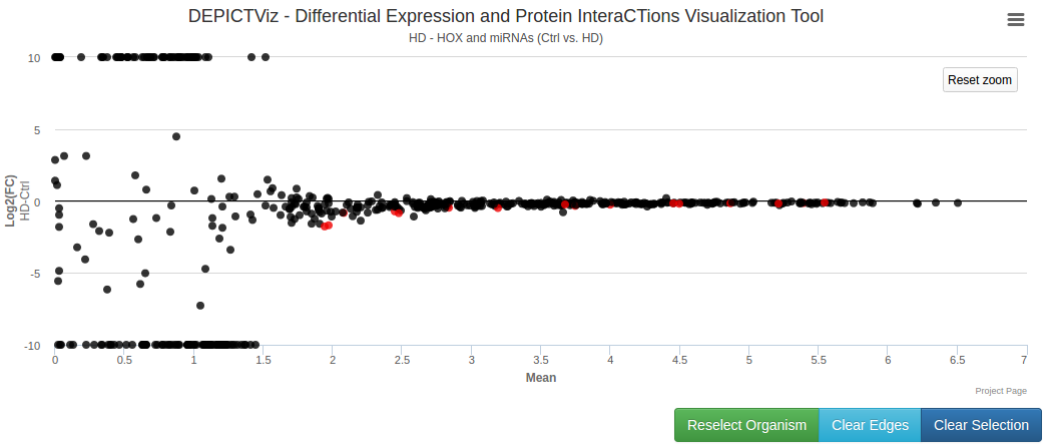


Figura 5.8: Visualização de miRNAs e genes HOX na `DEPICTViz` obtidos do experimento HD executado pelo *pipeline*.

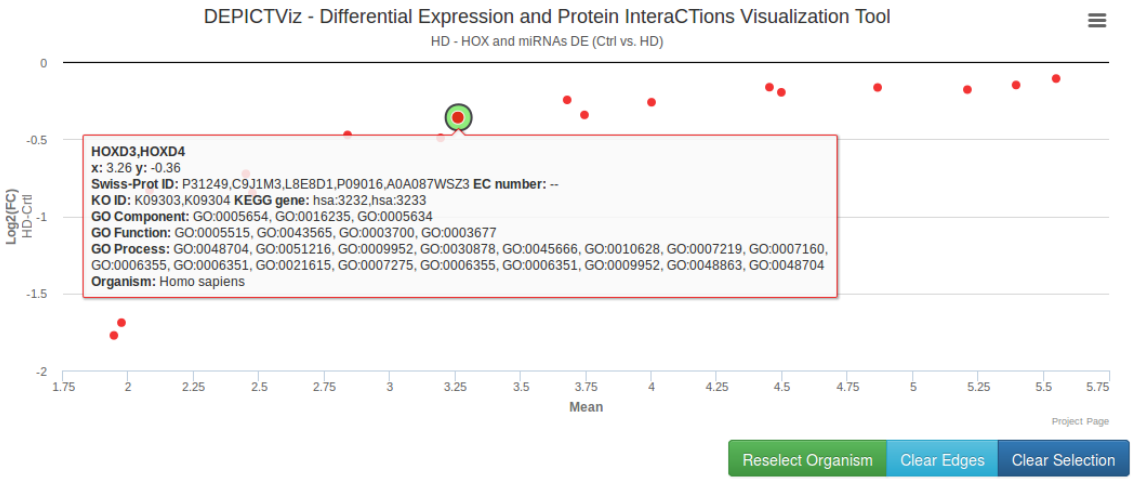


Figura 5.9: Visualização de genes HOX e miRNAs diferencialmente expressos para amostras HD em relação as de controle.

Esse estudo de caso foi importante para o trabalho demonstrando compatibilidade entre as duas diferentes ferramentas propostas nesse trabalho. Resultados obtidos até este ponto foram apresentados no Simpósio Brasileiro de Bioinformática de 2015 [72].

5.3 *Brachiaria decunbens*

O Brasil tem cerca de 169 milhões de hectares de pastagens, usadas para alimentar cerca de 208 milhões de cabeças de gado. As principais forrageiras usadas pertencem ao gênero *Urochloa*, popularmente conhecido como *Brachiaria* [80].

A espécie *Urochloa decumbens*, ou *Brachiaria decumbens*, tem ganho especial atenção, pelo fato de ser de alta produtividade, ter crescimento vigoroso e ser de fácil plantio. Além disso, se adapta facilmente em solos ácidos e não férteis. A alta presença de alumínio é uma das principais características de solos ácidos e é um dos fatores que mais prejudicam o crescimento dessa forrageira. Isso acontece porque o alumínio modifica a parede celular e a estrutura da membrana da planta, prejudicando a absorção de água e nutrientes, comprometendo assim seu crescimento. Dessa forma, a tolerância ao alumínio nesse tipo de solo tem sido alvo de estudos [94].

Em um projeto coordenado por pesquisadores da Embrapa Gado de Corte, o transcritoma de raízes de *U. decumbens* foi obtido, objetivando estudar os efeitos causados pela alta concentração de alumínio [94].

O experimento realizado com a *Brachiaria decunbens* foi o primeiro na qual utilizamos a `DEPICTViz` em um projeto prático, e com grande impacto em seu desenvolvimento, permitindo a inclusão de diversas funcionalidades através da grande quantidade de informações que estavam presentes nos dados do projeto.

Inicialmente, havia apenas uma tabela contendo genes representado por IDs, pelo fato do genoma não possuir uma anotação disponível, a contagem dos *reads* mapeados de cada gene e demais informações que foram atribuídas para cada gene como: Swiss-Prot ID, COG ID, COG *Description*, COG *Category*, KO ID, KEGG Gene, EC *number*, GO *Biological Process*, GO *Molecular Function*, GO *Cellular Component* e descrições para os três tipos de ontologias.

Como não havia a informação básica para a construção do gráfico (coordenadas e *p-value*), o caminho então tomado foi a execução do `EdgeR` utilizando as contagens já existentes, para então conseguirmos extrair as coordenadas (Log_2FC e LogCPM) e o *p-value* de cada gene. Posteriormente, os mesmos passos foram realizados utilizando o `DESeq2`.

A partir do momento que possuíamos todas as informações necessárias, selecionamos o genes de interesse para o experimento e construímos o arquivo de entrada, descrito na Seção 4.2. No total foram realizados dois experimentos. O primeiro experimento possuía 200 genes, 100 genes dos quais classificados como os mais diferencialmente expressos (*up-regulated*) para a condição `Final` em relação a `Inicial`; e 100 genes classificados como os mais diferencialmente expressos (*down-regulated*) para a condição `Inicial` em relação a `Final`.

O segundo experimento possuía 400 genes na mesma proporção do anterior, 200 *up-regulated* e 200 *down-regulated*. A razão de haver dois experimentos com diferentes números de genes foi testar o comportamento e navegação da ferramenta, utilizando diferentes volumes de dados. Embora tivesse o dobro de genes no gráfico, o segundo experimento mostrou boa usabilidade e não apresentou problemas durante sua execução, limitando-se apenas a uma inicialização ligeiramente maior, devido a leitura do arquivo de entrada maior.

A Figura 4.3, mostrada na Seção 4.3, corresponde ao gráfico carregado com os dados do segundo experimento (400 genes), enquanto que a Figura 5.10 mostra o transcrito Unigene66910 e todos os diferencialmente expressos que possuem a mesma categoria COG (categoria I - *lipid metabolism*).

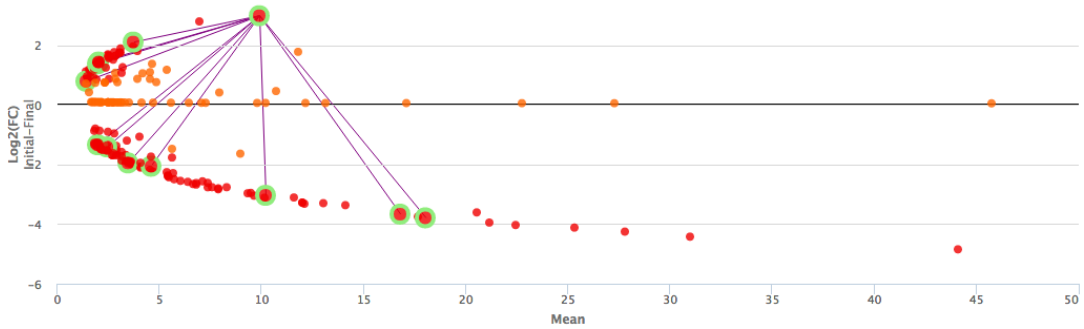


Figura 5.10: Todos os transcritos considerados diferencialmente expressos, com destaque para a escolha do transcrito 66910, a partir do qual vê-se todos com a mesma categoria COG I (*lipid metabolins*).

Pelo fato do organismo não possuir uma anotação para o genoma, a seleção do organismo deve ser feita como *Other* (Figura 4.2) e algumas funções não são disponíveis durante o uso, ficando inativas como o *Genome Browser* (Figura 4.10), por exemplo. Os resultados até aqui obtidos estão sendo usados na publicação submetida recentemente [94].

5.4 *Pseudomonas corrugata*

Pseudomonas corrugata é uma bactéria de plantas oportunista que causa a doença conhecida como *tomato pith necrosis*. Apesar disso, consegue proteger a planta de outros patógenos, sendo também usada como agente de controle biológico [105].

A produção de lipopeptídeos em *P. corrugata* é regulado por dois reguladores transcricionais pertencentes à superfamília LuxR, conhecidos como PcoR e RfiA. O interesse neste projeto é o estudo do papel que esses reguladores têm na interação com plantas e na atividade antimicrobiana. Especificamente, deseja-se investigar os genes controlados por PcoR e RfiA. Para isso, análise de expressão diferencial baseada em RNA-Seq foi feita usando mutantes PcoR e RfiA, em comparação com o *wild type* (`WT`) CFBP5454, cultivado normalmente, que foi sequenciado recentemente e descrito em [105].

Esse projeto tem a colaboração com pesquisadores da Grécia e da Itália e já geraram duas publicações de resumos em congressos [70, 71].

Neste caso, não houve grandes problemas ao montar o arquivo de entrada, visto que os dados já haviam sido processados pelo `EdgeR` e já possuíam os campos necessários. Diferente dos outros e experimentos realizados até então, para este haviam informações que até então não haviam sido usadas, sendo elas: *Start*, *End*, *Contig*, *Strand* e *Gene Product*. Estas informações não estão diretamente ligadas a funcionalidades como as outras existentes, porém elas servem como consulta ao usuário e são mostradas tanto na legenda ao passar o *mouse* por um ponto, quanto ao clicar no mesmo (Figura 5.11). Além disso, há ainda informações de Swiss-Prot ID presentes.

Embora os genes tenham sido já entregues anotados, após uma rápida pesquisa nas bases de dados utilizadas na ferramenta, não foram encontradas referências a *Pseudomonas corrugata*, logo, preferimos nos abster e recomendar a seleção de *Other* durante a seleção de organismos da `DEPICTViz`.

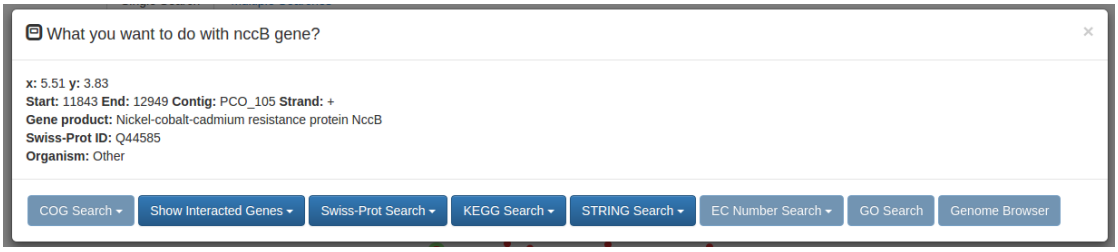


Figura 5.11: Conjunto de informações usadas nos experimentos de *Pseudomonas corrugata* que até então não haviam sido utilizadas.

Através do conjunto de dados obtidos, foram realizados quatro experimentos distintos, sendo dois experimentos entre as condições `WT vs DR` e dois entre `WT vs RfiA`. Para a primeira rodada, os dois primeiros experimentos (Figuras 5.12 e 5.13) continham apenas genes considerados diferencialmente expressos para alguma condição. Para alcançar esse objetivo, selecionamos os genes com FDR < 0.05. Durante os quatro experimentos realizados, o FDR foi utilizado ao invés do *p-value*. O FDR (*False Discovery Rate*) é uma forma ajustada do *p-value*, também chamada de *q-value*. Logo, o uso do mesmo como substituto não causa nenhum impacto no experimento. Ainda assim, caso fosse necessário usar outro indicador diferente de *p-value* ou FDR com diferentes limiares, este poderia normalmente preencher o campo de *p-value* no arquivo de entrada, e seu limiar de corte alterado no arquivo de configuração.

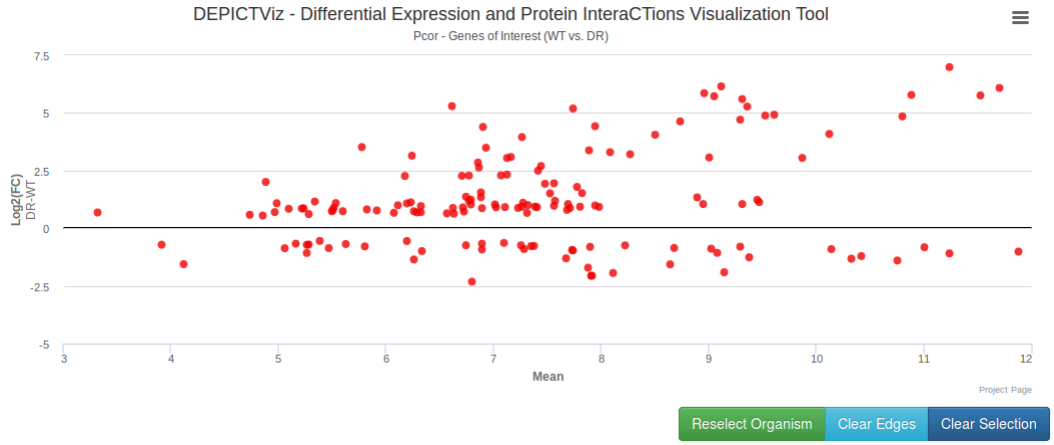


Figura 5.12: Experimento `WT vs DR` contendo apenas genes considerados diferencialmente expressos em alguma condição (FDR < 0.05). No total são 152 genes presentes no gráfico.

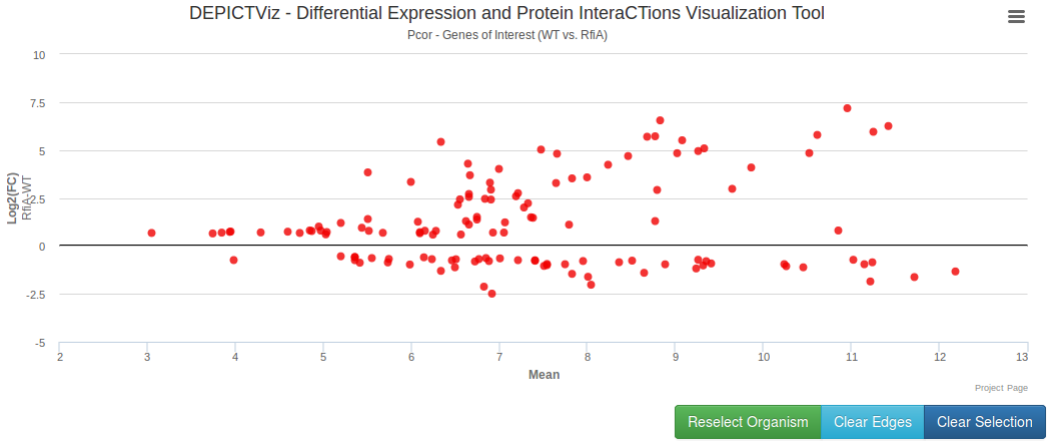


Figura 5.13: Experimento `WT vs RfiA` contendo apenas genes considerados diferencialmente expressos em alguma condição (FDR < 0.05). No total são 130 genes presentes no gráfico.

Para a segunda rodada também foram realizados dois novos experimentos (Figuras 5.14 e 5.15), utilizando as mesmas condições da rodada passada, porém incluindo todos os genes, seja qual for o valor de seu FDR. A Tabela 5.2 representa os experimentos executados e o total de genes presentes em cada experimento.

O conjunto de experimentos executados teve uma grande importância para o trabalho como um todo. Com ele foi possível por a prova a ferramenta para grandes volumes de dados e analisar seu comportamento. Seu comportamento somente é afetado drasticamente durante a inicialização da ferramenta, onde ocorre a leitura do

arquivo de entrada e construção do gráfico, levando alguns segundos para carregá-lo totalmente devido ao grande volume de dados.

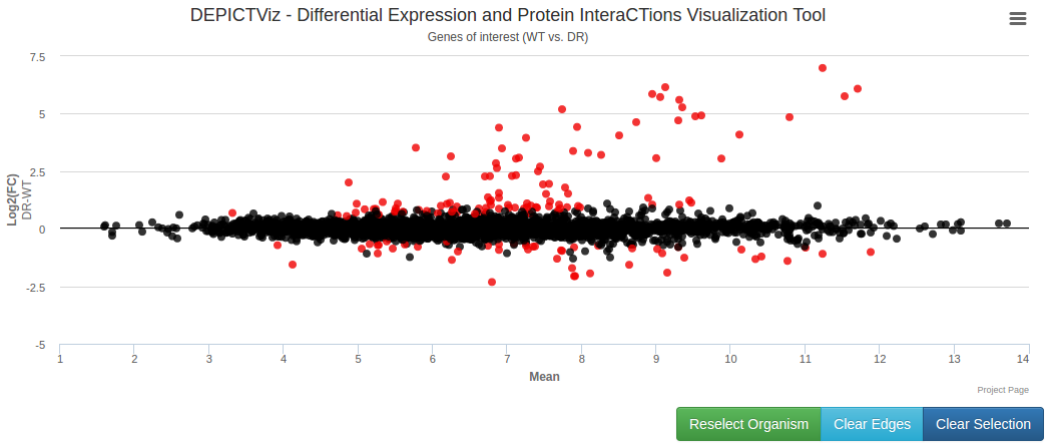


Figura 5.14: Experimento `WT vs DR` contendo todos os genes do experimento. Nota-se a alta concentração de genes próximo ao 0 no eixo x. No total são 3096 genes presentes no gráfico.

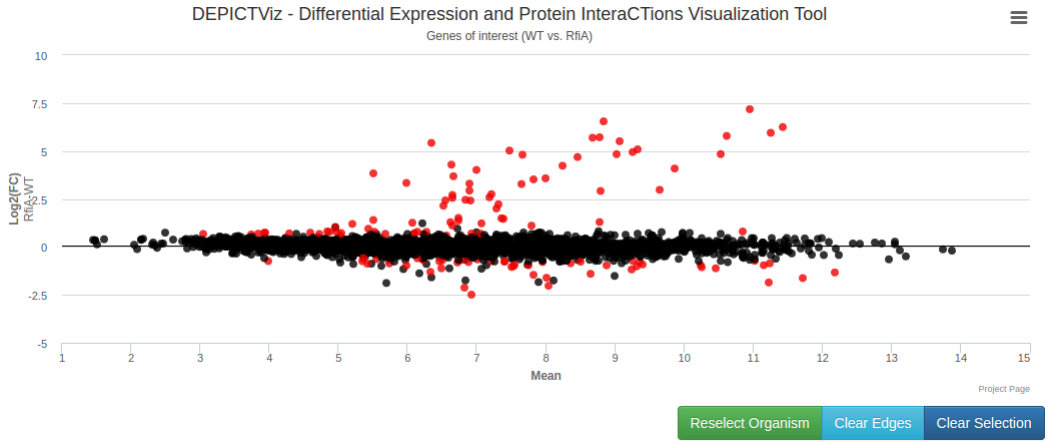


Figura 5.15: Experimento `WT vs RfiA` contendo todos os genes do experimento. Nota-se a alta concentração de genes próximo ao 0 no eixo x. No total são 3966 genes presentes no gráfico.

Tabela 5.2: Total de genes presentes em cada um dos experimentos.

Experimento	Condições	FDR	Total Genes
1	`WT vs DR`	< 0.05	152
2	`WT vs RfiA`	< 0.05	130
3	`WT vs DR`	Todos	3096
4	`WT vs RfiA`	Todos	3966

Capítulo 6

Conclusão

O trabalho aqui descrito e resultado de minha pesquisa no programa de Mestrado em Ciência da Computação consistiu no desenvolvimento de duas ferramentas computacionais: um pipeline para análise de expressão diferencial e outra para a visualização de resultados de expressão diferencial, chamada DEPICTViz, que agrega ao resultado da análise de expressão diferencial informações de interação funcional de proteínas.

O *pipeline* desenvolvido realiza o mapeamento das sequências no genoma de referência, a ordenação dos *reads* mapeados, a construção e união de transcritos e a análise de expressão diferencial.

A ferramenta `DEPICTViz`, por sua vez, foi desenvolvida para a visualização de experimentos de expressão diferencial de genes e interação de proteínas. A ferramenta *web* surgiu da necessidade de se visualizar pequenos conjuntos de genes de interesse em experimentos de expressão diferencial de RNA-Seq. Sua principal característica é a apresentação de um gráfico MA (*MA plot*), similar aos gráficos *Smear Plot*, presentes nos pacotes `EdgeR` e `DESeq2`, respectivamente. Não temos conhecimento na literatura de uma ferramenta semelhante.

O principal diferencial da `DEPICTViz` é o fato de ela ser interativa. Além disso, `DEPICTViz` é de fácil instalação, e também de fácil utilização. A entrada da ferramenta é basicamente uma tabela contendo todas as informações necessárias para a construção do gráfico e suas funcionalidades. Como resultado, a saída da ferramenta

é apresentada uma janela no navegador *web* do usuário, podendo este hospedá-la em um servidor ou mantê-la localmente.

As duas ferramentas foram avaliadas a partir de projetos que nos foram demandados por colaboradores externos à UFMS e resultaram nas publicações [70, 71, 72, 94], além de uma específica, descrevendo e disponibilizando o DEPICTViz, que está em preparação.

Trabalhos Futuros

O desenvolvimento e manutenção do *pipeline* e da ferramenta `DEPICTViz`, propostos nesse trabalho, não se limitam apenas à conclusão e obtenção do título de mestre proposto pelo programa de pós-graduação, visto que ainda há muito o que pode ser feito e melhorado, principalmente na `DEPICTViz`.

A cada experimento com o qual nos deparamos, demandas por novas funcionalidades foram aparecendo. Entre as prováveis novas funcionalidades que poderiam ser incluídas estão: ligação por arestas entre genes que possuem interação entre si segundo o STRING utilizando informações de Swiss-Prot, inclusão de novas bases de dados para pesquisas (GO, EC *number*, COG), além de melhorias de estabilidade e código.

O *pipeline* pode também ser expandido para análise de expressão diferencial de isoformas, sendo esse um dos pontos fortes do pacote `Cufflinks` e que poucos programas o fazem. Dentre outras possíveis inclusões estão a inserção de novas etapas como: filtragem de dados antes do mapeamento e inclusão de novas anotações antes da construção dos transcritos. A inclusão das novas anotações permitiria ao usuário do *pipeline* escolher o que deseja procurar, ou seja, possibilitar que ele próprio enriqueça a anotação do genoma, através do *download* de anotações contidas em bases de dados específicas de anotação.

A inclusão dessas anotações poderia ser feita utilizando um arquivo de configuração (novo ou o já existente), onde haveria opções de múltipla escolha, como por exemplo as bases de dados suportadas pelo *pipeline*. Essas bases podem ser, por exemplo, de miRNA ou ncRNAs, geralmente não presentes nas anotações originais. Exemplos

de bases de dados que poderão ser investigadas e utilizadas são: `miRWalk` [31] e `miRTarBase` [62].

Referências Bibliográficas

[1] Annotation of UniProtKB. http://www.uniprot.org/help/manual_curation. Acessado em 10 de março de 2016.

[2] ATOM - A Hackable text editor. Disponível em: https://atom.io/. Acessado em 27 de abril de 2016.

[3] Biological Pathways. National Human Genome Research Institute. http://www.genome.gov/27530687. Acessado em 14 de março de 2016.

[4] Bootstrap. Disponível em: http://getbootstrap.com/getting-started/#download. Acessado em 27 de abril de 2016.

[5] Bootstrap-select. Disponível em: https://github.com/caseyjhol/bootstrap-select. Acessado em 27 de abril de 2016.

[6] Highcharts. Disponível em: http://www.highcharts.com/download. Acessado em 27 de abril de 2016.

[7] jQuery. Disponível em: http://jquery.com/download/. Acessado em 27 de abril de 2016.

[8] jQuery-Autocomplete. Disponível em: https://github.com/devbridge/jQuery-Autocomplete. Acessado em 27 de abril de 2016.

[9] jQuery Caret. Disponível em: https://github.com/accursoft/caret. Acessado em 27 de abril de 2016.

[10] jQuery-tagEditor. Disponível em: https://github.com/Pixabay/jQuery-tagEditor. Acessado em 27 de abril de 2016.

[11] Nprogress. Disponível em: http://ricostacruz.com/nprogress/. Acessado em 27 de abril de 2016.

[12] SweetAlert. Disponível em: `https://github.com/t4t5/sweetalert`. Acessado em 27 de abril de 2016.

[13] Toastr. Disponível em: `https://github.com/CodeSeven/toastr`. Acessado em 27 de abril de 2016.

[14] Enzyme nomenclature 1992: recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the nomenclature and classification of enzymes. `http://www.chem.qmul.ac.uk/iubmb/enzyme/`, 1992. School of Biological and Chemical Sciences, Queen Mary University of London, UK.

[15] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[16] S. Anders and W. Huber. Differential expression analysis for sequence count data. *Genome Biol*, 11(10):R106, 2010.

[17] S. Anders, P. T. Pyl, and W. Huber. HTSeq – a Python framework to work with high-throughput sequencing data. *Bioinformatics*, 2014.

[18] S. Andrews. FastQC, A Quality Control tool for High Throughput Sequence Data. `http://www.bioinformatics.babraham.ac.uk/projects/fastqc/`, 2010. Acessado em 19 de fevereiro de 2016.

[19] A. Bateman. UniProt: a hub for protein information. *Nucleic Acids Research*, 43(D1):D204–D212, 2014.

[20] J. M. Berg, J. L. Tymoczko, and L. Stryer. *Biochemistry*. New York: W H Freeman, 2002.

[21] T. Berggård, S. Linse, and P. James. Methods for the detection and analysis of protein–protein interactions. *Proteomics*, 7(16):2833–2842, 2007.

[22] S. Bhatlekar, J.Z. Fields, and B.M. Boman. Hox genes and their role in the development of human cancers. *Journal of Molecular Medicine*, 92(8):811–823, 2014.

[23] D. Biau, B. Jolles, and R. Porcher. P Value and the Theory of Hypothesis Testing: An Explanation for New Researchers. *Clinical Orthopaedics and Related Research*, 468(3):885–892, 2009.

[24] J. Boer. Development of Quality Assessment Methods for De Novo Transcriptome Assemblies & Expression Analysis. Master's thesis, Leiden Institute of Advanced Computer Science (LIACS), 2013.

[25] M. R. Borges-Osório and W. M. Robinson. *Genética Humana*. Artmed, 2013.

[26] P. Braun and A. Gingras. History of protein-protein interactions: From egg-white to complex networks. *Proteomics*, 12(10):1478–1498, 2012.

[27] T. A. Brown. *Genomes 3*. Garland Science, 3rd edition, 2006.

[28] N. Campbell and J. Reece. *Biology*. Boston : Benjamin Cummings / Pearson, 9th edition, 2011.

[29] V. Costa, C. Angelini, I. Feis, and A. Ciccodicola. Uncovering the complexity of transcriptomes with RNA-seq. *Journal of Biomedicine and Biotechnology*, 2010:1–19, 2010.

[30] A. Dobin and C. A. Davis *et al.* STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, 29(1):15–21, 2012.

[31] H. Dweep, C. Sticht, P. Pandey, and N.Gretz. miRWalk – database: prediction of possible miRNA binding sites by walking the genes of 3 genomes. *Journal of Biomedical Informatics*, 44, 2011.

[32] Volker Egelhofer, Ida Schomburg, and Dietmar Schomburg. Automatic assignment of EC numbers. *PLoS Comput Biol*, 6(1):e1000661, 2010.

[33] A. D. Diehl *et al.* Ontology development for biological systems: immunology. *Bioinformatics*, 23(7):913–915, 2007.

[34] A. G. McDonald *et al.* ExplorEnz: a MySQL database of the IUBMB enzyme nomenclature. *BMC Biochemistry*, 8(1):14, 2007.

[35] J. Bullard *et al.* Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments. *BMC Bioinformatics*, 11(1):94, 2010.

[36] J. Huerta-Cepas *et al.* eggNOG 4.5: a hierarchical orthology framework with improved functional annotations for eukaryotic, prokaryotic and viral sequences. *Nucleic Acids Res*, 44(D1):D286–D293, 2015.

[37] L. J. Jensen *et al.* STRING 8–a global view on proteins and their functional interactions in 630 organisms. *Nucleic Acids Research*, 37(Database):D412–D416, 2009.

[38] M. Dillies *et al.* A comprehensive evaluation of normalization methods for illumina high-throughput RNA sequencing data analysis. *Briefings in Bioinformatics*, 14(6):671–683, 2012.

[39] M. Kanehisa *et al.* KEGG as a reference resource for gene and protein annotation. *Nucleic Acids Res*, 44(D1):D457–D462, 2015.

[40] M. L. Speir *et al.* The UCSC genome browser database: 2016 update. *Nucleic Acids Res*, 44(D1):D717–D725, 2015.

[41] M. R. Miller *et al.* TU-tagging: cell type–specific RNA isolation from intact complex tissues. *Nature Methods*, 6(6):439–441, 2009.

[42] M. Y. Galperin *et al.* Expanded microbial genome coverage and improved protein family annotation in the COG database. *Nucleic Acids Research*, 43(D1):D261–D269, 2014.

[43] R. Caspi *et al.* The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of Pathway/Genome Databases. *Nucleic Acids Research*, 42(D1):D459–D471, 2013.

[44] R. E. Foulger *et al.* Using the Gene Ontology to Annotate Key Players in Parkinson's Disease. *Neuroinform*, 2016.

[45] S. Carbon *et al.* AmiGO: online access to ontology and annotation data. *Bioinformatics*, 25(2):288–289, 2008.

[46] R. D. Finn, A. Bateman, J. Clements, P. Coggill, R. Y. Eberhardt, S. R. Eddy, A. Heger, K. Hetherington, L. Holm, J. Mistry, E. L. L. Sonnhammer, J. Tate, and M. Punta. Pfam: the protein families database. *Nucleic Acids Research*, 42(D1):D222–D230, 2014.

[47] W. M. Fitch. Distinguishing homologous from analogous proteins. *Syst. Zool*, 19:99–113, 1970.

[48] W. M. Fitch. Homology: a personal view on some of the problems. *Trends in Genetics*, 16(5):227–231, 2000.

[49] E. Gasteiger. ExPASy: the proteomics server for in-depth protein knowledge and analysis. *Nucleic Acids Research*, 31(13):3784–3788, 2003.

[50] L.Y. Geer, Marchler-Bauer A., R.C. Geer, L. Han, J. He, S. He, C. Liu, W. Shi, and S.H. Bryant. The NCBI BioSystems database. *Nucleic Acids Research*, 38:D492–6, 2010.

[51] W.J. Gehring. Master Control genes in development and evolution: The Homeobox story (Terry Lecture Series). *New Haven: Yale University Press*, 1998.

[52] Gene Ontology Consortium. The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Research*, 32(90001):258D–261, 2004.

[53] L. Goff, C. Trapnell, and D. Kelley. *cummeRbund: Analysis, exploration, manipulation, and visualization of Cufflinks high-throughput sequencing data.*, 2013. R package version 2.8.2.

[54] F.R. Goodman. Congenital abnormalities of body patterning: embryology revisited. *The Lancet*, 362(9384):651–662, 2003.

[55] N. Guex and M. C. Peitsch. SWISS-MODEL and the Swiss-Pdb Viewer: An environment for comparative protein modeling. *Electrophoresis*, 18(15):2714–2723, 1997.

[56] R. C. Hardison. Comparative Genomics. *PLoS Biology*, 1(2):e58, 2003.

[57] S. Hoffmann and C. Otto *et al. A segemehl manual*. Transcriptome Bioinformatics Group and Interdisciplinary Centre for Bioinformatics, Leipzig, Germany, 2012.

[58] S. Hoffmann and C. Otto *et al.* A multi-split mapping algorithm for circular RNA, splicing, trans-splicing and fusion detection. *Genome Biol*, 15(2):R34, 2014.

[59] S. Hoffmann, C. Otto, and S. Kurtz *et al.* Fast Mapping of Short Sequences with Mismatches, Insertions and Deletions Using Index Structures. *PLoS Comput Biol*, 2009.

[60] J. F. Holland and E. Frei *et al.* *CANCER MEDICINE*, volume 8. pmph usa, 2009.

[61] A.G. Hoss, V.K. Kartha, and X. Dong *et al.* MicroRNAs located in the hox gene clusters are implicated in huntington's disease pathogenesis. *PLoS Genetics*, 10(2):e1004188, 2014.

[62] S.D. Hsu and Y.T. Tseng *et al.* miRTarBase update 2014: an information resource for experimentally validated miRNA-target interactions. *Nucleic Acids Research*, pages D78–85, 2014.

[63] G. Karp. *Cell and Molecular Biology. Concepts and Experiments.* New Jersey: John Wiley, 5th edition, 2008.

[64] D. Kim and G. Pertea *et al.* TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome Biol*, 14(4):R36, 2013.

[65] The Hannon Lab. FASTX-Toolkit. `http://hannonlab.cshl.edu/fastx_toolkit/`. Acessado em 19 de fevereiro de 2016.

[66] B. Langmead and C. Trapnell *et al.* Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol*, 10(3):R25, 2009.

[67] B. Langmead and S. L. Salzberg. Fast gapped-read alignment with Bowtie 2. *Nature Methods*, 9(4):357–359, 2012.

[68] B. Lewin, J. E. Krebs, E. S. Goldstein, and S. T. Kilpatrick. *Lewin's Genes XI.* Jones and Bartlett Learning, 2014.

[69] H. Li, B. Handsaker, and A. Wysoker *et al.* The sequence alignment/map format and SAMtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[70] G. Licciardello, A. Caruso, P. Bella, R. Lima, C.P. Strano, E.A. Trantas, P.F. Sarris, N.F. Almeida, and V. Catara. RNA-Seq-based transcriptome analysis of *Pseudomonas corrugata* Luxr regulators PCOR and RFIA. In *XXII Congress of the Italian Society for Plant Pathology*, 2016. Roma, Italy.

[71] G. Licciardello, A. Caruso, R. Lima, C.P. Strano, E. Trantas, P.F. Sarris, P. Bella, R. Lo-Piero, N.F. Almeida, and V. Catara. Transciptome-based analysis of two Luxr regulators involved in *Pseudomonas corrugata* interaction

with host and non-host plants. In *60th Annual Congress of The Italian Society of Agricultural Genetics*, 2016. Catania, Italy.

[72] R. Lima, C. Nishibe, T. Raiol, and N.F. Almeida. Differential expression and functional associations of HOX genes. In *X-Meeting 2015 - 11th International Conference of th AB3C + Brazilian Symposium of Bioinformatics*, 2015.

[73] M. I. Love, W. Huber, and S. Anders. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biol*, 15(12), 2014.

[74] A. V. Lukashin and M. Borodovsky. GeneMark.hmm: New solutions for gene finding. *Nucleic Acids Research*, 26(4):1107–1115, 1998.

[75] M. Martin. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.journal*, 17(1):10, 2011.

[76] Y. Matsuta, M. Ito, and Y. Tohsato. ECOH: An Enzyme Commission number predictor using mutual information and a support vector machine. *Bioinformatics*, 29(3):365–372, 2012.

[77] A. Mortazavi and B. A. Williams *et al.* Mapping and quantifying mammalian transcriptomes by RNA-seq. *Nat Meth*, 5(7):621–628, 2008.

[78] C. Nishibe. *Pipelines automatizados para genômica e transcritômica*. Tese de doutorado, Faculdade de Computação, Universidade Federal de Mato Grosso do Sul, 2015.

[79] O.Brandenberg, Z. Dhlamini, A. Sensi, K. Ghosh, and A. Sonnino. *Introduction to molecular biology and genetic engineering*. Food and Agriculture Organization of the United Nations, 2011.

[80] Association of Brazilian beef exporters ABIEC. Brazilian beef pofile, 2013. http://www.brazilianbeef.org.br/texto.asp?id=9.

[81] World Health Organization. Cancer - World Health Organization, 2014.

[82] A. Oshlack, M. D. Robinson, and M. D. Young. From RNA-seq reads to differential expression results. *Genome Biology*, 220, 2010.

[83] H. Pearson. Genetics: What is a gene? *Nature*, 441(7092):398–401, 2006.

[84] C. Pedamallu and J. Posfai. Open source tool for prediction of genome wide protein-protein interaction network based on ortholog information. *Source Code Biol Med*, 5(1):8, 2010.

[85] J. Pevsner. *Bioinformatics and Functional Genomics*. Wiley-Blackwell, 2nd edition, 2009.

[86] E. M. Phizicky and S. Fields. Protein-protein interactions: methods for detection and analysis. *Microbiological Reviews*, 59(1):94–123, 1995.

[87] S. B. Primrose and R. M. Twyman. *Principles of Gene Manipulation and Genomics*. Blackwell Publishing, 7th edition, 2006.

[88] A. R. Quinlan and I. M. Hall. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, 26(6):841–842, 2010.

[89] V. Ramakrishnan. Ribosome structure and the mechanism of translation. *Cell*, 108(4):557–572, 2002.

[90] M. Ridley. *Genome. The Autobiography of a Species In 23 Chapters*. New York, NY: Harper Perennial, 5th edition, 2006.

[91] M. D. Robinson, D. J. McCarthy, and G. K. Smyth. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–140, 2009.

[92] M.D. Robinson and A. Oshlack. A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biol*, 11(3):R25, 2010.

[93] P. Russell. *Comparative Genomics*. Brooks Cole, 3rd edition, 2013.

[94] L.R. Salgado, R. Lima, B.F. dos Santos, K.T. Shirakawa, M.A. Vilela, N.F. Almeida, R.M. Pereira, A.L. Nepomuceno, and L.Chiari. De novo RNA sequencing and analysis of the transcriptional landscape of signalgrass (*Urochloa decumbens) roots exposed to aluminum*. Submitted, 2016.

[95] S. L. Salzberg, A. L. Delcher, S. Kasif, and O. White. Microbial gene identification using interpolated Markov models. *Nucleic Acids Research*, 26(2):544–548, 1998.

[96] A. Schneider, C. Dessimoz, and G. H. Gonnet. OMA Browser Exploring orthologous relations across 352 complete genomes. *Bioinformatics*, 23(16):2180–2182, 2007.

[97] I. Schomburg. BRENDA, the enzyme database: updates and major new developments. *Nucleic Acids Research*, 32(90001):431D–433, 2004.

[98] J. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology*. Brooks/Cole, 1997.

[99] L. Stein. Genome annotation: from sequence to biology. *Nat Rev Genet*, 2(7):493–503, 2001.

[100] R. L. Tatusov. The COG database: a tool for genome-scale analysis of protein functions and evolution. *Nucleic Acids Research*, 28(1):33–36, 2000.

[101] R. L. Tatusov, E. V. Koonin, and D. J. Lipman. A Genomic Perspective on Protein Families. *Science*, 278(5338):631–637, 1997.

[102] The Gene Ontology Consortium. The Gene Ontology project in 2008. *Nucleic Acids Research*, 36:D440–D444, 2007.

[103] The Uniprot Consortium. The universal protein resource (UniProt) in 2010. *Nucleic Acids Research*, 38(Database):D142–D148, 2009.

[104] J. Touchman. *Comparative Genomics*. Nature Education Knowledge, 2010.

[105] E.A. Trantas, G. Licciardello, N.F. Almeida, K. Witek, C.P Strano, Z. Duxbury, F. Ververidis, D.E. Goumas, J.D. Jones, D. Guttman, V. Catara, and P.F Sarris. Comparative genomic analysis of multiple strains of two unusual plant pathogens: Pseudomonas corrugata and Pseudomonas mediterranea. *Frontiers in Microbiology*, 6(811), 2015.

[106] C. Trapnell and A. Roberts *et al.* Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. *Nature Protocols*, 7(3):562–578, 2012.

[107] C. Trapnell and B. A. Williams *et al.* Transcript assembly and quantification by RNA-seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nat Biotechnol*, 28(5):511–515, 2010.

[108] C. Trapnell and D. G. Hendrickson *et al.* Differential analysis of gene regulation at transcript resolution with RNA-seq. *Nat Biotechnol*, 31(1):46–53, 2012.

[109] C. Trapnell, L. Pachter, and S. L. Salzberg. TopHat: discovering splice junctions with RNA-seq. *Bioinformatics*, 25(9):1105–1111, 2009.

[110] D. Voet, J. Voet, and C. Pratt. *Fundamentals of Biochemistry* . John Wiley & Sons, 2002.

[111] C. von Mering *et al.* Comparative assessment of large-scale data sets of protein–protein interactions. *Nature*, pages 399–403, 2002.

[112] Z. Wang, M. Gerstein, and M. Snyder. RNA-seq: a revolutionary tool for transcritomics. *Nature Reviewes Genetics*, 10:57–63, 2009.

[113] C. H. Wu. The Universal Protein Resource (UniProt): an expanding universe of protein information. *Nucleic Acids Research*, 34(90001):D187–D191, 2006.

[114] X. Xia. *Comparative Genomics*. Springer Berlin Heidelberg, 2013.

[115] M. Yanagida. Functional proteomics; current achievements. *Journal of Chromatography B*, 771(1-2):89–106, 2002.

[116] A. Zaha, H. B. Ferreira, and L. M. P. Passaglia. *Biologia Molecular Básica*. Artmed Editora, 5 edition, 2014.

[117] A. Zhang. *Protein Interaction Networks-Computational Analysis*. Cambridge University Press, New York, NY, USA, 2009.

Apêndice A

Pipeline User's Guide

Overview of the Pipeline

This document describes a specific pipeline for differential gene and transcript expression analysis of RNA-seq experiments. Code is available at `https://git.facom.ufms.br/bioinfo/pipeline_segemehl_cufflinks`.

The input of the pipeline is the set of reads with replicates in different conditions and a configuration file defined by the user. The output is the list of differential expression genes and graphics that shows these genes.

There are four stages in the pipeline:

1. **Aligning reads to a reference genome:** Using a reference genome, reads are mapped to find the point of origin for every read. The output is an alignment file showing the mapped reads and their mapping positions in the genome.

2. **Building and union of transcripts:** Using the genome reference, annotation and the mapped reads the goal is assemble transcriptomes from RNA-Seq data and merge these assemblies into a single master transcriptome to perform a differential expression analysis. The output is a single transcripts file.

3. **Differential expression analysis:** Using the genome reference, the union of transcripts and all the mapped reads files, the goal is calculate expression levels and identify genes that are differentially regulated. The output is a set of text files containing gene and transcript expression level changes and additional reports.

4. **Visualization of results:** Using the output of differential expression analysis, the goal is simplify the visualization of results and generating some graphics to better understand the result.

Requirements

To use the pipeline, the list above must be installed. Listed in parenthesis are the test versions used of each software.

- `Linux`
- `R` (v. 3.3)
- `Perl` (v. 5.18.2)
- `segemehl` (v. 0.2.0-418)
- `SAMtools` (v. 0.1.19)
- `Cufflinks` (v. 2.2.1)
- `Cuffmerge` (v. 2.2.1)
- `Cuffdiff` (v. 2.2.1)
- `cummeRbund` (v. 2.6.1)

Installation

Although the pipeline has several scripts and configuration files, the only thing that user needs to do in order to install it is to unpack from the site mentioned earlier in this chapter and set the permissions for all scripts are executable. The following commands will unpack the downloaded package and assure the permissions, respectively:

```
$  unzip pipeline_rod.zip

$  chmod -R 777 pipeline_rod/*
```

Unpacking results in two directories:

- `scripts`: The directory with all scripts files of the pipeline.
  - `diff2depictviz.pl`
  - `exec_cummeRbund.R`
  - `exec_pipeline.pl`
  - `getCoordinatesCuffdiffOut.R`
  - `separate_genes.pl`
- `configs`: The directory with all configuration files of the pipeline.
  - `cuffdiff.conf`
  - `cufflinks.conf`
  - `cuffmerge.conf`
  - `pipeline.conf`
  - `segemehl.conf`

To facilitate the pipeline execution, the user may add the directory of scripts in his/her path environment variable.

Executing the Pipeline

The first step to use the pipeline is the configuration of the file `pipeline.conf`. In this file important information should be provided such as the path of reads, reference genome and its annotation, the path of the configuration files of other programs used by the pipeline, the path where will be written the results and which pipeline stages will be executed.

The reads inside the directory given in the `pipeline.conf` must have a specific name. The name of each file use the condition, the number of the replicate and the type of read (only for paired-end reads) separated by "_".

- `paired-end reads`:
  - `condition_replicate_R1.fasta`
  - `condition_replicate_R2.fasta`

- `single-end reads`:
  - `condition_replicate.fasta`

The replicate must be numbered consecutively from 1 to n, where n is the number of replicates. It is important to keep track of the name of files because this informations is used to generates the outputs during the pipeline.

The next step is to fill the configuration files. For each software there is a configuration file with the parameters that must be used during the analysis.

For all configuration files, if a parameter expects a value, they must be separated by space or tab. Parameters expectig to receive paths (reads, genome or annotation) should omit this information, once already exists in the pipeline configuration file. No need to configure the files of the programs that will not be executed.

After specifying the parameters to be used in the analysis, just use the next command to execute the pipeline:

```
$  nohup ./exec_pipeline.pl ../configs/pipeline.conf > log.txt
```

This command will execute the entire pipeline automatically, but it is possible to execute manually steps of pipeline, or redo some steps just changing the values in the configuration file.

Steps of the Pipeline

This section describes each step of the pipeline. All the steps are executed by the `exec_pipeline.pl`. To run a specific step or a few steps, you need to set what you want in `pipeline.conf`.

Each step was implemented to receive a specific type of file.

Aligning Reads to a Reference Genome

The pipeline uses `segemehl` to align the reads. There is an option to create the index of your genome with `segemehl` by assigning the `build_index` information in configuration file to 0 (no execute) or 1 (execute). The search or creating of the genome index will be run in the path of the genome informed in configuration file. In order to know what kind of reads will be looked for (paired or single), there is an option named `paired` to be set to 0 or 1. The reads that will be align must have format `*.fasta`, `*.fa`, `*.fastq` or `*.fq`, no matter if they are single or paired-end.

The output of the alignment will be written in a directory inside the `path_project` defined in the configuration file and named `mapped_reads`. Each file will have the SAM format and his name will preserve the condition name and the number of the replicate (`condition_replicate.sam`).

After the aligment, the pipeline will sort the alignment and convert to the BAM format if the option `sort` was set to 1 in the configuration file. To execute the next step in the pipeline, the aligment must be sorted. The sorted files will be created in the same directory of `mapped_reads` and his name will be the same as before with the suffix "_sorted" (`condition_replicate_sorted.bam`).

Building and union of transcripts

The building and union of transcripts uses `Cufflinks` and `Cuffmerge`, respectively. To build the transcripts of each sample of reads `Cufflinks` uses the sorted alignment file found in `mapped_reads` directory. The transcripts built are storage in a different directory inside the `path_project` named `Cufflinks_out`. Each sample have your own directory with your transcript built with the name format `condition_replicate_Cufflinks`. The `Cufflinks` produces the following files:

- `genes.fpkm_tracking`: This file contains the estimated gene-level expression values in FPKM.
- `isoforms.fpkm_tracking`: This file contains the estimated isoform-level expression values in FPKM.

- `skipped.gtf`: This file contains the skipped loci if the parameter `max-bundle-frags` is used.
- `transcripts.gtf`: This GTF file contains `Cufflinks` assembled isoforms.

With all the transcripts built, `Cuffmerge` search for all `transcripts.gtf` across the directory and merge them in a single assembly file. The results are storage in the directory `Cuffmerge_out` inside the `path_project`. The `Cuffmerge` produces a log directory and the merged assemblies (`merged.gtf`) .

Differential expression analysis

The differential expression analysis to identify genes expressed in different quantities in distinct groups of samples we use `Cuffdiff`. The `Cuffdiff` take as input the merged assemblies in the last step, the labels of conditions (parameter -L) set in the `cuffdiff.conf` file separated by a coma (Ex: -L Cond1,Cond2) and a list with the sorted alignments of samples respecting the order of the labels (separating same conditions samples by coma and different conditions by space). It is very important to use the same names as the conditions presents in the samples of reads for the labels.

The output are written in a directory named `cuffdiff_out` in the `path_project`. The `Cuffdiff` produces the following files:

- `*.fpkm_tracking`: Calculates the FPKM of each transcript, primary transcript, and gene in each sample.
- `*.count_tracking`: Estimates the number of fragments that originated from each transcript, primary transcript, and gene in each sample.
- `*.read_count_tracking`: Calculates the expression and fragment count for each transcript, primary transcript, and gene in each replicate.
- `*.diff`: Tab delimited files lists the results of differential expression testing between samples for spliced transcripts, primary transcripts, genes, and coding sequences, isoform switching detected among its isoforms and the amount of overloading detected among its primary transcripts.

Visualization of results

The visualization step is important because it helps to understand the results of the differential expression analysis. The visualization can be done with `cummeRbund` and `DEPICTViz`. To visualize with `cummeRbund` we use the script `exec_cummeRbund.R` inside the `cuffdiff_out` directory, taking as input the name of conditions used as label in the step above. The script was developed in `R` and produces the following graphics:

- `MAplot_all_genes.pdf`: Graphic to examine intensity vs fold-change plots.
- `MDS.pdf`: Graphic to examine the level of similarity of individual cases of a dataset.
- `volcano.pdf`: The volcano plot is a useful visualization to compare fold change between any two conditions and significance (-log P-values).
- `dendrograma.pdf`: Dendrogram can provide insight into the relationships between conditions.

The script can be executed by the command:

```
$ Rscript exec_cummeRbund.R Condition1 Condition2
```

The visualization with DEPICTViz, the web tool proposed in this paper, can be done using three scripts: `getCoordinatesCuffdiffOut.R`, `diff2depictviz.pl` and `separate_genes.pl`. These scripts can generate a compatible input file for `DEPICTViz`. The first step is to extract the information from `Cuffdiff` output with the script `getCoordinatesCuffdiffOut.R`. The script must be executed inside the `cuffdiff_out` directory passing the name of conditions used as label in the differential expression analysis. The output is a file named `coordenadas.csv` with the coordinates of all genes.

The script can be executed by the command:

```
$ Rscript getCoordinatesCuffdiffOut.R Condition1 Condition2
```

Once with the coordinates file, we use the script `diff2depictviz.pl` to clean low coverage entries and set the correct headers to DEPICTViz read as a valid input

file. The scripts takes as input the `coordenadas.csv` and the result is a file named `in_depictviz.csv`. The script can be executed by the command:

```
$ ./diff2depictviz.pl coordenadas.csv
```

Although `in_depictviz.csv` file already is a valid input file for the tool, it can contain many genes and overload the tool, where it was made to handle small sets of genes. To work around this problem, there is one last script to extract a subset of genes of interest. The `separate_genes.pl` script takes as a parameter the `in_depictviz.csv` file and a list of names of the genes of interest. You can still filter the genes by using the p-value `-de` parameter with a value. The script can be executed by the following command, where the parameters `-o` and `-de` are optional:

```
$ ./separate_genes.pl -i in_depictviz.csv -o output_file -de 0.05
gene_name1 gene_name2...
```

Example

Suppose that all steps will be executed in the folder `/home/user/project/`. You have two conditions (A and B) and each one has duplicates. The reads are paired-end and and they are stored in `/home/user/reads/`, so your input data is:

- `A_1_R1.fastq` and `A_1_R2.fastq`
- `B_1_R1.fastq` and `B_1_R2.fastq`
- `A_2_R1.fastq` and `A_2_R2.fastq`
- `B_2_R1.fastq` and `B_2_R2.fastq`

The genome used to map the reads is `genome.fa` and it is stored in `/home/user/genomes/` and all configuration files used in the analysis are in `configs` folder. In this example we will use all steps of the pipeline, so we need to set several files.

First we need to set the `pipeline.conf` the path to genome, annotation, reads, configuration files, outputs and softwares used. In this example below, the execution builds the genome index, creates the directories and execute the entire pipeline. If you don't want to execute some step, just set 0 to the value.

```
path_project /home/user/project/
path_genome /home/user/genomes/genome.fa
path_annotation /home/user/genomes/genome.gtf
path_reads /home/user/reads/
path_segemehl_conf ../configs/segemehl.conf
path_cufflinks_conf ../configs/cufflinks.conf
path_cuffmerge_conf ../configs/cuffmerge.conf
path_cuffdiff_conf ../configs/cuffdiff.conf
build_index 1
paired 1
segemehl 1
cufflinks 1
cuffmerge 1
cuffdiff 1
sort 1
```

In `segemehl.conf`, we set the number of threads (`-t 8`) and others optional parameters like `-S` (detect split/spliced reads) and `-s` (silent to reduce the log messages).

```
-t 8
-S
-s
```

In `cufflinks.conf`, we tell to `Cufflinks` use 8 threads (`-p 8`), use an annotation to guide assembly (`-g`) and no print unnecessary log (`-q`). Note that we do not pass the path of annotation by parameter, as this information is already on file `pipeline.conf`.

```
-p 8
-g
-q
```

In `cuffmerge.conf`, we tell to `Cuffmerge` to use 8 threads (`-p 8`), a reference annotation (`-g`) and the genome file for reference (`-s`).

```
-p 8
-g
-s
```

In `cuffdiff.conf`, we use 8 threads (`-p 8`), the genome for bias correction (`-b`), the Labels (`-L`) of conditions (A and B) and no print unnecessary log (`-q`).

```
-p 8
-b
-L A,B
-q
```

The results of each step will be written in `/home/user/project/` directory.

Apêndice B

DEPICTViz User's Guide

Overview of the DEPICTViz

This document describes an interactive web tool showing a MA Plot, containing all genes of interest given by the user. All genes in the interface are clickable, linked to databases of protein interactions and gene ontology, like STRING and COG, allowing further GO functional enrichment. Code is available at `https://git.facom.ufms.br/bioinfo/depictviz`.

The input of the tool is an input file (tsv format) containing information about several genes of interest. The output is an interactive MA Plot loaded in the web browser.

Requirements

The web tool is written in `Javascript` and should work on any OS that supports a modern web browser and have it installed and running a web server, but our tool was specifically designed to run with a Linux OS and has been successfully tested on the following three platforms:

- Debian 8.2

- Ubuntu 14.04.3 LTS
- Linux Mint 17.2+

The tool also requires some third party software to run successfully (web browser and server). Listed in parenthesis are the versions used to test the tool:

- Google Chrome (v. 47.0.2526.80)
- Firefox (v. 42.0)
- Apache2 (v. 2.4.7)

There are several APIs used by the tool that is already present and integrated in the package. The list of these and his versions are:

- `Highcharts` (v. 4.1.9)
- `Bootstrap` (v. 3.3.5)
- `Bootstrap-select` (v. 1.6.4)
- `jQuery` (v. 2.1.4)
- `jQuery-Autocomplete` (v. 1.2.24)
- `jQuery-tagEditor` (v. 1.0.19)
- `jQuery Caret` (v. 1.3.3)
- `Nprogress` (v. 0.2.0)
- `Toastr` (v. 2.1.2)
- `SweetAlert` (v. 1.0.0)

Installation and Configuration

After downloading the package, the only thing the user needs to do in order to install it is to unpack and to move the content to the document root from web server or any other desirable subdirectory. In case of using apache2 on Linux, the document root should be in `/var/www/html/`.

The following command will unpack the downloaded package:

```
$  unzip depictviz.zip
```

The result of unpack are stored in two directories:

- `input`: This directory should keep the input files.
- `data`: This directory contains all the APIs, source code and support files.

There is a HTML (`index.html`) file in the main directory in which the tool is loaded in the browser.

After moving it, the next step is configure the tool for your own experiment. The input file should be copy to `input` directory. The file must be in tab-separated value (tsv) format and the header should be the same (no case-sensitive) as described below, as well as whether it is required or not.

Description of fields that must be present in the input file, in addition to its requirement to build the chart.

Header Name	**Description**	**Required**
ID	Gene identification (ID or name).	Yes
X	Coordinate value of the x axis. Is the average count of the gene in the experiment.	Yes
Y	Coordinate value of the y axis. It represents the Log_2FC between the conditions.	Yes
Pvalue	Value of p-value. Genes with p-value < 0.05 are classified as dif. expressed.	Yes
COG_ID	COG identifier for a gene.	No
COG_Description	COG function description.	No
COG_Category	Category each COG belongs.	No
Swiss_ID	Swiss-Prot identifier associated with the gene.	No
KO_ID	KEGG Orthology Identifier.	No
KEGG_Gene	Identifier gene in KEGG. In the absence of this is used the ID field.	No
EC	EC number associated with the gene.	No
GO_Process	GO identifier for the biological process ontology.	No
GO_P_Description	Description of function performed by the GO biological process ontology.	No
GO_Function	GO identifier for the molecular function ontology.	No
GO_F_Description	Description of function performed by the GO molecular function ontology.	No
GO_Component	GO identifier for the cellular component ontology.	No
GO_C_Description	Description of function performed by the GO cellular component ontology.	No
Start	*Loci* start position in the genome.	No
End	*Loci* end position in the genome.	No
Strand	DNA sense strand.	No
Gene_Product	Gene product.	No
Contig	Gene contig in the genome.	No

The order of the fields does not matter. The tool will not work if at least one of the required fields is not present. A small example input with the required fields can be found in `input` folder and works with all organisms. To represent the lack of information when a field is used, you should use "--" instead of leaving it empty.

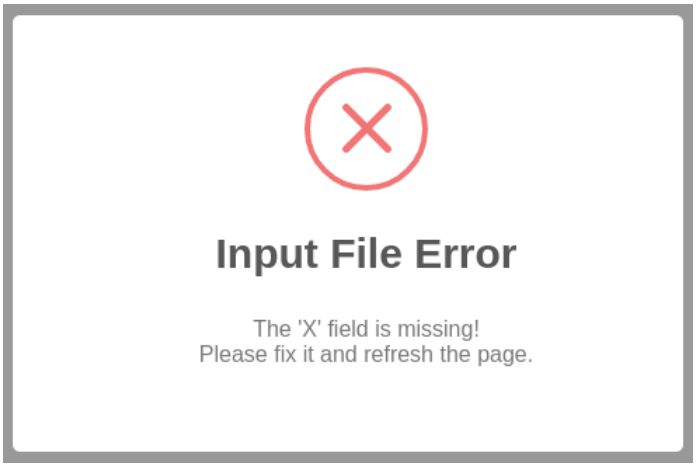
The GOs fields and descriptions, COG tags and descriptions, EC numbers, Swiss-Prot, KO and KEGG gene accept a list of values for each entry, where they shall be separated by commas. To avoid GO identifiers representation problems, three different forms of GO identifiers are accepted. For example, for 'GO: 0007018', the information can be represented in three different ways: GO: 0007018, 0007018 and 7018. Other examples of input files are available in the `input` directory.

The last step consists in editing `init.js` file inside `data/js/` directory. There you should set the input file name present in `input` directory to load the data, subtitle of your chart, name of the conditions, set the the p-value threshold used to classify differentially expressed genes and edit the description information of your chart if necessary.

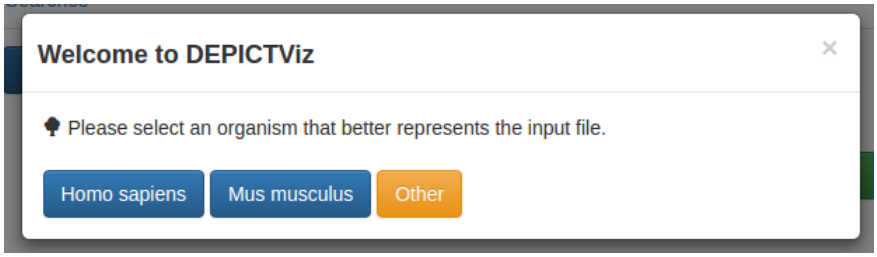
Running the Web Tool

After the step of installation and configuration, to run the web tool on your server you just need to access the path through the web browser. If there is any error related to the input file, either for lack of a required field for the construction of chart or incoherent values, an error message will be displayed to the user informing the type of error. After correction, just refresh the page again. If the error persists even after the correction, try clearing your browser cache.

If there is no problem with the input file, a window opens with the option to select the organism that best represents the genes present in the input file. Among the organisms are available *Homo sapiens* and *Mus musculus*. For other organizations exists the option `Other`. The selection of organisms aims to make the correct relationship between the data, searches and links to other databases, thereby allowing a more accurate search. If the user wants to reselect the organism, the `Reselect organism` button below the chart was added for this purpose.



Error alert when there is something wrong with the input file.



Organism selection window.

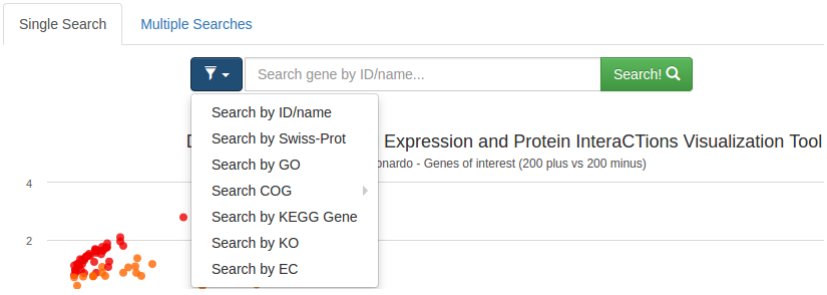
After selecting the organism, the tool is completely loaded and it can be divided into three areas: search, plot and description.

Search System

The search area is above the chart, consisting of a filter selector, a text field for search and a button to perform the search. By clicking on the filter, a selector is displayed giving the user the possibility to choose on which field you want to perform the search. You can only select a field that is present in the input file. By default the search is made by the gene ID.

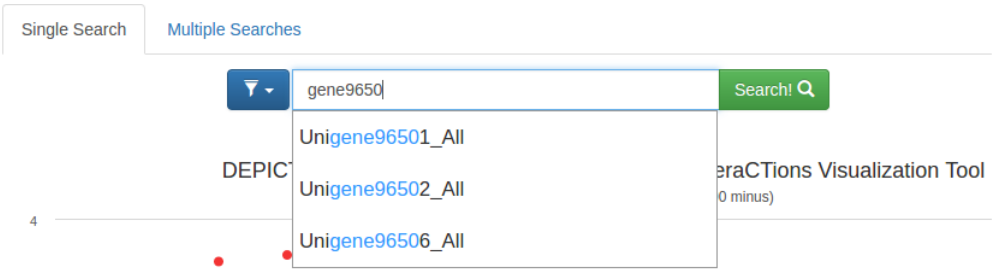
The search string should be inserted into text field which is located next to the filter. As the string is entered, suggestions are presented as autocomplete.

Tool completely loaded and separated into three areas: search (1), chart (2) and description (3).



Filter selector allows you to choose the field in which you want to perform the search.

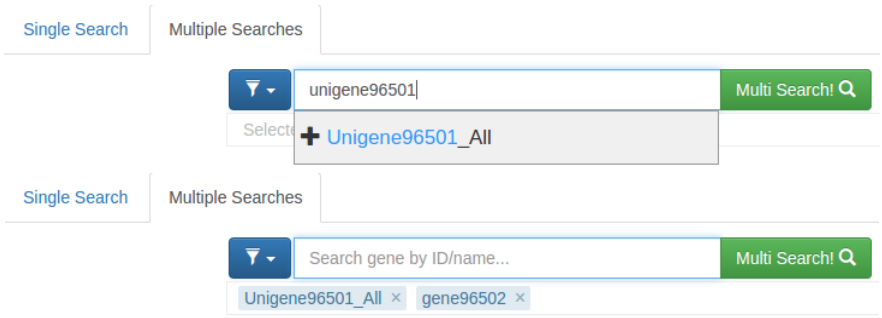
When you run the search, the gene found by the search will be selected in the chart area below. Otherwise, a message is displayed in the upper right corner of the screen indicating that results were not found.



The text field has Autocomplete that allows show off suggestions to the user.

As the single search, there is the possibility of multiple searches. The type of search can be toggled by selecting the tabs above the filter selection button. Multiple search runs in a slightly different manner. When you enter the search string, the user must click on a suggestion or press the `Enter` key to search the desired string or substring.

As the strings are searched and found in the chart, they are also stored in a tag field below the text field. That way, you can remove one or more strings without the need to redo all the searches by simply clicking the close icon of each tag (x).



Selection of search strings for the multiple search. The searched strings are stored in a tag field below the text field.

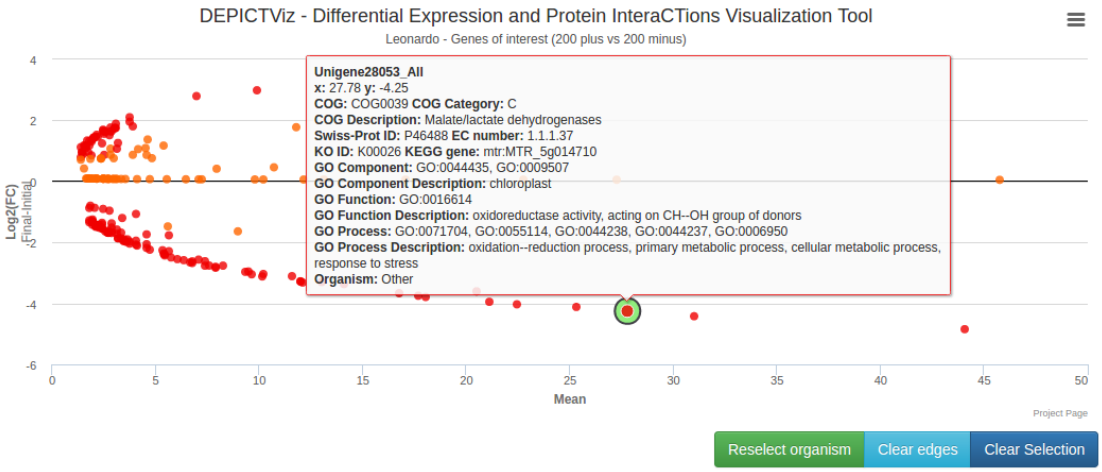
`Multi Search` button has a different action of its corresponding of the Single Search. When clicked, a window is displayed to the user that offers the option to redirect it to the STRING, where the interactions of genes and proteins belonging to the points that have been selected in multiple search are shown. This action is only valid for genes ID and Swiss-Prot searches, so you need to search for them through the filter.

Window open after the user performs multiple searches of Swiss-Prot identifiers and click on the button `Multi Search`. The action by clicking on the `Multi Search STRING` button takes you to the STRING in order to show off the interaction of genes and proteins related to the points that have been selected in the search.

Chart

The most important area of the tool is the chart area, due to the fact this show off all possible information. Since the graph is interactive, we can interact with any point present on it. When performing a search looking for a point or clicking on it, the point is selected and remains detached from others. To clear this selection, the user can click in the `Clear Selection` button below the chart.

Hovering the mouse over a point, you can view a tooltip that shows quickly the information present at that point.



Tooltip shown when the mouse hover a point.

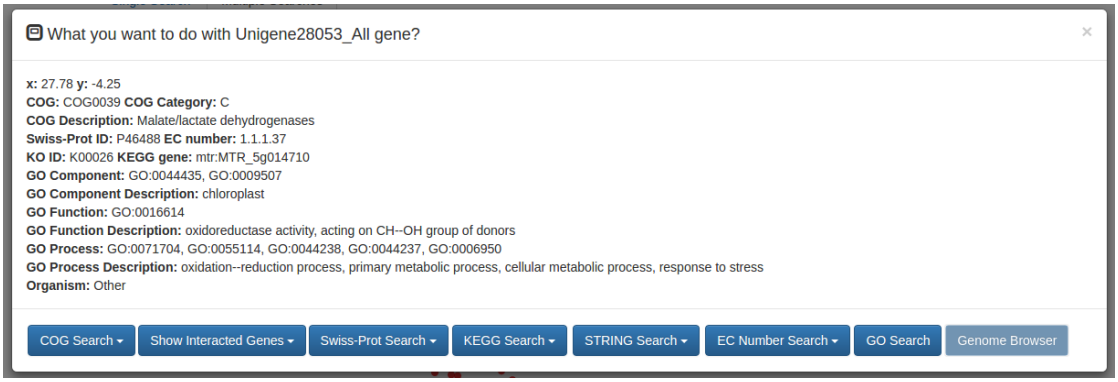
If the number of points on the chart is large or they are many close together, you can `zoom` in on a desired area. For this, the user must within the chart area click

with your mouse and drag it to create a selection area. Through these steps, the zoom will be applied in the selected area. To return to the original zoom, there is a button in the upper right corner of the chart area called `Reset zoom`.

Using zoom in (1) by creating a selection area with the mouse and zoom out (2) by clicking the Zoom Reset button in the upper right corner of the chart.

By clicking a point, a window opens showing all the information of the point and there are buttons to perform different actions. The availability of each option of the buttons depends exclusively of the provided input file. The first button, `COG Search`, aims to redirect the user to the pages of NCBI and EggNOG using a COG identifier, if present in the selected gene.

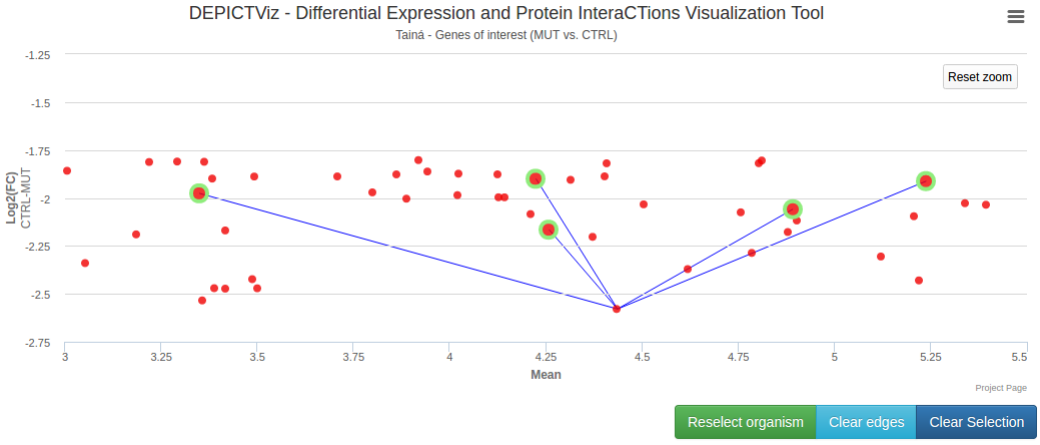
The second button, `Show Interacted Genes`, shows the interactions of the selected gene with others present. The genes that have interaction are connected with edges to each other. There are two possible types of interactions: Interaction between



Window open by clicking a point on the chart area.

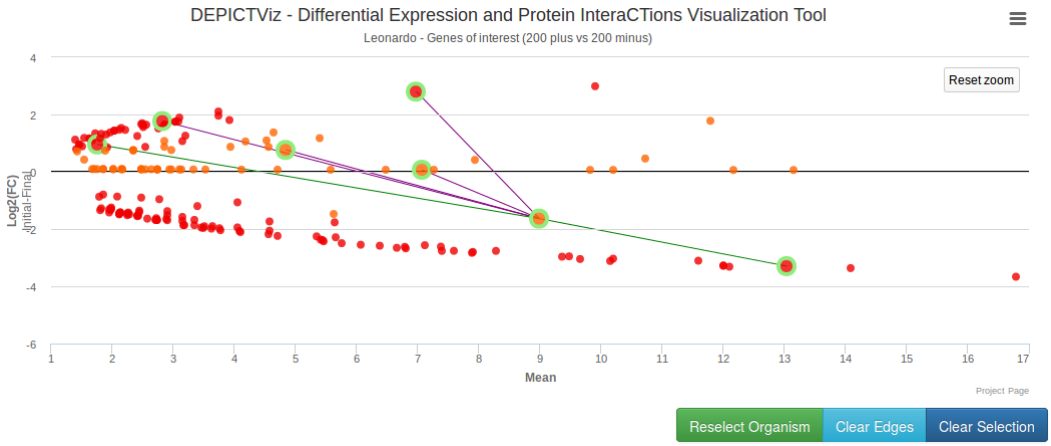
genes according to the STRING and genes that have the same category of the COG.

The interaction between genes according to the STRING is shown by blue edges between the points. The interaction of genes that share the same COG categories are made by purple edges, where the genes have exactly the same set of COG categories and green edges for genes that have at least one category in common. To clean the edges in the chart, the user can use the `Clear` button below the chart.



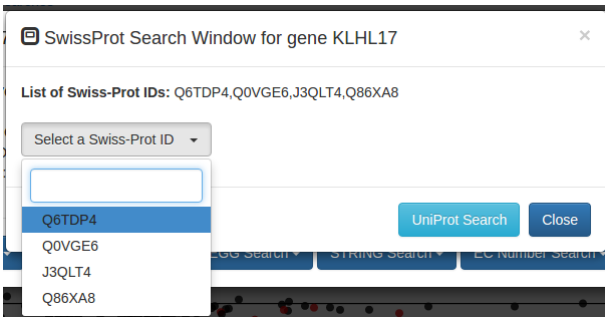
Edges in blue showing the interaction of the selected point with the others, according to STRING.

The `Swiss-Prot Search` button allows you to search in the online protein databases: UniProt, OMA Browser and SWISS-MODEL; using the Swiss-Prot identifiers of the gene. Likewise, `KEGG Search` and `STRING Search` buttons allows you to search for KO indentifiers and gene ID in the KEGG database and Swiss-Prot and gene ID in the STRING, respectively. How is possible have more than one Swiss-Prot identifier



Edges in purple showing the genes that have exactly the same category COG with selected gene, and edges in green for genes that share at least one COG category with the selected gene.

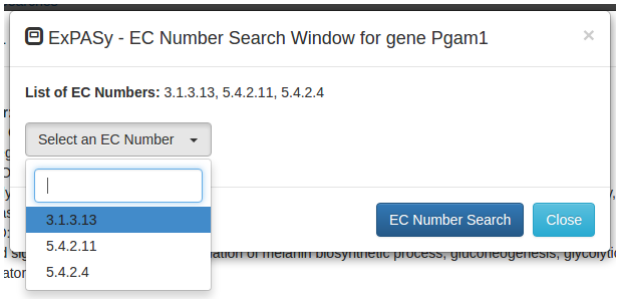
for a gene, when the user clicks a gene that has a list of Swiss-Prot identifiers, a window appears where the user choose which identifier would like to search.



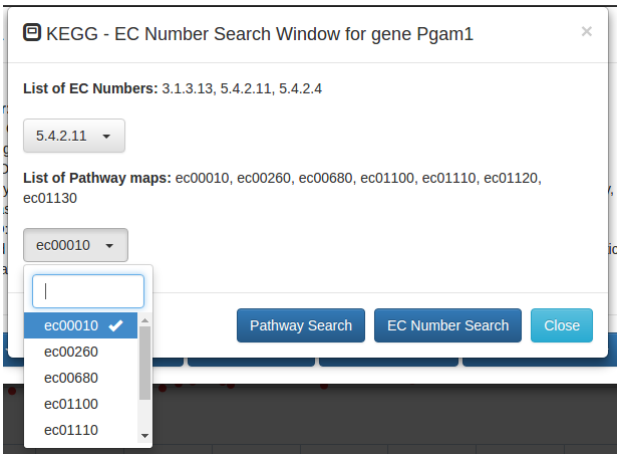
Window open to search for a Swiss-Prot identifier in the UniProt database.

One of the last options that have been developed and that brings more features is the `EC Number Search`. Through it is possible to search on several specialized databases of enzymes, which are: ExPASy, ExplorEnz, BRENDA Enzymes, MetaCyc and KEGG. When searching for an EC number in KEGG, it is also possible to select a biological `pathway` for this, if any.

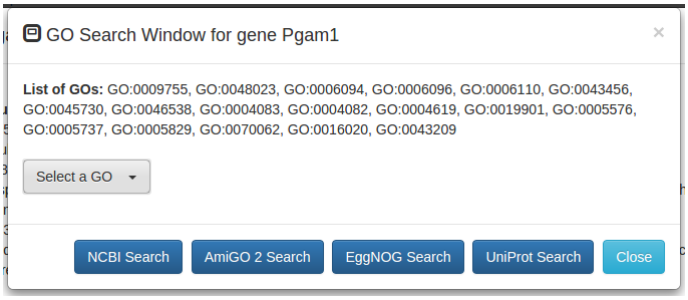
Similar to `EC Number Search`, `GO Search` allows searches for a GO in the databases: NCBI, friend, eggnog and UniProt. Likewise to search for an EC number, a new window opens where the user chooses a GO from the GOs list attributed to the gene.



Window open to search for an EC number in ExPASy database.



Window open to search for an EC number in the KEGG database. In addition to finding information of enzymes, you can do a search for biological `pathways` that are related to selected EC number.



Opened window to search for a GO. After selecting one of the listed GOs, the user is left to choose which database you want to search.

Finally, the last button that is available is the `Genome Browser`. In order for it is available, the user must have selected an organism (*Homo sapiens* or *Mus musculus*). Through it the user is redirected to the UCSC Genome Browser, where it can analyze the gene with a higher level of detail and create your own custom track.

Description

The last area of the tool composes a description or information that can be edited by the user. The text can be changed through the configuration file (`init.js`) mentioned above. By default, the text assigned the description briefly describes the graphic in general (coordinates, color points, chart interpretation).