Mendelova univerzita v Brně
Provozně ekonomická fakulta

Analýza a identifikace lokálních struktur DNA

Diplomová práce

Vedoucí práce:
prof. RNDr. Ing. Jiří Šťastný, CSc.

Bc. Jan Havlík

Brno 2020

Poděkování

Chtěl bych poděkovat vedoucímu mé diplomové práce, panu prof. RNDr. Ing. Jiřímu Šťastnému, CSc. za vedení práce a příležitost být součástí světa bioinformatiky. Dále pak konzultantovi práce, panu doc. Mgr. Václavovi Brázdovi, Ph.D., který mi pomohl s pochopením biologických procesů buď odpovědí na otázky, či dodáním potřebných materiálů. V neposlední řadě bych chtěl poděkovat mé rodině za trpělivost a podporu, mým přátelům za rozptýlení myšlenek a mým kolegům za mou náhradu v práci.

Čestné prohlášení

Prohlašuji, že jsem práci **Analýza a identifikace lokálních struktur DNA** vypracoval samostatně a veškeré použité prameny a informace uvádím v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a v souladu s platnou Směrnicí o zveřejňování závěrečných prací.

Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 20. května 2020 .....

podpis

Abstract

HAVLÍK, Jan. Analysis and identification of local structures in DNA. Brno, 2020. Master's thesis. Mendel University in Brno, Faculty of Business and Economics, Department of Informatics. Thesis supervisor Jiří Šťastný.

This master's thesis is focused on the integration of a new type of analysis into an existing server DNA-analyser. The main part is about identification and analysis of the local DNA structure R-loop, which has much more impact on genetic mutations and diseases in human body than previously thought (e.g. cancer, neurodegenerative diseases). The first part focuses on DNA structure relating to the R-loops. The second part is focused explicitly on R-loop forming and its effect on living organisms.

Key words: R-loop, DNA, RNA, local structures, analysis, genom, nucleotide, DNA-analyser, cancer, neurodegenerative disease

Abstrakt

HAVLÍK, Jan. Analýza a identifikace lokálních struktur DNA. Brno, 2020. Diplomová práce. Mendelova univerzita v Brně, Provozně ekonomická fakulta, Ústav informatiky. Vedoucí práce Jiří Šťastný.

Tato diplomová práce pojednává o integraci nového typu analýzy do stávajícího nástroje DNA-analyser. Jedná se o identifikaci a analýzu lokální struktury R-loop, jejíž význam pro vznik mutací a rozvoj onemocnění (rakovina, neurodegenerativní onemocnění) je v poslední době cílem mnoha studií. V první části je popsána obecná struktura DNA v souvislosti s výskytem R-loop, druhá část se zaměřuje přímo na výskyt a dopady vzniku R-loop na živé organismy.

Klíčová slova: R-loop, DNA, RNA, lokální struktury, analýza, genom, nukleotid, DNA-analyser, rakovina, neurodegenerativní onemocnění

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod a cíl práce

1.1 Úvod do problematiky

Tato závěrečná práce nastiňuje problematiku využití informatiky jako interdisciplinárního nástroje k usnadnění práce vědeckým pracovníkům pomocí zpracování velkého množství dat.

Čím dál vyšší výpočetní výkon nám umožňuje detailnější analýzu deoxyribonukleové kyseliny (2) (dále jen „DNA“). Například odhadovaná délka lidského genomu je až 3.2 miliardy párů bází (Karami, 2013). S odkazem na Gregora Johanna Mendela se tato práce věnuje fascinujícímu hledání lokálních struktur DNA, které snad v budoucnu povede k prohloubení našich znalostí v oboru genetiky.

Jako lokální struktury můžeme označit určité shluky nukleotidů (2.1), které se formují ve vzorcích, mají specificky dané pořadí, či počet jednotlivých typů nukleotidů. Velká část DNA je nekódující (2.2), ale stále častěji zkoumáním DNA nacházíme určité lokální struktury a sekvence nukleotidů, které hrají významnou roli v tom, jak jsou organismy odolné vůči mutacím a chorobám.

Lokální struktura, kterou se tato práce zabývá, se nazývá **R-loop**. Tato struktura je detailně popsána v kapitole 3. Jedná se o třívláknovou strukturu nukleové kyseliny, která se skládá z hybridu deoxyribonukleové kyseliny a ribonukleové kyseliny (pozn. označujeme ji jako RNA-DNA hybrid).

1.2 Cíl práce

Cílem práce je integrace modulu pro detekci a analýzu struktury R-loop do stávajícího serveru `DNA Analyser`[1], který vznikl na ústavu informatiky Provozně ekonomické fakulty Mendelovy univerzity. Server již obsahuje různé analýzy, například `G4Hunter` (Brázda; Kolomazník; Lýsek; Bartas et al., 2019), nebo `Palindrome analyser` (Brázda; Kolomazník; Lýsek; Hároníková et al., 2016). Tento server je provozován ve spolupráci s Akademií věd České republiky. Měl jsem tak v průběhu tvorby práce okamžitou zpětnou vazbu na zpracování analýz a dostatek vstupních dat k dispozici.

Výsledkem je modul, který provádí analýzu sekvencí nukleotidů z databáze, případně je možné analýzy provádět pomocí API. Analýza pak poskytne výzkumníkům z Akademie věd informace o tom, kde se v sekvenci struktura R-loop má vytvořit, a jak vypadá. Výstupem je pak vizualizace analýzy na webovém serveru. Pro detailnější informace o analýze je možné exporotvat data ve dvou podporovaných formátech, konkrétně `bedGraph` a `CSV`.

[1] `https://bioinformatika.pef.mendelu.cz`

Práce se skládá z následujících dílčích úkonů:

- Vývoj specializované platformy pro analýzu DNA sekvencí se zaměřením na velké objemy dat
- Analýza lokální DNA struktury R–loop a návrh implementace nástroje pro její zpracování
- Implementace nástroje pro analýzu struktury R-loop v jazyce Java a jeho integrace na webový server pro analýzu lokálních struktur DNA
- Rozšíření API rozhraní v prostředí Python pro analýzu struktury R-loop
- Modifikace R-loop algoritmu pro využití paralelismu
- Zhodnocení a otestování funkčnosti vytvořeného systému

2 Nukleové kyseliny

Při popisu nukleových kyselin se zaměřím převážně na ty biologické procesy a formování struktur, které pro nás mají význam z hlediska automatizace. V případě propojení informačních technologií a biologických procesů se zaměřuji na popis těch biologických procesů a pojmů, které následně používám v algoritmické části diplomové práce.

2.1 Deoxyribonukleová kyselina (DNA)

DNA je tvořena jednotlivými nukleotidy (2.1) a je nositelkou informace důležité pro růst, přežití a reprodukci živých organismů. Je také nositelkou dědičnosti buněčných organismů, což znamená, že téměř každá buňka (přesněji každé jádro buňky) v lidském těle obsahuje stejnou DNA.

Už v úvodě nacházíme podobné principy jako při použití programovacích jazyků, kterými jsou dědičnost (dědičnost DNA), zapouzdření (existence membrán jednotlivých buněk), nebo abstrakce (komunikace buněk pomocí signálů, nebo pomocí hormonů). DNA je zajímavá tím, že neustále pracuje jako malá výpočetní jednotka, která umožňuje replikaci a tvorbu proteinů na základě šablony (kterou jsou právě určité sekvence DNA). Replikaci se příliš tato práce nevěnuje, ale tvorbu proteinů je důležité nastínit kvůli možnému vzniku struktury R-loop při tomto procesu.

Struktura DNA

V této části popíši stejné struktury a prvky, se kterými pracuje výsledný modul. Základní stavební jednotkou molekuly DNA je *nukleová báze*.

Nukleotid vzniká spojením nukleové báze, deoxyribózy (cukru) a jednoho, nebo více zbytků kyseliny fosforečné.

Máme pět základních nukleových bází:

- Adenin (A)
- **Guanin (G)**
- Thymin (T)
- Cytosin (C)
- Uracil (U) - nachází se v RNA

Guanin je zvýrazněn, protože hraje významnou roli při hledání struktury R-loop v sekvenci (viz tabulka č. 3).

Tvorba DNA

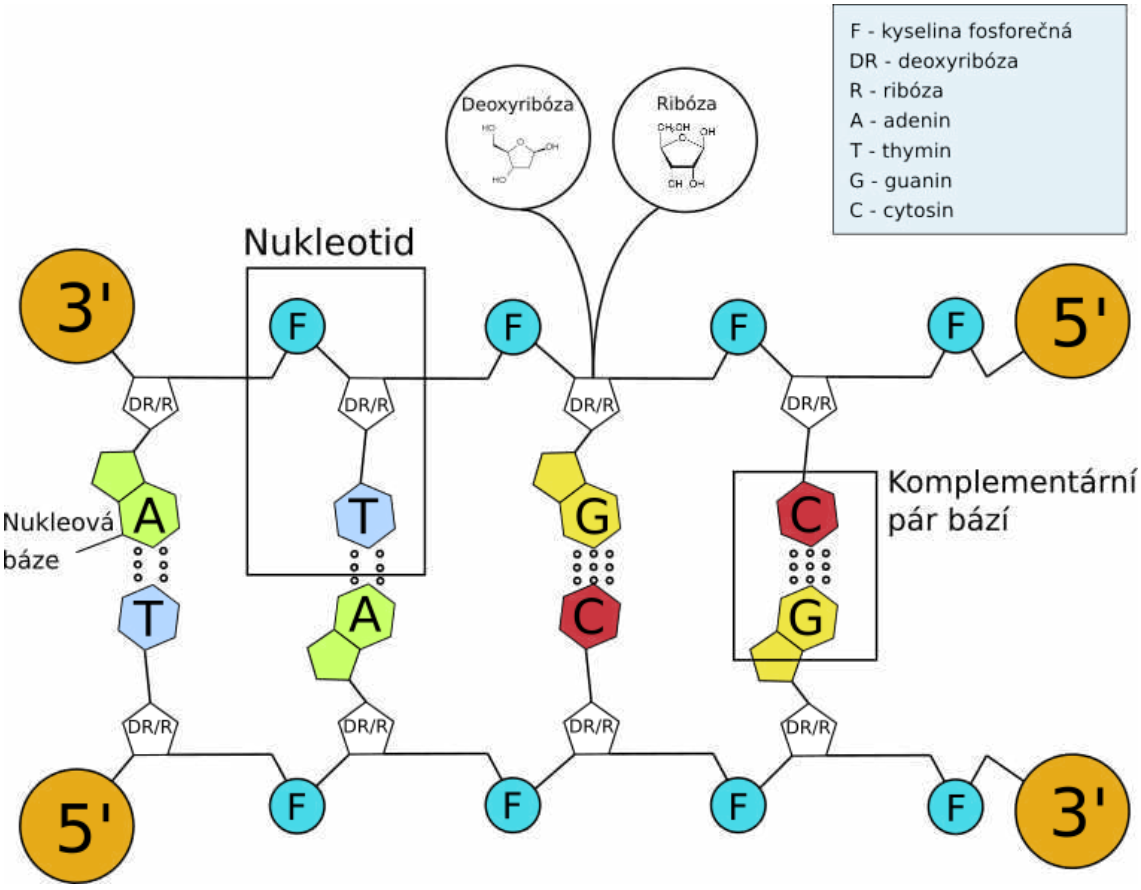
Při vzniku dvouřetězcové struktury DNA se každá báze se spojí s komplementární bází a vytvoří komplementární páry podle následujících pravidel:

Nukleová kyselina	Nukleové báze	Komplement
DNA	A, G, T, C	A=T G≡C
RNA	A, G, U, C	A=U G≡C

Tabulka 1: Pravidla komplementarity pro nukleové báze. Počet vodorovných čar mezi bázemi značí počet vodíkových vazeb.
Zdroj: (Pray, 2013)

Následující obrázek zobrazuje jednotlivé pojmy a vazby v molekule DNA. Obrázek jsem vytvořil, aby bylo zřejmé, se kterými částmi molekuly DNA algoritmus pracuje, protože sám o sobě zprostředkovává vysokou míru abstrakce. V obrázku je také vidět *direkcionalita*.

Direkcionalita je důležitá pro algoritmické zpracování vstupních i výstupních dat. Protože platí pravidlo komplementarity pro nukleové báze, na vstupu programu dostaneme pouze jeden směr řetězce nukleotidů. Druhý získáme právě dopočítáním komplementu a na výstup pak zaznamenáme jaká je direkcionalita sekvence nukleotidů, nad kterou provádíme analýzu.



Obrázek 1: Deoxyribonukleová kyselina s legendou. Direkcionalita je naznačena v obou směrech.

Direkcionalita

Direkcionalita určuje směrovitost struktury molekul nukleových kyselin. Vzniká důsledkem biosyntézy molekul. Značení `5'` a `3'` nám určuje pozici uhlíku molekuly ribózy, nebo deoxyribózy (cukru), na který je vázána kyselina trihydrogenfosforečná (H_3PO_4). Směr je také významný z biologického hlediska, ve směru `5' -> 3'` dochází k replikaci DNA a syntetizaci RNA (Lodish, 2008). Z toho důvodu i my předpokládáme vstupní data ve stejném směru. V případě výstupního formátu `bedGraph` pak používáme oba směry, ale s notací plus (`+`), nebo mínus (`-`). Možná označení direkcionality můžeme shrnout do následující tabulky:

Direkcionita	Alternativy
`5' -> 3'`	`+`, sense, coding
`3' -> 5'`	`-`, anti-sense, non-coding

Tabulka 2: Shrnutí možného označení direkcionality. U vstupních dat předpokládáme direkcionalitu `5' -> 3'`.
Zdroj: (Giudicelli et al., 1999)

2.2 Ribonukleová kyselina (RNA)

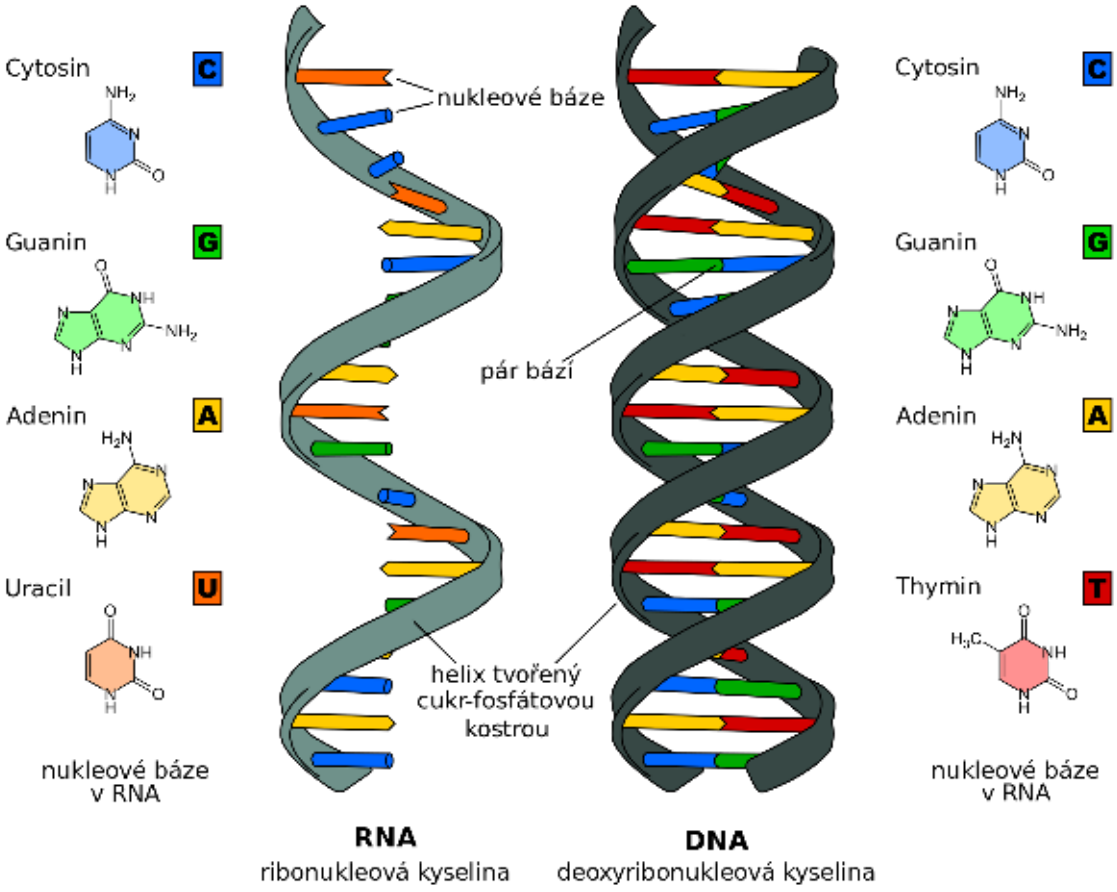
Jelikož se tato práce věnuje hlavně identifikaci a analýze struktury R-loop (RNA--DNA hybrid), je nutné zmínit i molekuly RNA a její funkce, případně jak ovlivňuje tvorbu R-loop.

RNA je jednovláknový polymer nukleové kyseliny, který se skládá z následujících nukleových bází:

- Adenin (A)
- Uracil (U)
- Cytosin (C)
- Guanin (G)

Rozdíl oproti DNA je ve výskytu nukleové báze - uracilu. RNA také oproti DNA obsahuje ribózu, což způsobuje její náchylnost k hydrolýze[2]. Důsledkem je nestabilita RNA oproti DNA, která je stabilní a uchovává informaci. To je pravděpodobně jeden z důvodů, proč je právě DNA nositelem genetické informace v organismech, nikoliv RNA. RNA slouží k syntéze proteinů a zastává přímou funkci v buněčném metabolismu (Roberts, 2020).

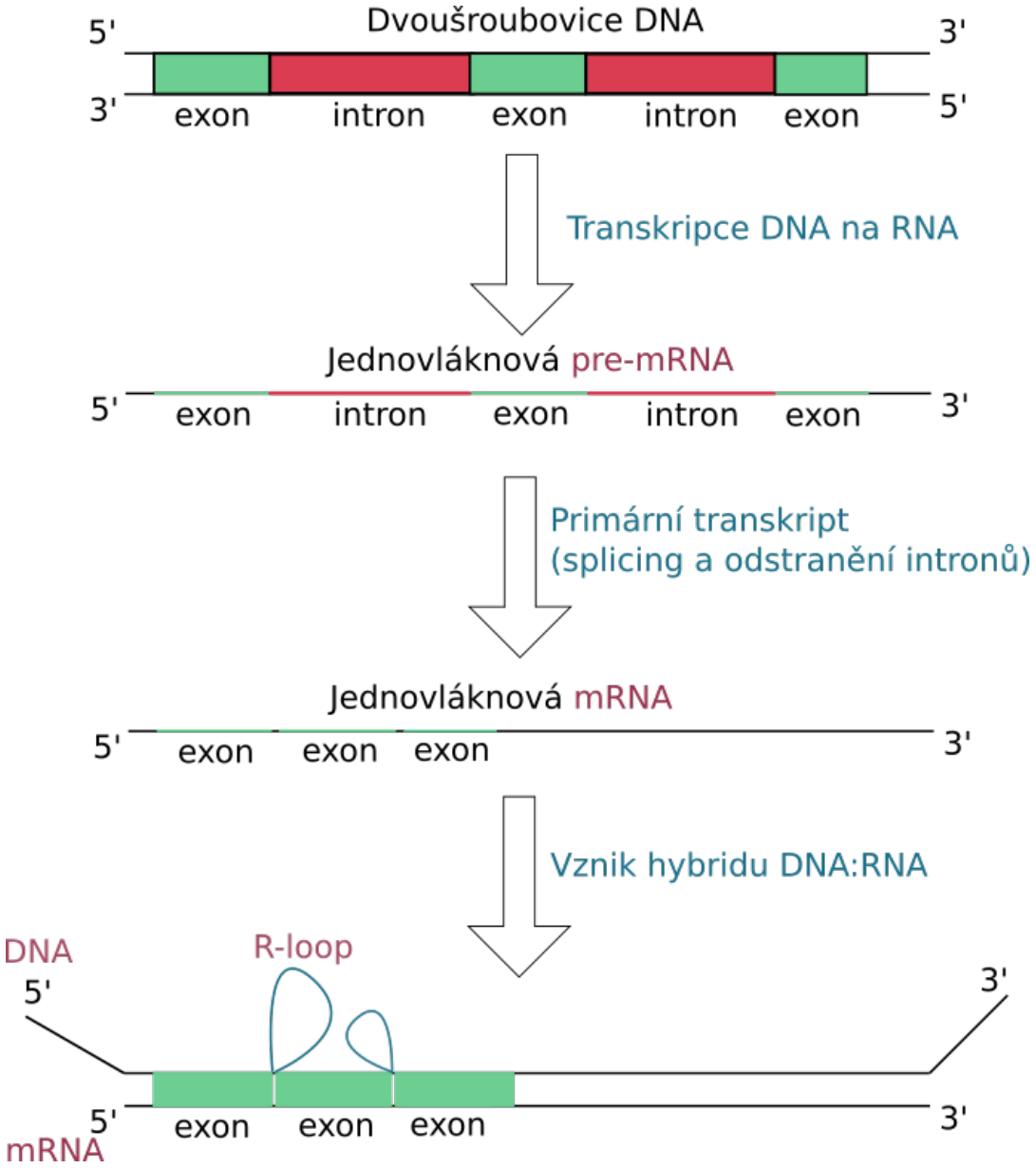
[2] *Hydrolýza:* rozkladná reakce, při které se spotřebovává voda.



Obrázek 2: Rozdíl mezi DNA a RNA.
Zdroj: (Sponk, 2014)

Procesy a posttranskripční modifikace RNA probíhající při transkripci DNA, mohou vést ke vzniku RNA-DNA hybridu.

Následující obrázek zjednodušeně popisuje proces přepisu DNA na mRNA, při kterém může vzniknout R-loop (pozn. proces transkripce a translace je mnohem složitější, použil jsem určitou míru abstrakce pro vyobrazení procesů, které se týkají přímo možného vzniku R-loop).



Obrázek 3: Proces transkripce DNA s možným vznikem struktur R-loop po odstranění intronů.

Exon

Aby mohlo dojít k syntéze proteinů, je nutné nejprve vyčlenit v sekvenci DNA takovou sekvenci, která je **kódující** . Kódující sekvence je právě ta sekvence, která kóduje výsledný protein. *Exony* jsou kódující sekvence DNA uvnitř genomu a zároveň tak označujeme i korespondující sekvenci RNA (Lodish, 2008). Výsledný exom je pak skupina exonů, které vzniknou *splicingem* (2.2) RNA.

Intron

Slovo *intron* pochází z anglického spojení *intragenic region*, tedy oblast uvnitř genu (Gilbert, 1978). Intronem rozumíme jakoukoli **nekódující** sekvenci obsaženou v DNA, nebo při transkripci RNA (Brown, 2012). Tyto sekvence jsou odstraněny při *splicingu* (2.2). Výsledná mediátorová RNA už žádné introny neobsahuje.

Vznik a význam intronů není úplně zřejmý. Tyto nekódující sekvence v lidském genomu tvoří až 25% genomu, což je zhruba 4-5x více, než exony (Sakharkar et al., 2004). Nekódující sekvence DNA však zastávají důležitou funkci, pomocí redundance totiž chrání genom před mutacemi (díky velikosti nekódujících sekvencí často dochází k chybnému přepisu či replikaci právě v nekódujících sekvencích, a tím je zachována správná sekvence kódující výsledný protein). Na druhou stranu je pro buňku přepis redundantních sekvencí velká energetická zátěž (Jo et al., 2015).

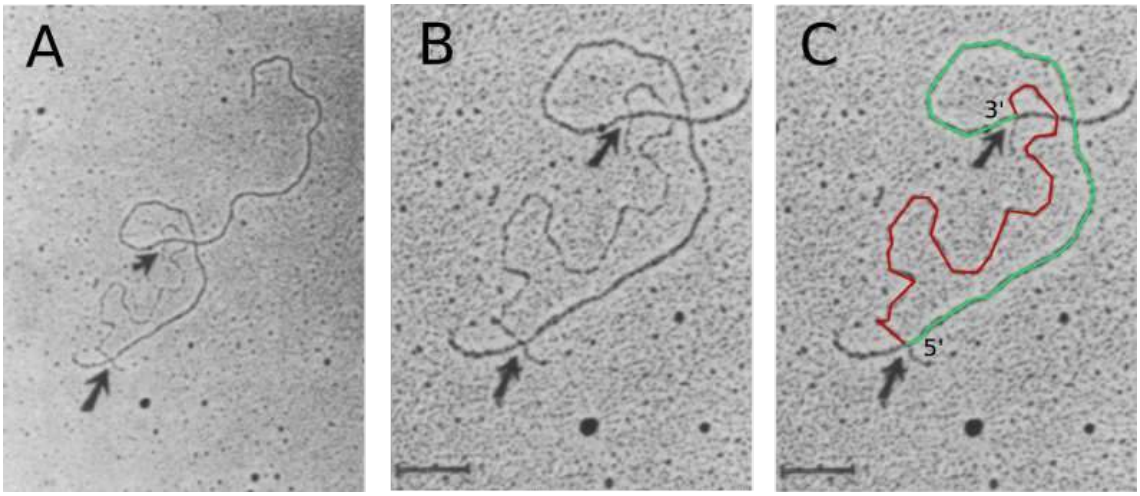
Splicing

Splicing je proces, při které se primární transkript (*pre-mRNA*) transformuje na mediátorovou RNA (*mRNA*), a při kterém dojde k odstranění intronů ze sekvence nukleotidů. Proces splicingu je zobrazen na obrázku 3.

3 R-loop

Jedná se o třívláknové struktury nukleových kyselin, které vznikají hybridizací DNA a RNA. Pokud máme dostatečně velké zastoupení guaninu ve vznikajícím RNA transkriptu, může dojít k hybridizaci s DNA vláknem. Poprvé byly R-loopy objeveny u prokaryot v roce 1976 (White et al., 1977a).

Po dlouhou dobu se R-loopy považovaly za vedlejší produkt transkripce, a tak se jim nevěnovala velká pozornost. V poslední dekádě se však objevuje velké množství studií, ve kterých se objevují důkazy o důležitosti a vlivu těchto struktur. Navíc se ukazuje, že se R-loopy tvoří velmi hojně a mohou vznikat až na 5% genomu savců (Sanz et al., 2016). Název R-loop pochází z termínu „D-loop“ (*displacement loop*), což je podobná struktura, ale každé ze tří vláken je tvořeno DNA.



Obrázek 4: Snímek RNA-DNA hybridu zachycený pomocí elektronového mikroskopu. Na obrázku **A** máme zachycenou kompletní sekvenci. Na obrázku **B** je pořízený detail záběru. Obrázek **C** je doplněním obrázku **B** o směr sekvence nukleotidů vlákna DNA (5’ → 3’) a červeně označenou jednovláknovou DNA. Zeleně je označena dvojice paralelních vláken RNA-DNA hybridu.
Podle: (White et al., 1977b)

3.1 Detekce RNA-DNA hybridu

Strategie určení místa, kde se hybrid vytvoří, je doposud stále prozkoumávána a ověřována. Aby bylo možné R-loopy systematicky predikovat a detekovat, je třeba využít matematických modelů založených na *in vitro* formování hybridů.

Laboratorně lze za určitých podmínek vytvořit prostředí pro formování R-loop. Tato práce proto vychází z predikce místa vzniku na základě již existujících naměřených dat. Tím, že se výzkum R-loop stále rozvíjí a probíhá převážně v in vitro podmínkách, mohou být některá data nepřesná a závislá na prostředí a podmínkách laboratorních testů při kterých vznikají. Existují však poměrně přesné modely založené na přítomnosti *G-clusterů* (pozn. jedná se o shluky nukleotidů bohaté na guanin). Už malé shluky guaninu mají za následek mnohem větší pravděpodobnost vytvoření inicializační zóny R-loop, než náhodná sekvence nukleotidů (Roy et al., 2009). Zónu, ve které se struktura R-loop vytváří, označujeme **R-loop Initiation Zone** (RIZ).

Pro vznik RNA-DNA hybridu je dále nutné vytvoření zóny prodloužení, která už není tolik závislá na přítomnosti G-clusterů, ale faktorem je zde vysoká hustota guaninu v sekvenci nukleotidů. Máme tedy dva faktory, které se od sebe částečně liší, ale oba závisí na přítomnosti guaninu. Zónu, ve které se struktura R-loop prodlužuje, označujeme **R-loop Elongation Zone** (REZ).

Mezi těmito zónami se může, ale nemusí vyskytovat zóna, která RIZ a REZ spojuje. Tu pak označujeme jako **linker**. Linker se skládá z náhodné sekvence nukleotidů a slouží především k posunutí okna, ve kterém měříme hustotu výskytu guaninu pro REZ. Pomocí linkeru je možné prodloužit okno pro detekci nejdelší R-loop s určenou hustotou guaninu.

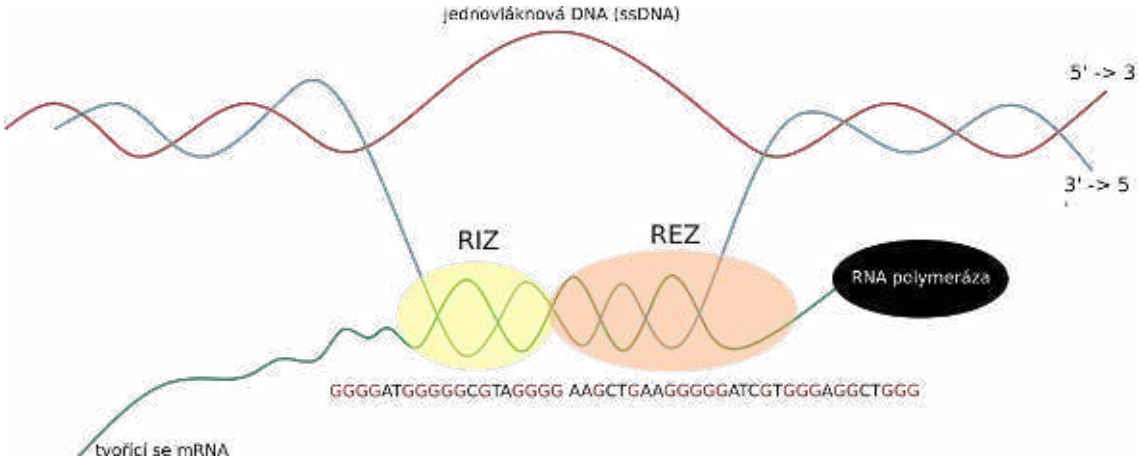
Při vzniku modelu pravděpodobného místa vzniku R-loop je podstatné, že takto můžeme zpracovávat **původní sekvenci nukleotidů jednotlivých genomů**, aniž bychom museli algoritmicky vytvářet transkript RNA, případně mediátorovou RNA. Jak tato vstupní data vypadají, popisuji v části týkající se technického zpracování algoritmu (5.4).

Následující tabulka vyobrazuje, jaký má dopad přítomnost G-clusterů v inicializační zóně a hustota guaninu v elongační zóně, na pravděpodobnost vzniku R-loop v daném segmentu nukleotidů. Bylo použito 9 substrátů (pozn. sekvence, se kterou enzym reaguje), které jsou seřazeny sestupně podle obsahu guaninu. Frekvence tvorby R-loop je počítána jako zastoupení nukleotidů podílejících se na tvorbě R-loop vzhledem k celkovému počtu nukleotidů v sekvenci.

Substrát	**RIZ**	**REZ**	**Frekvence vzniku R-loop** [%]
pDR18A	dva 4G-clustery	4G-clustery	12,3
pDR18C	jeden 4G-cluster	4G-clustery	5,9
pDR18B	žádný G-cluster	4G-clustery	0,8
pDR22A	dva 4G-clustery	max 3G-clustery	5,1
pDR22C	jeden 4G-cluster	max 3G-clustery	1,2
pDR22B	žádný G-cluster	max 3G-clustery	0,6
pDR26A	dva 4G-clustery	max 2G-clustery	0,2
pDR26C	jeden 4G-cluster	max 2G-clustery	<0,1
pDR26B	žádný G-cluster	max 2G-clustery	<0,1

Tabulka 3: Porovnání obsahu guaninu a přítomnosti G-clusterů v sekvenci nukleotidů s pravděpodobností vzniku RNA-DNA hybridu.
Zdroj: (Roy et al., 2009)

Vznikající třívláknové struktury pak zobrazuji na následujícím obrázku. Z tabulky lze vyčíst, že pro úspěšné vytvoření R-loop potřebujeme alespoň dva G4-clustery, nebo tři G3-clustery v inicializační zóně (RIZ) a dostatečné zastoupení guaninu v elongační zóně (REZ). Mutace vzniká spojením nekódujícího vlákna DNA (ve směru 3' → 5') s mediátorovu RNA.



Obrázek 5: Formace RNA-DNA hybridu za ideálních podmínek: Přítomnost G-clusterů v inicializační zóně a alespoň 40% obsahu guaninu v elongační zóně.
Podle: (Wongsurawat et al., 2011)

Výsledky výzkumu zabývající se výskytem RNA-DNA hybridů v genomech můžeme zapsat pomocí dvou následujících modelů:

$$\underbrace{G_{n1}N_xG_{n1}N_xG_{n1}}_{RIZ} \cdot \underbrace{N_y}_{linker} \cdot \underbrace{N_z}_{REZ} \begin{cases} RIZ \geq 50\%\, G \\ REZ \geq 40\%\, G \end{cases}, \quad (m_1) \tag{1}$$

$$\underbrace{G_{n2}N_xG_{n2}}_{RIZ} \cdot \underbrace{N_y}_{linker} \cdot \underbrace{N_z}_{REZ} \begin{cases} RIZ \geq 50\%\, G \\ REZ \geq 40\%\, G \end{cases}, \quad (m_2) \tag{2}$$

kde:

G_{n1} ... nukleotidy guaninu, kde $n1 \geq 3$,
G_{n2} ... nukleotidy guaninu, kde $n2 \geq 4$,
N_x ... libovolná sekvence nukleotidů, kde $1 \leq x \leq 10$,
N_y ... libovolná sekvence nukleotidů, kde $0 \leq y \leq 50$,
N_z ... libovolná sekvence nukleotidů, kde $100 \leq z \leq 2000$
Zdroj: (Jenjaroenpun et al., 2015)

Tyto dva modely pro svou analýzu používá nástroj vznikající v rámci mé diplomové práce a využívá je i doposud jediný existující nástroj pro detekci R-loop. Oba tyto nástroje srovnávám v kapitole (6.2).

Při srovnání s výsledky z laboratoří na základě dat o vzniku RNA-DNA hybridů vzniklých *in vitro* a *in vivo* mají tyto modely až 91% přesnost (Jenjaroenpun et al., 2015), proto jsme se rozhodli tyto modely uplatnit i v našem nástroji.

3.2 Vliv R-loop na živý organismus

Nyní víme jak se R-loopy tvoří, jak vypadají a jakým způsobem je možná jejich detekce v rámci dlouhých sekvencí nukleotidů. Na závěr kapitoly se pokusím krátce nastínit, proč se tato diplomová práce věnuje právě nástroji pro analýzu lokální struktury R-loop a jakých výsledků můžeme dosáhnout, pokud správně aplikujeme vyvinutý nástroj pro identifikaci a analýzu těchto struktur.

Negativní vliv

V živých organismech existuje mnoho ochranných mechanismů, které zapříčiňují rozpad RNA-DNA hybridů. Tyto ochranné mechanismy jsou důležité, protože nadměrný výskyt R-loop způsobuje nestabilitu genomu.

Častým jevem je mutace jednovláknové DNA (ssDNA) během transkripce, nebo působení RNA ve vzniklé struktuře R-loop jako primer (pozn. řetězec nukleové ky-

seliny DNA, nebo RNA, který slouží jako počáteční místo replikace), který může spustit neplánovanou rekombinaci (Costantino et al., 2015).

Ukazuje se, že zóny genomu, kde se tvoří tyto hybridy, mají určitou spojitost s podmínkami, při kterých dochází k rakovinnému bujení. Stabilita genomu je charakteristickým znakem rakovinných buněk. Při poruše, nebo vyřazení genu *BRCA2*, který potlačuje vznik nádoru, dochází k vyšší akumulaci RNA-DNA hybridů (Bhatia et al., 2014).

Stejně tak se výskyt těchto struktur spojuje s výskytem některých vývojových poruch. Je stále otázkou, zda se jedná o kauzalitu, nebo pouhou korelaci. Vědecké studie objevily souvislost s výskytem struktury R-loop a následujících vývojových poruch a neurodegenerativních onemocnění (Richard et al., 2017):

- Amyotrofická laterální skleróza (ALS)
- Aicardiho–Goutièrův syndrom (AGS)
- Praderův-Williho syndrom (PWS)

Pozitivní vliv

R-loopy mají vliv na *genovou expresi*, (tedy syntézu proteinu), při kterém dochází k transformaci informace uložené v genu na reálně existující buněčnou strukturu (např. protein).

Ukazuje se, že ovlivněním výskytu struktury R-loop v rakovinných buňkách můžeme ovlivnit jejich přežití. Správnou identifikací místa výskytu struktury R-loop a jejím následným zkoumáním je možnou cestou pro léčbu rakovinného bujení.

Výzkumem vzniku R-loop v rakovinných buňkách bylo zjištěno, že RNA-DNA hybridizace, která souvisí s destabilizací struktury DNA, může ovlivnit následující faktory (Boros-Oláh et al., 2019):

- Doba přežití rakovinné buňky
- Účinnost některých látek užívaných v léčbě nádorů

Nástroj pro detekci R-loop v rakovinných buňkách by tak mohl být užitečný při vývoji a analýze účinnosti onkologických léčiv.

4 Analýza existujících nástrojů

Výzkum v oblasti R-loop je teprve „v plenkách“ a většina článků, které řeší jejich lokalizaci a význam v laboratorních podmínkách, byla publikována až v posledních letech. Z existujících nástrojů jsem nalezl pouze jeden, který má algoritmus založený na laboratorních datech.

Jedná se o nástroj `QmRLFS-finder`, který predikuje tvorbu R-loop. `QmRLFS-finder` je volně dostupný na adrese `http://rloop.bii.a-star.edu.sg/?pg=qmrlfs-finder` buď jako webový server, nebo jako skript v jazyce Python verze 2. Nástroj už je delší dobu nevyužívaný a používá staré moduly.

4.1 QmRLFS-finder

Vyhledávací algoritmus

Algoritmus vyhledávání potenciálních míst tvorby R-loop je popsán dvěma modely (m_1, m_2). Formálně jsou modely zapsány pomocí vzorců 1 (m_1) a 2 (m_2).

Samotný algoritmus je následující:

Algorithm 1: QmRLFS-finder

Result: Array of strings representing R-loop
Data: Input FASTA file
if *Regular expression matches* m_1 *or* m_2 *and* $\%G \geq 50\%$ **then**
  set RIZ_{start} as position;
  store RIZ_{end}; set $linker_{start}$ as RIZ_{end};
  set REZ_{end} as $REZ_{start} + 100$;
  for $i \leftarrow 0$ **to** 50 **do**
    set $linker_{end}$ as i;
    set $REZ_{sequence}$ as REZ_{start} to REZ_{end};
    while *REZ length* $\leq$ *2000 and* $\%G \leq 40\%$ **do**
      `# finds longest REZ with given Guanine density`;
      add 1 nucleic base to $REZ_{sequence}$;
    end
  end
end

V následující tabulce je pak srovnání predikce tohoto algoritmu a laboratorních dat na jednotlivých genech. Ve valné většině případů je predikce velmi přesná.

Gen	Buněčná linie	Metoda identifikace R-loop	Experimentálně	Predikce
Immunoglobulin	lidský epiteliální karcinom	stopování r-loop	+	+
RHOH	in vitro	elektronová mikroskopie	+	+
Foxo4	testy na myších	DRIP-PCR	+	+
TP53	SKOV3	DRIPq-PCR	-	-
PBX1	SKOV3	DRIPq-PCR	-	+
APOE	Ntera2	DRIP-seq a stopování r-loop	+	+
Airn	kmenové buňky myší	stopování r-loop	+	+

Tabulka 4: Srovnání algoritmické predikce a experimentální predikce R-loop. Zdroj: (Jenjaroenpun et al., 2015)

Vstupní data

Jediným vstupním formátem, který nástroj poskytuje, je formát `FASTA`. `FASTA` formát umožňuje uložit do jednoho souboru dlouhou sekvenci společně s jejími metadaty. Počátečním identifikátorem metadat je symbol `>`, a za ním následuje popis sekvence. Obsahuje-li soubor více sekvencí pod sebou, pak se jedná o rozšířený formát `multiFASTA`, ten ale není nástrojem podporovaný (NCBI, 2001).

Obsah souboru může vypadat následovně:

```
>NM_010450.3 Mus musculus homeobox A11 (Hoxa11), mRNA
AAATTTCTACTTCACGGATCCGCTTCAAAGAGGCAGCTGCAGTGGAGAATCATGTTAAGCTCGGCTACTG
CGGAGAGCCCAAGGTAGCCCAATGATGGATTTTGATGAGCGTGGTCCCTGCTCCTCTAACATGTATTTGC
CAAGTTGTACTTACTACGTCTCGGGTCCAGATTTCTCCAGCCTCCCTTCTTTTTTGCCCCAGACCCCGTC
TTCGCGCCCAATGACATACTCCTACTCCTCCAACCTGCCCCAGGTCCAACCCGTGCGCGAAGTGACCTTC
```

Omezením nástroje pro vstupní data je maximální délka sekvence 300 000 znaků a maximální velikost `FASTA` souboru `300 kB`.

Ve formuláři je také na výběr jeden ze dvou modelů detekce struktur.

Výstupní data

Program `QmRLFS-finder` poskytuje následující výstupní formáty dat:

- `TABLE`
  Jednoduchý textový formát, kde jsou všechna data z analýzy. Řádky jsou oddělené znakem nového řádku (`CRLF`) a sloupce jsou odděleny mezerou.
- `FASTA`
  Formát, ve kterém jsou zapsány celé sub-sekvence původního genomu. Každý `FASTA` záznam je sekvence RIZ, linker a REZ. Začátek a konec této sekvence v původní sekvenci je uveden v hlavičce `FASTA` záznamu.

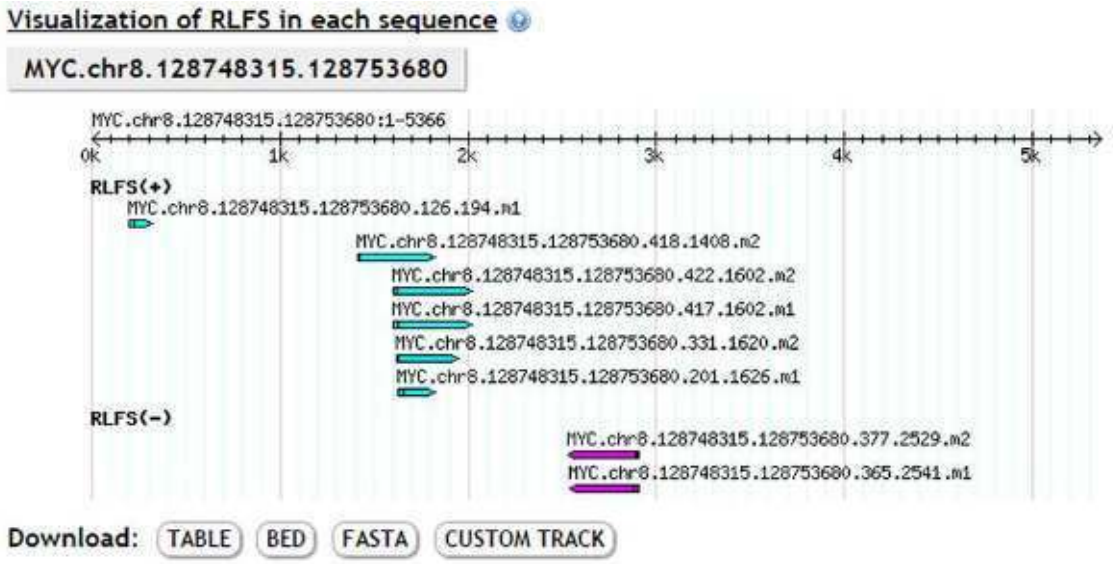
- `BED`
  Formát, ve kterém je přehledně vidět začátek a konec sekvence označující R-loop.
- `CUSTOM TRACK`
  Formát podobný `BED` formátu a je používán nástrojem `Genome browser` vyvinutým na University of California, Santa Cruz [3]. Slouží k vkládání vlastních *anotací* do `Genome browseru`. *Anotace* nám určuje kontext určité sekvence v daném místě. Jsou to metadata, která určují, co které geny v genomu dělají (Stein, 2001).

V každém formátu je jiný typ dat, ale celkově je možné získat informace o následujících metrikách, které nástroj počítá pro každou nalezenou strukturu R-loop:

- Použitý model pro detekci R-loop
- Začátek, konec a délka RIZ (pozice v genomu)
- Obsah guaninu v RIZ (hustota i počet)
- Počet 3G-clusterů v RIZ
- Počet 4G-clusterů v RIZ
- Sekvence RIZ
- Sekvence linkeru
- Sekvence REZ
- Začátek, konec a délka REZ (pozice v genomu)
- Obsah guaninu v REZ (hustota i počet)
- Počet 3G-clusterů v REZ
- Počet 4G-clusterů v REZ
- Směr DNA, ve kterém byla R-loop detekována (`+`, `-`)

[3] `https://genome.ucsc.edu/index.html`

Zajímavým výstupem je vizualizace výsledků v `Genome browseru`. Na celé sekvenci můžeme pozorovat směr struktury R-loop, jejich počet a délku velmi přehledně.



Obrázek 6: Zobrazení výsledků pomocí UCSC Genome browseru. V oblasti 128748315 - 128753680 chromozomu číslo 8. Celkově záznam obsahuje šest R-loop ve směru `+` a dvě R-loopy ve směru `-`

4.2 Databáze R-loop

Vědecká skupina, která nástroj `Qm-RLFS finder` vyvinula, také poskytuje `R-loop DB` s poslední aktualizací dat z roku 2016. Obsahuje různé genomy (člověk, šimpanz, octomilka, kvasinky, …), které ve své struktuře obsahují R-loopy s detailním popisem i náhledem, jak jednotlivé R-loopy vypadají.

Databáze je dostupná na adrese: `http://rloop.bii.a-star.edu.sg/`.

4.3 Zhodnocení

Jako velké pozitivum nástroje hodnotím integraci s UCSC Genome browserem. Vkládání anotací pro jednotlivé genomy je důležitá vlastnost nástroje pro srovnání s dalšími analýzami.

Databáze R-loop je pro nás také velice přínosná pro další rozšiřování našeho nástroje.

Nevýhodou je, že nástroj už není delší dobu udržovaný. Poslední aktualizace databáze je z roku 2016 a implementace je v `Pythonu` verze 2.7, která už od 1. ledna 2020 není dále vyvíjena a podporována (Python, 2019). Podpora vstupních formátů je řešitelná pomocí převodu těchto formátů, ale v rámci zpřístupnění práce s nástrojem pro vědecké pracovníky, je to také podstatná součást uživatelsky přívětivých nástrojů pro práci se sekvencemi.

Obecně tedy převládá potřeba aktuálního nástroje, který je nadále rozšiřitelný pomocí modulů pro detekci a analýzu neprobádaného světa R-loop (například agregátor statistik detekovaných R-loop).

5 R-loop-R

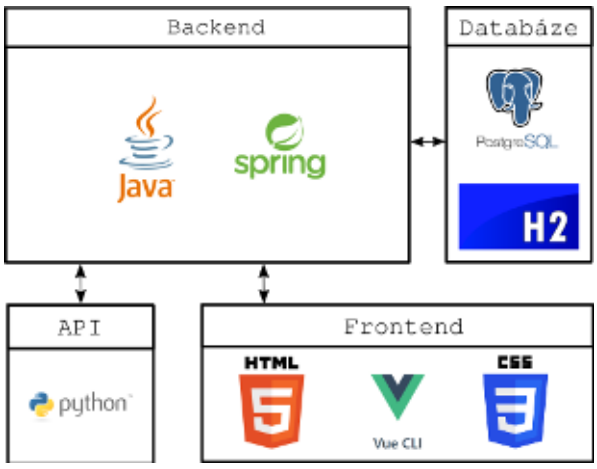
Nástroj `R-loop-R` vznikl jako výsledek této práce. Jedná se o modul, který je součástí serveru `DNA-Analyser`[4]. `R-loop-R` přidává novou analýzu R-loop pro sekvence, se kterými `DNA-Analyser` pracuje. Modul je optimalizován pro zpracování dlouhých sekvencí (celých chromozomů). Výstup je pak buď grafický, nebo ve specifickém formátu podle volby uživatele (`csv`, `bedGraph`).

5.1 Motivace

Motivací pro návrh a implementaci nástroje pro detekci a analýzu R-loop je možnost využití a rozšiřování takového nástroje podle představ Akademie věd České republiky. S panem doc. Mgr. Václavem Brázdou, Ph.D., který je konzultantem této práce jsme neustále v kontaktu a bylo tak možné modul ladit přesně podle představ a potřeb biofyzikálního ústavu. Zároveň díky tomu je zde možnost vyzkoušet si mnoho optimalizačních algoritmů pro práci s velkými daty, což právě sekvence nukleotidů nabízejí. V budoucnosti bychom rádi modul `R-loop-R` rozšířili o predikci za pomoci neuronové sítě.

5.2 Výběr technologií

`DNA Analyser` je poměrně komplexní webový server. Strukturu tohoto serveru znázorňuje obrázek níže:



Obrázek 7: Architektura webového serveru DNA Analyser zobrazující používané technologie.

[4] `https://bioinformatika.pef.mendelu.cz/`

Protože je nástroj integrován do webového serveru, měl jsem v rámci diplomové práce možnost osvojit si mnoho technologií které jsou v rámci serveru implementovány. Části, se kterými jsem pracoval, nebo je v rámci práce rozšiřoval, jsem rozdělil na následující celky:

Backend

- `Java Platform, Standard Edition 8`

  Hlavní část mé práce spočívala v implementaci a integraci algoritmu v jazyce `Java`. Během vývoje modulu navíc došlo k rozšíření platformy na verzi Java Standard Edition 8. To přineslo možnosti optimalizace buď použitím *Stream API*, nebo *lambda výrazů*, které jsou v této verzi nové.

  Stream API je součástí balíčku `java.util.stream`, který poskytuje rozhraní pro práci se streamy. Vytvořením streamových vláken z generických datových typů, které poskytuje jazyk Java můžeme docílit ohromného zvýšení efektivity programu, tak jako se to povedlo v této práci. Streamy také pracují s generátorovými funkcemi, proto tolik nezatěžují paměť. Klasické využití streamu vypadá následovně:

  1. Vytvoření streamu z datového typu `Array` pomocí operace `.stream()`
  2. Redukce streamu na konkrétní data pomocí filtru: `.filter()`
  3. Aplikace mapy na jednotlivá data: `.map()`
  4. Celková agregace dat: `.sum()`

  Lambda výrazy spolupracují se streamy většinou při filtrování, nebo mapování atributů, kdy potřebujeme specifické uživatelem definované chování. S lambda výrazy pracujeme následovně (příklad součtu dvou čísel):

  ```
  (int x, int y) -> x + y;
  ```

  Tyto dva koncepty se výborně doplňují při filtrování a mapování atribut.
  Konkrétní příklady použití v rámci diplomové práce uvádím v části týkající se implementace algoritmu (5.6)

- `Spring`

  Jedná se o framework, který zjednodušuje práci při psaní webové aplikace. Podporuje psaní `RESTful` aplikací založených na `CRUD` operacích. Díky definici endpointů, které server vynáší na rozhraní, je tak možné provázat jednotlivá

volání API na server s akcemi na backendu. Příkladem práce s `RESTful` rozhraním jsou následující endpointy (uvádím endpointy bez parametrů pro kratší zápis):

- `POST /api/api/analyse/rloopr HTTP/1.1` (spuštění analýzy)
- `GET /api/api/analyse/rloopr/{id}/average/length HTTP/1.1` (výpočet průměrné délky R-loop v dané analýze)
- `PUT /api/api/analyse/rloopr/{id}/tags HTTP/1.1` (úprava tagů týkajících se analýzy)
- `DELETE /api/api/analyse/rloopr/{id}/ HTTP/1.1` (smazání analýzy z databáze)

Pomocí dekorátorů je tak možné jednoduše namapovat `HTTP` metody společně s jejich endpointy a přidat například autorizaci nebo očekávaný návratový kód.

```
@GetMapping("/id/average/length")
@ResponseStatus(HttpStatus.OK)
@PreAuthorize("isAuthenticated()")
```

Užitečnou vlastností Spring frameworku je také možnost *batchování*. Batchování umožňuje automatizaci požadavků na server. Využíváme jej při analýze dlouhých sekvencí, kdy se batch worker dotazuje serveru, zda už je analýza hotová, dokud analýza nedoběhne.

- `H2`

Systém pro správu relační databáze, který je integrovaný v Java aplikaci. Tento systém používá podmnožinu příkazů SQL ke komunikaci (dotazování nad databází). Tabulky mohou být buď perzistentní, nebo dočasné, záleží podle nastavení. Na serveru DNA analyser jsou ve vývojovém prostředí dočasné a v produkční verzi perzistentní. Dočasné tabulky používají *hašovací tabulky a stromy* pro ukládání dat, perzistentní tabulky používají *b-stromy*.
Ukládání dat do databáze jsem v práci využil pro ty neobjemnější celky dat. H2 slouží k uložení hlavně:

- Sekvencí (vstupních dat analýzy)
- Analýz (výstupních dat analýzy)

Každá sekvence a analýza je uložena ve formě binárních dat jako samostatná databáze v databázovém systému H2.

- `PostgreSQL`

  Databázový systém, který funguje na většině operačních systémů. Jedná se o open-source projekt, který se drží SQL standardů. Databázové transakce `PostgreSQL` podporují vlastnosti *ACID*, tedy:

  – Atomicitu

  – Konzistenci

  – Izolaci

  – Durabilitu

  Pro indexaci dat používají *b-stromy*. Na serveru se do `PostgreSQL` databáze ukládají převážně metadata pro server a většina tabulek je vazebních.

- `Flyway`

  Nástroj flyway slouží k migraci databáze. Funguje na principu inkrementálních změn v databázi. Každá migrace (změna databázového schématu) je v odděleném souboru označeným verzí migrace a popisem, co migrace provádí:

  `V42__donot_panic.sql`

  Pokud máme prázdnou databázi, migrace proběhne bez problémů. U neprázdné databáze dochází k výpočtu *kontrolního součtu* pro jednotlivé verze migrací. Pokud někdo chybně upravil již stávající migraci, kontrola je neplatná a migrace skončí s chybovou hláškou. Je tak potřeba vždy každou novou migraci přidat do nového souboru.

  Také se nedoporučuje úprava dat pomocí migrace, protože se aplikuje při každém spuštění serveru.

Část týkající se backendu shrnuje obrázek na následující stránce, který znázorňuje propojení a strukturu databází pro modul R-loop-R (tabulky pro data uživatele nejsou součástí z důvodu zjednodušení obrázku):

Obrázek 8: Struktura databází pro R-loop-R analýzy. Znázorňuje vztah mezi PostgreSQL a H2 databázovými systémy.

Frontend

Veškeré frontendové komponenty jsou psány pomocí následujících technologií:

- `HTML`
- `CSS`
- `Javascript`
- `Vue CLI`

Z těchto technologií zmiňuji více informací pouze o `Vue CLI`, ostatní jsou poměrně dobře známé a rozšířené. Navíc nejsou hlavním předmětem práce, pokud nebereme v potaz finální vizualizaci výsledků.

`Vue CLI` umožňuje psaní komponent a předávání proměnných mezi těmito komponentami. Součástí je také možnost integrace *linteru*, což je nástroj, který pomáhá se zvyšováním čitelnosti a kvality kódu.

Continuous integration

Používáme několik nezávislých testů pro každou kategorii:

- **E2E testy (end-to-end)**

  Testy, ve kterých kontrolujeme průchod celým systémem. Netestujeme jednotlivé části, ale systém jako celek. Simuluje chování uživatele. Tato část není součástí systému.

- **Integrační testy**

  Testy, které kontrolují správnou funkci jednotlivých částí systému. Příkladem těchto testů může být služba `ExporterService`, která je součástí celého systému, ale má za úkol pouze exportovat data z databáze do výstupního souboru v požadovaném formátu. Tuto službu pak testuji jako součást systému, zda je správně integrována.

- **Unit testy**

  Testy lokálních metod a funkcí. Slouží k ověření, zda implementované algoritmy fungují správně. Například testy pro metodu `checkGRichness`, která v sekvenci *Rloop elongation zone* (RIZ) zjišťuje hustotu guaninu.

K realizaci těchto testů slouží jazyk `Groovy`. Jazyk je možné integrovat s Javou, aniž bychom museli řešit modifikátory přístupu jednotlivých metod. Stačí je pouze v hlavičce groovy souborů naimportovat.

Testy jsou velmi dobře čitelné a dá se z nich rychle vyčíst co právě testují i bez znalosti kontextu. To je dáno tím, že jazyk Groovy je psán pomocí agilní metodiky *Behavior driven development*, který je mezi testovacími nástroji stále více populární. Každý test je rozdělen na konkrétní části, které popisují co se má stát za jakých podmínek. Test se pak vyhodnocuje následovně:

1. `setup()` (nebo-li boilerplate)

   Znovupoužitelný kód, kde se inicializuje prostředí před každým testem.

2. `Konkrétní test`
   a) Blok `given:` (vstupní podmínky konkrétního testu)
   b) Blok `when:` (provedení akce, po které očekáváme výstup)
   c) Blok `then:` (Porovnání očekávaného a testovaného výstupu)

Uvádím konkrétní příklad unit-testu metody `checkGRichness`. Vstupní data nemají dostatek guaninu, proto vrací výsledek `false`:

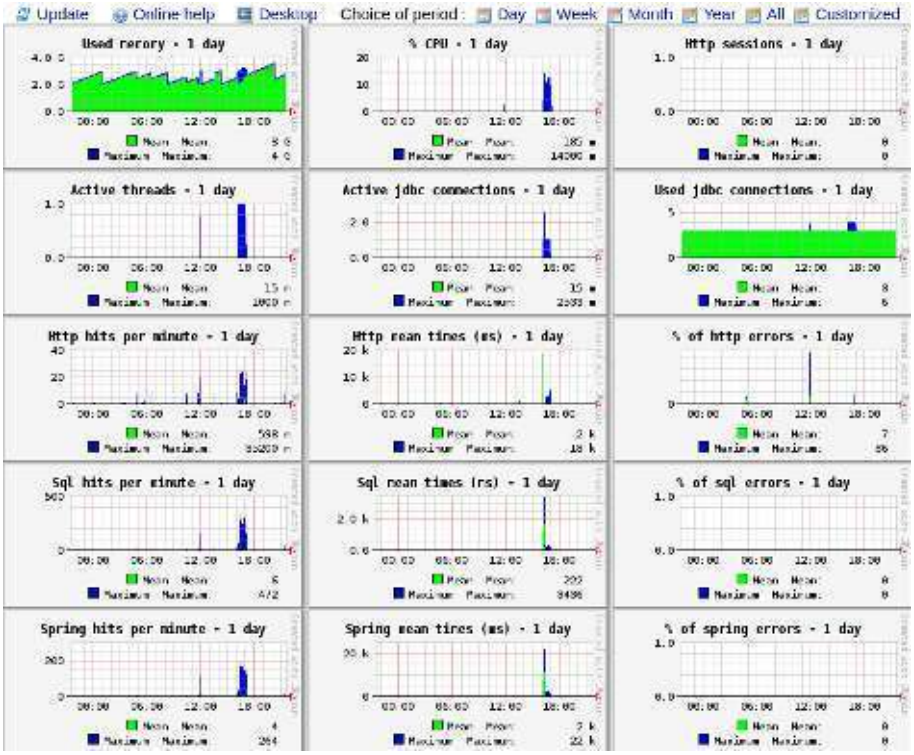
```
def "Guanine richness - below 40%"() {
    given:
    def sequenceString = "GGGGGGGAAAAAAAAAAAAA"; // 35 %
    when:
    def result = rloopr.checkGRichness(sequenceString, 0.4);
    then:
    result == false
}
```

Jednotlivé testy se spouští při každém odeslání do repozitáře `Gitlab`. Každá tato integrace je tak testována oproti stávajícím a přidaným testům. Mluvíme o *Continuous integration*, což je jeden ze základních principů udržitelnosti a funkčnosti velkých projektů, které mají více přispěvatelů.

Monitoring

V rámci diplomové práce jsem také do systému přidal nástroj `javamelody`. Jedná se o monitorování aplikace v reálném čase, které nabízí sledování využití zdrojů a monitoruje různé chybové události. Nástroj tak funguje částečně jako profiler.

Monitorovací dashboard je dostupný na adrese `https://bioinformatika.pef.mendelu.cz/monitoring`.



Obrázek 9: Ukázka monitoringu aplikace (pouze grafy). Vidíme, že paměť serveru operuje průměrně se třemi gigabajty a že mezi pátou a šestou hodinou bylo spuštěno velké množství analýz nad produkčním serverem. Monitorujeme také SQL dotazy a jejich odezvu, či přístupy do databáze H2.

5.3 Struktura modulu R-loop-R

Modul `R-loop-R` je součástí serveru podle architektonického vzoru *Model-view-controller* (MVC). `R-loop-R` tak můžeme strukturálně rozdělit na tři části:

1. **Model**

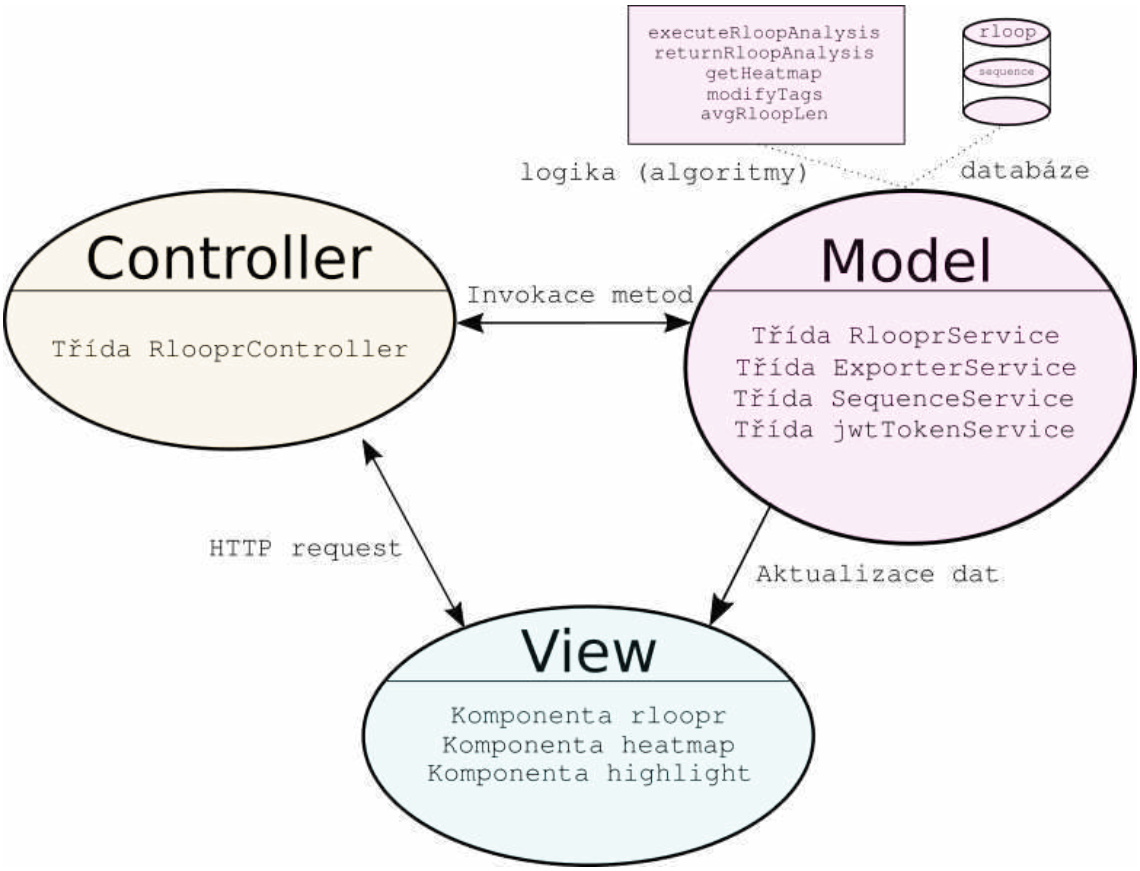
   Model se v aplikaci stará o dynamickou práci s daty (práce s databází, exporty do souborů, načítání dat ze souborů), logiku a pravidla, podle kterých se má aplikace chovat (algoritmická část). V modulu `R-loop-R` je model reprezentován třídami s příponou `Service`.

2. **View**

   Vrstva View se stará o zobrazování a reprezentaci dat uživateli. Může se jednat o výsledné tabulky, grafy, nebo heatmapu, kterou aplikace používá.

3. **Controller**

   Controller řídí tok aplikace a funguje jako prostředník mezi vrstvou Model a View. V modulu `R-loop-R` je reprezentován třídou `Controller`.



Obrázek 10: Využití MVC architektury pro modul R-loop-R.

Na MVC modelu vidíme strukturu modulu tak, jak je přítomna na webovém serveru. Samotný model pak obsahuje reference na úložiště dat a algoritmickou část. Řadič (Controller) je reprezentován jednou třídou, která zároveň definuje cílové URL adresy. Tyto adresy jsou pak dostupné z pohledu, nebo z API.

5.4 Vstupní data

Modul `R-loop-R` analyzuje sekvence nukleotidů, které je možné na server `DNA-Analyser` nahrát různými způsoby:

- **NCBI ID**

  NCBI (*National Center for Biotechnology Information*) je instituce, která mimo jiné spravuje databáze různých genomů. Každý genom je jednoznačně identifikován pomocí *NCBI ID*, které má následující podobu:

  `NC_X.Y`,

  kde čísla X a Y určují verzi sekvence podle přístupů. Číslo X je unikátní pro daný genom, číslo Y pak identifikuje revize a úpravy sekvence v databázi. NCBI ID identifikuje sekvenci v databázi `RefSeq`. Například pro lidský genom se může jednat o následující verze (*Sequence Identifiers*, 2014):

NCBI ID	Datum změny
NC_000001.11	3. února 2014, 23:01
NC_000001.10	13. srpna 2013, 00:15
NC_000001.9	3. března 2008 17:58

Tabulka 5: Verze lidského genomu v databázi NCBI, které jsou identifikovány číslem genomu a číslem přístupu (změna v sekvenci). Změny mohou být i minoritní, těm pak nezvyšujeme číslo změny, pouze zaneseme datum změny.

  Pro použití pak stačí nahrát sekvenci podle NCBI ID a můžeme se ní dále pracovat.

- **Soubor ve formátu txt (plain)**

  Dalším způsobem je nahrávání souborů. To nám může pomoci, pokud chceme analyzovat jen část sekvence, protože většina genomů je příliš dlouhá a budeme chtít analyzovat pouze určitý úsek. Formát je užitečný pro rychlé analýzy a testování. Jeho nevýhodou je ztráta informace o umístění sekvence v rámci genomu.

- **Soubor ve formátu FASTA**

  Formát `FASTA` (4.1) už zachovává informaci o pozici sekvence v genomu. Portál NCBI nabízí možnost exportu částí genomu ve formátu `FASTA`, který si následně můžeme upravit.

- **Vložení sekvence do formuláře**

  Poslední možností je rychlé ověření analýz krátkých sekvencí. Sekvenci vkládáme přímo do formuláře.

5.5 Výstupní data

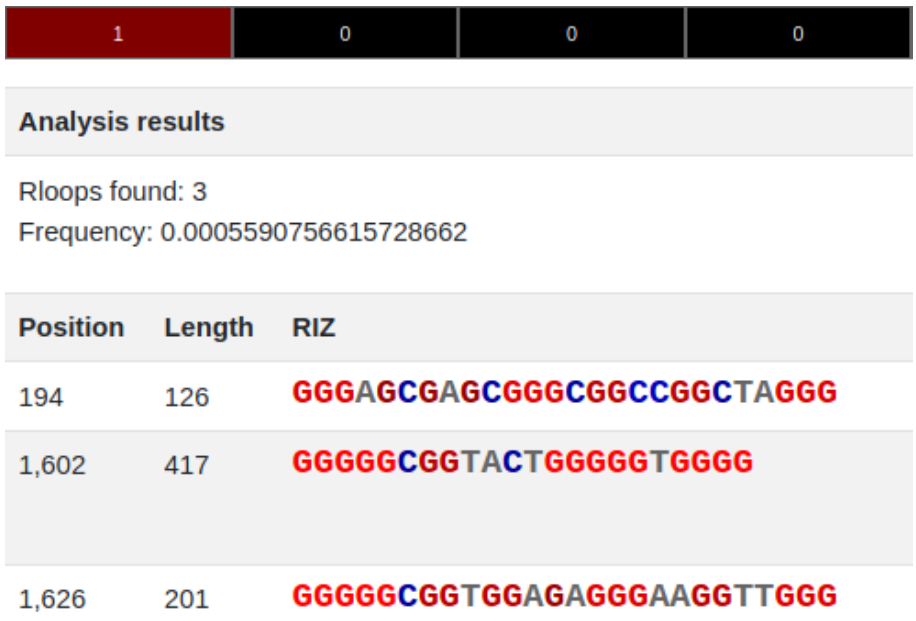
Výstupem analýzy je kolekce detekovaných R-loop. Každá R-loopa ve výstupní kolekci je reprezentována následujícím objektem:

```
public class RlooprResult implements Localised {

    protected int position; // počáteční pozice R-loopy v sekvenci
    protected int length; // celková délka R-loopy (RIZ + linker + REZ)
    protected String riz; // Nukleotidy, které jsou součástí RIZ
    protected double rizGRichness; // poměr guaninu v RIZ
    protected String linker; // Nukleotidy, které jsou součástí linkeru
    protected String rez; // Nukleotidy, které jsou součástí REZ
    protected double rloopGRichness; // poměr guaninu v celé R-loop sekvenci
    protected int g3cnt; // počet G3-clusterů
    protected int g4cnt; // počet G4-clusterů
    protected int gNcnt; // počet GN-clusterů, kde N > 4
    protected Long id; // ID analýzy, ke které se R-loopa vztahuje
    protected Short model; // použitý model pro detekci
    protected Character strand; // směr vlákna, ve kterém detekujeme R-loopy
}
```

Atributy týkající se počtu guaninových clusterů, nebo množství guaninu v R-loop, je možné vypnout pomocí přepínače. U velmi dlouhých sekvencí (více než 500 miliónů nukleotidů) by mohlo dojít ke zpomalení výpočtu.

Výsledky analýz je pak možné zobrazit na webu `https://bioinformatika.pef.mendelu.cz/rloopr-results`. K nahlédnutí přikládám část výsledků (na obrázku chybí zbytek heatmapy a sekvence linkeru a REZ, kvůli měřítku):



Obrázek 11: Zobrazení výsledků modulu R-loop-R. Zobrazená je jen iniciační zóna společně s daty o délce a pozici R-loopy.

Součástí modulu je pak napojení na službu `ExporterService`, kterou jsem v rámci práce vyčlenil a doplnil o další výstupní formát. Služba `ExporterService` se stará o převedení dat z databáze do souboru podle požadovaného formátu. Zatím služba podporuje dva výstupní formáty:

- Comma-separated-values (CSV)
- bedGraph

Formát CSV je obecně známý, proto jen zmíním, že použité sloupce jsou stejné, jako atributy třídy `RlooprResult` (konkrétně jeho reprezentace v databázi).

Formát bedGraph poskytuje jinou strukturu zobrazení výsledků. Data ve formátu bedGraph je možné vložit do nástroje *Genome browser* jako anotaci, a tím umožnit zobrazení struktury a lokalizace R-loop v rámci daného genomu. Strukturou také nabízí možnost překryvu jednotlivých analýz. Data uložená v bedGraph formátu jsou orientována podle řádků. Na začátku každého souboru je hlavička obsahující metadata - `track definition line`. Pokud chceme použít formát bedGraph pro nástroj *Genome Browser*, musí hlavičce předcházet informace pro browser.

Soubor můžeme rozdělit do čtyř částí:

- **Metadata pro Genome browser**

  ```
  browser position NM_001082575.2:2351-2631
  ```

  Pozice v browseru je identifikována číslem chromozomu, na kterém se vybraný gen nachází, počáteční pozicí a koncovou pozicí subsekvence, kterou chceme zobrazit.

- **Track definition line**

  Jedná se o definici pravidel, podle kterých se vykreslují naše anotace v Genome browseru. První parametr `type=bedGraph` je povinný, ostatní jsou volitelné.

  ```
  track type=bedGraph name="DNA analyser"
  description="BedGraph output of R-loop-R analysis"
  visibility=full color=200,100,0 altColor=0,100,200
  ```

- **Data**

  Samotná data jsou identifikována jejich pozicí v sekvenci, a pak následují naměřené hodnoty z analýzy. Co se týče identifikace analýzy, modul R-loop-R používá identifikaci podle NCBI ID. Pokud tento identifikátor není dostupný, použije se název sekvence.

  ```
  chromA  chromStartA  chromEndA  dataValueA
  chromB  chromStartB  chromEndB  dataValueB
  ```

  Modelový příklad výstupu analýzy ve formátu bedGraph je následující soubor:

  ```
  browser position NM_001082575.2:2351-2631
  track type=bedGraph name="DNA analyser"
  description="BedGraph output of R-loop-R analysis"
  visibility=full color=200,100,0 altColor=0,100,200
  NM_001082575.2 2351 2663  0.56 0.41 18.00 2.00 0.00
  NM_001082575.2 2444 2637  0.59 0.42 11.00 2.00 0.00
  NM_001082575.2 2464 2631  0.61 0.42 8.00 1.00 0.00
  ```

  V datech můžeme vidět, že modul našel tři R-loopy mezi nukleotidy 2351-2631 a žádná z nich neobsahovala G-clustery o větším počtu, než G4.

5.6 Implementační část

Tato část popisuje implementaci a modifikace algoritmu pro detekci R-loop (1) v jazyce Java. Hlavní část algoritmu je reprezentována metodou `getInitiationZones`, která přijímá dva parametry:

1. `short modelRIZ` - model detekce R-loop (mohou být zvoleny oba)
2. `Window window` - vybraná sekvence pro analýzu

Rozhraní `Window` je implementováno pomocí třídy `ByteBufferWindow`. Tato třída reprezentuje sekvenci nukleotidů jako stream bajtů. Například původní sekvence `GGGACGGTAGGGG` je v paměti reprezentována hexadecimálně jako `47474741434747544147474747`.

Datový typ string je v jazyce Java uložen do 16 bitů (záleží na kódování, ale výchozí kódování je `UTF-16`) (Oracle, 2020). Použitím datového typu `ByteBuffer` získáme dvakrát menší paměťovou náročnost.

R-loop initiation zone

Protože detekce RIZ je závislá na přítomnosti G-clusterů, mezi kterými může být až 10 různých nukleotidů, ideálním řešením pro detekci je použití *regulárních výrazů*.

Regulární výraz pro detekci R-loop podle modelu 1 (1)

```
(G{3,})([ACGT]{1,10}?)(G{3,})([ACGT]{1,10}?)(G{3,})
```

Regulární výraz pro detekci R-loop podle modelu 2 (2)

```
(G{4,})([ACGT]{1,10}?)(G{4,})
```

Pro detekci jsem musel použít kvantifikátor *non-greedy* regulárního výrazu. Pro skupiny nukleotidů mezi G-clustery je vhodné tento kvantifikátor použít, protože jinak regulární výraz hledá nejdelší možnou shodu, a tak bychom mohli zkrátit následující G-cluster. Uvádím příklad porovnání použití kvantifikátoru *non-greedy*:

Vstupní sekvence	Kvantifikátor	g1	g2	g3	g4	g5
GGGAGGGCTTGGGG	non-greedy	GGG	A	GGG	CTT	GGGG
GGGAGGGCTTGGGG	greedy	GGG	A	GGG	CTTG	GGG

Tabulka 6: Srovnání kvantifikátoru greedy a non-greedy. Při hladovém hledání najde regulární výraz menší poslední G-cluster.

Regulární výraz se před jeho použitím předkompiluje (`Pattern.compile()`), což také přispívá ke zlepšení výkonnosti algoritmu.

Použitím regulárního výrazu získáme iniciační zóny, tedy zóny s vysokou pravděpodobností vzniku R-loop. Struktura R-loop se vytvoří až po vzniku elongační zóny, kterou detekujeme v následujícím kroku. Práce s velkými sekvencemi by pro tento krok byla velmi náročná na paměť, proto je potřeba data určitým způsobem redukovat.

Z algoritmu, který je založený na laboratorních datech plyne, že maximální délka sekvence REZ je 2000 nukleotidů a k tomu může přibýt linker, který může mít až 50 nukleotidů. Původní sekvenci jsem proto rozdělil na malá okna, která začínají tam, kde končí RIZ a mají délku 2050 nukleotidů. Tím není potřeba pracovat s tak velkými sekvencemi.

Obrázek 12: Rozdělení původní sekvence na menší okna podle místa detekce RIZ. Rozdělení provádíme, protože práce s původní sekvencí by byla paměťově náročná.

R-loop elongation zone a linker

Další část algoritmu se zabývá hledáním elongační zóny v okně, které jsme získali při detekci RIZ. Každé okno (subsekvence) je analyzováno pomocí tohoto algoritmu:

Algorithm 2: Valid R-loop elongation zone search

```
Result: Detected R-loop
Data: RIZ, REZ Window
# gRichness() - returns guanine richness of the sequence;
if gRichness(RIZ) < 50% then
    set result to null;
else
    filter all possible REZ candidates and select the longest one;
    if found valid longest REZ, where gRichness(REZ) ≥ 40% then
        try to increment linker between RIZ and REZ;
        if gRichness(REZ + linker) ≥ 40% then
            set result to RIZ + linker + REZ;
        else
            set result to RIZ + REZ;
        end
    else
        set result to null;
    end
end
```

Úspěšný průchod algoritmem můžeme shrnout do čtyř kroků:

1. Ověření guaninové hustoty RIZ (alespoň 50%).
2. Nalezení nejdelší sekvence REZ, splňující hustotu guaninu pro REZ (alespoň 40%).
3. Prodloužení nejdelší sekvence REZ o linker, v případě, že zachová hustotu guaninu.
4. Analýza výsledné sekvence.

Pro části algoritmu, které hledají nejdelší sekvenci REZ a linker, jsem použil *paralelizované streamy*. Pro každý stream jsem předem připravil výchozí strukturu, na kterou aplikuji operaci filtrace a selekce. V zásadě jsem vytvořil datový typ `List`, do kterého jsem uložil původní okno rozdělené na další okna. Tyto okna mají počátek v nultém prvku původního REZ okna, ale zvětšuje se jejich délka. Pro každé původní okno tak existuje `List`, který obsahuje 2050 menších oken v následujícím rozsahu:

```
0 - 100
0 - 101
0 - 102
...
0 - 2049
0 - 2050
```

Může se zdát, že tento proces vyžaduje práci se zbytečně velkým množstvím operační paměti. Měřil jsem stav paměti před alokováním a po alokování objektu a výsledkem byla spotřeba paměti 4026528 B $\approx$ 4 MB pro výsledný `List` obsahující menší sekvence.

Hlavní výhoda předpřipravených struktur spočívá v aplikaci Java streamů. Při použití paralelního streamu nám sice výsledky mohou přijít v jiném pořadí, to ale nepředstavuje v této aplikaci problém a navíc dojde k extrémnímu zrychlení aplikace. Zrychlení popisuji v části (5.7).

Celková aplikace paralelního streamu vypadá následovně:

```
rezList.stream()
       .parallel()
       .filter(seq -> checkGRichness(seq, 0.4))
       .collect(Collectors.toList());
```

Postupně se tím provádějí následující operace:

1. Převedení datového typu `ArrayList` na `stream`.
2. Převedení streamu na paralelní stream → následující operace se provedou paralelně
3. Selekce takových oken (sekvencí), které obsahují minimálně 40 % guaninu.
4. Uložení těchto sekvencí do datového typu `List`.

Poté stačí ze sekvencí, které splňují podmínku, najít tu nejdelší. K tomu lze použít komparátor:

```
Collections.max(longestRezList, Comparator.comparing(String::length));
```

Stejné operace pak provádíme i při detekci linkeru, tam se ale jedná o menší počet dat, protože podle algoritmu je maximální délka linkeru stanovena na 50 nukleotidů.

Analýza R-loop

Pokud při detekci nalezneme alespoň jednu R-loopu, můžeme ji částečně zanalyzovat (statisticky). U detekovaných R-loop nás zajímají hlavně tyto hodnoty:

- *Obsah guaninu v RIZ* [%]
- *Obsah guaninu v celé R-loop* [%]
- *Počet G_3-clusterů*
- *Počet G_4-clusterů*
- *Počet G_N-clusterů, kde $N \geq 4$*

Tyto hodnoty jsou dostupné pří exportu analýzy do formátu `csv`, nebo `bedGraph`. Z hlediska přehlednosti webového rozhraní (kde tyto hodnoty nejsou), by bylo dobré zvážit možnost agregace těchto statistik do jedné hodnoty. Toto číslo by ale nemělo z biologického hlediska žádný význam, sloužilo by pouze jako charakteristika dané R-loopy. Proto jsem nakonec hodnoty neagregoval.

5.7 Benchmarking modulu

Na implementaci modulu jsem začal pracovat v době, kdy webový server `DNA-Analyser` používal verzi jazyka `Java < 8`. Původní implementace tak nepodporovala streamy a hledání R-loop elongation zone byl výpočetně náročný proces se složitostí $O(n^2)$ pro každé okno se sekvencemi v R-loop.

V této části porovnávám původní implementaci bez streamů a zrychlení po použití streamů z jazyka `Java verze 8`. Následně srovnávám zrychlení po použití *paralelismu*. Pro každou z těchto implementací jsem srovnal běh aplikace pro vstupní sekvenci, které řádově narůstala délka.

Pro benchmarking jsem použil webový server spuštěný na mém osobním notebooku. Specifikace testovacího notebooku je následující:

Operační systém	`Ubuntu 18.04 LTS`
CPU	`Intel® Core i5-8350U CPU @ 1.70GHz × 8`
RAM	`15,5 GiB`

Tabulka 7: Specifikace prostředí pro benchmark

Dobu běhu algoritmu jsem měřil pomocí modulů `Instant` a `Duration` z jazyka Java verze 8. Výsledek měření v milisekundách jsem postupně zapisoval do souboru, ze kterého vznikla výsledná tabulka.

```
Instant start = Instant.now();
/**
 * R-loop-R algorithm
**/
Instant end = Instant.now();
long timeBetween = Duration.between(start, end).toMillis();

try {
  // appends to a file
  FileWriter writer = new FileWriter("parallel_stream.bench", true);
  writer.write(resultCount + "\t" + timeBetween + "\n");
  myWriter.close();
} catch (IOException e) {
  e.printStackTrace();
}
```

Tabulka popisuje jednotlivé způsoby implementace a její dopad na dobu běhu analýzy. Účelem této tabulky bylo srovnání využití popisované implementace, z toho hlavně *paralelních streamů*. Jako referenční sekvenci jsem vybral lidský chromozom číslo 1[5], který je součástí genomu *GRCh38.p13*. Původní délka chromozomu je 248 956 422 nukleových bází, proto jsem data zkrátil do rozsahu 5000 - 10 000 000 nukleových bází.

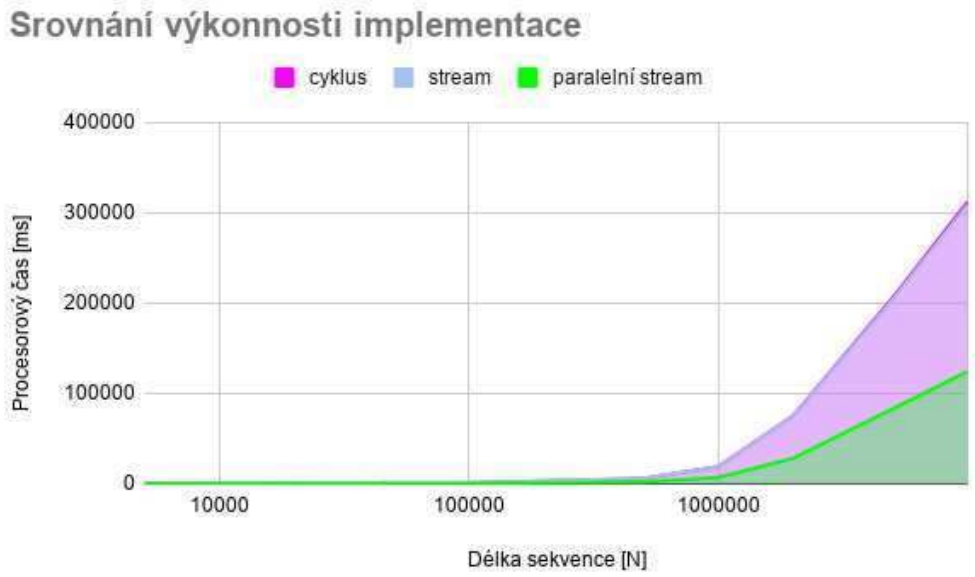
[5] `https://www.ncbi.nlm.nih.gov/nuccore/NC_000001.11`

	Délka sekvence	*Počet R-loop*	*Procesorový čas* [ms]
Paralelní streamy	5000	1	49
	10 000	4	134
	50 000	7	429
	100 000	7	475
	500 000	38	2229
	1 000 000	193	7050
	2 000 000	974	28369
	5 000 000	2130	82886
	10 000 000	2888	124279
Streamy bez paralelismu	5000	1	115
	10 000	4	342
	50 000	7	1145
	100 000	7	1276
	500 000	38	6150
	1 000 000	193	18881
	2 000 000	974	74499
	5 000 000	2130	203351
	10 000 000	2888	308461
Bez použití streamů a paralelismu (cyklus)	5000	1	123
	10 000	4	416
	50 000	7	1133
	100 000	7	1287
	500 000	38	5929
	1 000 000	193	19094
	2 000 000	974	75202
	5 000 000	2130	205427
	10 000 000	2888	311774

Tabulka 8: Srovnání postupné implementace algoritmu. Předpřipravením struktur pro paralelismus jsem dosáhl značného zrychlení oproti neparalelním variantám. Výsledky mohou být částečně zkreslené procesy běžícími na pozadí notebooku.

V tabulce vidíme, že čistě použitím streamů tolik čas neušetříme (výsledky se liší v řádu sekund). Pokud ale používáme paralelní streamy, dojde k masivnímu zrychlení analýzy.

Graf zobrazuje stejná data, jako jsou v tabulce pro zdůraznění časových rozdílů. Použil jsem logaritmické škálování, na kterém vidíme, že po překročení analýzy délky 1 000 000 dojde k zhoršení výkonu algoritmu.



Obrázek 13: Grafická reprezentace předchozí tabulky. Je použita logaritmická škála.

5.8 Příklady použití nástroje R-loop-R

V této části uvádím několik případů použití nástroje. Protože jsem modul `R-loop-R` integroval do RESTového API, je možné jej provolávat pomocí endpointů.

API

Vystavení endpointů definovaných třídou `Controller` pro `R-loop-R` již bylo integrováno ve webovém serveru `DNA-Analyser`. V rámci práce jsem tedy jen připsal cestu k jednotlivým metodám. Metody jsem vybral všechny, aby bylo zachováno rozhraní 1:1 s webovým serverem. S API je možné komunikovat pomocí unixového nástroje `curl`, z hlediska znovupoužitelnosti mi ale dávalo smysl vytvořit si sadu skriptů v jazyce `Python 3.6`, se kterými bych mohl do budoucna nadále pracovat.

Podle účelu dané metody mají jednotlivé endpointy přiřazené následující HTTP metody:

- `GET`
  - `/api/analyse/rloopr`
    Vrátí všechny provedené analýzy.
  - `/api/analyse/rloopr/{id}/analysis`
    Vrátí jednu analýzu podle parametru `id`.

  - `/api/analyse/rloopr/{id}/rloops`
    Vrátí všechny R-loopy pro analýzu podle parametru `id`.
  - `/api/analyse/rloopr/{id}/rloopr.csv`
    Vrátí informace o analýze ve formátu `CSV` podle parametru `id`.
  - `/api/analyse/rloopr/{id}/rloopr.bedgraph`
    Vrátí informace o analýze ve formátu `bedGraph` podle parametru `id`.
  - `/api/analyse/rloopr/{id}/average/length`
    Vrátí průměr délky všech R-loop v analýze podle parametru `id`.
  - `/api/analyse/rloopr/{id}/heatmap`
    Vrátí data analýzy pro heatmapu podle parametru `id`.
  - `/api/analyse/rloopr/{id}/tag`
    Vrátí všechny tagy analýzy podle parametru `id`.
- **`POST`**
  - `/api/analyse/rloopr`
    Provede detekci a analýzu R-loop. Je potřeba přiřadit metodě parametry `model`, `sequence` a `tags`.
- **`PUT`**
  - `/api/analyse/rloopr/{id}/tags`
    Přiřadí nebo odebere tagy u zvolené analýzy podle parametru `id`. Nové tagy je potřeba předat v parametru `tags`.
- **`DELETE`**
  - `/api/analyse/rloopr/{id}`
    Smaže analýzu podle parametru `id`.

Použití API je výhodné hlavně pro opakované zpracování a agregaci dat. Uvádím zde část skriptu, který jsem použil pro zhodnocení výsledků. Celý skript je značně rozsáhlý i pro přílohu, protože následně generuje informace o rozložení R-loop v sekvenci. Jedná se o repetitivní činnost, pro kterou je vhodné použití API. Skript je psán v jazyce `Python` a používá modul `waiting`, který umožňuje exponenciální čekání na událost. Je to výborná forma aktivního čekání na dokončení analýzy. Část skriptu se skládá z těchto kroků:

1. Analýza sekvence podle ID.
2. Aktivní čekání, dokud analýza úspěšně neskončí.
3. Získání informace o analýze (extrakce tokenu pro stažení dat).

```
# Initialize the analysis
request = requests.post(
    host + '/api/analyse/rloopr',
    data=json.dumps(rloopr_params),
    headers=headers
)
analysis_id = request.json()["payload"]["id"]

try: # waiting for the analysis batch to finish
  predicate = lambda : wait_for_analysis(analysis_id)
  wait(predicate, sleep_seconds=(1, 100))
except TimeoutExpired:
  print("Timeout for sequence upload exceeded.")
  exit(1)

# Download finished analysis
finished = requests.get(
    host+"/api/analyse/rloopr/" +
    analysis_id + "/analysis",
    headers=headers
)
```

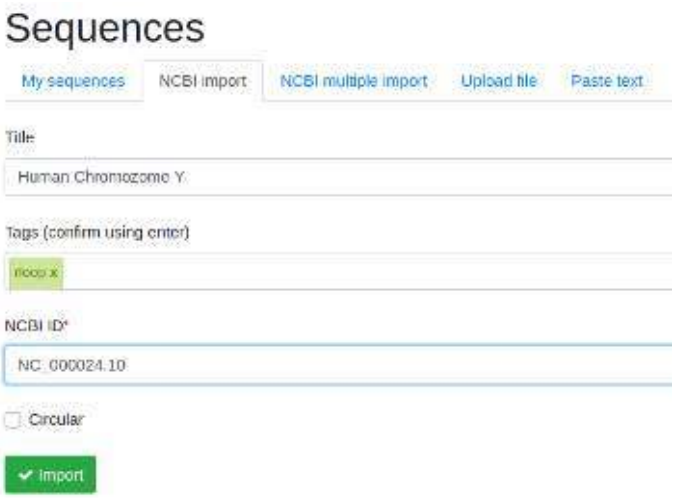
Webový server

Modul `R-loop-R` plně integrován do webového serveru `DNA-Analyser` jako nová analýza. Pro úplnost popisu práce s modulem přikládám jednotlivé kroky pro analýzu sekvencí a hledání R-loop.

Jako příklad analýzy sekvence jsem si vybral lidský chromozom Y, který má `NCBI ID NC_000024.10`.

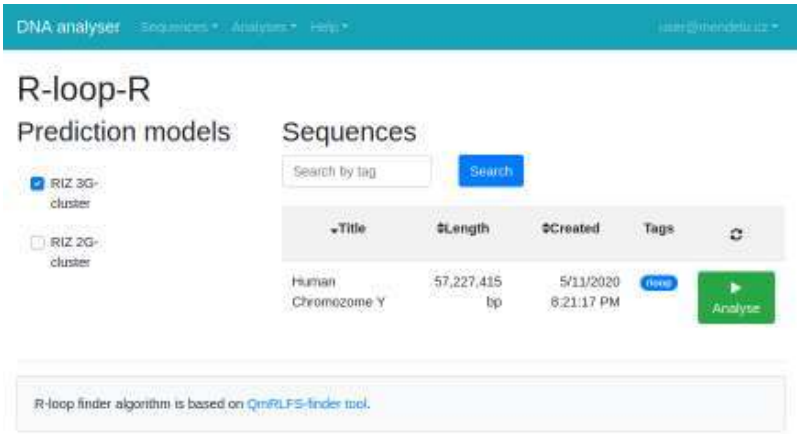
Webový server je dostupný na adrese `https://bioinformatika.pef.mendelu.cz`. Pro spuštění v lokálním prostředí je možné postupovat podle návodu v příloze A.

1. Vložení sekvence, kterou chceme analyzovat[6]



Obrázek 14: Vložení sekvence pomocí formuláře. V našem případě používám import pomocí `NCBI ID`.

2. Spuštění detekce a analýzy R-loop [7]
   Analýzu lze spustit s detekcí dvou modelů. Model `RIZ 3G-cluster` detekuje v oblasti RIZ skupinu **tří** G-clusterů, které obsahují alespoň tři guaninové báze. Druhý model, `RIZ 2G-cluster` detekuje skupinu **dvou** G-clusterů, které obsahují alespoň čtyři guaninové báze. Je možné zvolit oba modely pro analýzu.



Obrázek 15: Spuštění analýzy nahrané sekvence se zvoleným modelem `RIZ 3G-cluster`.

[6] `https://bioinformatika.pef.mendelu.cz/#/sequence/tools`
[7] `https://bioinformatika.pef.mendelu.cz/#/analyse/rloopr`

3. Zobrazení výsledné analýzy
   Po dokončení batch jobu se uživateli zobrazí výsledky. Je možné získat detailnější informace o analýze pomocí tlačítek pro export (`bedGraph`, `csv`). Ve webovém rozhraní je zobrazena výsledná sekvence, která reprezentuje R-loop a její umístění v rámci původní analyzované sekvence.

   Nad daty vidíme i heatmapu, podle které dokážeme určit, kde se R-loopy v sekvenci nacházejí. Z obrázku vidíme, že zrovna pro lidský chromozom 19 se většina R-loop nachází v první čtvrtině chromozomu.

Obrázek 16: Výsledek analýzy. V lidském chromozomu číslo 19 se nachází celkem 1422 R-loop. guaninové clustery jsou vyznačeny rudou barvou.

6 Zhodnocení výsledků

Zhodnocení výsledků jsem se rozhodl rozdělit na dvě části. Ta první se věnuje výstupům nástroje jako takového. Druhá se věnuje porovnání nástroje `R-loop-R` a nástroje `Qm-RLFS finder` (4). Vybral jsem pokusné sekvence, na kterých demonstruji rozložení R-loop napříč sekvencí nukleotidů.

6.1 Výstupy nástroje

Pro analýzu jsem vybral následující sekvence nukleotidů:

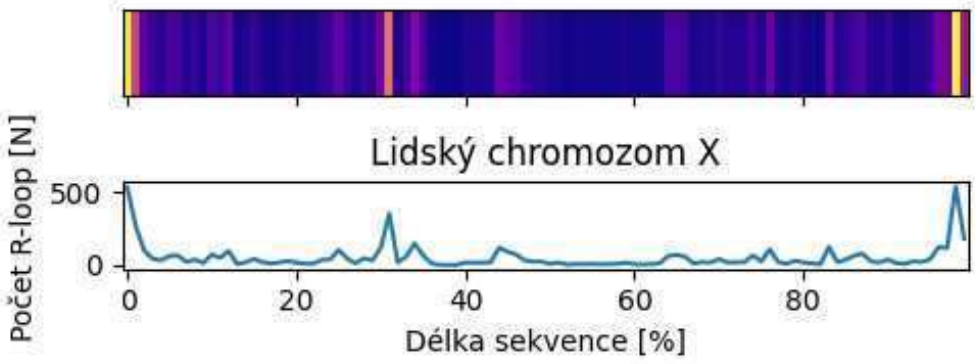
- Lidský chromozom X [`NC_000023.11`]
- Lidský chromozom Y [`NC_000024.10`]
- Lidský chromozom 11 [`NC_000011.10`]
- Octomilka (chromozom X) [`NC_004354.4`]
- Kvasinky (chromozom 11) [`NC_001143.9`]
- Pes domácí (chromozom 1) [`NC_006583.3`]

Cílem zhodnocení byla demonstrace výkonnosti nástroje u dlouhých sekvencí nukleotidů a zhodnocení výstupů nástroje.

Pro analýzu sekvence jsem využil i API, ke kterému jsem se připojoval pomocí `Python` skriptu. Tento skript nahrál sekvenci, nad kterou následně spustil analýzu. Jakmile se analýza dokončila, skript získal statistická data o R-loopách a vykreslil graf rozložení R-loop v chromozomu.

Lidský chromozom X

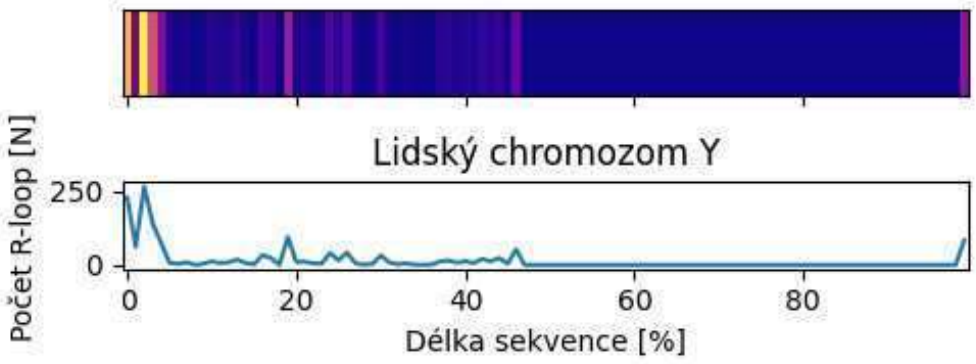
Délka sekvence [bp]			**Počet R-loop** [N]		
156 040 895			5231		
Podíl guaninu v RIZ [%]			**Podíl guaninu v R-loop** [%]		
avg	*min*	*max*	*avg*	*min*	*max*
0.69	0.5	1.0	0.42	0.39	0.63
3G-clustery [N]			**4G-clustery** [N]		
sum	*min*	*max*	*sum*	*min*	*max*
158 693	2	223	63 615	0	165



Obrázek 17: Chromozom X z lidského genomu. Jeden z pohlavních chromozomů, který mají ženy i muži. R-loopy se soustředí na začátku sekvence, na jejím konci a pak asi ve 30 % délky. 1% délky sekvence zde odpovídá 1 560 409 nukleovým bázím.

Lidský chromozom Y

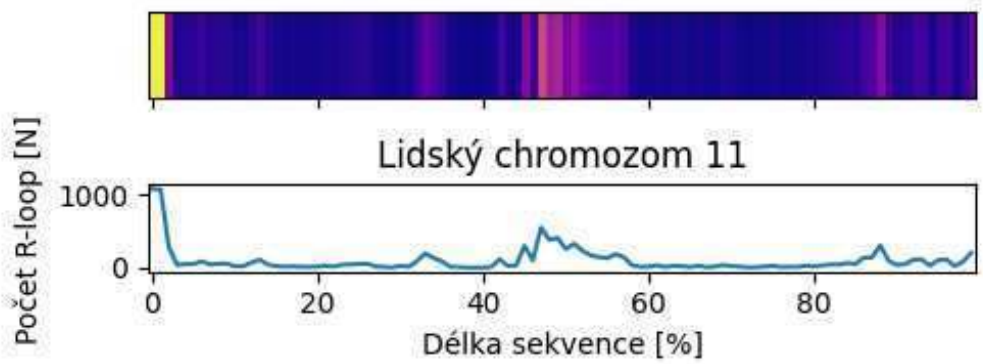
Délka sekvence [bp]			**Počet R-loop** [N]		
57 217 458			1422		
Podíl guaninu v RIZ [%]			**Podíl guaninu v R-loop** [%]		
avg	*min*	*max*	*avg*	*min*	*max*
0.69	0.5	1.0	0.43	0.39	0.63
3G-clustery [N]			**4G-clustery** [N]		
sum	*min*	*max*	*sum*	*min*	*max*
82 572	1	240	24 552	0	165



Obrázek 18: Chromozom Y z lidského genomu. Další z pohlavních chromozomů, který mají jen muži (dvojice chromozomů XY). R-loopy se zde soustředí převážně na začátku sekvence. Zhruba v 50 % délky sekvence jejich výskyt téměř ustane a objeví se až ke konci. 1% délky sekvence zde odpovídá 572 174 nukleovým bázím.

Lidský chromozom 11

Délka sekvence [bp]			**Počet R-loop** [N]		
135 006 516			9454		
Podíl guaninu v RIZ [%]			**Podíl guaninu v R-loop** [%]		
avg	*min*	*max*	*avg*	*min*	*max*
0.67	0.5	1.0	0.42	0.38	0.55
3G-clustery [N]			**4G-clustery** [N]		
sum	*min*	*max*	*sum*	*min*	*max*
231 636	2	161	91 825	0	97



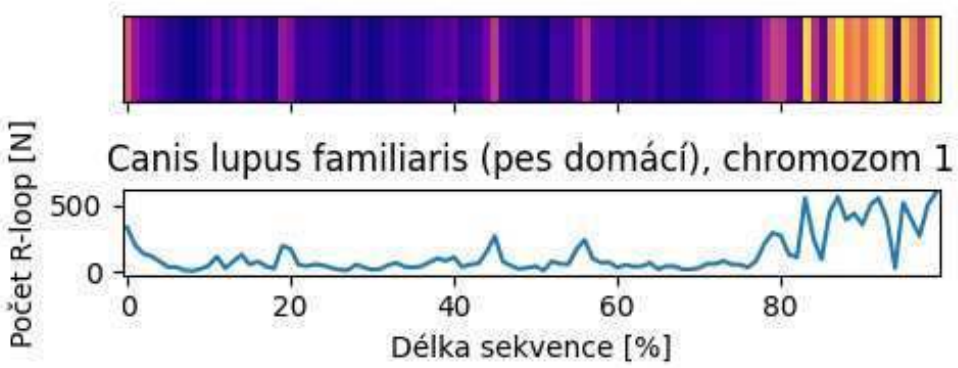
Obrázek 19: Chromozom číslo 11 z lidského genomu. V tomto chromozomu je anotováno spousta proteinů spojených s onkologickými onemocněními (například BRCA2), nebo s výskytem leukémie. Zde je opět nejvíce R-loop na začátku chromozomu. V polovině ale oproti poklesu vidíme poměrně razantní nárust výskytu těchto struktur. 1% délky sekvence zde odpovídá 1 350 065 nukleovým bázím.

Každý z lidských chromozomů, na které jsem se zaměřil má trochu jiné rozložení R-loop. U všech je stejné to, že koncentrace jejich výskytu je vysoká na začátku celého chromozomu.

Jako kontrast jsem analyzoval i některé typické vzorky ze zvířecí říše. Další sekvencí je genom psa domácího, konkrétně zmapovaného plemena boxer.

Pes domácí (Canis lupus familiaris), chromozom 1

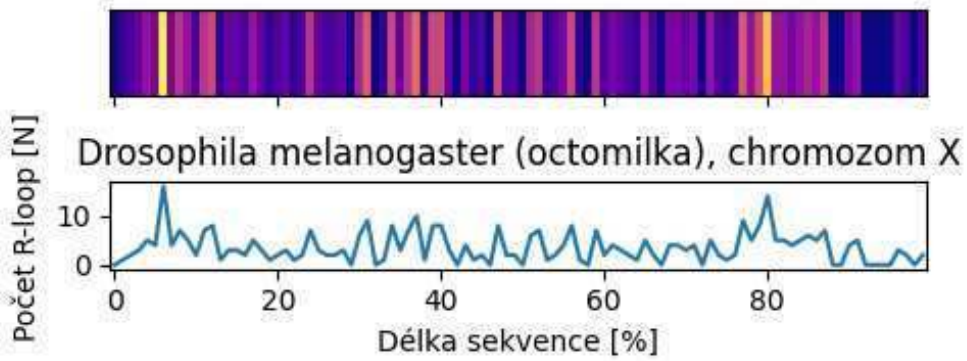
Délka sekvence [bp]			**Počet R-loop** [N]		
122 678 785			13 300		
Podíl guaninu v RIZ [%]			**Podíl guaninu v R-loop** [%]		
avg	*min*	*max*	*avg*	*min*	*max*
0.67	0.5	1.0	0.42	0.38	0.51
3G-clustery [N]			**4G-clustery** [N]		
sum	*min*	*max*	*sum*	*min*	*max*
309 315	1	148	135 568	0	68



Obrázek 20: Chromozom číslo 1 z genomu psa domácího. Od 80 % délky sekvence se objevují velké počty R-loop. 1% délky sekvence zde odpovídá 1 226 788 nukleovým bázím.

Octomilka obecná (Drosophila melanogaster), chromozom X

Délka sekvence [bp]			**Počet R-loop** [N]		
23 542 271			359		
Podíl guaninu v RIZ [%]			**Podíl guaninu v R-loop** [%]		
avg	*min*	*max*	*avg*	*min*	*max*
0.67	0.5	1.0	0.43	0.40	0.49
3G-clustery [N]			**4G-clustery** [N]		
sum	*min*	*max*	*sum*	*min*	*max*
3238	3	59	1260	0	12

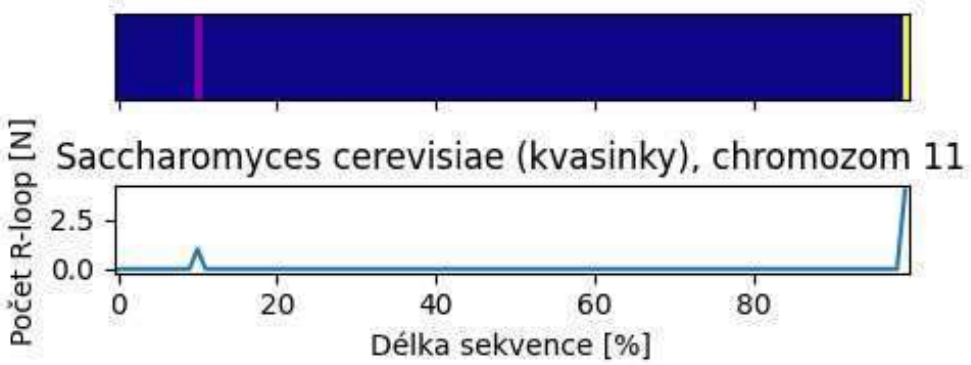


Obrázek 21: Chromozom X z genomu octomilky obecné. Můžeme pozorovat téměř palindromatický výskyt R-loop při průchodu celým chromozomem. 1% délky sekvence zde odpovídá 235 423 nukleovým bázím.

Pro poslední vizualizaci jsme si vybral chromozom kvasinek, na kterém jde vidět, že ne vždy se v chromozomu musí vyskytovat R-loopy v poměru k celkové délce sekvence.

Kvasinky (Saccharomyces cerevisiae), chromozom 11

Délka sekvence [bp]			**Počet R-loop** [N]		
666 816			5		
Podíl guaninu v RIZ [%]			**Podíl guaninu v R-loop** [%]		
avg	*min*	*max*	*avg*	*min*	*max*
0.65	0.56	0.7	0.57	0.41	0.61
3G-clustery [N]			**4G-clustery** [N]		
sum	*min*	*max*	*sum*	*min*	*max*
68	6	20	4	0	4



Obrázek 22: Chromozom číslo 11 z genomu kvasinek (pivní, pekařské, vinné). Frekvence R-loop je velmi nízká. 1% délky sekvence zde odpovídá 6668 nukleovým bázím.

U kvasinek jsem zkoušel takto analyzovat i některé další chromozomy dané kvasinky, získal jsem ale prakticky stejné výsledky.

6.2 Porovnání s nástrojem QmRLFS-finder

Ke srovnání jsem použil nástroj `QmRLFS-finder`, jehož vlastnosti popisuji v kapitole (4). Algoritmus obou nástrojů je stejný, založený na matematických modelech, které vychází z experimentálních dat naměřených v laboratoři (1, 2). Porovnávám tak převážně použitelnost nástroje a implementaci.

Výhody nástroje R-loop-R

1. *Udržitelnost software*
   Tím, že modul vzniká podle specifických požadavků Akademie věd České republiky a je součástí webového serveru, který je aktivně používán i pro jiné analýzy, je možné modul udržovat stále aktuální. Navíc pomocí rychlé zpětné vazby je zde možnost úpravy a optimalizace analýz.

2. *Spojení s API*
   Díky napojení modulu na API máme možnost psát automatické skripty pro analýzu. Skripty je možné psát i s verzí `QmRLFS-finderu` pro příkazovou řádku, není ale možné pohodlně pracovat s formátem výstupu. S API se nám nabízí mnohem vyšší versatilita skriptů, se kterými můžeme pracovat (volba různých výstupů, spojení analýz a podobně). Můžeme tak získat konkrétní data z analýzy a ty pak vizualizovat tak, jako v předchozí části.

3. *Výkonnost modulu*
   `R-loop-R` je na vstupu schopen přijmout mnohem delší sekvence. Řádově je nástroj testovaný pro stamilionové sekvence, ale je možné pracovat i s delšími sekvencemi nukleotidů podle potřeby. Díky přítomnosti batch modulu je možné spustit více analýz současně.

Výhody nástroje QmRLFS-finder

1. *Databáze R-loop*
   Součástí nástroje je i databáze R-loop. S touto databází pracují molekulární biologové z výzkumné skupiny, která nástroj vyvíjela, proto mohou případná data experimentálně ověřit v laboratoři.

2. *Anotace a vizualizace*
   Další výhodou je integrace nástroje s Genome browserem. Uživatel má k dispozici výstupní formát, který je možné vložit do browseru jako anotaci genomu a zaznamenat tak zajímavá místa nebo analýzy.

3. *Paralelní vlákno*
   `Qm-RLFS finder` hledá R-loopy nejen ve směru 5'→3', ale dopočítá i opačný směr.

	R-loop-R	**Qm-RLFS finder**
Vstup	fasta plain text NCBI ID	fasta
Maximální velikost vstupní sekvence	2 GB $\sim$ 2 000 000 000 nukleotidů	300 kB $\sim$ 300 000 nukleotidů
Výstup	web csv bedGraph	table bed fasta customTrack
API	ano	ne
Současné spuštění analýz	ano	ne
Paralelní vlákno	ne	ano
Databáze výsledků	ne	ano

Tabulka 9: Srovnání dvou dostupných nástrojů pro detekci a analýzu R-loop. Tabulka sumarizuje výhody a nevýhody popsané na předchozí straně.

7 Závěr

V rámci této diplomové práce jsem vytvořil modul `R-loop-R`, který je součástí webového serveru `DNA-Analyser`[8]. Modul slouží jako interdisciplinární prostředek pro detekci a analýzu lokálních struktur DNA, konkrétně struktury R-loop.

Lokální struktura R-loop je v posledních letech předmětem mnoha vědeckých výzkumů v oblasti molekulární biologie. Ukazuje se, že výskyt R-loop může mít vztah k rozvoji neurodegenerativních onemocnění, rakovinného bujení či celé řady dalších onemocnění. Zároveň práce popisuje procesy probíhající v nukleových kyselinách (RNA, DNA), které vedou ke vzniku struktur R-loop. Popis biologické části koresponduje s implementační částí, aby byl srozumitelnější kontext implementace analýzy.

Tento nástroj pro detekci struktur R-loop ve zvoleném genomu bude sloužit potřebám Akademie věd České republiky. Oproti jedinému stávajícímu nástroji umožňuje analýzu dlouhých sekvencí (celých chromozomů) a díky implementaci paralelizace jsou výsledky analýzy rychle dostupné v mnoha výstupních formátech (`csv`, `bedGraph`, nebo webové rozhraní). Webové rozhraní zpřístupňuje uživateli základní vizualizaci výsledků, export do souboru pak poskytuje detailní informace o analýze.

Pomocí skriptů je možné také používat API, pokud je nutné si modifikovat výsledky pro vlastní potřebu nebo agregovat data. Modul `R-loop-R` je možné nadále rozšířit pro optimální vyhledávání R-loop v sekvencích nukleotidů.

Během práce na modulu jsem využil značné množství technologií. Od těch front-endových (`vue`), přes back-end s databázemi (`Java`), až po psaní podpůrných skriptů (`Python`) a testů (`Groovy`). Zároveň jsem měl možnost pracovat s velkým množstvím dat, což podnítilo optimalizaci algoritmu. Na konci vznikl funkční modul, který je schopen analyzovat dlouhé sekvence nukleotidů.

[8] `https://bioinformatika.pef.mendelu.cz/`

8 Literatura

Odkazy

BHATIA, Vaibhav; BARROSO, Sonia I.; GARCÍA-RUBIO, María L.; TUMINI, Emanuela; HERRERA-MOYANO, Emilia; AGUILERA, Andrés, 2014. BRCA2 prevents R-loop accumulation and associates with TREX-2 mRNA export factor PCID2. *Nature*. Roč. 511, č. 7509, s. 362–365. Dostupné z DOI: 10.1038/nature13374.

BOROS-OLÁH, Beáta; DOBOS, Nikoletta; HORNYÁK, Lilla; SZABÓ, Zoltán; KARÁNYI, Zsolt; HALMOS, Gábor; ROSZIK, Jason; SZÉKVÖLGYI, Lóránt, 2019. Drugging the R-loop interactome: RNA-DNA hybrid binding proteins as targets for cancer therapy. *DNA Repair*. Roč. 84, s. 102642. Dostupné z DOI: 10.1016/j.dnarep.2019.102642.

BRÁZDA, Václav; KOLOMAZNÍK, Jan; LÝSEK, Jiří; BARTAS, Martin; FOJTA, Miroslav; ŠŤASTNÝ, Jiří; MERGNY, Jean-Louis, 2019. G4Hunter web application: a web server for G-quadruplex prediction. *Bioinformatics*. Roč. 35, č. 18, s. 3493–3495. Dostupné z DOI: 10.1093/bioinformatics/btz087.

BRÁZDA, Václav; KOLOMAZNÍK, Jan; LÝSEK, Jiří; HÁRONÍKOVÁ, Lucia; COUFAL, Jan; ŠT'ASTNÝ, Jiří, 2016. Palindrome analyser – A new web-based server for predicting and evaluating inverted repeats in nucleotide sequences. *Biochemical and Biophysical Research Communications*. Roč. 478, č. 4, s. 1739–1745. Dostupné z DOI: 10.1016/j.bbrc.2016.09.015.

BROWN, T. A., 2012. *Introduction to genetics : a molecular approach*. New York: Garland Science. ISBN 978-0-8153-6509-9.

COSTANTINO, Lorenzo; KOSHLAND, Douglas, 2015. The Yin and Yang of R-loop biology. *Current Opinion in Cell Biology*. Roč. 34, s. 39–45. Dostupné z DOI: 10.1016/j.ceb.2015.04.008.

GILBERT, Walter, 1978. Why genes in pieces? *Nature*. Roč. 271, č. 5645, s. 501–501. Dostupné z DOI: 10.1038/271501a0.

GIUDICELLI, V.; LEFRANC, M.-P., 1999. Ontology for immunogenetics: the IMGT-ONTOLOGY. *Bioinformatics*. Roč. 15, č. 12, s. 1047–1054. Dostupné z DOI: 10.1093/bioinformatics/15.12.1047.

JENJAROENPUN, Piroon; WONGSURAWAT, Thidathip; YENAMANDRA, Surya Pavan; KUZNETSOV, Vladimir A., 2015. QmRLFS-finder: a model, web server and stand-alone tool for prediction and analysis of R-loop forming sequences. *Nucleic Acids Research*. Roč. 43, č. W1, s. W527–W534. Dostupné z DOI: 10.1093/nar/gkv344.

JO, Bong-Seok; CHOI, Sun Shim, 2015. Introns: The Functional Benefits of Introns in Genomes. *Genomics & Informatics*. Roč. 13, č. 4, s. 112. Dostupné z DOI: 10.5808/gi.2015.13.4.112.

KARAMI, Ali, 2013. Largest and Smallest Genome in the World.

LODISH, Harvey F. (ed.), 2008. *Molecular cell biology*. 6th ed. New York: W.H. Freeman. ISBN 9780716743668 9780716776017. OCLC: ocm83758878.

NCBI, 2001. *Basic Local Alignment Search Tool*. U.S. National Library of Medicine. Dostupné také z: https://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=BlastHelp.

ORACLE, 2020. Dostupné také z: https://docs.oracle.com/javase/8/docs/api/java/lang/String.html.

PRAY, Leslie A., 2013. Discovery of DNA Structure and Function: Watson and Crick. In:

PYTHON, 2019. *Sunsetting Python 2*. Dostupné také z: https://www.python.org/doc/sunset-python-2/.

RICHARD, Patricia; MANLEY, James L., 2017. R Loops and Links to Human Disease. *Journal of Molecular Biology*. Roč. 429, č. 21, s. 3168–3180. Dostupné z DOI: 10.1016/j.jmb.2016.08.031.

ROBERTS, Richard J., 2020. *Nucleic acid*. Encyclopædia Britannica, inc. Dostupné také z: https://www.britannica.com/science/nucleic-acid.

ROY, Deepankar; LIEBER, Michael R., 2009. G Clustering Is Important for the Initiation of Transcription-Induced R-Loops In Vitro, whereas High G Density without Clustering Is Sufficient Thereafter. *Molecular and Cellular Biology*. Roč. 29, č. 11, s. 3124–3133. Dostupné z DOI: 10.1128/mcb.00139-09.

SAKHARKAR, M. K.; CHOW, V. T.; KANGUEANE, P., 2004. Distributions of exons and introns in the human genome. *In Silico Biol. (Gedrukt)*. Roč. 4, č. 4, s. 387–393.

SANZ, Lionel A.; HARTONO, Stella R.; LIM, Yoong Wearn; STEYAERT, Sandra; RAJPURKAR, Aparna; GINNO, Paul A.; XU, Xiaoqin; CHÉDIN, Frédéric, 2016. Prevalent, Dynamic, and Conserved R-Loop Structures Associate with Specific Epigenomic Signatures in Mammals. *Molecular Cell*. Roč. 63, č. 1, s. 167–178. Dostupné z DOI: 10.1016/j.molcel.2016.05.032.

2014. *Sequence Identifiers*. U.S. National Library of Medicine. Dostupné také z: https://www.ncbi.nlm.nih.gov/genbank/sequenceids.

SPONK, 2014. *Rozdíly mezi DNA a RNA*. Dostupné také z: ttps://commons.wikimedia.org/w/index.php?curid=32081792 CC BY-SA 3.0.

STEIN, Lincoln, 2001. Genome annotation: from sequence to biology. *Nature Reviews Genetics*. Roč. 2, č. 7, s. 493–503. Dostupné z DOI: 10.1038/35080529.

WHITE, Raymond L.; HOGNESS, David S., 1977a. R loop mapping of the 18S and 28S sequences in the long and short repeating units of drosophila melanogaster rDNA. *Cell*. Roč. 10, č. 2, s. 177–192. Dostupné z DOI: 10.1016/0092-8674(77)90213-6.

WHITE, Raymond L.; HOGNESS, David S., 1977b. R loop mapping of the 18S and 28S sequences in the long and short repeating units of drosophila melanogaster rDNA. *Cell*. Roč. 10, č. 2, s. 177–192. Dostupné z DOI: 10.1016/0092-8674(77)90213-6.

WONGSURAWAT, T.; JENJAROENPUN, P.; KWOH, C. K.; KUZNETSOV, V., 2011. Quantitative model of R-loop forming structures reveals a novel level of RNA-DNA interactome complexity. *Nucleic Acids Research*. Roč. 40, č. 2, s. e16–e16. Dostupné z DOI: 10.1093/nar/gkr1075.

Přílohy

A Návod pro lokální spuštění serveru

Pokud z nějakého důvodu není možné využít server `https://bioinformatika.pef.mendelu.cz`, je možnost si webový server sestavit na lokálním stroji.

A.1 Prerekvizity

Pro lokální sestavení serveru je nutné si zažádat o přístup do repozitáře `GitLab`. Zdrojové kódy projektu jsou dostupné na adrese `https://git.pef.mendelu.cz/bioinformatics/dna-analyser`. Celý modul týkající se R-loop je zahrnutý jako součást merge requestu `https://git.pef.mendelu.cz/bioinformatics/dna-analyser/merge_requests/237`.

Pro úspěšné sestavení je nutné mít nainstalované v systému následující technologie:

- `docker`
- `docker-compose`
- `Gradle`
- `Java SE 8`
- `NodeJS`
- `npm`
- `git`

A.2 Sestavení (build) serveru

1. Nejprve je nutné naklonovat si repozitář

   ```
   git clone https://git.pef.mendelu.cz/bioinformatics/dna-analyser
                           && cd dna-analyser
   ```

2. Build lokálních databází (složka `/backend`)

   Databáze používají docker image, stačí tedy pouze image stáhnout z docker registru.

   ```
   cd ./backend && docker-compose build
   ```

3. Sestavení front-endových závislostí (složka `/frontend`)

   Nejprve musíme stáhnout všechny knihovny, které front-end používá.

   ```
   cd ./frontend && npm install
   ```

   Pak můžeme sestavit soubory pro front-end.

   ```
   npm run build
   ```

4. Sestavení back-endu (`root` projektu)

   Zdrojové soubory pro server se sestavují pomocí utility `gradle`. Nejprve musíme inicializovat nový gradle build a pak je možné sestavit soubory pro back-endový server.

   ```
   ./gradlew init && ./gradlew build
   ```

A.3 Spuštění (run) serveru

Protože jsou jednotlivé komponenty (databáze, back-end, front-end) na sobě závislé, je nutné je spouštět v přesně daném pořadí.

1. Spuštění databází

   ```
   cd ./backend && docker-compose run
   ```

2. Spuštění back-endu

   ```
   ./gradlew bootRun
   ```

3. Spuštění front-endu

   ```
   cd ./frontend && npm run serve
   ```

Server je pak dostupný na portu `8080` přes loopback interface:

```
http://localhost:8080
```