UNIVERSIDADE FEDERAL DE UBERLÂNDIA
INSTITUTO DE BIOTECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM BIOTECNOLOGIA

VINICIUS SOARES SILVA MARQUES

ENGENHARIA DE SOFTWARE APLICADA A SOFTWARES BIOLÓGICOS: UM ESTUDO DE CASO EM DOCUMENTAÇÃO DE SOFTWARE BIOINFORMÁTICO

PATOS DE MINAS – MG
DEZEMBRO DE 2021

VINICIUS SOARES SILVA MARQUES

ENGENHARIA DE SOFTWARE APLICADA A SOFTWARES BIOLÓGICOS: UM ESTUDO DE CASO EM DOCUMENTAÇÃO DE SOFTWARE BIOINFORMÁTICO

Dissertação de mestrado apresentada ao Programa de Pós-graduação em Biotecnologia como requisito parcial para obtenção do título de Mestre em Biotecnologia.

Orientador: Prof. Dr. Laurence Rodrigues do Amaral

PATOS DE MINAS – MG
DEZEMBRO DE 2021

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

M357
2021

Marques, Vinicius Soares Silva, 1983-
Engenharia de Software aplicada a softwares biológicos [recurso eletrônico] : Um estudo de caso em documentação de software bioinformático / Vinicius Soares Silva Marques. - 2021.

Orientador: Laurence Rodrigues do Amaral.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Biotecnologia.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2022.19
Inclui bibliografia.
Inclui ilustrações.

1. Biotecnologia. I. Amaral, Laurence Rodrigues do, 1978-, (Orient.). II. Universidade Federal de Uberlândia. Pós-graduação em Biotecnologia. III. Título.

CDU: 60

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:

Gizele Cristine Nunes do Couto - CRB6/2091

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Biotecnologia
Av. Getúlio Vargas, 230, 3º andar, Sala 308 - Bairro Centro, Patos de Minas-MG, CEP 38700-128
Telefone: (34) 3823-3714 - Ramal 39 - www.ppgbiotec.ibtec.ufu.br - ppgbiotec@ibtec.ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Biotecnologia				
Defesa de:	Dissertação de Mestrado Acadêmico				
Data:	20 de dezembro de 2021	Hora de início:	14:00	Hora de encerramento:	15:40
Matrícula do Discente:	41912BTC012				
Nome do Discente:	Vinicius Soares Silva Marques				
Título do Trabalho:	ENGENHARIA DE SOFTWARE APLICADA A SOFTWARES BIOLÓGICOS: UM ESTUDO DE CASO EM DOCUMENTAÇÃO DE SOFTWARE BIOINFORMÁTICO				
Área de concentração:	Biociência				
Linha de pesquisa:	Bioinformática e Biologia Molecular aplicada à genômica, transcriptômica e proteômica				
Projeto de Pesquisa de vinculação:					

Reuniu-se por Webconferência, Campus Patos de Minas, da Universidade Federal de Uberlândia, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Biotecnologia, assim composta: Professores Doutores: Joslaine Cristina Jeske de Freitas (UFJ), Pedro Luiz Lima Bertarini (UFU) e Laurence Rodrigues do Amaral, orientador do discente.

Iniciando os trabalhos o presidente da mesa, Dr. Laurence Rodrigues do Amaral apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao discente a palavra para a exposição do seu trabalho. A duração da apresentação do discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor(a) presidente concedeu a palavra, pela ordem sucessivamente, aos(às) examinadores(as), que passaram a arguir o(a) candidato(a). Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o(a) candidato(a):

Aprovado.

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre .

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Laurence Rodrigues do Amaral**, **Professor(a) do Magistério Superior**, em 20/12/2021, às 15:27, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Pedro Luiz Lima Bertarini**, **Professor(a) do Magistério Superior**, em 20/12/2021, às 15:27, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Joslaine Cristina Jeske de Freitas**, **Usuário Externo**, em 20/12/2021, às 15:28, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3269930** e o código CRC **B12E9CB9**.

Referência: Processo nº 23117.088265/2021-52 SEI nº 3269930

Dedico esta dissertação à minha esposa, Eliza, e meus filhos, Frederico e Liz, pelo apoio, paciência e incentivo em todos os momentos.

AGRADECIMENTOS

A Deus, pelo dom da vida e do aprendizado.

À minha esposa, Eliza, por ter me incentivado desde a inscrição no Programa de Pós-Graduação, até a defesa, estando ao meu lado em todos os momentos e não me deixando desistir frente às inúmeras dificuldades pessoais, profissionais e acadêmicas.

Aos meus filhos, Frederico e Liz, por serem o motivo pelo qual busco crescimento diário em todos os apectos da minha vida, e por serem fonte de amor, paz, alegria e energia.

Aos meus pais, Renildo e Florisbet, meu irmão Virgílio e sua esposa Isabela, minha tia Bernadete, meus sogros Lázaro (*in memorian*) e Nadir, e demais familiares que foram apoio durante esta caminhada, de variadas formas.

Aos colegas de mestrado, por compartilhar das angústias e realizações durante este percurso.

Ao Prof. Dr. Laurence Rodrigues do Amaral, pela confiança, orientação, paciência, compreensão e dedicação.

A todo o corpo docente do Instituto de Biotecnologia, por todo o conhecimento repassado e pela paciência.

À Universidade Federal de Uberlândia, por ter proporcionado condições de realização segura dos trabalhos e aulas em meio à catástrofe sanitária que assolou o planeta.

Aos pesquisadores e cientistas de todas as nacionalidades, que contribuíram com o mundo durante a pandemia de Covid-19, combatendo o vírus e a desinformação.

A todos os que apoiaram de alguma forma este trabalho e torceram pelo seu melhor desfecho.

RESUMO

A Engenharia de *Software* é a área do conhecimento que trata da construção de sistemas computacionais. Uma de suas disciplinas trata da documentação do sistema, imprescindível para o bom desenvolvimento do projeto, a correta compreensão do funcionamento do *software*, sua manutenção e orientação ao usuário. *Softwares* biológicos são sistemas computacionais empregados em diversas atividades ligadas à biologia e biotecnologia, utilizados principalmente por pesquisadores das áreas biológicas. Documentar eficientemente um *software* biológico pode facilitar sua utilização por equipes interdisciplinares. Estima-se que a Engenharia de *Software*, aliada a um trabalho de exploração do código-fonte, pode ser utilizada como suporte para a documentação de um *software* de bioinformática. Este trabalho propõe identificar os artefatos mais úteis para o usuário de *softwares* biológicos, por meio de um estudo de caso em que se realizou a documentação de um sistema bioinformático voltado à identificação de microRNAs em um dado genoma. Os documentos gerados auxiliarão os usuários finais na utilização do sistema analisado, e fornecerão subsídios a outros pesquisadores que desejem ou precisem realizar sua manutenção.

Palavras-chave: engenharia de *software*, projetos de *software*, documentação de *software*, *softwares* biológicos, bioinformática

ABSTRACT

Software Engineering is the knowledge field that deals with the construction of computacional systems. One of its disciplines deals with system documentation, essential for the proper development of the project, the correct understanding of the software's operation, its maintenance and user orientation. Biological software are computacional systems used in various activities related to biology and biotechnology, mainly used by researchers in biological areas. Efficiently documenting biological software can ease its use by interdisciplinary teams. It is assumed that Software Engineering, combined with an exploratory effort over the source code, may serve as support for developing bioinformatics software documentation. This work proposes to identify the most useful artifacts for the user of biological software, through a case study in which the documentation of a bioinformatics system aimed at identifying microRNAs in a given genome was carried out. The documents generated will help end users in using the analyzed system, and will provide means to other researchers who wish or need to perform its maintenance.

Palavras-chave: software engineering, software project, software documentation, biological software, bioinformatics

LISTA DE ABREVIATURAS E SIGLAS

AGO – Argonauta

cDNA – DNA complementar

DNA – *Deoxyribonucleic Acid*

DSDM – *Dynamic System Development Model*

EST – *Expressed Sequence Tags*

HP-GAS – *prediction of Human Protein based on Genetic Algorithm driven Stacking method*

MFE – *minimum free energy*

miRISC – *miRNA-induced silencing complex*

miRNA – microRNA

mRNA – RNA mensageiro

OODA – Observar, Orientar, Decidir, Agir

RF – *Random Forests*

RNA - *Ribonucleic Acid*

SVN – *Support Vector Machine*

UML – *Unified Modelling Language*

SUMÁRIO

INTRODUÇÃO

Softwares biológicos ou bioinformáticos são sistemas computacionais utilizados em diversas aplicações em áreas como Biologia e Biotecnologia (Jawdat, 2006). Esses *softwares* são geralmente utilizados por equipes interdisciplinares, que podem envolver pessoas de diversas formações, como Biologia, Bioquímica, Genética, Farmácia, Medicina e Computação (Kumar; Chordia, 2017). Muitas vezes, porém, a maior parte da equipe – ou toda ela – é formada por profissionais de áreas não afins à Engenharia de *Software*, que é a área do conhecimento que trata da construção de sistemas computacionais (Kumar; Dudley, 2017). Profissionais de áreas diferentes muitas vezes usam terminologias semelhantes para se referir a coisas distintas, ou com sentido mais amplo ou mais restrito (Chen, 2012).

Nesse cenário, torna-se imprescindível que os *softwares* biológicos possam ser melhor compreendidos, não em nível de linguagem de programação, mas de funcionamento geral, e corretamente utilizados por profissionais das mais diversas formações. Questiona-se, portanto, como se pode tornar um desses sistemas mais compreensível tendo em vista a visão do usuário de bioinformática.

Uma das disciplinas da Engenharia de *Software* que pode auxiliar a solucionar de forma eficaz o problema apresentado é a Documentação de *Software*. Englobando a produção de artefatos que orientam desde o bom desenvolvimento do projeto, passando pela correta compreensão do funcionamento do *software* e sua manutenção, até a capacitação do usuário, essa disciplina possibilita diversas visões voltadas a cada tipo envolvido em um projeto de *software*.

O objetivo deste estudo foi identificar artefatos úteis ao usuário do *software* biológico *Biopipeline* (Gomes *et al.*, 2011), que sirvam de base para a construção da documentação do sistema, que por sua vez possa ser utilizada por profissionais de diversas áreas. Para isso, foi necessário entender o funcionamento do *software* estudado, por meio da análise do código fonte, o que levou à geração de artefatos de documentação que possibilitaram a compreensão do *software*, e, por fim, à geração de documentação voltada para o usuário final.

Dada a importância dos *softwares* biológicos para a pesquisa em Biologia, Biotecnologia e áreas afins, gerar uma documentação voltada à sua compreensão e correta utilização é de grande valia. Como as equipes envolvidas nessas pesquisas

costumam ser interdisciplinares, e nem sempre contam com profissionais versados em computação ou mesmo bioinformática, torna-se muito importante que os sistemas computacionais voltados a essas pesquisas sejam bem documentados, tendo em vista sua correta utilização pelos pesquisadores.

A identificação dos artefatos úteis ao usuário de um *software* bioinformático específico, e a confecção dessa documentação trarão benefícios à comunidade científica na medida em que colaboram para uma maior aproximação entre a Engenharia de *Software* e a Biotecnologia, com a intenção de disseminar boas práticas de documentação de *software* voltada ao usuário final.

REFERENCIAL TEÓRICO

Documentação de Software

Documentação de *software* é uma das atividades mais importantes da Engenharia de *Software* (Kipyegen; Korir, 2013), embora também seja a mais negligenciada (Parnas, 2011). Seja por restrições de tempo ou de custos, muitas vezes é produzida uma documentação pobre ou incompleta (Lethbridge *et al.*, 2003). Porém, Parnas (2011) aponta que, tanto do ponto de vista da manutenção do *software*, quanto sob a ótica do usuário, a documentação fornece informações extremamente valiosas. Ela possibilita desde uma visão geral sobre o sistema, até um detalhamento técnico sobre seu funcionamento, abrangendo todo o público do *software* (desenvolvedores, mantenedores, usuário final e comunidade em geral).

Há vários artefatos que podem ser produzidos em cada fase do desenvolvimento na atividade de documentar um *software*. Um estudo de 2006 (De Souza *et al*.) procurou identificar aqueles considerados mais importantes para os próprios desenvolvedores considerando a manutenção do *software*, sob a ótica de dois paradigmas de desenvolvimento: análise estruturada (mais tradicional) e processo unificado (mais moderno, utilizado em projetos orientados a objeto). Os artefatos avaliados foram:

- Fase de elicitação de requisitos:
  - Análise estruturada: lista de requisitos, diagrama de contexto, descrição de requisitos;
  - Processo unificado: documento de visão, diagrama de casos de uso;
  - Ambos: modelo de dados conceitual, glossário.

- Fase de análise:
  - Análise estruturada: funções derivadas dos requisitos, diagrama hierárquico de funções, diagrama de fluxo de dados;
  - Processo unificado: especificação de casos de uso, diagrama de classes, diagrama de atividades, diagrama de sequência, diagrama de estado;
  - Ambos: protótipo não-funcional, modelo lógico de dados, dicionário de dados.

- Fase de projeto:
  - Análise estruturada: modelo de arquitetura, diagrama geral de transações, especificação de componentes;
  - Processo unificado: diagrama de colaboração, diagrama de componentes, diagrama de distribuição;
  - Ambos: modelo físico de dados, protótipo funcional.

- Codificação:
  - Ambos: código-fonte e comentários no código-fonte.

- Teste:
  - Ambos: plano de teste unitário, plano de teste sistêmico, plano de teste de aceitação.

- Transição:
  - Ambos: plano de migração de dados, plano de transição, manual de usuário.

Esse estudo identificou que, para a manutenção de um *software*, os artefatos mais importantes são o código-fonte com seus comentários, um modelo de dados (seja lógico, físico ou o diagrama de classes), e a especificação dos requisitos. Embora métodos ágeis de desenvolvimento possam apoiar-se mais na comunicação informal do que na documentação, o estudo deixa claro que esse não é o caso da atividade de manutenção de *software*, que ainda necessita de documentação farta e atualizada.

Outro estudo (Lethbridge *et al.,* 2003) analisou se realmente toda a documentação precisa estar atualizada para ser relevante na manutenção do *software*. A primeira conclusão desse estudo foi de que, em geral, não se costuma atualizar documentação de *software*, a não ser artefatos relativos a testes ou qualidade do *software*. Por outro lado, identificou-se que mesmo a documentação desatualizada pode ser útil, e que existe uma correlação entre a frequência com que um artefato é comumente consultado e sua exatidão em relação ao estado atual do *software*, ou seja, a atualização do artefato acompanha a atualização do *software* quanto mais ele é demandado na sua manutenção. Além disso, quanto mais próxima do código-fonte, mais a documentação precisa estar

atualizada para ser útil, como as conclusões de De Souza *et al.* (2006) apontadas acima parecem corroborar.

Esses estudos foram produzidos levando-se em conta duas metodologias de produção de *software*. Primeiro, o processo tradicional de desenvolvimento de *software*, conhecido como "cascata" (Royce, 1987), que se inicia com a definição do escopo do projeto e a elicitação de requisitos. Com isso, uma grande quantidade de documentação é produzida antes mesmo do *software* em si começar a ser desenvolvido. De um lado, essa documentação inicial é muito importante para nortear o desenvolvimento, lançando as bases de onde partem todas as tarefas a seguir. Durante todo o processo, mais documentação é produzida e cada fase possui seus artefatos típicos.

Por outro lado, quanto mais documentação é produzida, menos se atualiza aquela que já existe (Forward; Lethbridge, 2002). Uma vez que todo projeto está sujeito a mudanças durante todo o ciclo de desenvolvimento, é interessante que sua documentação reflita isso. Porém, dadas as restrições de tempo e orçamento, geralmente esse esforço de atualização fica restrito ao *software* em si, mesmo porque mais documentação é gerada a cada etapa do ciclo de desenvolvimento. Além disso, muitas vezes parte dos requisitos levantados na primeira fase do projeto mostram-se inúteis ou equivocados à medida que o projeto avança (Sharon, 1996).

O outro processo mencionado nos estudos acima é conhecido como "processo unificado" (Jacobson *et al.*, 1999), e refere-se a um conjunto de modelos de construção de *software*, geralmente atrelados ao paradigma da orientação a objetos, que utilizam como notação de apoio a *Unified Modeling Language* (UML). Ao contrário do método em "cascata", o processo unificado implementa vários ciclos de iterações de desenvolvimento. Dessa maneira, todas as fases do desenvolvimento são repetidas de forma que, a cada iteração, seja gerado um produto testado, integrado e executável, mas ainda não pronto para ser colocado em produção, levando ao crescimento incremental do sistema. Ao final de cada iteração, há o refinamento e adaptação dos requisitos e do projeto para a próxima iteração. Neste modelo, ainda há um foco muito grande na geração de farta documentação que deveria ser atualizada constantemente durante o processo, o que muitas vezes não ocorre.

Pensando em uma forma de reduzir o impacto das mudanças no processo de desenvolvimento, em meados da década de 1990 foi criado o *Scrum* (Beedle *et al.*, 1999), uma forma mais rápida, confiável e eficiente de desenvolver *software*. Baseado no Sistema Toyota de Produção e no ciclo OODA da aviação de combate, o Scrum foca em

reduzir o escopo do projeto a várias unidades menores, que resultam na entrega de uma funcionalidade por vez, porém já em funcionamento. Trabalhando dessa forma, é possível reduzir custos e prazos, entregar valor constantemente e tornar as mudanças uma parte do projeto. Outras iniciativas, como a *eXtreme Programming* (Beck, 1999), o DSDM (Howard, 1997), e o *Adaptive Software Development* (Highsmith, 2013) surgiram em torno dessa mesma época, e um encontro de seus criadores em fevereiro de 2001 culminou na redação do Manifesto Ágil (Beck *et al.*, 2001), que é uma coleção de princípios ou valores desejáveis em projetos ágeis de *software*: indivíduos em vez de processos; produtos que funcionam em vez de documentação dizendo como deveriam funcionar; colaboração com o cliente em vez de negociação com ele; resposta às mudanças em vez de um plano rígido.

Embora possa parecer, essas iniciativas ágeis não são avessas à documentação. Pelo contrário, reconhecem sua importância para a qualidade do produto desenvolvido, mas também estabelecem situações em que a comunicação por outros meios pode ser mais produtiva e gerar menos pendências. Ao invés de começar por uma documentação em grande parte estática, as propostas ágeis iniciam pela definição dinâmica do que é esperado de cada entrega. Na prática, grande parte do que seria extensivamente documentado no método em cascata, acaba sendo comunicado à equipe por meio de desenhos em quadro branco, reuniões em pé, etiquetas autocolantes ("*post-its*"). Ainda assim, toda a documentação considerada relevante para a compreensão futura do sistema e sua manutenção é gerada, porém em formatos um pouco diferentes dos tradicionais.

Assim, a utilização de estruturas ágeis no desenvolvimento de *software* (e, hoje, em vários outros tipos de projeto) permite à equipe se autogerenciar, definindo a cada iteração a quantidade de trabalho que será capaz de produzir. Além disso, ajuda a eliminar possíveis obstáculos que a impedem de produzir mais, obtendo avaliação constante de quem de fato utilizará o produto que está sendo desenvolvido e gerando documentação atualizada, simples de compreender e em quantidade suficiente para a correta manutenção do sistema, porém eliminando os excessos dos métodos em cascata e unificado.

Este trabalho investiga quais artefatos de documentação se aplicam com maior aderência ao contexto do *software* bioinformático estudado. Ainda, reconhecendo a enorme importância da documentação para a correta compreensão desse tipo de *software*, e considerando que sua utilização se dá em equipes multidisciplinares, gerará

artefatos a partir desse *software* voltado à identificação de microRNAs em um dado genoma.

Engenharia de Software aplicada à Bioinformática

A aplicação da Engenharia de Software à Bioinformática tem sido neglicenciada, em parte por uma crença de que o uso de software em contextos científicos difere de sua aplicação em situações comerciais (Lawlor; Walsh, 2015). Essa crença parece vir de uma confusão entre os conceitos de programação de computadores e da Engenharia de *Software* propriamente dita, que leva pesquisadores de outras áreas (mas que possuem habilidades de programação) a desenvolver *software* sem o uso dos modernos padrões de desenvolvimento. Ainda segundo Lawlor e Walsh (2015), isso se traduz em lentidão para realizar descobertas, que por sua vez são menos confiáveis do que poderiam ser.

Esses pesquisadores da área da Bioinformática geralmente são autodidatas em programação (Verma et al., 2013), e, embora no método científico seja considerada uma boa prática documentar todos os processos, a falta de algum tipo de treinamento formal em Engenharia de *Software* leva à falta de documentação das ferramentas de *software* desenvolvidas nesse contexto.

A documentação seria importante não apenas para guiar o processo de desenvolvimento, mas para orientar o usuário final na utilização da ferramenta. Dessa forma, não apenas mais pessoas poderiam entender o processo de desenvolvimento de uma dada solução, como também mais pessoas seriam capazes de utilizá-la em suas pesquisas, levando a um maior número de citações e a uma maior disseminação da solução proposta. Nesse contexto, embora a aplicação de uma metodologia de desenvolvimento de *software* fosse a solução definitiva para o problema, isso demandaria que todos os pesquisadores de Bioinformática envolvidos na produção de *software* conhecessem a fundo pelo menos uma dessas metodologias, o que geralmente não acontece (Verma et al., 2013).

Como forma de simplificar a adoção de boas práticas de Engenharia de *Software* sem a necessidade de treinamento formal na área, pode-se adotar uma série de recomendações como as sugeridas por Karimzadeh e Hoffman (2018), com foco na documentação para o usuário final, que têm sido utilizadas em diversos projetos de Bioinformática. Esse trabalho traz várias considerações sobre a criação de documentação

de *software* de bioinformática, destacando, entre outras, a disponibilização do código-fonte e a confecção de arquivo “leia-me” contendo guia de instalação, guia de início rápido e formatos de entrada e saída; e a oferta de um manual de referência com a descrição detalhada de cada parâmetro configurável pelo usuário.

Em 2017, quando essas recomendações foram aceitas para publicação e já estavam acessíveis em versão preliminar, Davide Chicco (2017) se baseou nas mesmas para a criação de suas dez sugestões para o uso de Aprendizado de Máquina em contextos de Bioinformática. Na sugestão de número 9, o autor recomenda utilizar ferramentas e código *open source*, juntamente com a confecção de notas sobre o desenvolvimento e, citando as recomendações de Karimzadeh e Hoffman, documentação completa e detalhada do *software*. No mesmo ano, outro artigo (Taschuk; Wilson, 2017) trouxe dez recomendações para que *software* produzido para pesquisa possa ser utilizado pela comunidade, eliminando problemas comuns que impedem sua utilização fora da instituição original ou mesmo em outra máquina que não a do desenvolvedor. Uma das recomendações do artigo envolve documentar o código e o uso do *software* desenvolvido, na qual os autores ainda sugerem a leitura do artigo de Karimzadeh e Hoffman para referência sobre a escrita de documentação de qualidade.

As considerações de Karimzadeh e Hoffman foram utilizadas por Saelens et al. (2018) para construir alguns itens de um sistema de avaliação da qualidade de ferramentas computacionais para a inferência da trajetória de células únicas. Esse tipo de ferramenta ordena as células de uma dada população de acordo com o ponto em que cada uma se encontra no seu ciclo de vida. As categorias do sistema de avaliação incluem: Disponibilidade, Qualidade do Código, Garantia de Código, Documentação, Comportamento e Artigo. Em todas as categorias da avaliação houve referência às considerações de Karimzadeh e Hoffman, exceto em Garantia de Código e Comportamento. Isso revela a importância da documentação de *software* e como essa atividade influi na qualidade de uma ferramenta de bioinformática.

O artigo de Hart (2018) compara a revolução provocada pela genômica à que potencialmente pode ocorrer a partir da digitalização da avaliação patológica, e destaca erros cometidos no transcorrer do estudo da genômica que atrasaram a dita revolução. O autor cita as recomendações de Karimzadeh e Hoffman quando fala sobre o estágio atual da interface de linha de comando que desenvolve para avaliação patológica digital. Ele destaca que esse tipo de ferramenta demanda uma documentação bastante completa para melhor compreensão do sistema e maior adoção pela comunidade, e que portanto

esse novo ramo seria melhor atendido por ferramentas com interfaces gráficas, mas as interfaces de linha de comando ainda teriam sua importância. Assim, entende-se que as sugestões de Karimzadeh e Hoffman seriam de grande valia no desenvolvimento de ferramentas para avaliação patológica digital.

O estudo de Sumonja et al. (2019) desenvolveu um método para predição automatizada de interações proteína-proteína utilizando Aprendizado de Máquina e Algoritmos Genéticos. Esse método foi disponibilizado como um *software* gratuito. As recomendações de Karimzadeh e Hoffman são utilizadas para argumentar sobre a complexidade dos procedimentos de métodos computacionais para predição de interações proteína-proteína. Os autores destacam que, apesar do grande potencial dessas ferramentas, sua acessibilidade e uso amplo são comprometidos por esse problema. Dessa forma, justificam o desenvolvimento de uma nova ferramenta, o *software* HP-GAS, que levaria em consideração essas recomendações sobre usabilidade, documentação, exemplos de uso, arquivo “leia-me”, instruções de instalação e operação, dentre outros.

Georgeson et al. (2019) criaram uma ferramenta de desenvolvimento de *softwares* bioinformáticos com suporte a 12 linguagens de programação e orientada a boas práticas de desenvolvimento. Karimzadeh e Hoffman são citados quando é destacada a importância de identificar problemas rapidamente a partir de qualquer mudança introduzida no *software*. Também são citados na descrição da documentação de usuário gerada pela ferramenta (arquivo “leia-me” e ajuda em linha de comando). Novamente, na seção em que é descrito o controle de revisão da ferramenta, o trabalho de Karimzadeh e Hoffman é citado, destacando a importância do gerenciamento de versões de um *software*. Suas contribuições também são lembradas na seção que trata da padronização do código de acordo com a linguagem de programação escolhida, realçando que isso possibilita maior integração com o ecossistema da linguagem, evita erros comuns e encoraja contribuições de outros desenvolvedores.

Um estudo (Mangul et al., 2019) sobre princípios para garantir a usabilidade e estabilidade de arquivamento de *softwares* bioinformáticos, com sugestões para o desenvolvimento dessas ferramentas e de abordagens padronizadas de verificação e arquivamento de *software*, faz referência ao estudo de Karimzadeh e Hoffman quando trata de soluções propostas para guiar o desenvolvimento de *software* científico. Isso é especialmente abordado na seção que sugere que esse tipo de ferramenta seja

distribuída com um guia rápido cobrindo a instalação e uso do programa, além de um manual mais detalhado com configurações avançadas.

As recomendações de Karimzadeh e Hoffman foram seguidas na criação de toda a documentação do *software* desenvolvido por Carper et al. (2020), para a criação de comunidades sintéticas de micróbios para uso em experimentações *in vivo*. Zobolas et al. (2020) também destacam que a documentação adequada é uma prática saudável de desenvolvimento de *software*, e mencionam as recomendações de Karimzadeh e Hoffman como vitais para o alcance dos objetivos do estudo que propôs a implementação de portais de busca para termos e conceitos das ciências da vida, ajudando biocuradores a encontrar termos para anotação e outros recursos.

Softwares bioinformáticos para identificação de miRNAs

miRNAs são pequenas cadeias de aproximadamente 21 nucleotídeos, responsáveis por regular a expressão genética em quase todos os processos celulares (Filipowicz et. al, 2008). Foram descobertos a partir da identificação da sequência *lin-4* em *Caenorhabditis elegans* (Lee et al., 1993), que regula a passagem do primeiro estágio larval para o segundo, através da supressão progressiva da proteína *lin-14*. Posteriormente, um segundo miRNA, *let-7*, foi identificado em *C. elegans* (Reinhardt *et al.*, 2000), e demonstrou-se que essa molécula era conservada evolutivamente (Pasquinelli *et al.*, 2000). Com essa descoberta, foi possível entender que os miRNAs, apesar de não codificarem proteínas, possuem papel muito importante na regulação genética dos organismos, atuando de forma negativa – ou seja, silenciando ou inibindo a expressão de outras sequências.

Sua importância vem sendo, dia após dia, confirmada e ampliada. Eles interagem com as proteínas da família Argonauta (AGO), formando a base dos complexos de silenciamento induzido por miRNA (miRISCs), que por sua vez serão mediadores do silenciamento pós-transcricional de RNA mensageiros (mRNAs) que contenham sequências total ou parcialmente complementares aos miRNAs (Jonas; Izaurralde, 2015).

Essas moléculas podem ser manipuladas e utilizadas de diversas formas, incluindo o silenciamento de genes responsáveis pelo aparecimento de doenças (Cubillos-Ruiz *et al*., 2012); a reprogramação do crescimento de células tumorais, levando a sua autodestruição (Jain *et al.*, 2018); o carregamento de drogas especificamente até as

células doentes (Pindiprolu *et al.*, 2017); ou sua utilização como biomarcadores de diagnóstico e prognóstico (Di Leva; Croce, 2013).

Desde o início dos anos 2000, várias ferramentas computacionais foram criadas com o objetivo de identificar miRNAs (Mendes *et al.*, 2009). Essas ferramentas fazem uso de diversas estratégias. Uma delas é a comparação com características estruturais conservadas identificadas em outras espécies. Um dos primeiros métodos a fazer uso desse tipo de filtro foi o de Grad *et al.* (2003). Nele, inicialmente definiu-se um conjunto de 39 *stem-loops* imperfeitos obtidos de regiões intergênicas do genoma de *C. elegans*, que foram filtrados a partir de características de miRNAs já conhecidos desse organismo, chegando-se a 29 candidatos. Em seguida, os *loops* foram avaliados por critérios de conservação a partir de estruturas semelhantes presentes em outros dois genomas, e apenas 6 foram validados nesse critério.

Outro método, chamado MiRscan (Lim *et al.*, 2003), começava escaneando o genoma de *C. elegans* por meio de uma janela de 110 nucleotídeos, utilizando filtros mais permissivos quanto à estrutura. Depois, foram buscados homólogos em *C. briggsae*, chegando-se a 36.000 candidatos. Dos 53 miRNAs conhecidos na época e conservados entre as duas espécies, 50 foram recuperados nesse procedimento e utilizados para classificar os 36.000 *hairpins*, porém o método só foi capaz de recuperar metade dos miRNAs conhecidos. Esta abordagem foi posteriormente aperfeiçoada para levar em consideração a presença de motivos e blocos de conservação de sequências acima e abaixo dos *loops* presumidos, resultando no método MiRscanII.

Outro estudo (Berezikov *et al.*, 2005) fez uso da detecção de regiões conservadas no entorno dos precursores. Embora não tenham conseguido encontrar motivos claramente conservados em regiões imediatamente adjacentes aos precursores de miRNA, os autores puderam observar um padrão de conservação decrescente que foi utilizado como perfil.

Um outro método, conhecido como miRseeker (Lai *et al.*, 2003), utilizou genomas de duas espécies de *Drosophila*. As regiões intergênicas e intrônicas não anotadas foram alinhadas, e então *stem-loops* potenciais foram identificados e avaliados nas regiões conservadas. Nessa avaliação foram considerados o comprimento do maior braço dos *loops*, sua Energia Mínima Livre (MFE na sigla em inglês), e um conjunto de métricas para penalizar *loops* internos (especialmente *loops* assimétricos e protuberâncias).

Todos esses métodos focados na comparação de regiões conservadas foram capazes de recuperar grande parte dos miRNAs conhecidos e identificar novos

reguladores, mas não tiveram sucesso em produzir regras capazes de recuperar todos os miRNAs conhecidos sem levar a muitos resultados falso-positivos. Dessa forma, logo começaram a surgir abordagens baseadas em Aprendizado de Máquina, partindo da ideia de analisar um único genoma de cada vez.

Uma das tentativas resultou no método ProMIR (Nam *et al.*, 2005), que utilizou um conjunto inicial de candidatos formado por *stem-loops* presentes em etiquetas de sequências expressas (ESTs na sigla em inglês para *Expressed Sequence Tags*). ESTs são segmentos de sequências de um clone de DNA complementar (cDNA) que correspondem a um RNA mensageiro (mRNA) (Adams *et al.*, 1991), e dessa forma o ProMIR se restringiria a buscar em sequências com expressão confirmada. Os candidatos seriam filtrados por meio de critérios estruturais bastante permissivos envolvendo comprimento, tamanho do *loop* e MFE. Seriam modeladas as características da porção haste do *stem-loop* como uma sequência pareada, considerando o padrão de pareamento de base e a localização do miRNA maduro. Dois conjuntos de dados de treinamento foram definidos, sendo o positivo consistindo em todos os pre-miRNAs humanos conhecidos, e o negativo correspondendo a 1.000 *stem-loops* estendidos extraídos aleatoriamente do genoma humano. O *stem-loop* seria considerado um bom candidato a pre-miRNA se contivesse uma sequência com probabilidade de ser um miRNA maduro acima de determinado valor. Como o conjunto de candidatos se revelou muito grande, foram utilizados filtros adicionais, incluindo a abordagem utilizada por Berezikov *et al.* (2005), com a verificação de um padrão de conservação decrescente por meio da comparação com outros genomas de vertebrados.

Houve um método que procurou identificar miRNAs em genomas virais (Pfeffer *et al.,* 2005), nos quais critérios de conservação não são passíveis de serem utilizados, já que grande parte dos pre-miRNAs virais não mostram indícios de conservação em relação a outros pre-miRNAs virais ou aos precursores do hospedeiro infectado. Esse método foi, portanto, o primeiro a ter sucesso na análise de um único genoma. Foram identificados inicialmente *stem-loops* robustos, que em seguida foram analisados quanto à energia livre, quantidade de nucleotídeos no *stem* simétrico, e número de bases pareadas.

Outros métodos baseados em Aprendizado de Máquina foram desenvolvidos, geralmente empregando *Support Vector Machines* (SVMs) (Cortes; Vapnik, 1995) para identificar as características mais distintivas dos miRNAs. As características analisadas

envolvem a composição das sequências, propriedades topológicas dos *stem-loops*, estabilidade termodinâmica e métricas de entropia (Ng; Mishra, 2007).

Alguns métodos (Sheng *et al.*, 2007 e Xue *et al.*, 2005) chegaram a utilizar uma cadeia de três SVMs, cada uma focada em um aspecto diferente, como a conservação da sequência, conservação da estrutura secundária, localização e estrutura do miRNA maduro na estrutura do *hairpin*.

Uma outra abordagem (Jiang *et al.*, 2007) envolvendo Aprendizado de Máquina utilizou *Random Forests* (RF) (Breiman, 2001) e comparou seus resultados com métodos que utilizaram SVMs, conseguindo melhores resultados. As RF são um conjunto de árvores de decisão em que, a cada nó de cada árvore, um conjunto de características é escolhido para a tarefa de classificação. Este método, denominado MiPred, consegue predizer se uma sequência é um *hairpin* típico de pre-miRNA e, em caso positivo, se é um pre-miRNA real ou um pseudo pre-miRNA.

Combinando o MiPred a outras ferramentas computacionais, Gomes *et al.* (2011) desenvolveu um método capaz de identificar miRNAs conservados no genoma de *Schistosoma mansoni.* Em versões posteriores, o MiPred deixou de ser utilizado, e o método foi batizado de Biopipeline.

O presente estudo pretende estudar essa abordagem supracitada, identificando como a Engenharia de *Software* poderia se aplicar ao *software* proposto, documentando o funcionamento do sistema com o objetivo de facilitar a utilização por pesquisadores das mais diversas áreas.

Briefings in Bioinformatics, 2021, pp. 1–7

doi: DOI HERE
Advance Access Publication Date: Day Month Year
Paper

PAPER

Documenting Bioinformatics Software Via Reverse Engineering

Vinicius Marques[1,] and Laurence Amaral[2,*]

[1]Institute of Biotechnology (IBTEC), Federal University of Uberlandia (UFU), Getulio Vargas ave, 230, 38700-103, Minas Gerais, Brazil and [2]Faculty of Computation (FACOM), Federal University of Uberlandia (UFU), Getulio Vargas ave, 230, 38700-103, Minas Gerais, Brazil
*Corresponding author. laurence@ufu.br
FOR PUBLISHER ONLY Received on Date Month Year; revised on Date Month Year; accepted on Date Month Year

Abstract

Documentation is one of the most neglected activities in Software Engineering, though it's an important way of assuring quality and understanding. Bioinformatics software is generally written by researchers of various fields other than Computing who usually don't provide documentation. Documenting bioinformatics software may ease its adoption in multidisciplinary teams, as well as expand its impact on the community. In this paper, we highlight how one can document software that is already finished, using reverse engineering and thinking of the end-user.

Key words: Bioinformatics, biological software, Software Engineering, software documentation

Introduction

Bioinformatics software are computational systems built for Biology and Biotechnology applications [2]. They are usually written and used by multidisciplinary teams, which often include researchers from fields as diverse as Biology, Biochemistry, Genetics, Pharmacy and Medicine [5]. While some of these teams do have Computing researchers, some are comprised of other fields scientists who learned to write software. They may not be aware of the best Software Engineering practices, and usually don't provide the documentation alongside their software. Thus, even though the software itself may be efficient and effective, it may be difficult for other researchers outside the original team to use it.

Software documentation is one of the most important tasks in Software Engineering [4], though it is one of the most neglected [10]. Time and cost restraints often lead to poor or incomplete documentation [9]. But, considering both maintaining and end-user points of view, documentation provides valuable information, from an overview of the system to technical detailing of how the software works, being useful for developers, maintainers, end-user, and the community [10].

The application of Software Engineering to Bioinformatics has been also neglected, partly because of a creed that the use of software in scientific contexts is different than in commercial situations [7]. This seems to come from a misunderstanding of concepts from computer programming and Software Engineering itself, which leads researchers from other fields who know how to code to develop software without the best practices in mind, slowing discoveries and making them less reliable than they could be. Bioinformatics researchers usually are self-taught programmers [11] that know the scientific method considers documenting all processes a good practice, but due to the lack of formal training in Software Engineering don't document the software they develop.

Documentation is important not only to guide the development process but also to conduct the end-user correctly, in the use of the software. Thus, not only more people could understand the development of a given solution, but even more, people would be able to use it in their researches, increasing the number of citations of the original work and leading to a wider spread of the solution. Adoption of a software development methodology could definitively solve this problem, but it would demand deep knowledge of one of these methodologies by Bioinformatics researchers, which usually is not the case [11]. As a way of simplifying the adoption of good Software Engineering practices without formal training, one could adopt recommendations like Karimzadeh and Hoffman's [3], focusing in the end-user documentation.

In this study, we analyzed Biopipeline [1], a bioinformatics software used to predict miRNAs within a genome, starting from the source code and building documentation, for both

maintenance and the end-user. We followed Karimzadeh and Hoffman recomendations where applicable, and formatted the end user documentation as simple as possible.

Materials and methods

The Biopipeline software

The analyzed software is named Biopipeline and was written to identify conserved miRNAs within a genome. The source code of the Biopipeline software was kindly provided by Laboratório de Informática e Análises Moleculares (LBAM) of the Federal University of Uberlândia, at Patos de Minas. It was imported in Atom text editor, using the Perl extension. Preliminary analysis showed that it was written in Perl using the Structured Programming paradigm. It is comprised of a series of procedures called functions and numbered from F1 to F50, each one of them performing a specific task.

Technical documentation

For a better understanding of the functionalities, and to guide the development of the end-user documentation, we built a block diagram using Dia 0.97, and a two-level data flow diagram using LibreOffice Draw. We made these documents available so that they can also be useful for software maintenance.

Guidelines for end user documentation

In order to provide a simpler way for researchers to document software already implemented, we followed the considerations for the creation of bioinformatics software documentation proposed by Karimzadeh and Hoffman. Their work provide various considerations about bioinformatics software documentation, being the minimum set of documentation comprised of a GitHub or Bitbucket page with the source code and an issue tracker; a *Readme* file containing an installation guide, a quick start guide, and the file formats used for inputs and outputs; and a reference manual with a detailed description of each user-configurable parameter. Since the authors of Biopipeline are not releasing the source code as open-source at the moment, we have launched a GitHub page (available at https://github.com/marques-vinicius/biopipeline-docs) only as a documentation repository, not providing source code or issue tracker. Additionally, as the authors have stated that there are no user-configurable parameters, and all parameters are set within the source code with optimized values, we also have not written a reference manual.

Results

Source code analysis and technical documentation

Analysis of the source code revealed that the software was written in Perl using the Structured Programming paradigm. It is comprised of a series of procedures called functions and numbered from F1 to F50, each one of them performing a specific task. In addition, no Object Oriented Programming was identified, like classes or instances.

After realizing what methodology was used when programming, we identified that aside from the processing conducted by original code, additional, external tools are used to perform important parts of the overall processing, using a concept called pipeline [6], which means that various steps of processing are executed in a specific sequence.

This so-called "reverse engineering" made it possible to understand exactly how the software does its work. The first input of the system is a fasta file containing a complete genome. This file is conducted to a preliminary analysis using *einverted EMBOSS* and *BLASTn* software, configured with optimized parameters identified by the authors. This step returns as output the structures identified as hairpins and homolog pre-miRNAs from miRBase. Those sequences are archived in various files for parallel processing.

Those files are then processed by *RNAfold*, then *BLASTn* again, to identify similarities between the found sequences and known miRNAs. Next, the sequences that resemble coding genes, non-coding RNA (ncRNA), and repeating sequences are discarded. Then, another selection is performed by *BLASTn* with more specific parameters. After all these steps are executed, the files with the sequences are joined, resulting in a single fasta file containing all miRNAs found. The overall processing is represented in Picture 1, by means of a block diagram which was divided into three parts for better visualization.

The block diagram is useful to translate the source code to a visual language, but showed itself very large and complex, and not the best representation of the system, serving instead as a starting point for the understanding of the functionalities and how Biopipeline uses external software to perform specific tasks.

A data flow diagram is a better, clearer representation of the system operation. It shows the relationships between Biopipeline and its external dependencies and the data flow in a more direct way. We built it as a two-level diagram. Level 0, showed in Figure 2 provides an overview of what the software does and makes clear that it uses external tools for additional processing. It also shows the inputs and outputs in a simplified way, making it clear that the main input is the genome, which is processed by Biopipeline and external software, and the output is the set of miRNAs found.

Nonetheless, this level of detail is insufficient to show how the software analysis the genome until finding the miRNAs. In order to reach this detailing, a second level of the data flow diagram is presented in Figure 3. This Level 1 diagram shows what kind of processing is performed on the initial genome file, and exactly when external tools are called. There is no direct reference to the actual functions in the algorithm, contrary to the block diagram.

End-user documentation

The technical documentation above was produced to allow a better understanding of the software. Both block diagram and data flow diagram were useful to retrieve important information used in the production of the end-user documentation and are also useful for maintenance tasks.

In order to produce relevant yet understandable documentation, we followed Karimzadeh and Hoffman recommendations where applicable. This led us to build a *Readme* file containing an overview, installation guide, and user guide. Nonetheless, we have opted to launch a GitHub/Bitbucket page only as a documentation repository, and not as recommended, because the authors are not releasing the source code as open-source at the moment; and a reference manual is not provided because all parameters were optimized and set in the source code by the authors and are not user-configurable.

The *Readme* file was formatted as simple as possible, yet containing all information needed in order to solve

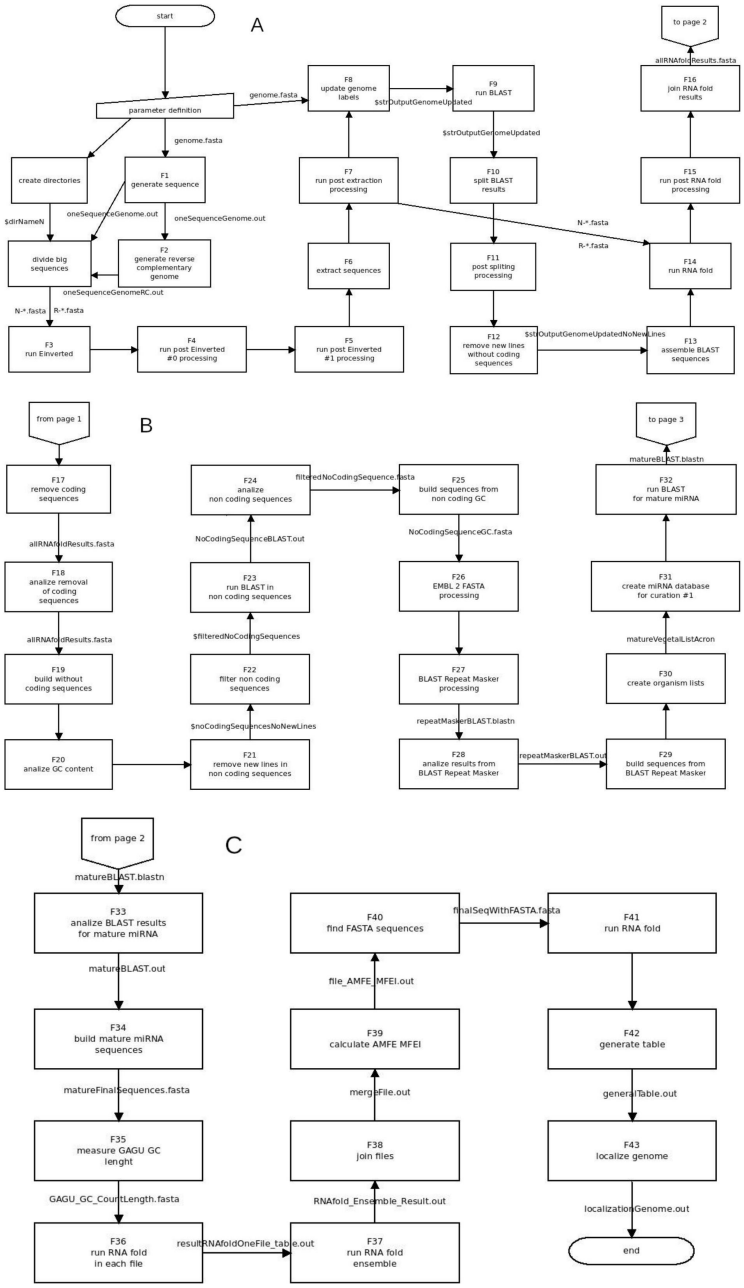


Fig. 1. Block diagram showing an overview of the system, with inputs and outputs. It was divided into three parts (A)(B)(C) for better visualization. Each "FXX" represents a function defined in the Biopipeline algorithm, and it's possible to see that some of them are performed by external software.

dependencies, as shown in Figure 4, install and run the software, as shown in Figure 5. It provides the minimum requirements of the system, examples of command lines,

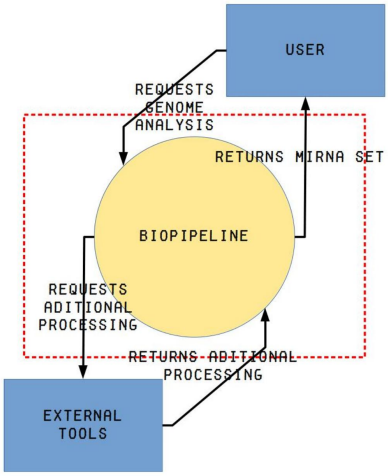


Fig. 2. Level 0 Data Flow Diagram, showing an overview of the system and the existence of additional processing done by external tools

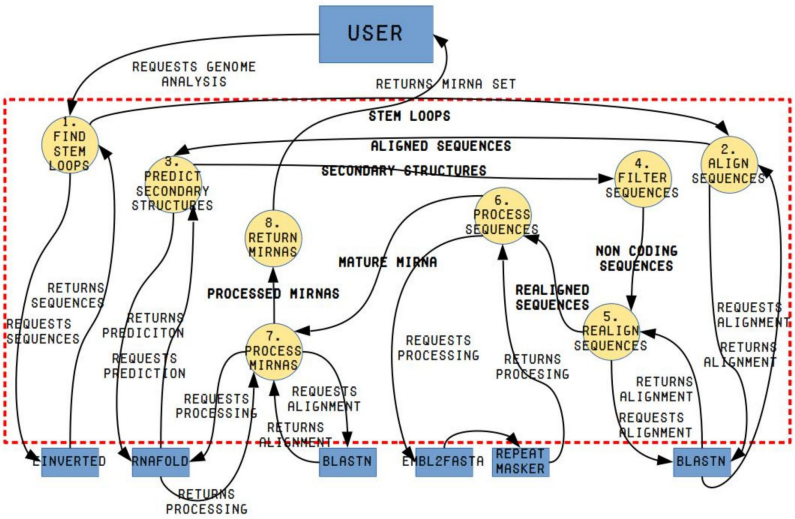
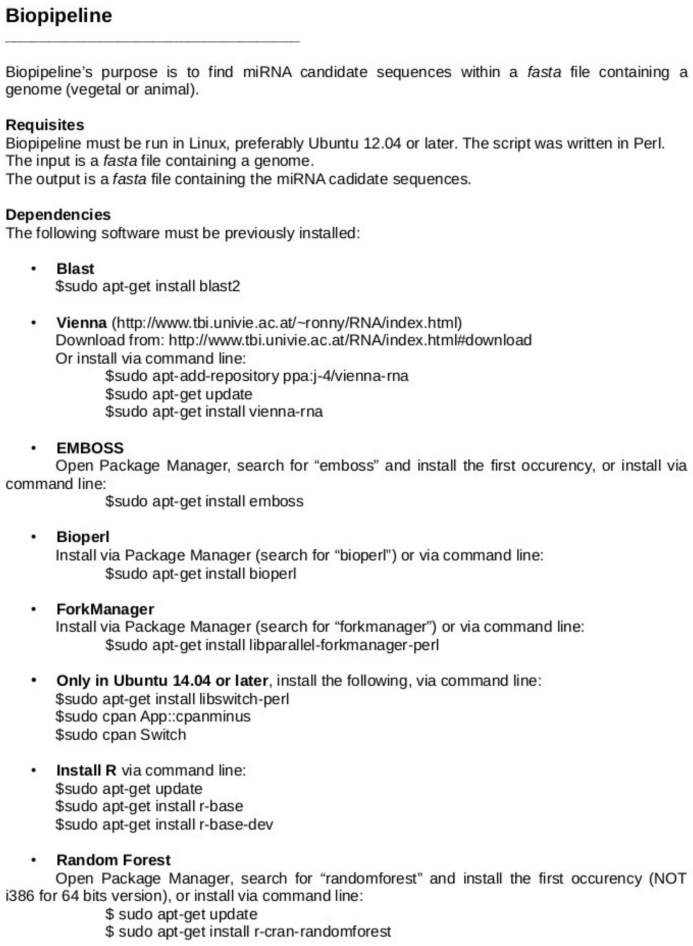


Fig. 3. Level 1 Data Flow Diagram showing details of the processing performed on the file containing the genome, including what and when external tools (*einverted*, *RNAFold*, *Blastn*, *Embl2Fasta*, and *Repeat Masker*) are called. One of these tools, *Blastn*, is shown twice for aesthetic reasons

and references the technical documentation and the original Biopipeline article.

Discussion

Documenting software can be a tedious task when one is anxious to finish development, but it is a task better done during this period. However, reverse engineering of a piece of software can lead to the building of good documentation, both maintaining and end-user documentation. Many authors have demonstrated the importance of documenting bioinformatics software, such as Leprevost and collaborators [8], Karimzadeh and Hoffman, Kumar and Dudley.



Biopipeline

Biopipeline's purpose is to find miRNA candidate sequences within a *fasta* file containing a genome (vegetal or animal).

Requisites
Biopipeline must be run in Linux, preferably Ubuntu 12.04 or later. The script was written in Perl.
The input is a *fasta* file containing a genome.
The output is a *fasta* file containing the miRNA cadidate sequences.

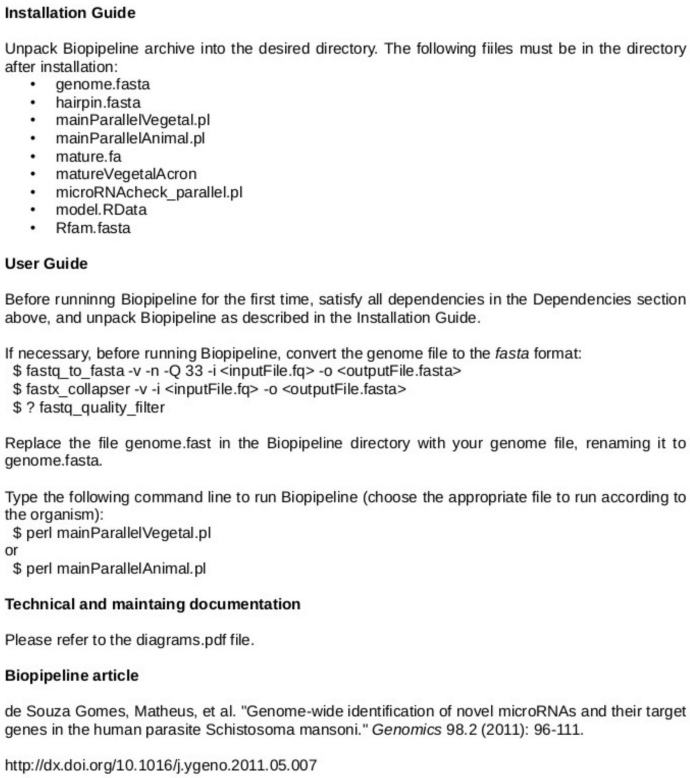
Dependencies
The following software must be previously installed:

- **Blast**
  $sudo apt-get install blast2
- **Vienna** (http://www.tbi.univie.ac.at/~ronny/RNA/index.html)
  Download from: http://www.tbi.univie.ac.at/RNA/index.html#download
  Or install via command line:
  $sudo apt-add-repository ppa:j-4/vienna-rna
  $sudo apt-get update
  $sudo apt-get install vienna-rna
- **EMBOSS**
  Open Package Manager, search for "emboss" and install the first occurency, or install via command line:
  $sudo apt-get install emboss
- **Bioperl**
  Install via Package Manager (search for "bioperl") or via command line:
  $sudo apt-get install bioperl
- **ForkManager**
  Install via Package Manager (search for "forkmanager") or via command line:
  $sudo apt-get install libparallel-forkmanager-perl
- **Only in Ubuntu 14.04 or later**, install the following, via command line:
  $sudo apt-get install libswitch-perl
  $sudo cpan App::cpanminus
  $sudo cpan Switch
- **Install R** via command line:
  $sudo apt-get update
  $sudo apt-get install r-base
  $sudo apt-get install r-base-dev
- **Random Forest**
  Open Package Manager, search for "randomforest" and install the first occurency (NOT i386 for 64 bits version), or install via command line:
  $ sudo apt-get update
  $ sudo apt-get install r-cran-randomforest

Fig. 4. The end-user documentation includes an overview of the software along with a list of dependencies that must be satisfied before installing it. The document was formatted in a simple readable way

Amongst the reasons to invest time and efforts in the documentation task are: improvement of software usability, continuity of the software development, reproducibility of experiments, better software maintenance, reducing of support requests for the authors, and increasing of citations of the original work. Thus, documenting bioinformatics software can benefit the authors, researchers, scientific community, and general community.

In our efforts to document Biopipeline, we showed that is possible to write useful documentation for already implemented software. We also showed that this can be done by means of reverse engineering, and maybe done on software written by



Installation Guide

Unpack Biopipeline archive into the desired directory. The following fiiles must be in the directory after installation:

- genome.fasta
- hairpin.fasta
- mainParallelVegetal.pl
- mainParallelAnimal.pl
- mature.fa
- matureVegetalAcron
- microRNAcheck_parallel.pl
- model.RData
- Rfam.fasta

User Guide

Before runninng Biopipeline for the first time, satisfy all dependencies in the Dependencies section above, and unpack Biopipeline as described in the Installation Guide.

If necessary, before running Biopipeline, convert the genome file to the *fasta* format:
$ fastq_to_fasta -v -n -Q 33 -i <inputFile.fq> -o <outputFile.fasta>
$ fastx_collapser -v -i <inputFile.fq> -o <outputFile.fasta>
$? fastq_quality_filter

Replace the file genome.fast in the Biopipeline directory with your genome file, renaming it to genome.fasta.

Type the following command line to run Biopipeline (choose the appropriate file to run according to the organism):
$ perl mainParallelVegetal.pl
or
$ perl mainParallelAnimal.pl

Technical and maintaing documentation

Please refer to the diagrams.pdf file.

Biopipeline article

de Souza Gomes, Matheus, et al. "Genome-wide identification of novel microRNAs and their target genes in the human parasite Schistosoma mansoni." *Genomics* 98.2 (2011): 96-111.

http://dx.doi.org/10.1016/j.ygeno.2011.05.007

Fig. 5. The installation guide contains straight forward instructions. The user guide shows command line examples and instructions on the input format. There is a reference to another document containing the technical documentation and a reference to the original Biopipeline manuscript

others. Additionally, we showed that the documentation can (and sometimes must) be simple, understandable, and yet bring useful information. These results have the potential to help encourage researchers to document their software (and software developed by other researchers), thus benefiting the community.

Competing interests

There is NO Competing Interest.

Author contributions statement

V.M. and L.A. conceived the experiment(s), V.M. conducted the experiment(s), V.M. and L.A. analysed the results. V.M. and L.A. wrote and reviewed the manuscript.

Acknowledgments

The authors like to thanks CAPES, CNPq, FAPEMIG, and PROPP/UFU for financial support.

References

1. Matheus de Souza Gomes, Mohan Kumar Muniyappa, Sávio Gonçalves Carvalho, Renata Guerra-Sá, and Charles Spillane. Genome-wide identification of novel microrna and their target genes in the human parasite schistosoma mansoni. *Genomics*, 98(2):96–111, 2011.
2. Dana Jawdat. The era of bioinformatics. In *2006 2nd International Conference on Information & Communication Technologies*, volume 1, pages 1860–1865. IEEE, 2006.
3. Mehran Karimzadeh and Michael M Hoffman. Top considerations for creating bioinformatics software documentation. *Briefings in Bioinformatics*, 19(4):693–699, 2018.
4. Noela Jemutai Kipyegen and William PK Korir. Importance of software documentation. *International Journal of Computer Science Issues (IJCSI)*, 10(5):223, 2013.
5. A Kumar and N Chordia. Role of bioinformatics in biotechnology. *Res Rev Biosci*, 12(1):116, 2017.
6. Sudhir Kumar and Joel Dudley. Bioinformatics software for biologists in the genomics era. *Bioinformatics*, 23(14):1713–1717, 2007.
7. Brendan Lawlor and Paul Walsh. Engineering bioinformatics: building reliability, performance and productivity into bioinformatics software. *Bioengineered*, 6(4):193–203, 2015.
8. Felipe da Veiga Leprevost, Valmir C Barbosa, Eduardo L Francisco, Yasset Perez-Riverol, and Paulo C Carvalho. On best practices in the development of bioinformatics software. *Frontiers in genetics*, 5:199, 2014.
9. Timothy C Lethbridge, Janice Singer, and Andrew Forward. How software engineers use documentation: The state of the practice. *IEEE software*, 20(6):35–39, 2003.
10. David Lorge Parnas. Precise documentation: The key to better software. In *The Future of Software Engineering*, pages 125–148. Springer, 2011.
11. Dhawal Verma, Jon Gesell, Harvey Siy, and Mansour Zand. Lack of software engineering practices in the development of bioinformatics software. *ICCGI*, 2013:57–62, 2013.

Vinicius Marques is a M.S. student at the Institute of Biotechnology (IBTEC), Federal University of Uberlandia (UFU), Minas Gerais, Brazil. His research areas include Software Engineering and Bioinformatics.

Laurence Amaral is associate professor Faculty of Computing (FACOM) at the Federal University of Uberlândia (UFU), located in Patos de Minas - MG. He holds a doctorate (DC / UFSCar), a master's degree (FACOM / UFU) and a bachelor's degree (FACOM / UFU) in Computer Science, as well as a specialization in Biotechnology from the Federal University of Lavras (UFLA).

CONCLUSÃO

A Bioinformática é uma área interdisciplinar, e como tal, oferece desafios e oportunidades diversas. A convergência entre a Computação e a Biotecnologia, inerente a essa área, passa pela conjugação de interesses, formações, experiências e especialidades muito diferentes. Não é raro que um pesquisador precise desenvolver novas habilidades para alcançar os objetivos de sua pesquisa. Como os resultados precisam ser apresentados, muitas vezes, em tempo exíguo, não é possível se realizar um treinamento formal em outra área de formação, por exemplo. Isso é particularmente comum quando um pesquisador das áreas biológicas se depara com a necessidade de implementar um *software* e não dispõe de pessoas com formação em Computação em sua equipe, mas possui interesse em aprender a programar e consegue chegar ao resultado que almejava.

Essa necessidade leva ao aprendizado de linguagens de *script*, que costumam oferecer uma curva de aprendizagem menos íngreme que as linguagens “de mercado”, embora também ofereçam menos recursos ao pesquisador. Isso pode levar a um código de compreensão mais difícil para terceiros, especialmente por ser comum a utilização da programação como forma de juntar funcionalidades de várias ferramentas em um fluxo direcionado (o chamado *pipeline*). Esse contexto, em que o pesquisador sem formação em Computação conhece bem várias ferramentas de bioinformática e identifica que elas podem se relacionar de uma maneira particular para prover uma funcionalidade maior que a soma de suas funcionalidades individuais, costuma ser o que leva à necessidade de desenvolver código que agregue essas funções de forma satisfatória, como ocorreu com o *Biopipeline*.

Muito embora o código desenvolvido possa realizar corretamente o fluxo que o pesquisador tinha em mente, o fato de se usar uma linguagem de *script*, aliado à comum negligência em torno da documentação do *software* implementado podem levar a um produto final que funciona, mas depende muito de seu autor para ser utilizado e mantido. Isso, além de dificultar a disseminação do trabalho realizado, pode acarretar na sobrecarga do autor com suporte e manutenção do *software*.

É essa lacuna que este trabalho procurou preencher. Gerou-se a documentação voltada ao usuário final do *software* Biopipeline, além de alguns artefatos voltados para a manutenção, evidenciando-se a importância da Engenharia de *Software* e, mais

especificamente, da documentação de *software* para a melhor compreensão do sistema e de seu funcionamento. Houve, porém, dificuldade em implementar mesmo o conjunto mínimo de recomendações de Karimzadeh e Hoffman, considerando que o autor do *software* analisado definiu todos os parâmetros de forma fixa no código-fonte (impedindo a geração do manual de referência contendo os parâmetros editáveis pelo usuário), e também optou por não disponibilizar livremente o código-fonte (o que levou à criação da página no GitHub[1] apenas para servir de repositório da documentação, e não do código-fonte, e sem a funcionalidade de rastreamento de defeitos). Ainda assim, os documentos gerados têm potencial de ajudar a disseminar o uso do *Biopipeline*, contribuindo para os objetivos de pesquisadores interessados na predição de miRNA em genomas, além de possibilitar que terceiros mantenham o *software*, caso necessário.

1 Disponível em: https://github.com/marques-vinicius/biopipeline-docs

REFERÊNCIAS

ADAMS, Mark D. et al. Complementary DNA sequencing: expressed sequence tags and human genome project. **Science**, v. 252, n. 5013, p. 1651-1656, 1991. https://doi.org/10.1126/science.2047873

BECK, K. Embracing change with extreme programming. **Computer**, v. 32, p. 70-77, out, 1999. *https://doi.org/10.1109/2.796139*

BECK, K. M. *et al*. **Manifesto for Agile Software Development**. Agile Alliance. 2001. Disponível em <http://agilemanifesto.org/>. Acesso em: 18 dez. 2019.

BEEDLE, M. *et al*. SCRUM: An extension pattern language for hyperproductive software development. **Pattern languages of program design**, v. 4, p. 637-651, 1999.

BEREZIKOV, E. *et al.* Phylogenetic shadowing and computational identification of human microRNA genes. **Cell**, v. 120, p. 21–24, jan, 2005. https://doi.org/10.1016/j.cell.2004.12.031

BREIMAN, Leo. Random forests. **Machine learning**, v. 45, n. 1, p. 5-32, 2001. https://doi.org/10.1023/A:1010933404324

CARPER, Dana L. et al. DISCo-microbe: design of an identifiable synthetic community of microbes. **PeerJ**, v. 8, p. e8534, 2020. https://doi.org/10.7717/peerj.8534

CHEN, Z. Development of a Database Course for Bioinformatics. **Procedia Computer Science**, v. 9, p. 532-539, 2012. https://doi.org/10.1016/j.procs.2012.04.057

CHICCO, Davide. Ten quick tips for machine learning in computational biology. **BioData mining**, v. 10, n. 1, p. 1-17, 2017. https://doi.org/10.1186/s13040-017-0155-3

CORTES, C; VAPNIK, V. Support-Vector Networks. **Machine Learning**, v. 20, p. 273-297, mar, 1995. https://doi.org/10.1007/BF00994018

CUBILLOS-RUIZ, Juan R. et al. Reprogramming tumor-associated dendritic cells in vivo using miRNA mimetics triggers protective immunity against ovarian cancer. **Cancer research**, v. 72, n. 7, p. 1683-1693, 2012. https://doi.org/10.1158/0008-5472.CAN-11-3160

DE SOUZA, S. C. B. *et al*. Which documentation for software maintenance?. **Journal of the Brazilian Computer Society**, v. 12, n. 3, p. 31-44, 2006. https://doi.org/10.1007/BF03194494

DI LEVA, Gianpiero; CROCE, Carlo M. miRNA profiling of cancer. **Current Opinion in Genetics & Development**, v. 23, p. 3-11, mar, 2013. https://doi.org/10.1016/j.gde.2013.01.004

FILIPOWICZ, W. *et al*. Mechanisms of post-transcriptional regulation by microRNAs: are the answers in sight? **Nature Reviews – Genetics**, v. 9, p. 102-114, jan, 2008. https://doi.org/10.1038/nrg2290

FORWARD, A.; LETHBRIDGE, T. C. The relevance of software documentation, tools and technologies: a survey. In: **Proceedings of the 2002 ACM symposium on Document engineering**. p. 26-33, 2002. https://doi.org/10.1145/585058.585065

GEORGESON, Peter et al. Bionitio: demonstrating and facilitating best practices for bioinformatics command-line software. **GigaScience**, v. 8, n. 9, p. giz109, 2019. https://doi.org/10.1093/gigascience/giz109

GOMES, M. de S. *et al.* Genome-wide identification of novel microRNAs and their target genes in the human parasite Schistosoma mansoni. **Genomics**, v. 98, p. 96-111, mai, 2011. https://doi.org/10.1016/j.ygeno.2011.05.007

GRAD, Y. *et al.* Computational and experimental identification of C. elegans microRNAs. **Molecular Cell**, v. 11, p. 1253–1263, mai, 2003. https://doi.org/10.1016/S1097-2765(03)00153-9

HART, Steven N. Will digital pathology be as disruptive as genomics?. **Journal of pathology informatics**, v. 9, 2018. https://doi.org/10.4103/jpi.jpi_25_18

HIGHSMITH, J. A. **Adaptive software development: a collaborative approach to managing complex systems**. NY: Addison-Wesley, 2013.

HOWARD, A. A new RAD-based approach to commercial information systems development: the dynamic system development method. **Industrial Management & Data Systems**, v. 97, p. 175-177, ago, 1997. https://doi.org/10.1108/02635579710785456

JACOBSON, I. *et al*. The Unified Process. **IEEE Software**, v. 16, p. 96-102, jun, 1999.

JAIN, R. *et al*. MicroRNAs Enable mRNA Therapeutics to Selectively Program Cancer Cells to Self-Destruct. **Nucleic Acid Therapeutics**, v. 29, p. 285-296, jun, 2018. https://doi.org/10.1089/nat.2018.0734

JAWDAT, D. The Era of Bioinformatics. In: **2006 2nd International Conference on Information & Communication Technologies**. IEEE, 2006. p. 1860-1865. https://doi.org/10.1109/ICTTA.2006.1684672

JIANG, P. *et al*. MiPred: classification of real and pseudo microRNA precursors using random forest prediction model with combined features. **Nucleic acids research**, v. 35, n. suppl_2, p. W339-W344, 2007. https://doi.org/10.1093/nar/gkm368

JONAS, S; IZAURRALDE, E. Towards a molecular understanding of microRNA-mediated gene silencing. **Nature Reviews – Genetics**, v. 16, p. 421-433, jun, 2015. https://doi.org/10.1038/nrg3965

KARIMZADEH, Mehran; HOFFMAN, Michael M. Top considerations for creating bioinformatics software documentation. **Briefings in Bioinformatics**, v. 19, n. 4, p. 693-699, 2018. https://doi.org/10.1093/bib/bbw134

KIPYEGEN, N. J.; KORIR, W. P. K. Importance of Software Documentation. **International Journal of Computer Science Issues,** v. 10, p. 223-228, set, 2013.

KUMAR, A.; CHORDIA, N. Role of bioinformatics in biotechnology. **Research and Reviews in Biosciences**, v. 12, n. 1, p. 1-6, 2017.

KUMAR, S.; DUDLEY, J. Bioinformatics software for biologists in the genomics era. **Bioinformatics**, v. 23, n. 14, p. 1713-1717, 2007. https://doi.org/10.1093/bioinformatics/btm239

LAI, E.C. *et al.* Computational identification of Drosophila microRNA genes. **Genome Biology**, v. 4, p. R42.1-R42.20, jun 2003. https://doi.org/10.1186/gb-2003-4-7-r42

LAWLOR, Brendan; WALSH, Paul. Engineering bioinformatics: building reliability, performance and productivity into bioinformatics software. **Bioengineered**, v. 6, n. 4, p. 193-203, 2015. https://doi.org/10.1080/21655979.2015.1050162

LEE, R. C. *et al*. The C. elegans heterochronic gene lin-4 encodes small RNAs with antisense complementarity to lin-14. **cell**, v. 75, n. 5, p. 843-854, 1993. https://doi.org/10.1016/0092-8674(93)90529-Y

LETHBRIDGE, T. G. *et al*. How software engineers use documentation: The state of the practice. **IEEE Software**, v. 20, p. 35-39, dez, 2003. https://doi.org/10.1109/MS.2003.1241364

LEPREVOST, Felipe da Veiga et al. On best practices in the development of bioinformatics software. **Frontiers in genetics**, v. 5, p. 199, 2014. https://doi.org/10.3389/fgene.2014.00199

LIM, L.P. *et al.* The microRNAs of Caenorhabditis elegans. **Genes & Development**, v. 17, p. 991–1008, fev, 2003. https://doi.org/10.1101/gad.1074403

MANGUL, Serghei et al. Challenges and recommendations to improve the installability and archival stability of omics computational tools. **PLoS biology**, v. 17, n. 6, p. e3000333, 2019. https://doi.org/10.1371/journal.pbio.3000333

MENDES, N. D. *et al.* Current tools for the identification of miRNA genes and their targets. **Nucleic Acids Research**, v. 37, p. 2419-2433, mar, 2009. https://doi.org/10.1093/nar/gkp145

NAM,J.-W. *et al.* Human microRNA prediction through a probabilistic co-learning model of sequence and structure. **Nucleic Acids Research**, v. 33, p. 3570–3581, jun, 2005. https://doi.org/10.1093/nar/gki668

NG, K. L. S.; MISHRA, S. K. De novo SVM classification of precursor microRNAs from genomicpseudo hairpins using global and intrinsic folding measures. **Bioinformatics**, v. 23, p. 1321-1330, jan, 2007. https://doi.org/10.1093/bioinformatics/btm026

PARNAS, D. L. Precise Documentation: The Key To Better Software. In: **The Future of Software Engineering**. Springer, Berlin, Heidelberg, 2011. p. 125-148. https://doi.org/10.1007/978-3-642-15187-3_8

PASQUINELLI, A. E. et al. Conservation of the sequence and temporal expression of let-7 heterochronic regulatory RNA. **Nature**, v. 408, n. 6808, p. 86-89, 2000. https://doi.org/10.1038/35040556

PINDIPROLU, S. K. S. *et al*. Nanocarrier based approaches for targeting breast cancer stem cells. **Artificial Cells, Nanomedicine, and Biotechnology**, v. 46, p. 885-898, ago, 2017. https://doi.org/10.1080/21691401.2017.1366337

PFEFFER, S. *et al.* Identification of microRNAs of the herpesvirus family. **Nature Methods**, v. 2, p. 269–276, abr, 2005. https://doi.org/10.1038/nmeth746

REINHART, B. J. et al. The 21-nucleotide let-7 RNA regulates developmental timing in Caenorhabditis elegans. **nature**, v. 403, n. 6772, p. 901-906, 2000. https://doi.org/10.1038/35002607

ROYCE, W. Managing the development of large software systems: Concepts and techniques. In: **Proceedings of the 9th international conference on Software Engineering**. 1987. p. 328-338.

SAELENS, Wouter et al. A comparison of single-cell trajectory inference methods: towards more accurate and robust tools. **BioRxiv**, p. 276907, 2018. *https://doi.org/10.1101/276907*

SHARON, D. Meeting the challenge of software maintenance. **IEEE Software**, v. 13, n. 1, p. 122-125, 1996. *https://doi.org/10.1109/52.476304*

SHENG, Y. *et al.* Mammalian MicroRNA prediction through a support vector machine model of sequence and structure. **PLoS ONE**, v. 2, p. e946.1-e946.15, set, 2007. https://doi.org/10.1371/journal.pone.0000946

SUMONJA, Neven et al. Automated feature engineering improves prediction of protein–protein interactions. **Amino acids**, v. 51, n. 8, p. 1187-1200, 2019. https://doi.org/10.1007/s00726-019-02756-9

TASCHUK, Morgan; WILSON, Greg. Ten simple rules for making research software more robust. 2017. https://doi.org/10.1371/journal.pcbi.1005412

VERMA, Dhawal et al. Lack of software engineering practices in the development of bioinformatics software. **ICCGI**, v. 2013, p. 57-62, 2013.

XUE, C. *et al.* Classification of real and pseudo microRNA precursors using local structure-sequence features and support vector machine. **BMC Bioinformatics**, v. 6, p. 310.1-310.7, dez, 2005. https://doi.org/10.1186/1471-2105-6-310

ZOBOLAS, John et al. UniBioDicts: Unified access to biological dictionaries. 2020. *https://doi.org/10.20944/preprints202007.0586.v1*

ANEXO – NORMA DO PERIÓDICO “BRIEFINGS IN BIOINFORMATICS”

OXFORD ACADEMIC

Briefings in Bioinformatics

Instructions to Authors

Authors will be asked to 'sign' their licence to publish online

Please contact Alison Bentley in the Editorial Office at briefings@oup.com with any submission queries in the first instance.

Journal online-only since 2018

Briefings in Bioinformatics is published online-only. Authors are able to access their published articles and content via the Oxford Academic platform. All print editions have been discontinued.

Author Self-Archiving/Public Access Policy

For information about this journal's policy, please visit our Author Self-Archiving policy page.

Aims

The aim of *Briefings in Bioinformatics* is to provide an indispensable resource for the experimental practitioner seeking awareness of the disparate sources of data and analytical tools of contemporary biology, biotechnology and medicine based on the molecular level. This includes all areas of genomics, proteomics, lipidomics, glycomics, metabolomics, interactomics and network biology, imaging, systems biology, phenomics, chemoinformatics, computational biology and clinical/medical informatics that have a molecular foundation to the study. Large-scale instrumentation and computerisation is reducing the time that needs to be spent in the laboratory.

Instead, the rate-limiting step is the analysis and interpretation of data. The journal provides topical reviews of new methodologies as they become established. Reviews of the related education and training in the biomedical sciences are included in the scope.

The Editors welcome the submission of review articles and case studies for publication. We publish reviews of clearly defined subject areas for both experimental biologists and for bioinformatics specialists. Reviews may be broader or more narrowly focussed but must cover a variety of approaches to a very well specified biological problem or research area. The underlying concepts must be clearly explained along with the selection of the correct tools for a problem, their limitations, and the interpretation of the results. Descriptions of existing analytical solutions of biological problems should be transparent and user friendly and may include mathematical and statistical methods for high dimensional and high throughput data. We do not publish work on new methods that have not yet been described elsewhere except in the context of a wider review of that specific subject area.

Methodological approaches of interest include software comparison and benchmarking, data cleaning and curation, accuracy of predicted and extracted information, ontologies and text-mining, solutions that allow for the large-scale analysis of biological data in reasonable time (high performance computing solutions and cloud systems), standards, training and change management activities, and the determination of causal relationships from data. Ontologies for semantic-based analysis of molecular data and interaction networks, methods and tools for the automatic or semiautomatic annotation of biological data with terms extracted from ontologies, and methods and tools for enrichment analysis are all relevant to the objectives. Articles focusing on illuminating bottleneck problems in important bioinformatics approaches will be especially helpful to readers. There should be more studies focussing on the replicability and reproducibility of bioinformatics methodologies and realistic sample sizes required for this. We encourage papers that provide an independent evaluation of software tools for common tasks. In bioinformatics there are typically multiple different tools and parameter settings that can be selected for a given problem and these should be evaluated and compared to one another using simulation if appropriate and empirical real datasets if available, preferably by authors who are not originators of the software. As the science advances details of what is important changes and the Editors, Editorial Board and Reviewers will be flexible in their policies.

Types of Manuscript

Submissions of the following types are accepted for review in the Journal:

- Reviews: surveys of previously published research in specified areas giving an overview and making recommendations for the optimal approaches to problem solving. Do not include new results from your own work (2000-7000 words)
- Problem solving protocols: protocols for solving a specific problem using different sets of programs (2000-5000 words)
- Case studies in biological research applied to clinical practice (2000-5000 words)
- Opinion articles: topical or controversial areas that do not warrant a full review (500-1000 words)
- Letters to the Editors: critique of an article previously published in the journal (500-1000 words)

- Software and website reviews: these must be written by authors other than the originators (500-1000 words)
- Book reviews (500-1000 words)

If your article does not focus on bioinformatics you may consider submitting to *Briefings in Functional Genomics* which publishes high quality peer reviewed articles that focus on the use, development or exploitation of genomic approaches, and their application to all areas of biological research. Please note that *Briefings in Bioinformatics* is a review journal and does not publish articles that are pure original research. Authors may instead wish to consider submitting to *Biology Methods and Protocols*.

Submission

All material to be considered for publication in *Briefings in Bioinformatics* must be submitted in electronic form via the journal's online submission system at Manuscript Central. New authors should create an account prior to submitting a manuscript for consideration. Once you have prepared your manuscript according to the instructions below, instructions on how to submit your manuscript online can be found.

Submissions should be typewritten, double-spaced, on A4 or US letter paper and supplied electronically as Word or rich-text files.

This journal is supported by the OUP LaTeX template. If you are using LaTeX to write your paper, please see the 'OUP LaTeX template' section on this page to find instructions and to access the template:
https://academic.oup.com/journals/pages/authors/preparing_your_manuscript.

Articles received by the Editors will undergo a pre-screening process to increase the efficiency of the publication process. Papers that are considered to be of minor importance to the readership of the Journal are not reviewed. Papers selected for review are sent out to three referees, who agree to undertake the refereeing within a short period of time.

Licence and Permissions

It is a condition of publication that authors grant an exclusive licence to Oxford Journals or the society of ownership. This ensures that requests from third parties to reproduce articles are handled efficiently and consistently, and will also allow the article to be as widely disseminated as possible. In assigning copyright, authors may use their own material in other publications, provided that the journal is acknowledged as the original place of publication, and Oxford Journals is notified in writing and in advance.

Upon receipt of accepted manuscripts at Oxford Journals authors will be invited to complete an online copyright licence to publish form.

Please note that by submitting an article for publication you confirm that you are the corresponding/submitting author and that Oxford University Press ("OUP") may retain your email address for the purpose of communicating with you about the article. You agree to notify OUP immediately if your details change. If your article is accepted for publication OUP will contact you using the email address you have used in the registration process. Please note that OUP does not retain copies of rejected articles.

Work submitted for publication must be original, previously unpublished, and not under consideration for publication elsewhere. If previously published figures, tables, or parts of text are to be included, the copyright-holder's permission must have been obtained prior to submission.

The author bears the responsibility for checking whether material submitted is subject to copyright or ownership rights, e.g. photographs, illustrations, trade literature and data. Where use is so restricted, the Editors and the Publisher must be informed with the submission of the material.

Third-Party Content in Open Access papers

If you will be publishing your paper under an Open Access licence but it contains material for which you do not have Open Access re-use permissions, please state this clearly by supplying the following credit line alongside the material:

Title of content

Author, Original publication, year of original publication, by permission of [rights holder]

This image/content is not covered by the terms of the Creative Commons licence of this publication. For permission to reuse, please contact the rights holder.

Open Access

Briefings in Bioinformatics offers the option of publishing under either a standard licence or an open access licence. Please note that some funders require open access publication as a condition of funding. If you are unsure whether you are required to publish open access, please do clarify any such requirements with your funder or institution.

Should you wish to publish your article open access, you should select your choice of open access licence in our online system after your article has been accepted for publication. You will need to pay an open access charge to publish under an open access licence.

Details of the open access licences and open access charges.

OUP has a growing number of Read and Publish agreements with institutions and consortia which provide funding for open access publishing. This means authors from participating institutions can publish open access, and the institution may pay the charge. Find out if your institution is participating.

Availability of Data and Materials

Where ethically feasible, *Briefings in Bioinformatics* strongly encourages authors to make all data and software code on which the conclusions of the paper rely available to readers. Authors are required to include a Data Availability Statement in their article.

We suggest that data be presented in the main manuscript or additional supporting files, or deposited in a public repository whenever possible. For information on general repositories for all data types, and a list of recommended repositories by subject area, please see Choosing where to archive your data.

Data Availability Statement

The inclusion of a Data Availability Statement is a requirement for articles published in *Briefings in Bioinformatics*. Data Availability Statements provide a standardised format for readers to

understand the availability of data underlying the research results described in the article. The statement may refer to original data generated in the course of the study or to third-party data analysed in the article. The statement should describe and provide means of access, where possible, by linking to the data or providing the required unique identifier.

The Data Availability Statement should be included in the endmatter of your article under the heading ‘Data availability’.

More information and examples of Data Availability Statements.

Data Citation

Briefings in Bioinformatics supports the Force 11 Data Citation Principles and requires that all publicly available datasets be fully referenced in the reference list with an accession number or unique identifier such as a digital object identifier (DOI). Data citations should include the minimum information recommended by DataCite:

- [dataset]* Authors, Year, Title, Publisher (repository or archive name), Identifier

*The inclusion of the [dataset] tag at the beginning of the citation helps us to correctly identify and tag the citation. This tag will be removed from the citation published in the reference list.

Preprint policy

Authors retain the right to make an Author’s Original Version (preprint) available through various channels, and this does not prevent submission to the journal. For further information see our Online Licensing, Copyright and Permissions policies. If accepted, the authors are required to update the status of any preprint, including your published paper’s DOI, as described on our Author Self-Archiving policy page.

ORCID

Briefings in Bioinformatics requires submitting authors to provide an ORCID iD at submission to the journal. More information on ORCID and the benefits of using an ORCID iD is available. If you do not already have an ORCID iD, you can register for free via the ORCID website.

Authorship

All persons designated as authors should qualify for authorship. The order of authorship should be a joint decision of the co-authors, and should be agreed upon before submission to the journal. Each author should have participated sufficiently in the work to take public responsibility for the content. Authorship credit should be based on substantial contribution to conception and design, execution, or analysis and interpretation of data. All authors should be involved in drafting the article or revising it critically for important intellectual content, and must have read and approved the final version of the manuscript. Assurance that all authors of the paper have fulfilled these criteria for authorship should be given in the covering letter.

Competing Interests

At the point of submission, *Briefings in Bioinformatics'* policy requires that each author reveal any financial interests or connections, direct or indirect, or other situations that might raise the question of bias in the work reported or the conclusions, implications, or opinions stated - including pertinent commercial or other sources of funding for the individual author(s) or for the associated

department(s) or organization(s), personal relationships, or direct academic competition. When considering whether you should declare a conflicting interest or connection please consider the conflict of interest test: Is there any arrangement that would embarrass you or any of your co-authors if it was to emerge after publication and you had not declared it?

It is mandatory for authors to declare any potential conflicts of interest. You will be prompted to declare these when you make your submission on Manuscript Central. As the submitting author, it is your responsibility to be aware of your co-authors' conflicts of interest and to declare these. If, however, you are unable to speak on behalf of your co-authors, you and your co-authors will need to complete and return a Conflict of Interest form (Conflict of Interest Form) to the editorial office (fax: +44 (0) 1865 355907, email: briefings@oup.com). It is the Corresponding author's responsibility to ensure that all authors adhere to this policy. Please note that your manuscript will not be peer-reviewed until all conflicts have been declared.

If the manuscript is published, Conflict of Interest information will be communicated in a statement in the published paper.

Material Disclaimer

The opinions expressed in *Briefings in Bioinformatics* are those of the authors and contributors, and do not necessarily reflect those of the editors, the editorial board, Oxford University Press or the organization to which the authors are affiliated.

Crossref Funding Data Registry

In order to meet your funding requirements authors are required to name their funding sources, or state if there are none, during the submission process. For further information on this process or to find out more about CHORUS, visit the CHORUS initiative.

Preparation of Manuscripts

General

Papers must be clearly written in English.

All articles should be submitted with a title page. The title page should include the following: (1) the title, (2) the name(s) of authors, (3) the authors' institutional affiliations (4) the telephone number, fax number, and e-mail address of the corresponding author.

All review articles and software/website reviews should be accompanied by a short abstract, outlining the aims and subject matter, up to 5 key points (see below) and up to six keywords should be provided for indexing purposes.

All papers, articles and reviews should be accompanied by a short (about 30 words) description of the author(s) and, if appropriate, the organisation of which he or she is a member.

Authors should avoid the use of language or slang which is not in keeping with the academic and professional style of the Journal. Authors should not also seek to use the Journal as a vehicle for marketing any specific product or service.

Authors should follow the conventions of the CSE Style Manual (Council of Science Editors, Reston, VA, 2006). Chemical Abstracts and its indices should be followed for chemical names. For biochemical terminology the recommendations issued by the IUPAC-IUB Commission on Biochemical Nomenclature, as given in Biochemical Nomenclature and Related Documents, published in 1992 by the Biochemical Society, UK should be followed. For enzymes, the recommended name assigned by the IUPAC-IUB Committee on Biochemical Nomenclature, 1978, as given in Enzyme Nomenclature, published by Academic Press, New York, 1992 should be used. Wherever possible, the recommended SI units should be used. Genotypes should be italicised. Phenotypes should not be italicised. For bacterial agents nomenclature Demerec et al. (1966) Genetics, 54, 61-76 should be followed and The Trends In Genetics Nomenclature Guide (1998), Elsevier, Cambridge. Titles of organisations etc should be written out first in full followed by the organisation's initials in brackets, eg. Food and Drug Administration (FDA) and thereafter the initials only should be used.

References

All references must be cited in the text and should be denoted using numbers in square brackets before the punctuation., e.g. [1, 3–5]. References should be numbered consecutively in the order in which they appear.

Style in the References section should be as follows (this is to be consistent with PubMed):

1. Attwood T.K. The role of pattern databases in sequence analysis. Brief Bioinform 2000;1:45–59.
2. Long HC, Blatt MA, Higgins MC et al.. Medical Decision Making. Boston: Butterworth-Heinemann, 1997.
3. Manners T, Jones R, Riley M. Relationship of overweight to haitus hernia and reflux oesophagitis. In: Newman W (ed). The Obesity Conundrum. Amsterdam: Elsevier Science, 1997,352–74.
4. Hou Y, Qiu Y, Vo NH et al. 23-O derivatives of OMT: highly active against H. influenzae. In: Programs and Abstracts of the Forty-third Interscience Conference on Antimicrobial Agents and Chemotherapy, Chicago, IL, 2003. Abstract F-1187, p.242. American Society for Microbiology, Washington, DC, USA.
5. Public Health Laboratory Service. Antimicrobial Resistance in 2000: England and Wales. http://www.hpa.org.uk./infections/topics_az/antimicrobial_resistance/amr.pdf (7 January 2004, date last accessed).

If there are four or more authors, then use the first three followed by et al. Papers in preparation or submitted for publication should not be in the reference list.

Authors are asked to ensure the references to named people and/or organisations are accurate and without libellous implications.

Key Points

Key Points are displayed at the end of the article, and should consist of 3-5 brief sentences. Key Points are an opportunity to summarise the key messages of the article and can include anything that is felt to be important for the reader to particularly note.

Funding & NIH Funding

Details of all funding sources for the work in question should be given in a separate section entitled 'Funding'. This should appear before the 'Acknowledgements' section.

Oxford Journals will deposit all NIH-funded articles in PubMed Central. See Author Resources for details. Authors must ensure that manuscripts are clearly indicated as NIH-funded using the guidelines below.

The following rules should be followed:

- The sentence should begin: ‘This work was supported by …’
- The full official funding agency name should be given, i.e. ‘National Institutes of Health’, not ‘NIH’ (full RIN-approved list of UK funding agencies) Grant numbers should be given in brackets as follows: ‘[grant number xxxx]’
- Multiple grant numbers should be separated by a comma as follows: ‘[grant numbers xxxx, yyyy]’
- Agencies should be separated by a semi-colon (plus ‘and’ before the last funding agency)
- Where individuals need to be specified for certain sources of funding the following text should be added after the relevant agency or grant number 'to [author initials]'.

An example is given here: ‘This work was supported by the National Institutes of Health [AA123456 to C.S., BB765432 to M.H.]; and the Alcohol & Education Research Council [hfygr667789].’

Figures

Figures should be supplied in an electronic format at a suitable size for printing with the following resolutions: 600 dots per inch (dpi) for line drawings and combinations; 300 dpi for greyscale. Please ensure that the prepared electronic image files print at a legible size and are of a high quality for publication. Online colour figures are free of charge.

Figures should be referred to in the text and numbered consecutively. They should be supplied separately from the main body of the text, with their approximate final positions, and legends marked within the main text. Figure legends should describe the figure content and should be understood independently from the text. Abbreviations should be avoided in figures. If abbreviations or symbols are used in the figures they should be explained in the figure legend, if they have not been explained in a key.

Line charts, bar charts and pie charts should be two-dimensional, with single categories, a generous margin, and grey-scaled backgrounds (with a 25% tint). Appropriate scales should be used and sources should be quoted. Bar charts should have two categories or more and at least five observations; otherwise the data should be presented in a table. Horizontal lines should be used to mark the major values on the y-axis. Line charts should show changes over long time spans and should have at least ten observations. Pie charts should be used to show proportions and have a minimum of four segments, and a maximum of twelve.

Photographs

Photos can be supplied as good quality black and whites. They must be of sufficient quality with respect to detail, contrast and fineness of grain to withstand the unavoidable loss of contrast inherent in the printing process. Their approximate final positions should be indicated in the text. Electronic copies of photos should be provided, where possible, as GIF, TIFF or BITMAP files (minimum acceptable resolution 300 dpi).

Internet Screen Dumps

Internet screen dumps must be provided in greyscale and should have a white background to increase the contrast between the illustration and the background. They should be provided electronically as BITMAP, with a minimum acceptable resolution of 300 dpi. Their approximate final positions should be indicated in the margin of the text. Authors should be aware that graphics supplied with low resolution are not guaranteed to reproduce well and should be avoided whenever possible.

Tables

Tables should be submitted in electronic form, preferably in MS Word or Excel. Tables should be referred to in the text and numbered consecutively. They should be supplied separately from the main body of the text, with their approximate final positions indicated in the text. Each column should have a short heading and, where appropriate, the units should be stated. Table legends should describe the content and should be understood independently from the text. Data columns should be right-hand aligned, or aligned by decimal place, where appropriate; data should be sorted where possible. Footnotes should be included on the same pages as the tables themselves and should be used to explain any abbreviations used in the table and denote them by letter. Footnotes should also be used to quote sources.

Proofs

All manuscripts will undergo some editorial modification, so it is important to check proofs carefully. PDF page proofs will be sent via e-mail to the corresponding author for checking. To avoid delays in publication, proofs should be checked and returned within 48 hours. Corrections should be returned by annotated PDF, e-mail or fax. Extensive changes to the text may be charged to the author.

English Language Editing

Language editing, if your first language is not English, to ensure that the academic content of your paper is fully understood by journal editors and reviewers is optional. Language editing does not guarantee that your manuscript will be accepted for publication. Further information on this service. Several specialist language editing companies offer similar services and you can also use any of these. Authors are liable for all costs associated with such services.

LaTeX

Please see the guidance on LaTeX files and formatting which contains the OUP LaTeX template.

Please see also further information on working with LaTeX.

Oxford University Press is a department of the University of Oxford. It furthers the University's objective of excellence in research, scholarship, and education by publishing worldwide