UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Banco não-relacional para armazenamento de dados genéticos

Rodrigo de Andrade Santos Weigert

São Carlos – SP

Banco não-relacional para armazenamento de dados genéticos

Rodrigo de Andrade Santos Weigert

Orientadora: **Profa. Dra. Elaine Parros Machado de Sousa**

Monografia final de conclusão de curso apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.
Área de Concentração: Banco de dados

USP – São Carlos
Novembro de 2019

Weigert, Rodrigo de Andrade Santos
Banco não-relacional para armazenamento de dados genéticos / Rodrigo de Andrade Santos Weigert. - São Carlos - SP, 2019.
52 p.; 29,7 cm.

Orientadora: Elaine Parros Machado de Sousa.
Monografia (Graduação) - Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos - SP, 2019.

1. Banco de dados. 2. NoSQL. 3. MongoDB. 4. SNP. I. Sousa, Elaine Parros Machado de. II. Instituto de Ciências Matemáticas e de Computação (ICMC/USP). III. Título.

“Nem Júpiter traz de volta
oportunidades perdidas.”
(Phaedrus)

RESUMO

WEIGERT, R. A. S.. **Banco não-relacional para armazenamento de dados genéticos**. 2019. 52 f. Monografia (Graduação) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

Este trabalho explora a utilização de um sistema de banco de dados não-relacional para gerenciar dados oriundos de processos de sequenciamento e genotipagens comerciais, particularmente os chamados SNPs (pronunciado "snips", do inglês *single-nucleotide polymorphisms* – polimorfismos de nucleotídeo único). Foi proposta uma solução de armazenamento orientado a documentos capaz de lidar com o grande volume de dados e flexível o bastante para aceitar a importação e exportação de arquivos em diferentes formatos, além de oferecer suporte eficiente a consultas relevantes na área de aplicação. Foi utilizado o *MongoDB* para um *backend* no topo do qual foi implementada uma aplicação em *Python* com interface por linha de comando. Testes foram realizados para avaliação do desempenho do sistema em consultas, exportações e importações, estas avaliadas tanto com relação a tempo quanto a espaço ocupado em comparação com os arquivos originais. Os resultados indicaram que a solução proposta é viável, sobretudo quando versatilidade e poder de consulta forem mais prioritários do que eficiência, a qual sofre algum impacto.

Palavras-chave: Banco de dados, NoSQL, MongoDB, SNP.

ABSTRACT

WEIGERT, R. A. S.. **Banco não-relacional para armazenamento de dados genéticos**. 2019. 52 f. Monografia (Graduação) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

This paper explores a way to use a non-relational database system to manage data originating from sequencing and commercial genotyping processes, particularly single-nucleotide polymorphisms (SNPs, pronounced “snips”). A document-oriented solution capable of dealing with the large volume of data, flexible enough to allow importing and exporting of files in different formats, and which supports relevant queries from the field of application was proposed. *MongoDB* was used for a backend on top of which a Python application with a command-line interface was implemented. Tests were performed in order to evaulate the system’s performance for queries, exports and imports, the evaulation of the latter taking into account not only the time required, but also the storage space used in comparison to the original files. Results indicated that the proposed solution is viable mainly when versatility and querying capabilities have higher priority over efficiency, which is impacted to an extent.

Key-words: Databases, NoSQL, MongoDB, SNP.

LISTA DE ILUSTRAÇÕES

LISTA DE ARQUIVOS EXEMPLO

SUMÁRIO

Capítulo 1

INTRODUÇÃO

1.1 Motivação e Contextualização

Profissionais ligados à bioinformática, além de lidar com grandes volumes de dados, comumente precisam trabalhar com os mesmos em uma variedade de formatos. Este cenário é consequência da profusão de técnicas, ferramentas e aplicações envolvidas neste relativamente jovem mas importantíssimo campo multidisciplinar da ciência. Com o avanço rápido da tecnologia, e aprimoramento dos processos envolvidos, como o de sequenciamento de DNA, é importante que se mantenham os esforços para que a parte informática se mantenha parelha com a biológica.

A pecuária é uma das áreas beneficiadas pelos avanços da biologia. Graças à genômica e seus estudos de associação genética, é possível desenvolver programas de criação de gado mais efetivos na ampliação e manutenção de características de importância econômica, como produção de leite e resistência a doenças (KOOPAEE; KOSHKOIYEH, 2014). Tais estudos estão intimamente relacionados ao conceito de polimorfismo de nucleotídeo único, ou **SNP** (pronunciado "snip"), do inglês *single-nucleotide polymorphism*. Simplificadamente, um SNP é uma diferença em um único nucleotídeo (A, T, C, G) entre sequências de DNAs de dois indivíduos de uma mesma espécie ou dois cromossomos homólogos do mesmo indivíduo. Por exemplo, se na maior parte da população de certa espécie encontra-se a sequência ACATAGC, mas em 5% dos indivíduos encontra-se ATATAGC, então diz-se que há um SNP na posição sublinhada e C e T são chamados de **alelos**. Em certos casos, um único SNP pode ser o responsável por grandes efeitos, como, por exemplo, anemia falsiforme em humanos (CLANCY, 2008).

Segundo o Prof. Dr. Ricardo Ventura[1] da Faculdade de Medicina Veterinária e Zootecnia da Universidade de São Paulo (FMVZ-USP), profissionais que fazem uso de dados moleculares em suas pesquisas, em particular os oriundos de plataformas de sequenciamento genômico, podem se deparar com cenários onde o volume de dados alcança a ordem das dezenas de milhões, armazenados em arquivos de dados que rapidamente ultrapassam o gigabyte, chegando, às vezes, a exceder 150 GB mesmo após compressão. Um tipo comum de arquivo é aquele que, dado um conjunto de SNPs (um **mapa**[2]), traz dados sobre um certo conjunto de **amostras** de material genético analisadas. Para cada par amostra-SNP, o dado diz respeito ao alelo que a amostra

[1] <http://vnp.fmvz.usp.br/docentes/ricardo-vieira-ventura/>

[2] Também chamado de painel. Este trabalho não usará esse termo.

apresenta na posição do SNP. Como há varios formatos diferentes de arquivo, há variação com relação a em que exatamente tal dado consiste: pode ser apenas um caractere[3] representando o alelo presente (no exemplo anterior, C ou T), ou algo mais complexo, que traga também, por exemplo, informações sobre confiança/probabilidade de corretude. Em cenários típicos, um arquivo pode conter centenas ou milhares de amostras, e milhões de SNPs.

As amostras supracitadas podem ser associadas a **indivíduos**, aos quais também pode ser de interesse vincular outros tipos de arquivos, por exemplo imagens, áudios e vídeos que sirvam para expor uma característica em estudo, ou arquivos que contenham sequências brutas de DNA. A fim de gerenciar tamanha variedade de arquivos, não raramente pesquisadores utilizaram apenas o *file system*, de forma que a organização e a consulta de dados tenham de ser realizadas manualmente. Existem, no entanto, sistemas com o objetivo de aprimorar o trabalho nesse contexto, como é o caso do *TheSNPPit* (GROENEVELD; LICHTENBERG, 2016), uma solução de alta eficiência que tem como um dos principais objetivos lidar com a proliferação de cópias de arquivos produzidas para processos de filtragem. Este trabalho, realizado em parceria com o Prof. Ricardo, possui, fundamentalmente, similaridades com o *TheSNPPit*, contudo apresenta foco diferente, voltado para a flexibilidade de ser possível lidar com essa variedade de arquivos e formatos no mesmo lugar (*TheSNPPit* trabalha com apenas um), e para o oferecimento de maior poder de consulta sobre dados de SNPs, amostras e indivíduos, tudo isso enquanto tenta-se reduzir o quanto possível a perda de performance natural a soluções que visam maior grau de generalidade.

1.2 Objetivos

Este trabalho tem como objetivo principal a criação de um sistema para gerenciamento de dados de SNPs e amostras analisadas. O mesmo deve atender aos seguintes requisitos, elaborados com auxílio do Prof. Ricardo:

1. Ser flexível a ponto de permitir a importação e exportação de dados a partir de variados formatos de arquivos de texto.

2. Ser capaz de lidar com grandes volumes de dados sem que o desempenho se deteriore proibitivamente.

3. Responder eficientemente a consultas relevantes sobre SNPs, mapas, amostras e indivíduos nele contidos.

4. Ser capaz de armazenar e indexar arquivos “anexos” relacionados a indivíduos, como imagens e arquivos brutos de sequenciamento.

[3] Na verdade, um par de caracteres, já que trabalha-se com espécies diploides, isto é, que carregam dois exemplares de cada cromossomo em suas células.

1.3 Organização

No capítulo 2 será apresentada uma discussão sobre as escolhas do sistema de banco de dados e a linguagem de programação utilizadas neste projeto.

No capítulo 3, os dados tratados no projeto e as consultas que deseja-se atender serão expostos com mais detalhes. Além disso, a solução desenvolvida e os resultados de testes que demonstrem sua funcionalidade e performance serão discutidos.

O último capítulo discutirá os objetivos alcançados e possíveis trabalhos futuros, além de trazer a análise sobre o curso de graduação e a jornada do autor por ele.

Capítulo 2

MÉTODOS, TÉCNICAS E TECNOLOGIAS UTILIZADAS

A utilização de um banco de dados não-relacional, também chamado *NoSQL*, foi cogitada desde o início do projeto em razão de sua alta flexibilidade e simplicidade, a qual harmoniza bem com a ideia de buscar um grau de generalização que consiga lidar com a importação dos diferentes formatos de arquivos de dados. O movimento *NoSQL* (hoje interpretado como "Not Only SQL") questiona a ideia de que o gerenciamento de dados modelado pelas relações tabulares do modelo relacional é adequada para todos os propósitos, buscando alternativas (STRAUCH, 2012). Dependendo da alternativa escolhida, podemos identificar quatro tipos principais de bancos não-relacionais, conforme discutem Perkins, Redmond e Wilson (2018):

- *Key-value stores*: armazena pares chave-valor, tal qual *maps* ou *hashtables* em linguagens de programação. Podem ser úteis em diversos cenários, mas deixam a desejar a partir do momento em que surgem necessidades complexas de consulta e agregação.
- *Columnar databases*: enquanto os relacionais são orientados a linhas, estas são orientadas a colunas, armazenando juntos os dados de cada uma. Isso possibilita baixo custo na inserção de novas colunas e que duas linhas possuam colunas diferentes associadas, o que elimina o custo de armazenamento de valores nulos. Considerada um meio caminho entre relacional e *key-value*.
- *Document stores*: armazena documentos, os quais são compostos por um campo com identificador único e quaisquer outros campos adicionais, de tipos variados de dados, inclusive outros documentos. A possibilidade de aninhamento confere às *document stores* alta flexibilidade. Sistemas desse tipo apresentam abordagens variadas com relação indexação, consultas *ad hoc*, replicação, consistência e outras decisões de design.
- *Graph databases*: armazenam dados na forma de nós e arestas representando o relacionamento entre eles, sendo, assim, apropriadas para dados muito interconectados. O poder desses sistemas está na capacidade de travessia dos nós pelas arestas – ambos capazes de terem propriedades associadas a si – para responder consultas.

A maior desvantagem dos não-relacionais, para o caso deste projeto, é provavelmente a perda de garantias de consistência, as quais precisariam ser migradas para a aplicação, ficando

mais complicadas de realizar. Esse problema, no entanto, é em parte compensado pela simplicidade do modelo advinda da flexibilidade do banco, e também pelo fato de que está-se assumindo que operações de atualização sejam raras e pontuais. Outra desvantagem notável, porém não irremediável, é o maior *overhead* de espaço utilizado devido à ausência (ou opcionalidade) de *schema*.

Entre os gêneros de bancos *NoSQL* citados, optou-se por procurar uma solução nas *document stores*. Os outros tipos puderam ser rapidamente descartados: *key-value stores* são *document stores* mais simples e menos versáteis; não existe modelagem clara em vértices e arestas para justificar a escolha de uma *graph database*; e o mesmo se aplica às *columnar databases*.

Em geral, as *document stores* cogitadas para o *backend* diferiam entre si muito mais em detalhes do que em funcionalidades chave. Para listar exemplos de relevância para o projeto, verificou-se que quase todas:

- aceitam índices secundários, essenciais para a garantir que as consultas de maior relevância possam ser atendidas eficientemente;
- são *schema-less* (mas permitem algum grau de validação dos dados), o que descomplica o armazenamento dos dados variados que diferentes formatos de arquivos de entrada podem trazer;
- aceitam campos, índices e consultas sobre *arrays*, sem os quais a modelagem discutida no capítulo seguinte não teria sido possível.

As diferenças maiores entre os bancos, quando apareciam, eram pouco relevantes para os propósitos aqui em questão. No final, fatores como a facilidade de uso e a altíssima popularidade (DB-ENGINES, 2019) contribuíram para a escolha do *MongoDB*. Entre as outras alternativas consideradas, estavam:

- *Couchbase*: traz um sistema de índices poderoso[1] mas de manipulação mais complexa. Não oferece um modo nativo para lidar com importação de arquivos binários na íntegra no banco (é preciso uma solução customizada[2]), o que impacta o objetivo 4 mencionado na seção 1.2.
- *CouchDB*: a construção de seus índices é baseada em *views MapReduce*[3], o qual, por não combinar com o domínio atual, adicionaria complexidade evitável ao projeto. Além disso, está voltado principalmente para aplicações web, como sugere o fato de que sua API é HTTP[4].

[1] <https://docs.couchbase.com/server/4.5/indexes/indexing-overview.html>. Acesso em 30/10/2019.
[2] <https://blog.couchbase.com/storing-blobs-in-couchbase-for-content-management/>. Acesso em 06/11/2019.
[3] <https://docs.couchdb.org/en/stable/ddocs/views/intro.html>. Acesso em 30/10/2019.
[4] <https://docs.couchdb.org/en/stable/api/basics.html#api-basics>. Acesso em 30/10/2019.

- *RavenDB*: perde em facilidade de uso para o *MongoDB*, o qual permite a criação de índices com uma única linha[5]. Nesse quesito, além de o *RavenDB* ser menos sucinto, não se pôde confirmar, por sua documentação[6], se ele é capaz de lidar facilmente com índices sobre *arrays*, funcionalidade que é fundamental para tratamento de entidades que possam admitir mais de um identificador, como é o caso para indivíduos.

- *RethinkDB*: similar ao *MongoDB* em vários aspectos, mas menos capaz de lidar com importação de arquivos grandes na íntegra (ver abaixo discussão sobre tamanho máximo de documentos), novamente entrando em conflito com objetivo 4 da seção 1.2.

O *MongoDB* trabalha com documentos em formato *JSON*. Eles são armazenados em coleções, as quais são análogas a tabelas num banco relacional, com a diferença importante da ausência de *schema*: de forma geral, campos podem ser criados e deletados à vontade. Associados a cada coleção podem ser criados um número arbitrário de índices, os quais podem ser sobre um campo ou conjunto ordenado de campos de qualquer tipo, incluindo documentos aninhados e *arrays*, caso este em que cada elemento do *array* gera uma entrada no índice. Ao processar uma consulta, o *MongoDB* irá levar em conta os índices existentes e os campos consultados para elaborar o melhor plano de execução.

Uma restrição comum nas *document stores* diz respeito ao tamanho máximo dos documentos. Com exceção do *CouchDB*, todos os sistemas mencionados limitam (ou recomendam que o usuário limite) esse valor a 20 MB ou menos[7,8,9,10]. Consequentemente, não seria apropriado o carregamento de arquivos grandes em bancos que os transformam em documento para os armazenar, como é o caso do *RavenDB*, *RethinkDB* e *MongoDB*. No entanto, este último oferece um módulo – *GridFS*[11] – que gerencia arquivos dividindo-os em múltiplos documentos pequenos de maneira transparente.

Para o desenvolvimento da aplicação de demonstração e teste, foi escolhida a linguagem *Python 3*[12], pois:

- Combina "em espírito" com *document stores* por ser também flexível, com o dicionário nativo (*dict*) servindo como representação conveniente do documento no banco. Permite um código mais sucinto e compreensível, o que facilita sua edição inclusive por profissionais sem formação em computação ou grande *background* em programação.

5 <https://docs.mongodb.com/manual/indexes/#create-an-index>. Acesso em 30/10/2019.
6 <https://ravendb.net/docs/article-page/4.1/csharp/indexes/what-are-indexes>. Acesso em 06/11/2019.
7 <https://ayende.com/blog/156865/ravendb-net-memory-management-and-variable-size-obese-documents>. Acesso em 30/10/2019.
8 <https://rethinkdb.com/limitations/>. Acesso em 30/10/2019
9 <https://docs.mongodb.com/manual/reference/limits/#bson-documents>. Acesso em 30/10/2019
10 <https://docs.couchbase.com/server/current/learn/clusters-and-availability/size-limitations.html>. Acesso em 30/10/2019
11 <https://docs.mongodb.com/manual/core/gridfs/>. Acesso em 30/10/2019.
12 <https://docs.python.org/3/>

- Apresenta driver oficial para o *MongoDB*, *PyMongo*.
- Apresenta o módulo oficial *argparse* que facilita a criação de interfaces de linha de comando (CLIs).
- Tem ganhado grande espaço e sendo lider em popularidade para computação científica e diversas outras aplicações (INFOWORLD, 2019).

A maior desvantagem do uso de *Python* é a performance reduzida em comparação com linguagens como *C/C++*, o que pode fazer diferença significativa na leitura e manipulação de arquivos grandes. Isso pode ser parcialmente mitigado tendo-se cuidado para escrever código eficiente, o que envolve, por exemplo, favorecer o uso de iteradores e geradores, e evitar laços explícitos com muitas iterações.

Capítulo 3

DESENVOLVIMENTO

3.1 O Problema

Dados os arquivos de dados de SNPs e amostras, pertencentes a diferentes formatos, deseja-se criar um sistema capaz de importá-los e disponibilizar seu conteúdo para consultas relevantes para a área de aplicação. Tais consultas devem ser executadas eficientemente. Além disso, o sistema também deve oferecer capacidades de exportação de seus dados. Finalmente, também deve ser possível o carregamento de arquivos "anexos" (por exemplo, de mídia), os quais, assim como amostras, devem ser associáveis a indivíduos.

3.1.1 Arquivos de Dados

Os arquivos cujo conteúdo o banco precisará incorporar trazem informações sobre SNPs e amostras analisadas, conforme descrito na seção 1.1. Um arquivo típico trará dois tipos de dados: sobre amostras analisadas e sobre os SNPs que compõem o mapa segundo o qual foi feita a análise. Esses dados podem vir em arquivos separados, caso no qual serão aqui chamados de arquivos de pedigree (ou de amostras) e de mapa, respectivamente.

Todo SNP vem com um identificador, o qual não necessariamente é único pois, posto simplificadamente, fontes distintas podem usar nomes diferentes para o mesmo SNP. Entre os outros dados que podem vir associados a um SNP, os mais importantes são provavelmente seu cromossomo e sua posição, os quais servem como coordenadas do local (*locus*) exato do material genético onde o SNP se encontra. Outros dados possíveis são, por exemplo, sobre a qualidade das leituras feitas e frequência de alelos.

Já com relação às amostras, os principais dados associados serão sobre o alelo apresentado para cada SNP. Retomando o que foi dito na seção 1.1, estes podem vir escritos de diferentes maneiras e trazer quantidades variáveis de informação: nas versões mais simples, pode ser apenas um ou dois caracteres representando o alelo presente; nas mais complexas, vários outros dados numéricos podem estar acompanhando. Outros possíveis dados associados a uma amostra incluem sexo do indivíduo do qual a mesma foi extraída e indicação se este é ou não afetado por certa condição (e. g. doença) que esteja em estudo.

Além do mapa e do pedigree, em alguns casos pode haver um arquivo adicional com duas colunas, associando cada identificador de amostra presente no pedigree a um identificador de

indivíduo, chamado *tatoo*. Assim se conhecem as relações entre os dois, um indivíduo podendo ter múltiplas amostras associadas a si.

Para este projeto serão considerados quatro formatos de arquivos a serem importados para o banco. Para melhor mostrar as diferenças entre eles, os mesmos serão descritos brevemente nas subseções a seguir, e exemplos serão compilados no apêndice A. Ressalta-se que não raramente variações podem ser encontradas entre exemplares do mesmo formato. Além disso, a ideia é que a solução desenvolvida seja facilmente expansível para quaisquer outros formatos.

3.1.1.1 Formato 0125

Os arquivos *0125* são os mais simples. O arquivo de mapa (arquivo exemplo 1 do apêndice A) traz uma linha para cada um de seus *M* SNPs, contendo seu nome, cromossomo e posição. O arquivo de pedigree (arquivo exemplo 2) traz uma linha por amostra, contendo o identificador da mesma e uma cadeia de *M* caracteres (0, 1, 2 ou 5) representando os alelos dos SNPs. Por ser o formato mais denso (ocupa apenas 1 byte por SNP), é um dos que costuma trazer maior volume de dados.

Aqui e nas próximas subseções, não é relevante detalhar o significado dos dados associados aos SNPs – os mesmos podem ser abstraídos.

3.1.1.2 PLINK

Trata-se de arquivos de mapa e pedigree (*.map* e *.ped*) produzidos pelo pacote de software de análise *PLINK* (PURCELL *et al.*, 2007). O mapa (arquivo exemplo 3) é muito similar ao 0125, trazendo apenas uma coluna a mais, não obrigatória (0 indica ausência de dado). O pedigree (arquivo exemplo 4) também é similar, mas cada linha traz dois caracteres por SNP em vez de um, além de algumas informações extras, como o sexo do indivíduo ao qual pertence a amostra.

3.1.1.3 Illumina Final Report

Este é um dos formatos gerados pelo software comercial *Illumina GenomeStudio*[1]. Diferentemente dos anteriores, traz um *header* com metadados e uma linha para cada par amostra-SNP contendo vários dados além do próprio alelo encontrado. As colunas presentes não são fixas, podendo haver variação entre arquivos, mas para este projeto foram tratados apenas arquivos similares ao exemplo. Note que não são trazidos a posição e o cromossomo do SNP, apenas seu nome. Nada impede, no entanto, o *Final Report* de vir acompanhado por um arquivo de mapa de um dos formatos citados acima, por exemplo.

[1] <https://www.illumina.com/>

3.1.1.4 VCF

O Variant Call Format – VCF – é o formato mais complexo. Pode conter centenas de linhas de metadados, as quais determinam quais dados o arquivo traz e sua formatação. Traz uma linha por SNP: as nove primeiras colunas contêm informações sobre o mesmo, incluindo nome, cromossomo e posição do SNP. Em seguida há uma coluna por amostra, trazendo dados do SNP para cada uma. A *string* associada a um par SNP-amostra é dividida em campos separados por dois pontos (":"). Os campos presentes e seus significados são determinados, respectivamente, pela coluna *FORMAT* e as linhas iniciais de metadados. O campo que traz informação dos alelos presentes, identificado por *GT*, é o único obrigatório.

3.1.2 Arquivos de Mídia

Os arquivos da seção anterior terão seus dados efetivamente incorporados pelo banco e disponibilizados para consultas complexas. Conforme foi dito na seção 1.1 e no começo deste capítulo, também é de interesse que o sistema seja capaz de importar arquivos na íntegra e associá-los a um indivíduo. A ideia é que se possa organizar tudo sobre ele num lugar só, sendo possível recuperar facilmente a totalidade do que estiver disponível.

Embora seja mais eficiente armazenar apenas caminhos para arquivos em disco no lugar dos bytes propriamente ditos, considerou-se que a última opção tornaria a solução mais completa e fechada.

3.1.3 Funcionalidades

Importados os arquivos, alguns tipos de consultas a serem atendidas são:

1. Encontrar SNPs com base em sua localização (cromossomo e posição) e/ou mapas a que pertence.

2. Encontrar mapas com base em seu tamanho (número de SNPs contidos) e seu formato de importação original.

3. Encontrar o valor de certo SNP para certa amostra.

4. Encontrar mapas, amostras e arquivos associados a um indivíduo.

Além de buscas mais simples como encontrar SNPs, amostras, mapas, indivíduos com base em identificadores, não deixando de considerar o fato de que algumas entidades podem ter mais de um.

Note que consultas mais complexas podem ser derivadas por meio de combinações das mencionadas acima.

Por fim, é de interesse que haja a possibilidade de exportar arquivos a partir do banco. Na forma mais básica, isso se traduz como a capacidade de recuperar arquivos importados integralmente. Há, no entanto, funcionalidades a mais que podem ser desejáveis, como a exportação de um subconjunto dos dados de um arquivo (tanto com relação a SNPs quanto a amostras) e capacidade de exportar dados em formatos diferentes do de origem.

3.2 Atividades Realizadas

3.2.1 Modelagem do Banco

A ideia principal envolvida na modelagem foi a de particionamento dos dados em blocos. A primeira razão para tal é a limitação mencionada no capítulo 2: o *MongoDB* não aceita documentos com tamanho excedendo 16 MB, valor que pode ser facilmente excedido caso tente-se colocar dados sobre milhões de SNPs num único documento. No entanto, mesmo se essa limitação não existisse, é provável que o particionamento continuasse sendo benéfico, já que permite que consultas que retornam poucos dados sejam executadas sem a necessidade de se trazer centenas de MBs para a memória principal.

A importação de dados é feita em duas partes: primeiro importa-se um mapa, depois amostras associadas a este. A figura 1 mostra as coleções utilizadas no banco. Para ilustrar o papel de cada uma delas, considere o que acontece durante a importação de um par de arquivos de mapa e pedigree:

1. O arquivo de mapa é escaneado e gera-se um documento por SNP, os quais serão inseridos na coleção SNPs. Cada novo SNP recebe um identificador numérico interno.

2. É criado um *array* contendo os identificadores internos dos SNPs do mapa, preservando a ordem original do arquivo. Esse *array* é dividido em blocos e guardado em documentos na coleção Bloco de SNPs de Mapa. Uma cópia dele, desta vez ordenada pelo valor numérico do identificador interno, também é armazenada. Apesar de os identificadores internos serm gerados em ordem crescente seguindo a ordem do mapa, admite-se a possibilidade de que os *arrays* citados não tenham a mesma ordem, pois isso permite que mapas diferentes referenciem o mesmo SNP no banco.

3. O documento descritor do mapa é criado e inserido na coleção Mapas. O mapa foi importado com sucesso e agora é reconhecido pelo banco.

4. O arquivo de amostras é escaneado. Para cada amostra:

   4.1. Insere-se seu documento descritivo na coleção Amostras.

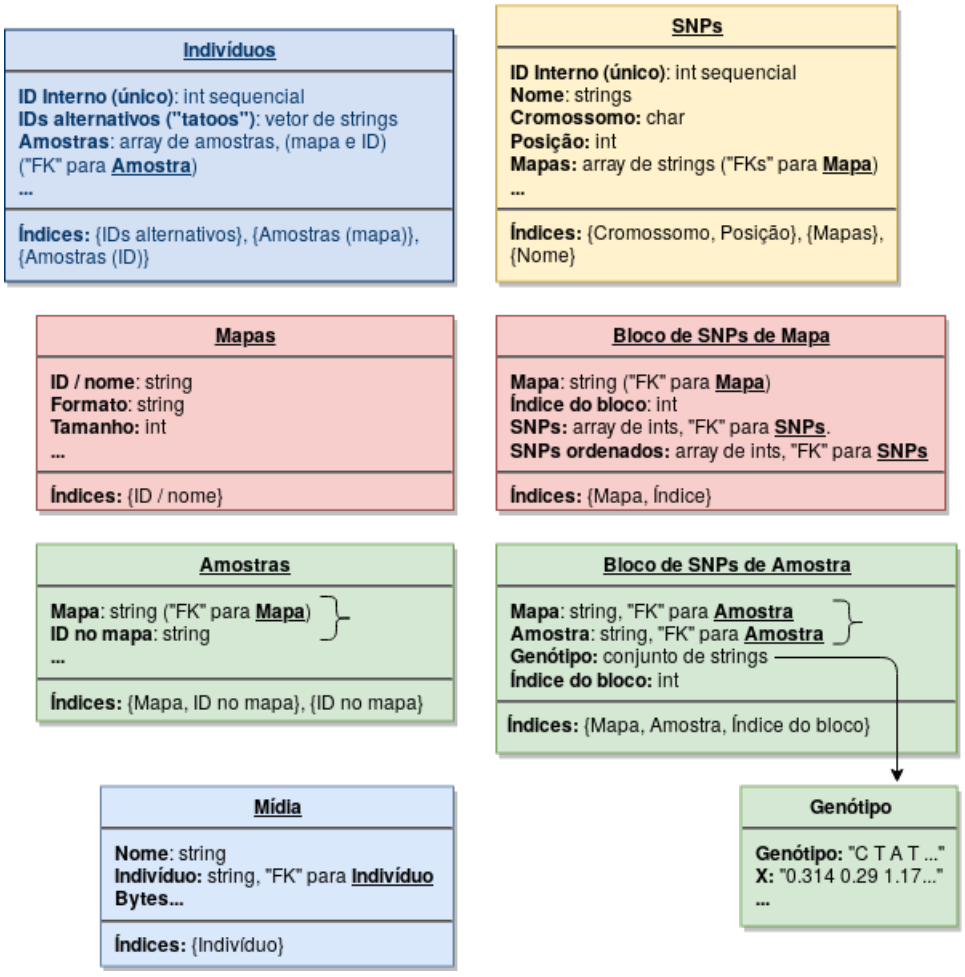
   4.2. Cria-se um ou mais arrays com os dados dos SNPs da amostra.

   4.3. Ordena-se esse array pelos identificadores internos dos SNPs, como no passo 2..

4.4. Particiona-se esse array em blocos e armazena-se em documentos na coleção Bloco de SNPs de Amostra, no campo *genótipo*.

O passo 2. menciona mapas diferentes contendo o mesmo SNP. Uma das maneiras de atingir esse efeito envolve a inserção de um passo a mais opcional entre o 1. e o 2. no qual é feita uma busca por SNPs similares aos que estão sendo importados e, se houver resultado, utilizá-lo em vez de criar um novo SNP, possivelmente após consentimento do usuário. Para essa abordagem é necessário definir critérios de "similaridade" entre SNPs, como, por exemplo, correspondência exata de cromossomo e posição.

Figura 1 – Coleções e índices usados na solução final. $\{A, B\}$ denota um índice ordenado primeiro pelo campo A, desempatando pelo campo B. "FK" (foreign key) está entre aspas pois a integridade referencial não é garantida pelo MongoDB.



Fonte: Elaborada pelo autor.

A organização mostrada na figura 1 é suficiente para realizar quase todas as consultas mencionadas na seção 3.1.3 de maneira direta. Por exemplo, pode-se pedir ao *MongoDB* os SNPs pertencentes ao mapa X cujo cromossomo seja 22, 23 ou 24 e a posição esteja entre 1000000 e 2000000. Para responder a essa consulta, o *MongoDB* usará critérios internos para decidir se utiliza o índice {Mapas} ou o {Cromossomo, Posição}.

Contudo, destaca-se pela maior complexidade a busca por certo SNP de certa amostra, a qual pode ser executada da seguinte maneira:

1. Se não for dado, encontra-se o ID interno do SNP por meio de busca usando os parâmetros dados (nome e/ou região).
2. Obtém-se o documento descritor do mapa ao qual pertence a amostra.
3. Descobre-se o índice do SNP em questão dentro do mapa, considerando a ordenação por valor numérico de ID interno (e não a original do arquivo).
4. De posse dessa informação, é possível descobrir o bloco na coleção Bloco de SNPs de Amostra que contém os dados do SNP desejado.
5. Recupera-se o bloco e retorna-se os dados.

As reticências na figura 1 simbolizam locais onde a inclusão de mais campos é apropriada para acomodar os diferentes dados trazidos pelos arquivos de entrada. Exemplos: na coleção SNPs são armazenados os dados das colunas “REF”, “ALT” e “QUAL” do formato VCF. Este também pode trazer metadados sobre o mapa para serem guardados na coleção Mapas. O formato PLINK traz dados extras sobre as amostras descritas, os quais ficam disponíveis nos documentos da coleção Amostras. *Arrays* com quaisquer dados possíveis sobre um SNP de amostra podem ficar no campo *genótipo* dos documentos da coleção Bloco de SNPs de Amostra.

3.2.2 Criação da Aplicação Básica

Como não há suporte no *MongoDB* para armazenar toda a lógica necessária para realização da solução descrita (e tal prática seria desencorajada[2]), foram desenvolvidas a aplicação e outros *scripts* em *Python* para alcançá-la. Além de servir como prova de conceito, os mesmos permitiram a realização de atividades de validação e teste. A figura 2 mostra o texto de ajuda principal da interface de linha de comando (CLI), desenvolvida com o módulo *argparse*[3], para executar operações no banco. Abaixo segue uma lista das funcionalidades implementadas:

- Importação de mapas e amostras a partir de arquivos nos formatos *0125*, *PLINK*, *Final Report* e *VCF*, com a ressalva de que nem todas as variações possíveis desses formatos foram consideradas.
- Possibilidade de vincular amostras a indivíduos via importação de arquivo associador, mencionado na seção 3.1.1, concomitantemente ao de amostras.

[2] Ver nota em <https://docs.mongodb.com/manual/tutorial/store-javascript-function-on-server/>. Acesso em 02/11/2019.

[3] <https://docs.python.org/3/library/argparse.html>. Acesso em 04/11/2019.

- *Upload*, *download* e busca de arquivos na íntegra, os quais podem ser associados a indivíduos. Os filtros possíveis para a busca são por nome original do arquivo e identificador de indivíduo associado.

- Buscas de SNPs por nome, identificador interno, cromossomo, posição, e mapa.

- Buscas de amostras por mapa e identificador dentro do mapa.

- Buscas de mapas por identificador, formato original e tamanho.

- Buscas de indivíduos por nome (*tatoo*), amostra ou mapa.

- Obtenção dos dados associados a determinado SNP de determinada amostra, explicada na seção 3.2.1.

- Exportação de arquivos de mapa nos formatos *0125* e *PLINK*. Os mapas podem ter sido originalmente importados em formato *0125*, *PLINK*, ou *VCF*. O *Final Report* exclue-se porque, isoladamente, traz dados insuficientes – faltam cromossomo e posição dos SNPs.

- Exportação de amostras nos formatos *0125* e *PLINK*. Diferente do item anterior, aqui é exigido que o formato de exportação seja o mesmo de importação, uma vez que a implementação de conversão entre formatos requereria maior conhecimento de domínio. Outra diferença diz respeito ao fato de que aqui é permitida a exportação de subconjuntos arbitrários das amostras associadas a um mapa – não é necessário que todas sejam incluídas.

Figura 2 – Textos de ajuda na aplicação desenvolvida.

```
$ ./cli.py --help
usage: cli.py [-h]
              {import-map,import-samples,find-snps,find-maps,find-individuals,find-samples,get-snp-genotype,put-file,find-files,get-files,export-map,e
xport-samples}
              ...

positional arguments:
  {import-map,import-samples,find-snps,find-maps,find-individuals,find-samples,get-snp-genotype,put-file,find-files,get-files,export-map,export-sample
s}
    import-map          import map from proper map or ped files
    import-samples      import samples from file
    find-snps           search snps in the database
    find-maps           search maps in the database
    find-individuals    search individuals in the database
    find-samples        search samples in the database
    get-snp-genotype    retrive the genotype given a sample and a SNP
    put-file            upload file to the database
    find-files          search files in the database
    get-files           download files from the database
    export-map          export map from database to file
    export-samples      export samples from database to file

optional arguments:
  -h, --help            show this help message and exit
$ ./cli.py import-map --help
usage: cli.py import-map [-h] [-q] [--force-create-new | --force-use-existing]
                         {0125,pl,fr,vcf} mapfile mapname

positional arguments:
  {0125,pl,fr,vcf}      map file format
  mapfile               map file path
  mapname               id of the map to be stored

optional arguments:
  -h, --help            show this help message and exit
  -q, --quiet           omit all output
  --force-create-new    always create new snps
  --force-use-existing  always use existing snps when available (may ask user
                        to decide which)
```

Fonte: Elaborada pelo autor.

Um ponto notável sobre aplicação é o foco na facilidade de se estendê-la para novos formatos ou fazer modificações no comportamento para os já implementados. Para cada formato possível de ser importado, há classes responsáveis por lê-lo e retornar seus dados estruturados da maneira que a aplicação espera receber para importar. O mesmo vale, analogamente, para exportação. Ou seja, para dar suporte a importação e/ou exportação em novo formato, é suficiente implementar a classe que saiba lê-lo/escrevê-lo corretamente e fazer uma atualização pontual na aplicação para que tal classe seja conhecida.

Ressalta-se também que a implementação de novas consultas é pelo menos tão simples quanto seria com um *backend* relacional, contando ainda com o diferencial da facilidade de se acrescentar novos campos.

3.2.3 Experimentos

O desempenho da solução, tanto em tempo quanto espaço, foi avaliado por meio de vários testes, elaborados com a ajuda do Prof. Ricardo e discutidos em mais detalhes no apêndice B. Todos eles foram executados localmente em um servidor com 1 TB de RAM e 4 processadores Intel Xeon Gold 6140, totalizando 72 *cores*. Os arquivos de teste foram todos gerados aleatoriamente, e os tempos medidos são reais (*wall clock time*). Para os tamanhos de bloco, foram utilizados os valores 100.000 (mapas) e 10.000 (amostras) salvo onde especificado diferentemente. Nos testes de importação realizados, o passo opcional mencionado na seção 3.2.1 não foi considerado.

No apêndice B, em alguns momentos, são feitas comparações com o *TheSNPPit* (GROENEVELD; LICHTENBERG, 2016). Ele e este projeto atuam num momento similar do fluxo de trabalho, porém o primeiro possui foco em eficiência de espaço, tendo entre suas principais funcionalidades a capacidade de trabalhar, a um baixo custo de armazenamento, com subconjuntos de mapas importados (relembre o que foi mencionado na seção 1.1: um de seus objetivos é prover uma alternativa melhor do que cópia bruta e manual de arquivos para trabalho com subconjuntos). O *TheSNPPit* tem um *backend* relacional, e trabalha apenas com um formato, *PLINK*.

Todas as comparações com o *TheSNPPit* tem como base os dados apresentados no artigo citado. Adicionalmente, é importante destacar que a máquina usada para testá-lo é muito inferior à aqui utilizada.

3.3 Discussão dos Resultados

O projeto pôde ser desenvolvido, de modo geral, com todas as funcionalidades e características propostas. A flexibilidade se destaca como um dos pontos fortes: não só a expansão para novos formatos é relativamente simples, como também há grande liberdade para se determinar os detalhes de como a importação/exportação de um formato será feita. Pode-se optar por ignorar ou não certos dados, ou mudar sua representação no banco, tornando-a mais legível ou eficiente.

Nada impede que a técnica de armazenamento de dados *PLINK* empregada pelo *TheSNPPit*, a qual ocupa apenas 2 bits por SNP de amostra (GROENEVELD; LICHTENBERG, 2016), seja utilizada aqui. Esta poderia até mesmo ser expandida ou adaptada para os outros formatos, caso seja razoável. Tudo isso requereria uma quantidade de linhas de código *Python* que depende apenas da complexidade do efeito que se está tentando alcançar, e nada mais.

A compressão realizada pelo *MongoDB* foi mais que suficiente para compensar os *overheads* envolvidos, o que torna a utilização da aplicação vantajosa, no mínimo, em relação a manter os dados em arquivos sem compressão no sistema, desde que os tempos de importação e exportação envolvidos não sejam considerados obstáculos proibitivos.

Não se encontrou evidência de que as velocidades de importação e exportação se deterioram significativamente conforme o banco é carregado. Quando comparadas aos valores obtidos pelo *TheSNPPit*, as mesmas – principalmente a última – podem ser consideradas baixas. Enquanto isso pode significar que esta abordagem ainda não está pronta para enfrentar os maiores volumes de dados trabalhados atualmente, também não é suficiente para desqualificá-la para todos os cenários e propósitos.

A mesma afirmação sobre não deterioração pode ser feita sobre o desempenho das consultas, porém com uma exceção notável: a recuperação de SNP de amostra, a qual não só se deteriora significativamente com o aumento do tamanho da amostra como também apresenta uma performance base ruim em comparação com exportações normais. Trata-se da maior evidência de que a solução atual ainda requer aprimoramento.

3.4 Dificuldades e Limitações

A maior dificuldade associada ao projeto esteve ligada à compreensão do domínio e do problema, já que havia pouco conhecimento prévio do autor sobre eles. Mesmo após ser atingido um nível de entendimento suficiente para o desenvolvimento deste trabalho, o fato de ele não estar sendo elaborado por um especialista pode ter consequências significativas, como priorização inadequada de tarefas, funcionalidades ou detalhes, a qual pode resultar em aproveitamento subótimo de tempo disponível. Esse tipo de deficiência poderia ser eliminado tendo-se um especialista disponível para consultoria 100% do tempo, o que não é razoável.

Houve dificuldade também na decisão da *document store* a ser empregada. A escolha do banco de dados mais adequado para uma solução é algo que exige livros inteiros (como o autorado por Perkins, Redmond e Wilson (2018)) de estudo. Para este projeto, não houve tempo sequer para análises profundas da documentação dos sistemas considerados. Consequentemente, a justificativa para emprego do *MongoDB* não é forte, e não se pode descartar definitivamente a hipótese de que outras alternativas permitem resultados melhores.

A implementação da solução apresenta limitações importantes de performance e funcio-

nalidade, como:

- Incapacidade de um SNP de ser identificado por mais de um nome. Tal funcionalidade complica a lógica de exportação de dados pois, se um SNP tem mais de um nome, é necessário saber qual deles usar (excluindo-se o formato VCF, todos os outros trabalham com um identificador por SNP). Esse problema pode ser resolvido de várias formas, como delegar a escolha ao usuário, mas acabou sendo despriorizado em favor da implementação de funcionalidades de maior relevância. Ressalta-se que o problema não se aplica a indivíduos, pois estes não podem ser exportados diretamente.

- Ineficiência do *parser VCF*. Devido ao fato de trazer os dados de amostra em colunas em vez de linhas, é o único *parser* que requer carregamento total do arquivo em memória primária. Soluções que não façam isso e sejam rápidas não foram exploradas.

- Baixa velocidade da recuperação de SNP de amostra, citada na seção anterior e ilustrada na figura 10, o que limita a aplicabilidade do banco em cenários que requeiram "acesso aleatório" a muitas amostras num curto período de tempo.

- Incapacidade de se realizar operações de atualização e deleção pela interface. Essa funcionalidade foi considerada de baixa prioridade para esta prova de conceito, de forma que atualmente só podem ser executadas manualmente (por exemplo, via *shell* do *MongoDB*), sem nenhuma garantia de manutenção de consistência.

Também é importante notar que o uso que está sendo feito do *MongoDB* não está totalmente alinhado com o que é idealizado para *document stores*, uma vez que parte da filosofia envolvida é "guardar dados relacionados no mesmo lugar". O uso de "chaves estrangeiras", como está na figura 1, não é ideal nem suportado pelo banco, no entanto foi fundamental sobretudo para otimização de espaço, já que elimina a necessidade de replicação de dados, por exemplo, de SNPs.

Capítulo 4

CONCLUSÃO

Este projeto explorou o uso de um sistema de banco de dados não-relacional para o *backend* de uma aplicação gerenciadora de dados de SNPs. Alguns objetivos puderam ser definitivamente alcançados, como a alta flexibilidade e generalidade (objetivo 1 da seção 1.2, discutido na seção 3.3), enquanto outros têm avaliação mais dependente do usuário e do uso, como a qualidade da performance (objetivo 2). De forma geral, no entanto, todos eles puderam ser atingidos.

Embora o *TheSNPPit* tenha apresentado desempenho superior, não há evidência o bastante para se isolar definitivamente as razões para tal. Talvez exista um *tradeoff* inevitável entre eficiência e versatilidade. Ou, talvez, as diferenças de performance possam ser eliminadas simplesmente com o aprimoramento da execução da solução aqui apresentada.

O código fonte desenvolvido está disponível em <https://github.com/rodrigo-weigert/snpdb>[1]. Há muitos trabalhos futuros, com variados graus de complexidade e abrangência, que podem ser desenvolvidos a partir deste:

- Aprimoramento de funcionalidades. Há aquelas que são passíveis de serem polidas (e.g. interface para *download* de arquivos anexos), otimizadas (e.g. recuperação de SNP de amostra) e expandidas (e.g. permitir exportação de amostras para subconjuntos de SNPs de mapas).

- Implementação de novas funcionalidades, buscando-se determinar as de maior relevância considerando contextos de maior ou menor grau de especificidade, conforme apropriado.

- Expansão para mais formatos, a qual foi simplificada, conforme discutido na seção 3.2.2.

- Ampliação da generalidade dos *parsers* para os formatos já tratados, permitindo que aceitem mais variações.

- Aprimoramento da importação e exportação para os formatos já tratados, por exemplo reescrevendo-as numa linguagem mais rápida, usando abordagens paralelas para o *parsing*, ou encontrando representações mais eficientes para os dados no banco.

[1] Na data de depósito deste trabalho, a elaboração da documentação do código e do repositório, bem como a limpeza e organização final deste, ainda estavam em andamento.

- Exploração dos ganhos de desempenho obtíveis com alterações nas configurações do *MongoDB* ou parâmetros como o tamanho dos blocos.
- Criação de interface gráfica.

As próximas seções, escritas em primeira pessoa, estão menos diretamente relacionadas ao projeto desenvolvido mas são escritas conforme instruções do ICMC.

4.1 Sobre conhecimentos utilizados

Abaixo as disciplinas cursadas por mim que mais forneceram conhecimentos utilizados no projeto acima:

- Bases de Dados: trouxe conceitos que ajudam no entendimento dos não relacionais, embora estes não tenham sido abordados diretamente.
- Introdução ao Desenvolvimento Web: graças aos trabalhos, proveu um contato inicial bem vindo com *NoSQL* e *Javascript*, linguagem usada pelo *MongoDB*.
- Processamento de Imagens: foi inteiramente ministrada com *Python 3*, promovendo ótima prática com a linguagem e alguns de seus principais módulos, como o *NumPy* e o *matplotlib*, os quais foram utilizados para criação dos gráficos no apêndice B.
- Programação Orientada a Objetos: ajudou na criação de um código mais facilmente extensível por terceiros.

4.2 Sobre a graduação do autor

Passou, de forma geral, sem maiores problemas. Tive boas notas, o que sugere dedicação adequada mas não necessariamente significa que eu saiba o que estou fazendo. Provavelmente não aproveitei bem meu potencial. Fui bastante chato.

O Bacharelado em Ciências da Computação fez justiça à sua classificação de curso de tempo integral. Não foi, de fato, um curso em que se poderia entrar esperando abundância de tempo livre, o qual é consumido num piscar de olhos pela quantidade de trabalhos pedidos sobretudo nas disciplinas de computação. Pela minha experiência, não parece difícil, com as combinações certas de disciplinas e professores, surgirem semestres “mortais”.

Encontrei professores de péssimos a ótimos. Os ótimos, para mim, foram aqueles que dominavam o conteúdo da disciplina e sabiam transmití-lo bem. Os bons e ruins perdiam pontos nesses quesitos, e os péssimos falhavam definitivamente em ambos. Não vi, no entanto, nenhum professor que não parecesse ter boas intenções, mas já houve quem me chamasse de naïve por causa dessa opinião.

Nada tenho a dizer sobre infraestrutura, o que deve ser bom, já que, se eu tivesse sentido falta de algo, eu lembraria.

Eu gostei do meu tempo no ICMC. Sentirei falta. Não sei se das aulas que por um motivo ou outro pareciam se arrastar por horas, ou dos momentos de sofrimento com acúmulo de provas e trabalhos, mas certamente de todo o resto, principalmente daqueles que estavam sofrendo comigo.

REFERÊNCIAS

CLANCY, S. Genetic mutation. **Nature Education**, v. 1, n. 1, p. 187, 2008. Disponível em: <https://www.nature.com/scitable/topicpage/genetic-mutation-441/>. Acesso em: 08/11/2019. Citado na página 15.

DB-ENGINES. **DB-Engines Ranking - popularity ranking of document stores**. 2019. Disponível em: <https://db-engines.com/en/ranking/document+store>. Acesso em: 17/10/2019. Citado na página 20.

GROENEVELD, E.; LICHTENBERG, H. **TheSNPpit-A High Performance Database System for Managing Large Scale SNP Data**. Public Library of Science, 2016. e0164043-e0164043 p. 27780248[pmid]. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/27780248>. Citado 5 vezes nas páginas 16, 30, 31, 43 e 44.

INFOWORLD. **Python's popularity surges as a mainstay language**. 2019. Disponível em: <https://www.infoworld.com/article/3331603/pythons-popularity-surges-as-a-mainstay-language.html>. Acesso em: 19/10/2019. Citado na página 22.

KOOPAEE, H. K.; KOSHKOIYEH, A. E. Snps genotyping technologies and their applications in farm animals breeding programs: review. **Brazilian Archives of Biology and Technology**, scielo, v. 57, p. 87 – 95, 02 2014. ISSN 1516-8913. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1516-89132014000100013&nrm=iso>. Citado na página 15.

PERKINS, L.; REDMOND, E.; WILSON, J. **Seven databases in seven weeks: a guide to modern databases and the NoSQL movement**. [S.l.]: Pragmatic Bookshelf, 2018. Citado 2 vezes nas páginas 19 e 31.

PURCELL, S.; NEALE, B.; TODD-BROWN, K.; THOMAS, L.; FERREIRA, M. A. R.; BENDER, D.; MALLER, J.; SKLAR, P.; BAKKER, P. I. W. de; DALY, M. J.; SHAM, P. C. **PLINK: a toolset for whole-genome association and population-based linkage analysis.** 2007. Disponível em: <http://zzz.bwh.harvard.edu/plink/>. Citado na página 24.

STRAUCH, C. **NoSQL Databases**. 2012. Disponível em: <https://www.christof-strauch.de/nosqldbs.pdf>. Citado na página 19.

APÊNDICE A

EXEMPLOS DE ARQUIVOS DE DADOS

Arquivo Exemplo 1: Mapa 0125 com 12 SNPs. Este formato traz apenas o nome, cromossomo e posição dos SNPs.

```
1 Name  Chromosome  TEMPPOS
2 Hapmap43437-BTA-101873  1 135098
3 ARS-BFGL-NGS-16466  1 267940
4 Hapmap34944-BES1_Contig627_1906 1 393248
5 ARS-BFGL-NGS-98142  1 471078
6 Hapmap53946-rs29015852  1 516404
7 ARS-BFGL-NGS-65067  1 883895
8 ARS-BFGL-NGS-98203  1 1009504
9 Hapmap53766-ss46526150  1 1359951
10 ARS-BFGL-NGS-29653  1 2049400
11 ARS-BFGL-NGS-115671 1 2313042
12 BTB-02081949  1 3079342
13 ARS-BFGL-NGS-43924  1 3148834
```

Arquivo Exemplo 2: Pedigree 0125 mostrando duas amostras com 12 SNPs cada. O conteúdo em cada SNP é representado por um único dígito.

```
1 ID CALL...
2 1251799 111125115022
3 1251446 222121221012
```

Arquivo Exemplo 3: Mapa PLINK com 4 SNPs. Muito similar ao mapa 0125. A terceira coluna pode trazer uma informação extra (“distância genética”), que está ausente neste caso.

```
1 1 Hapmap43437-BTA-101873 0 135098
2 1 ARS-BFGL-NGS-16466 0 267940
3 1 ARS-BFGL-NGS-19289 0 305793
4 1 ARS-BFGL-NGS-105096 0 353745
5 1 Hapmap34944-BES1_Contig627_1906 0 393248
6 1 BTA-07251-no-rs 0 434180
7 1 ARS-BFGL-NGS-98142 0 471078
```

Arquivo Exemplo 4: Pedigree PLINK com 4 amostras e 7 SNPs por amostra. Zeros significam valores ausentes. As primeiras 6 colunas são, respectivamente: identificadores da família, da amostra, paterno, materno, sexo (1 para macho, 2 para fêmea) e indicação de afetado ou não afetado. As próximas colunas são os valores dos SNPs, um par de colunas para cada.

```
1 ACE ACE_616520730 0 0 1 U G G G G 0 0 0 0 C C 0 0 G G
2 ACE ACE_616520740 0 0 1 U G G G G 0 0 0 0 C C 0 0 G G
3 ACE ACE_616520750 0 0 2 U G G G G 0 0 0 0 A A 0 0 G G
4 ACE ACE_616520760 0 0 1 U G G G G 0 0 0 0 A C 0 0 G G
```

Arquivo Exemplo 5: Final Report mostrando 4 SNPs de uma amostra.

```
1 [Header]
2 GSGT Version    1.9.4
3 Processing Date 4/1/2013 5:09 PM
4 Content     GGPv2_SemiPrivate.bpm
5 Num SNPs    8810
6 Total SNPs  9323
7 Num Samples 178
8 Total Samples   179
9 [Data]
10 SNP Name    Sample ID   Allele1 - Forward   Allele2 - Forward
     Allele1 - Top   Allele2 - Top   Allele1 - AB    Allele2 - AB    GC
      Score    X   Y
11 ARS-BFGL-BAC-10975  2013OG1852  A   A   A   A   A   A   0.8234  1.452
       0.054
12 ARS-BFGL-BAC-11025  2013OG1852  T   G   A   C   A   B   0.7956  0.307
       0.501
13 ARS-BFGL-BAC-11044  2013OG1852  C   C   G   G   B   B   0.9203  0.012
       0.860
14 ARS-BFGL-BAC-11193  2013OG1852  A   A   A   A   A   A   0.9101  0.452
       0.000
```

Arquivo Exemplo 6: VCF com 3 amostras e 5 SNPs. Formato muito mais complexo e flexível, e o único que traz as amostras nas colunas, não nas linhas.

```
1 ##fileformat=VCFv4.0
2 ##fileDate=20090805
3 ##source=myImputationProgramV3.1
4 ##reference=1000GenomesPilot-NCBI36
5 ##phasing=partial
6 ##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples
     With Data">
7 ##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
8 ##INFO=<ID=AF,Number=.,Type=Float,Description="Allele Frequency">
```

```
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build
    129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50\% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality
    ">
#CHROM POS     ID         REF ALT    QUAL FILTER INFO
                FORMAT      NA00001         NA00002         NA00003
20     14370   rs6054257 G       A       29   PASS   NS=3;DP=14;AF
    =0.5;DB;H2           GT:GQ:DP:HQ 0|0:48:1:51,51 1|0:48:8:51,51
    1/1:43:5:.,.
20     17330   .          T       A       3    q10    NS=3;DP=11;AF
    =0.017               GT:GQ:DP:HQ 0|0:49:3:58,50 0|1:3:5:65,3
    0/0:41:3
20     1110696 rs6040355 A       G,T     67   PASS   NS=2;DP=10;AF
    =0.333,0.667;AA=T;DB GT:GQ:DP:HQ 1|2:21:6:23,27 2|1:2:0:18,2
    2/2:35:4
20     1230237 .          T       .       47   PASS   NS=3;DP=13;AA=T
                       GT:GQ:DP:HQ 0|0:54:7:56,60 0|0:48:4:51,51
    0/0:61:2
20     1234567 microsat1 GTCT    G,GTACT 50   PASS   NS=3;DP=9;AA=G
                       GT:GQ:DP    0/1:35:4       0/2:17:2
    1/1:40:3
```

APÊNDICE B

RESULTADOS DOS EXPERIMENTOS

As primeiras três figuras dizem respeito a testes de importação de mapa, sem incluir dados de amostras, embora estes sejam os maiores exigentes de recursos por grande margem. Mapas foram importados a velocidades médias entre 10.000 e 20.000 SNPs por segundo[1], havendo diminuição para os mapas maiores (fig. 3). A causa desta é, provavelmente, a lotação crescente do banco, como também sugere o aumento da frequência de picos na figura 5. Quanto ao armazenamento, houve aumento de 100 a 200% em relação aos arquivos originais (fig. 4).

Sobre amostras, a figura 6 exibe velocidades de importação para todos os formatos, considerando mapas de tamanho 10.000 e 1.000.000. Algumas velocidades ficaram na mesma ordem de grandeza dos apresentados por Groeneveld e Lichtenberg (2016) com o *TheSNPPit*, mas o formato *Final Report* foi destacadamente mais lento. Isso se explica pelo fato de ser o formato com mais dados por SNP. Lógica oposta se aplica ao *0125*.

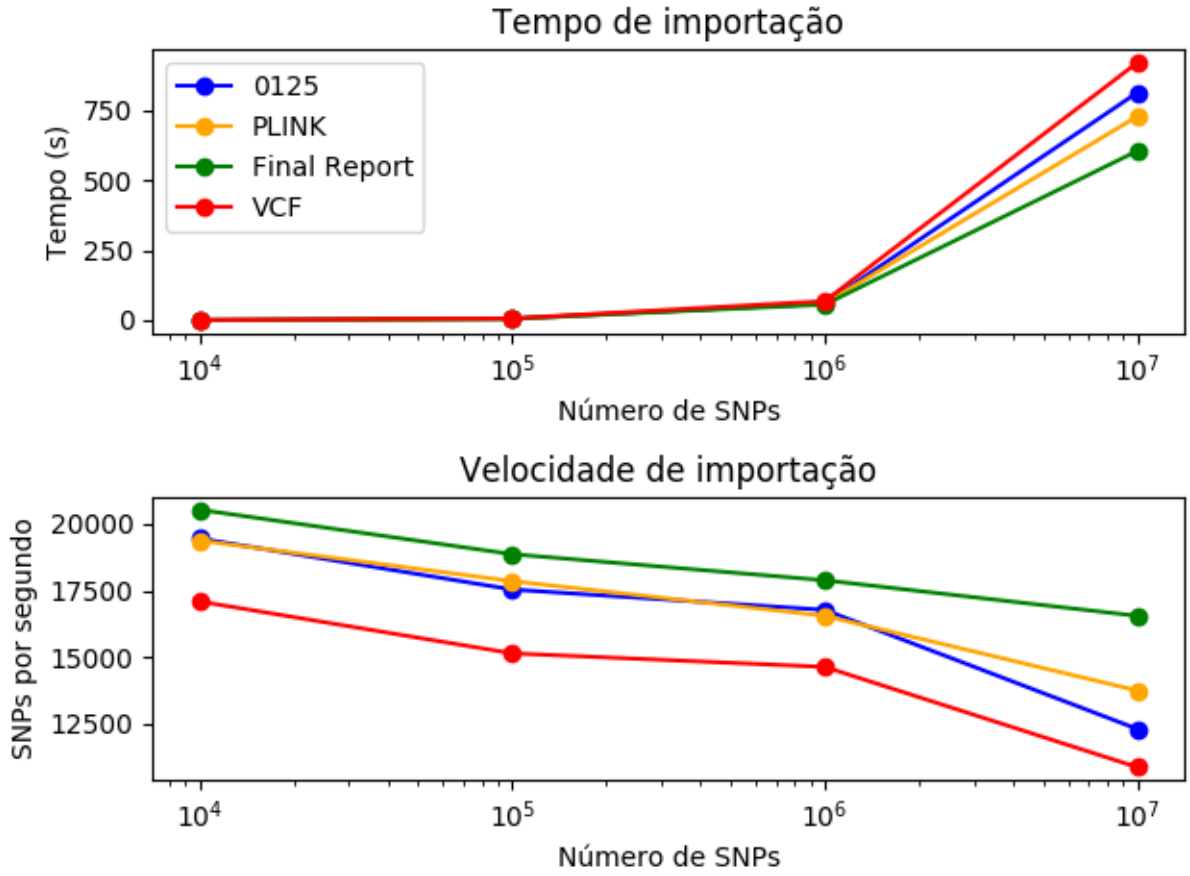
Os espaços ocupados pelos dados de mapa e amostra juntos são comparados na figura 7. Observa-se que o armazenamento menos eficiente do mapa não é o bastante para prejudicar a economia de espaço advinda da compressão feita pelo *MongoDB* e da forma como os dados de amostra são representados internamente, a qual pode ser mais eficiente que alguns dos formatos, neste caso, nomeadamente, *PLINK* (possui muitos caracteres espaço desnecessários) e *Final Report* (repete identificadores de SNP e amostra em cada linha). A redução de espaço média foi de 59%, com a mais alta no formato *Final Report* (68%) e a mais baixa no *0125* (41%).

A figura 8 mostra as velocidades obtidas para a importação de arquivos na íntegra. Considerando *overheads* e a compressão (praticamente nula neste caso), o aumento de espaço de armazenamento não superou 0,5%.

As figuras 9 e 10 dizem respeito a testes com consultas. Na primeira, observa-se que o tempo de resposta a uma consulta de SNPs por região permaneceu abaixo do milissegundo, mesmo com o banco carregado com 100 milhões de SNPs. A segunda figura trata da busca pelo dado do SNP de uma amostra específica, a consulta mais pesada aqui tratada e que foi descrita na seção 3.2.1. Observa-se que ela sofre impacto significativo com o aumento do tamanho do mapa associado (e, portanto, aumento do número de blocos em que uma amostra é dividida). Não se verificou perda de desempenho significativa ligada ao aumento da quantidade de amostras.

[1] Para referência, os maiores mapas, em 2016, chegavam a 3.000.000 SNPs (GROENEVELD; LICHTENBERG, 2016).

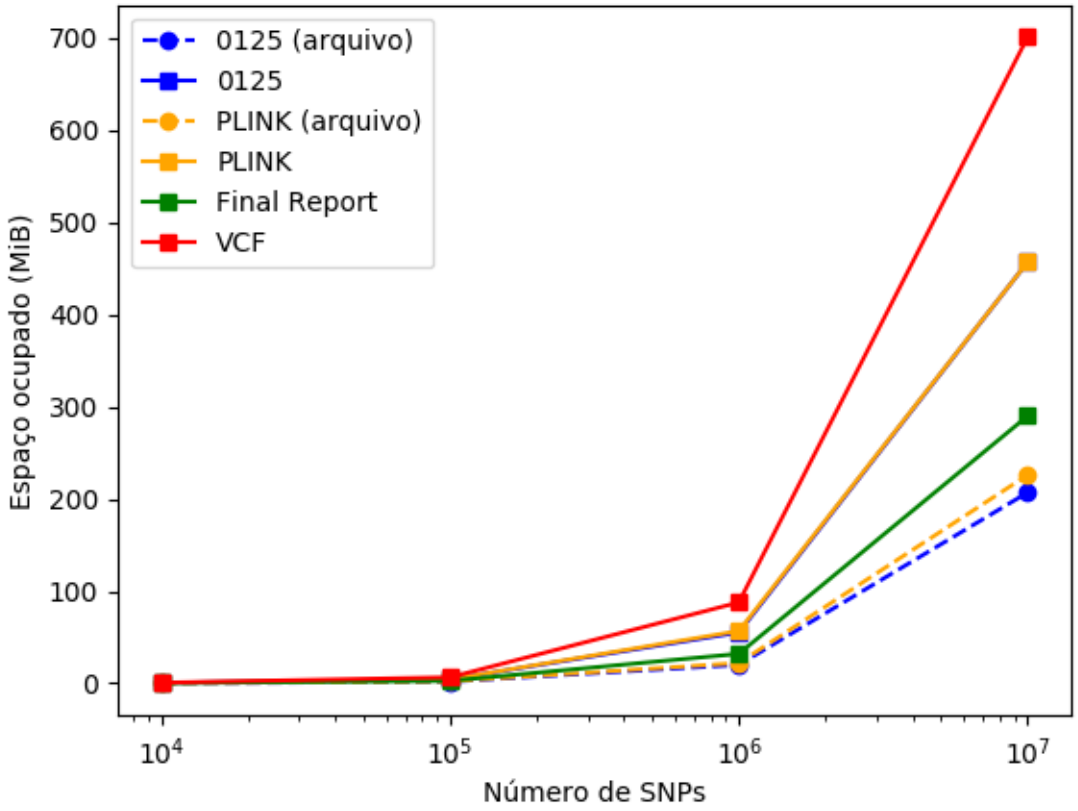
Figura 3 – Tempo e velocidade média de importação para um conjunto de mapas de diferentes formatos e tamanhos. O banco foi inicializado sem dados.



Fonte: Elaborada pelo autor.

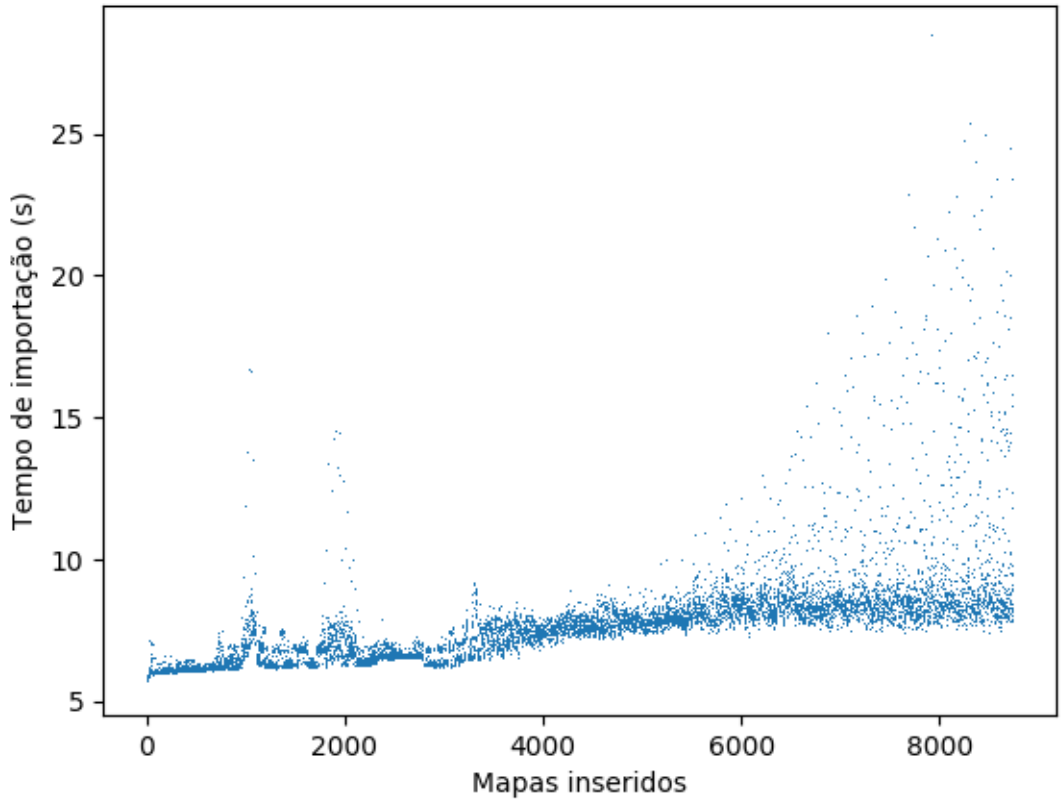
Por fim, a figura 11 mostra o desempenho de exportação de mapas e amostras para arquivos no HD. Para as últimas, atingiu-se velocidades próximas de 1,4 milhões de SNPs por segundo. O *TheSNPPit*, para referência, chega a operar a 120 milhões de SNPs por segundo (GROENEVELD; LICHTENBERG, 2016).

Figura 4 – Utilização de espaço em disco ocupado após importação de mapas de diferentes formatos e tamanhos, incluindo a compressão interna feita pelo motor de armazenamento do *MongoDB*. As linhas tracejadas representam os tamanhos de arquivos originais (só duas estão presentes, pois os outros dois formatos não possuem arquivo de mapa separado).



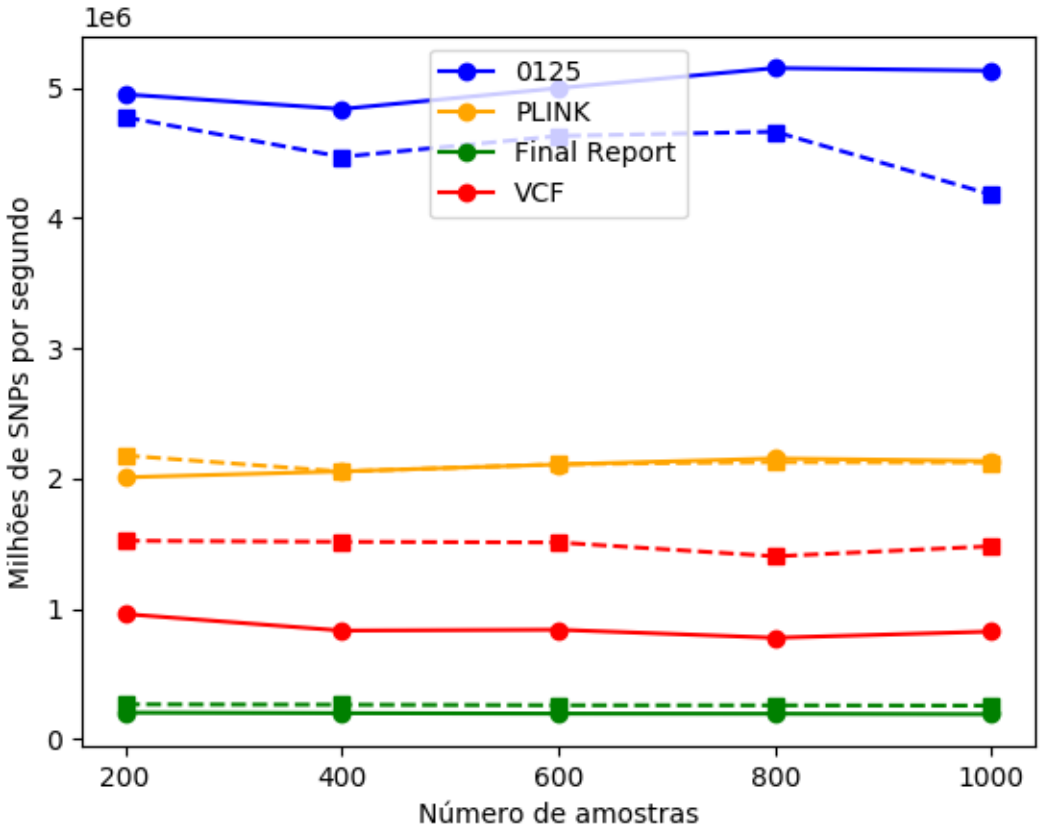
Fonte: Elaborada pelo autor.

Figura 5 – Deterioração do tempo de importação de mapa conforme o banco fica carregado. Foram inseridos, em sequência, cerca de 8700 mapas com 100000 SNPs cada a partir do banco vazio. O gráfico mostra o tempo de importação de cada mapa, em ordem.



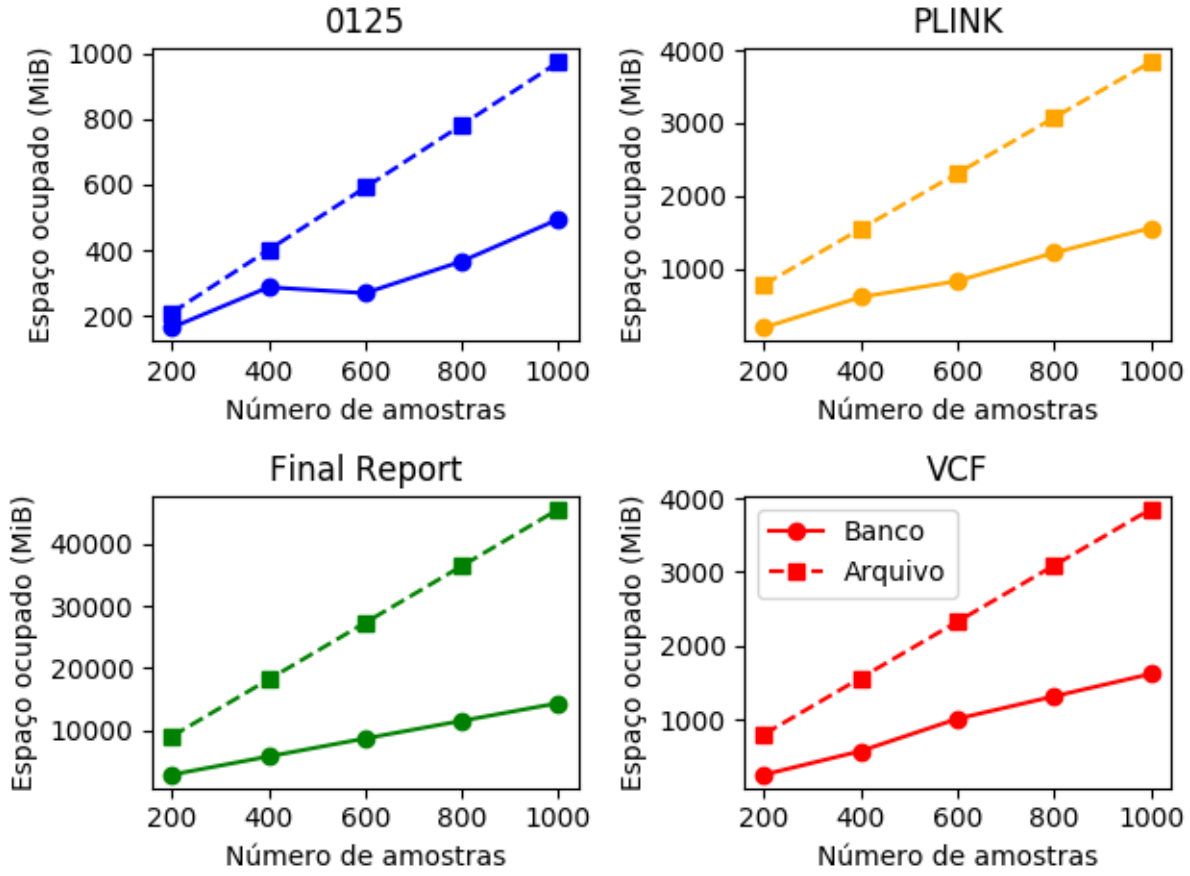
Fonte: Elaborada pelo autor.

Figura 6 – Velocidade de importação de arquivos de amostras. As linhas cheias correspondem a mapas de tamanho 1.000.000, e, as tracejadas, 10.000. Todas importações foram feitas a partir do banco vazio.



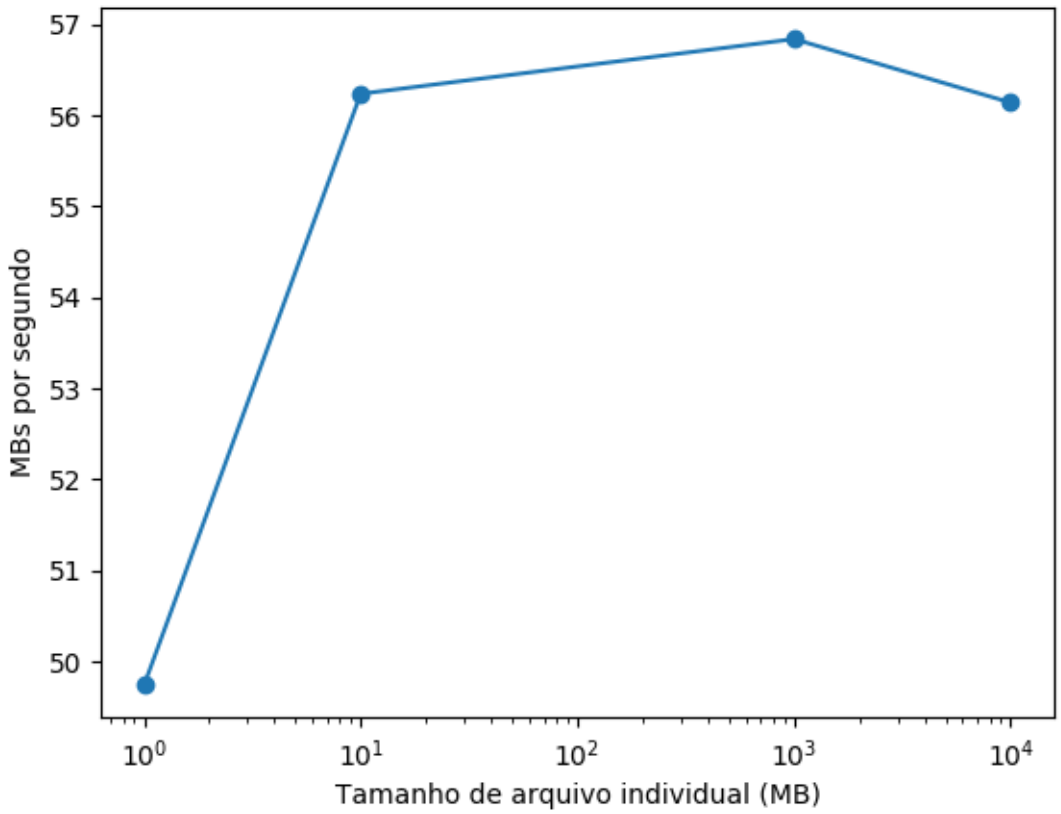
Fonte: Elaborada pelo autor.

Figura 7 – Espaço de armazenamento ocupado pelo banco (linhas cheias) contendo diferentes números de amostras importadas. O tamanho do mapa foi fixado em 1.000.000 e o espaço ocupado por ele também está contado. As linhas tracejadas representam o tamanho dos arquivos originais.



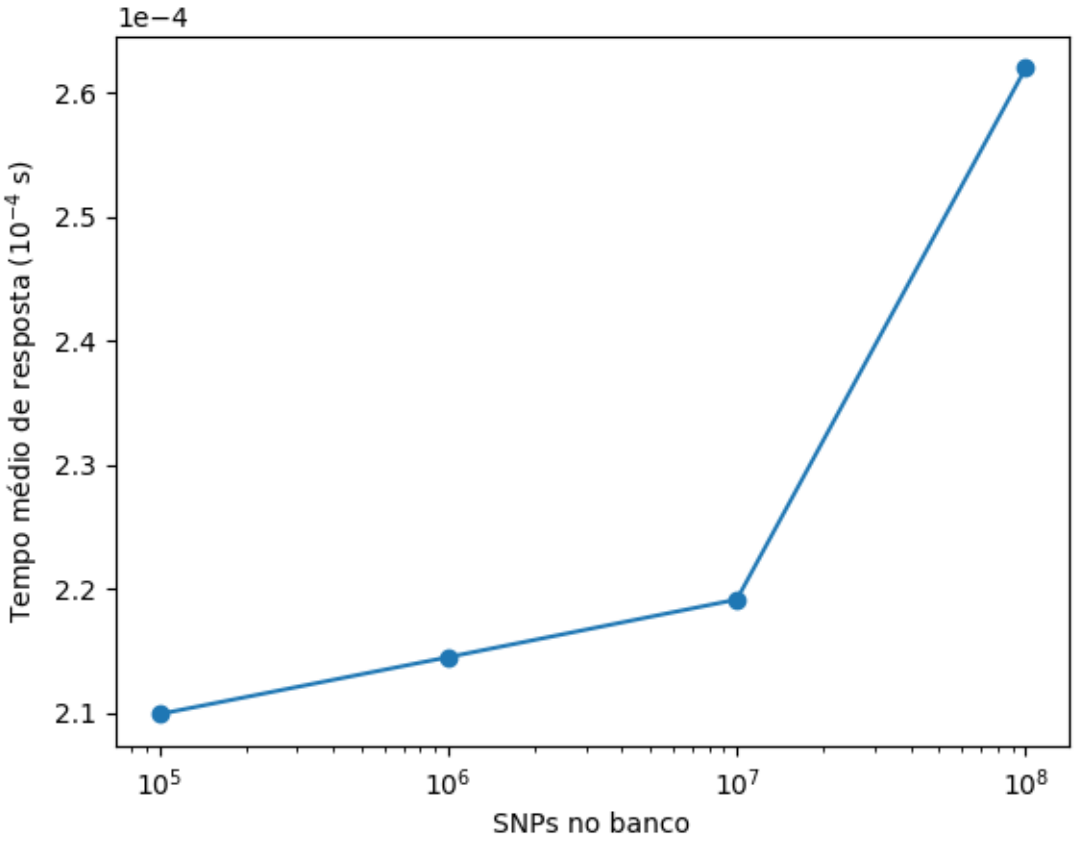
Fonte: Elaborada pelo autor.

Figura 8 – Velocidade média de importação de arquivos na íntegra. Para cada tamanho, foram importados 100 GB de arquivos binários gerados aleatoriamente.



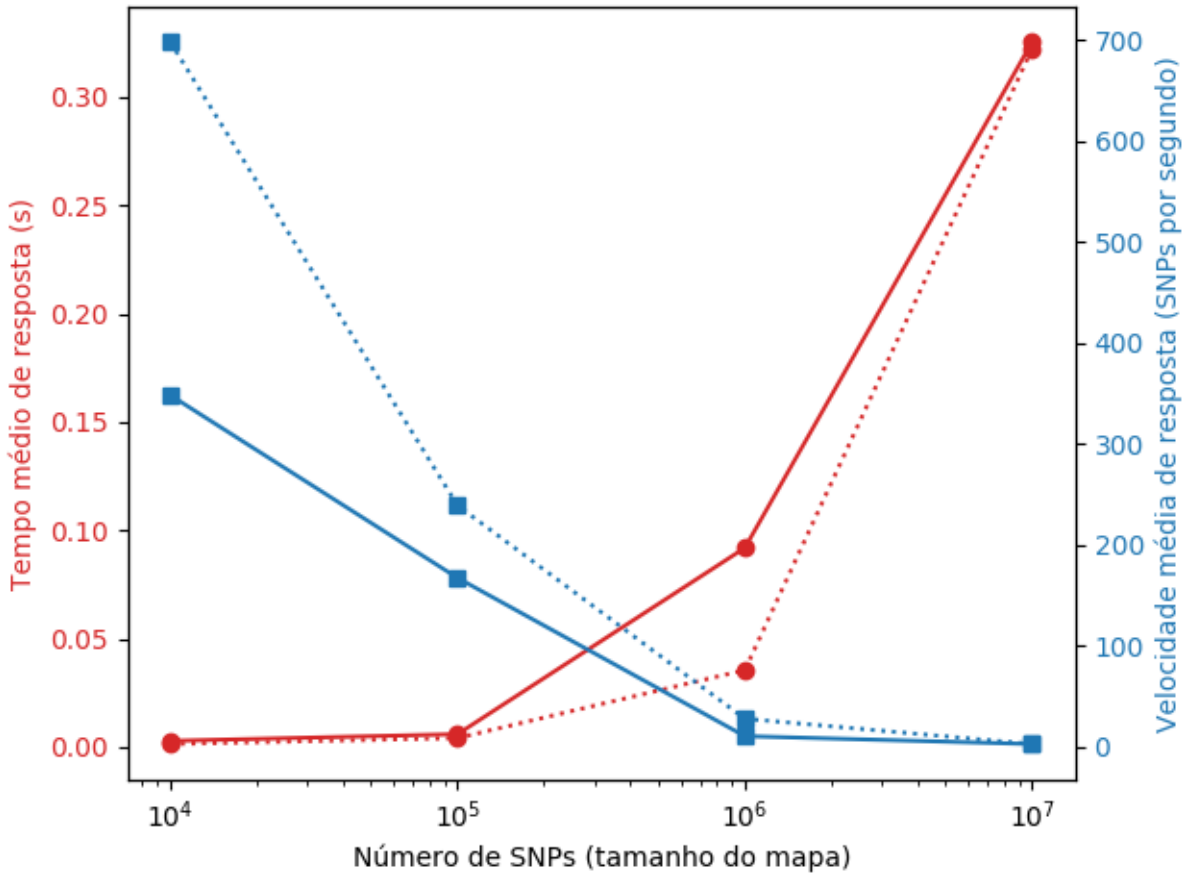
Fonte: Elaborada pelo autor.

Figura 9 – Tempo médio de resposta a busca de SNP por região exata (cromossomo e posição), para diferentes quantidades de SNPs no banco. Foram feitas 100.000 consultas geradas aleatoriamente, de forma que todas retornassem ao menos uma resposta.



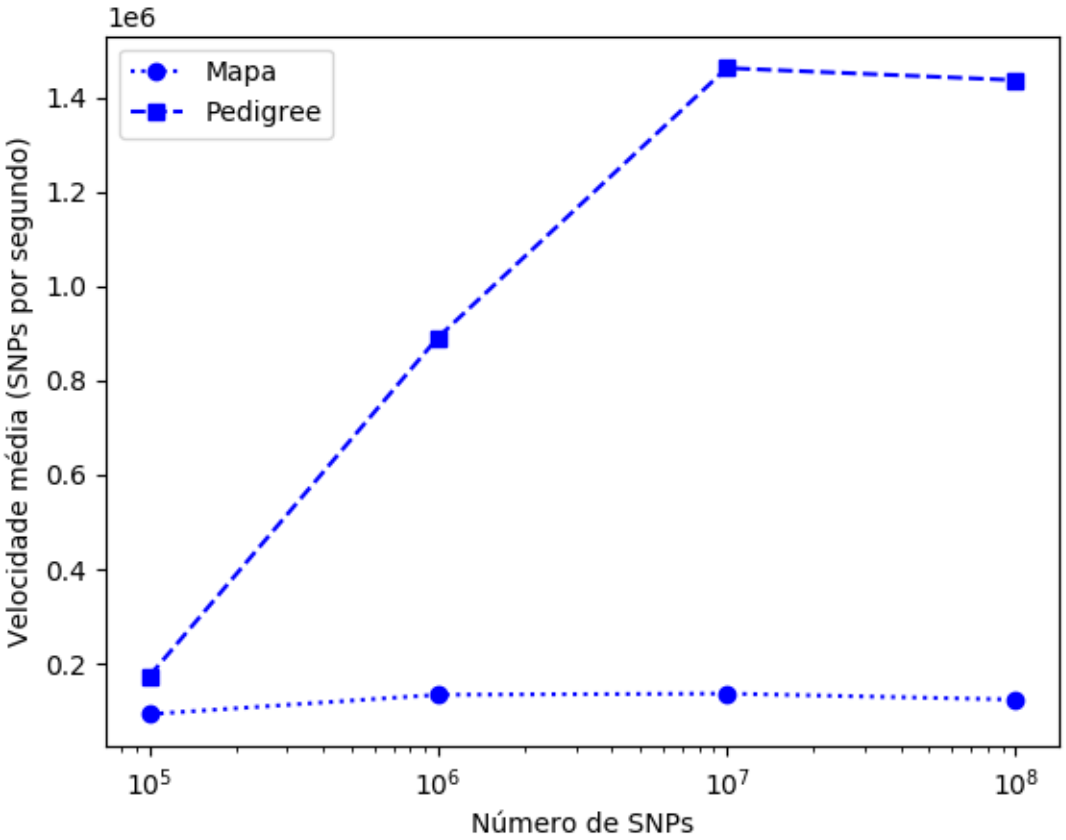
Fonte: Elaborada pelo autor.

Figura 10 – Tempo e velocidade médias de resposta a busca de dados de SNP de certa amostra, ambos selecionados aleatoriamente, variando-se o tamanho do mapa (SNPs por amostra). Os SNPs foram buscados via nome. O banco foi carregado com 100 amostras de cada tamanho testado, e foi feita média entre 10000 consultas. Foi utilizado o formato Final Report, pois é o que contém mais dados por SNP. As linhas pontilhadas representam o mesmo teste, mas realizado com tamanho de bloco de amostra fixado em 1.000 em vez de 10.000.



Fonte: Elaborada pelo autor.

Figura 11 – Velocidade de exportação de mapa e amostra 0125. O banco foi carregado com apenas o mapa e a amostra em questão. Valores muito similares foram obtidos para o formato PLINK.



Fonte: Elaborada pelo autor.