SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 285

STRUKTURA PODATAKA ZA EFIKASNO SPREMANJE OČITANJA DOBIVENIH SEKVENCIRANJEM GENOMA

Ivan Klabučar

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 285

STRUKTURA PODATAKA ZA EFIKASNO SPREMANJE OČITANJA DOBIVENIH SEKVENCIRANJEM GENOMA

Ivan Klabučar

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 12. ožujka 2021.

ZAVRŠNI ZADATAK br. 285

Pristupnik:	**Ivan Klabučar (0036513702)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Mile Šikić

Zadatak: **Struktura podataka za efikasno spremanje očitanja dobivenih sekvenciranjem genoma**

Opis zadatka:

Uređaji za sekvenciranje DNA u stanju su proizvesti velike količine podataka koje je potrebno na neki način obraditi. Razni alati u bioinformatici parsiraju skup podataka jednom i spreme ga u cijelosti, što za veće genome iziskuje ogromnu količinu radne memorije. Smanjenje potrebne memorije lako je izvedivo parsiranjem dijela ulaznog skupa podataka po potrebi, ali to pak znatno povećava vrijeme izvođenja. Idealno rješenje je pronaći način sažimanja koji je kompromis smanjenja memorije i gubitka informacije. Potrebno je implementirati strukturu podataka koja sažima FASTQ format, za podatke dobivene drugom i trećom generacijom uređaja za sekvenciranje, te evaluirati utjecaj sažimanja prilikom poliranja sastavljenih genoma alatom Racon. Rješenje mora biti napisano koristeći programski jezik C++. Programski kod je potrebno komentirati i pri pisanju pratiti neki od standardnih stilova. Napisati iscrpne upute za instalaciju i izvođenje. Kompletno programsko rješenje postaviti na GitHub pod jednom od OSI-odobrenih licenci.

Rok za predaju rada: 11. lipnja 2021.

POPIS SLIKA

POPIS TABLICA

SADRŽAJ

1. Uvod

Zbog prirode problema, rješavanje bilo kakvih zadataka na području genomike zahtjeva iznimno velike memorijske kapacitete. Genomi su sami po sebi gigantski, pa je tako i broj očitanja potrebnih za njihovo sastavljanje ili bilo kakvu drugu obradu jednako velik. Uobičajen način skladištenja ovih podataka jest u FASTQ formatu,[1] koji osim genetskog slijeda opisuje i niz vrijednosti koje predstavljaju ocjenu kvalitete (engl. *Phred quality scores*) FASTQ zapisa. Te vrijednosti, niz nukleotida i pripadni niz ocjena kvaliteta, generiraju uređaji za sekvenciranje genetskih sljedova tako da za svaki očitani nukleotid procjene vjerojatnost greške. Nukleotidi poprimaju samo četiri diskretne vrijednosti[2], pa se efektivno mogu pohraniti koristeći dva bita za svaku bazu, no isto se ne može reći i za ocjene kvalitete. Uzrok tome je mnogo širi raspon vrijednosti koje mogu poprimiti, otprilike $[0, 93]$. Pohrana takvih vrijednosti u memoriji iziskuje čak osam bitova za svaku ocjenu kvalitete. Dakle, primjenom opisanih načina kodiranja, ocjene kvalitete iziskuju čak četiri puta više radne memorije nego sami genetski sljedovi. Potrebu za memorijom može se smanjiti parsiranjem dijela skupa podataka po potrebi, ali to pak znatno povećava vrijeme izvođenja te zato nije primjenjivo. Pitanje kompresije prirodno se nameće kao moguće rješenje.

Cilj ovog rada razviti je algoritam sažimanja niza nukleotida i ocjena kvaliteta očitanja koji će postići dobar kompromis između smanjenja memorije, gubitka informacije i povećanja kompleksnosti. Osim samih karakteristika kompresije bitno je i ispitati njen učinak na rezultate postojećih alata. Drugim riječima, kako bi procijenili korisnost kompresije moramo utvrditi vezu između izgubljenih informacija i pada točnosti krajnjih rezultata bioinformatičkih alata. U tu svrhu alat Racon ćemo opremiti s razvijenim algoritmom kompresije te zatim evaluirati točnost njegovih rezultata.

[1] Pregled FASTQ formata napravljen je prema [1].

[2] Najčešće zapisane kao A, G, C ili T.

2. Pregled pojmova

2.1. Opis problema

Pri parsiranju skupova očitanja zapisanih u FASTQ formatu najočitije rješenje jest u radnoj memoriji svaki nukleotid zapisati s dva bita te svaku kvalitetu s osam bitova. Zadatak ovog rada pružiti je alternativu takvom pristupu nadogradnjom razreda *NucleicAcid* postojeće biblioteke *Biosoup*[1] s novim algoritmom kompresije vrijednosti kvaliteta. Fokus ovog rada bit će simulirana očitanja te očitanja dobivena Oxford Nanopore (ONT) i Pacific Biosciences (PacBio) tehnologijama. Potrebno je proučiti distribucije njihovih vrijednosti kvaliteta kako bi im mogli prilagoditi algoritam. Nadalje, postupak sažimanja treba evaluirati na širokom skupu podataka kako bi se procijenila njegova svojstva. Trebat ćemo izmjeriti količinu izgubljene informacija te utvrditi utjecaj na performanse alata pri korištenju dotične kompresije. U tu svrhu integrirat ćemo nadograđenu kolekciju *Biosoup* s alatom *Racon*. Kao što je opisano u [11], *Racon* služi za ispravljanje grešaka sastavljenih sljedova (engl. *assemblies*) dobivenih metodama bez konsenzus koraka. Uspješnost kompresije utvrđivat ćemo na temelju sličnosti između rezultata alata Racon i referenca dostupnih u izvornom skupu podataka, u čiju točnost, prema [9], možemo biti vrlo sigurni.

2.1.1. FASTQ format

```
@ime_ocitanja
CGGGGCGGGGTCTCTACGGGGTCTCT
+
<,<8@F@FF,,,<EC96+6::,,+7F
```

Slika 2.1: Primjer očitanja u FASTQ formatu. Plavo: niz nukleotida, narančasto: niz kvaliteta

Prema [6], skupovi DNA očitanja opisuju se FASTQ datotekama. To su jednos-

[1] Ova kolekcija opisana je u potpoglavlju 2.4.

tavne tekstualne datoteke koje zapisuju svako očitanje u četiri retka. Prvi red započinje znakom `@` iza kojeg slijedi ime očitanja. Drugi redak sadrži niz nukleotida kodiranih slovima A, G, C i T,[2] a u trećem redu nalazi se separator +. Poslije separatora, u četvrtom redu zapisan je niz kvaliteta očitanja svake pojedine baze iz drugog retka. Kvalitete su zapisane kao ascii znakovi, koji odgovaraju ocjeni kvalitete dotične baze uvećane za 33.[3] Na slici 2.1 prikazan je primjer jednog kratkog očitanja iz FASTQ datoteke. Dakle, ocjena kvalitete zadnje baze očitanja sa slike jest $70 - 33 = 37$ jer `ord('F')=70`.

2.1.2. Ocjena kvalitete

Prema [2], ocjena kvalitete jest zapravo logaritamska skala vjerojatnosti pogrešnog očitanja. Dakle, ako je P vjerojatnost da je neka baza bila pogrešno očitana, ocjenu kvalitete Q te baze dobiva se sljedećom formulom:

$$Q = -10 \log_{10} P \tag{2.1}$$

Iz formule 2.1 slijedi da svakim povećanjem ocjene kvalitete za 10, vjerojatnost pogrešnog očitanja pada za faktor 10.

Tablica 2.1: Ocjene kvalitete i odgovarajuće vjerojatnosti pogreške prema formuli 2.1

Kvaliteta	Vjerojatnost pogreške	Točnost očitanja
10	$1 \,/\, 10$	90%
20	$1 \,/\, 100$	99%
30	$1 \,/\, 1000$	99.9%
40	$1 \,/\, 10,000$	99.99%
50	$1 \,/\, 100,000$	99.999%
60	$1 \,/\, 1,000,000$	99.9999%

2.1.3. Potrebna svojstva kompresije

Pri razmatranju različitih pristupa sažimanja opisanih nizova kvaliteta moramo biti svjesni nekoliko preduvjeta koje novi algoritam mora zadovoljiti. Za učinkovit rad alata *Racon*, bitno je da nova kompresija ne premaši vremensku kompleksnost trenutne implementacije kolekcije *Biosoup*. To znači da svojstvo pristupa kvaliteti bilo kojeg nukleotida u $\mathcal{O}(1)$ vremenu mora biti očuvano, te da se samo sažimanje može

[2] Mogu se pojaviti još neka određena slova.

[3] Vrijednost prvog ispisivog znaka *'!'* jest 33.

izvršiti u linearnom vremenu.[4] Bilo kakvo povećanje kompleksnosti nije prihvatljivo jer bi previše usporilo već itekako vremenski intenzivne postupke. Nadalje, sam proces sažimanja mora biti relativno jednostavan iz istog razloga. Omjer smanjenja potrebnog prostora te izgubljene informacije također mora biti takav da se točnost alata minimalno naruši.

[4] U linearnom vremenu s obzirom na duljinu niza kvaliteta.

2.2. Postojeće tehnologije

Trenutna implementacija kolekcije *Biosoup*[5] u radnoj memoriji FASTQ očitanja sprema tako da nukleotide kodira s dva bita, dok kvalitete pohranjuje s osam bitova. Međutim, ne pohranjuje svaku pojedinačnu kvalitetu, nego samo prosjek svake 64 kvalitete. Ovaj jednostavan algoritam kompresije brz je i smanjuje potrebu za prostorom čak 64 puta ako razmatramo samo ocjene kvaliteta. Sveukupno, trenutna implementacija pruža faktor kompresije od 7.53.[6] Ipak, velika mana jest da gubi poprilično puno informacija o kvalitetama. Razvitkom novog algoritma kompresije ne nadamo se doseći isti faktor sažimanja, no očekujemo manje izgubljenih informacija. Ove usporedbe provest ćemo kasnije u radu.

Alati tvrtke Illumina također koriste kompresiju kvaliteta. Iz tablice 2.2, preuzete s [4], može se vidjeti da se kvalitete dijele u osam grupa te se kvalitete u istoj grupi poistovjećuju. U daljnjim izračunima, za svaku grupu empirijski se odabire specifična vrijednost koja ju najbolje predstavlja. Na taj se način ocjena kvalitete ograničava na osam diskretnih vrijednosti koje se onda mogu kodirati s tri bita. Faktor takve trobitne kompresije jest 2.66.

Tablica 2.2: Primjer grupacije kvaliteta u Illumina alatima [4]

Grupa ocjena kvaliteta	Odabrana ocjena kvalitete
N *(bez odluke)*	N *(bez odluke)*
$2-9$	6
$10-19$	15
$20-24$	22
$25-29$	27
$30-34$	33
$35-39$	37
≥ 40	40

Kao što je opisano u [4], konkretne granice između grupa izabiru se tako da empirijski optimiziraju gubitak informacije o kvalitetama. Iz tog razloga, tablica 2.2 podložna je promjenama jer se granice grupa kalibriraju u skladu s promjenama tehnologije.

[5] Kolekcija *Biosoup* opisana je u potpoglavlju 2.4.

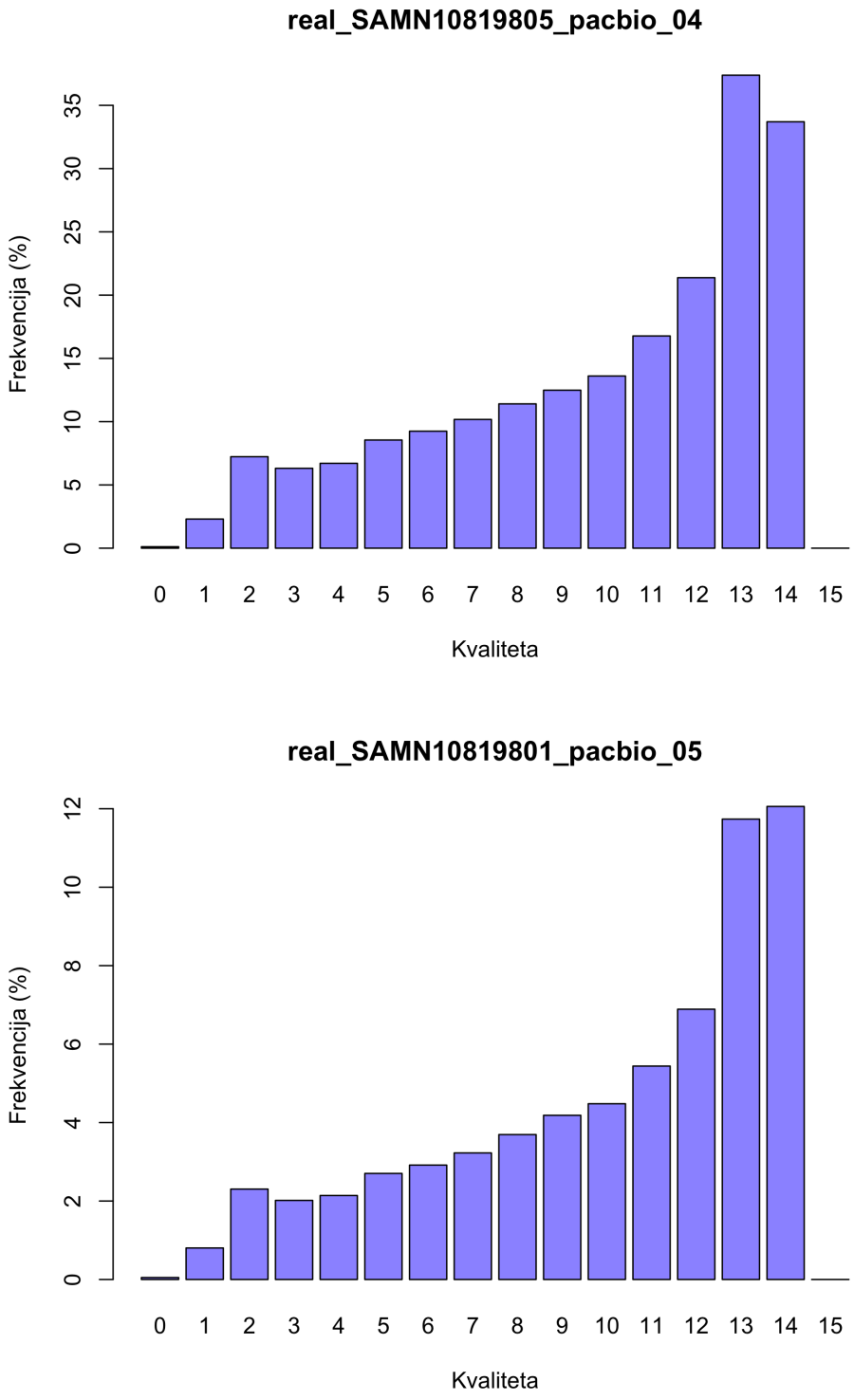
[6] Ako promatramo prozor od 64 baze: $\frac{64 \cdot 8 + 64 \cdot 8}{64 \cdot 2 + 1 \cdot 8} = 7.53$

2.3. Opis korištenog skupa podataka

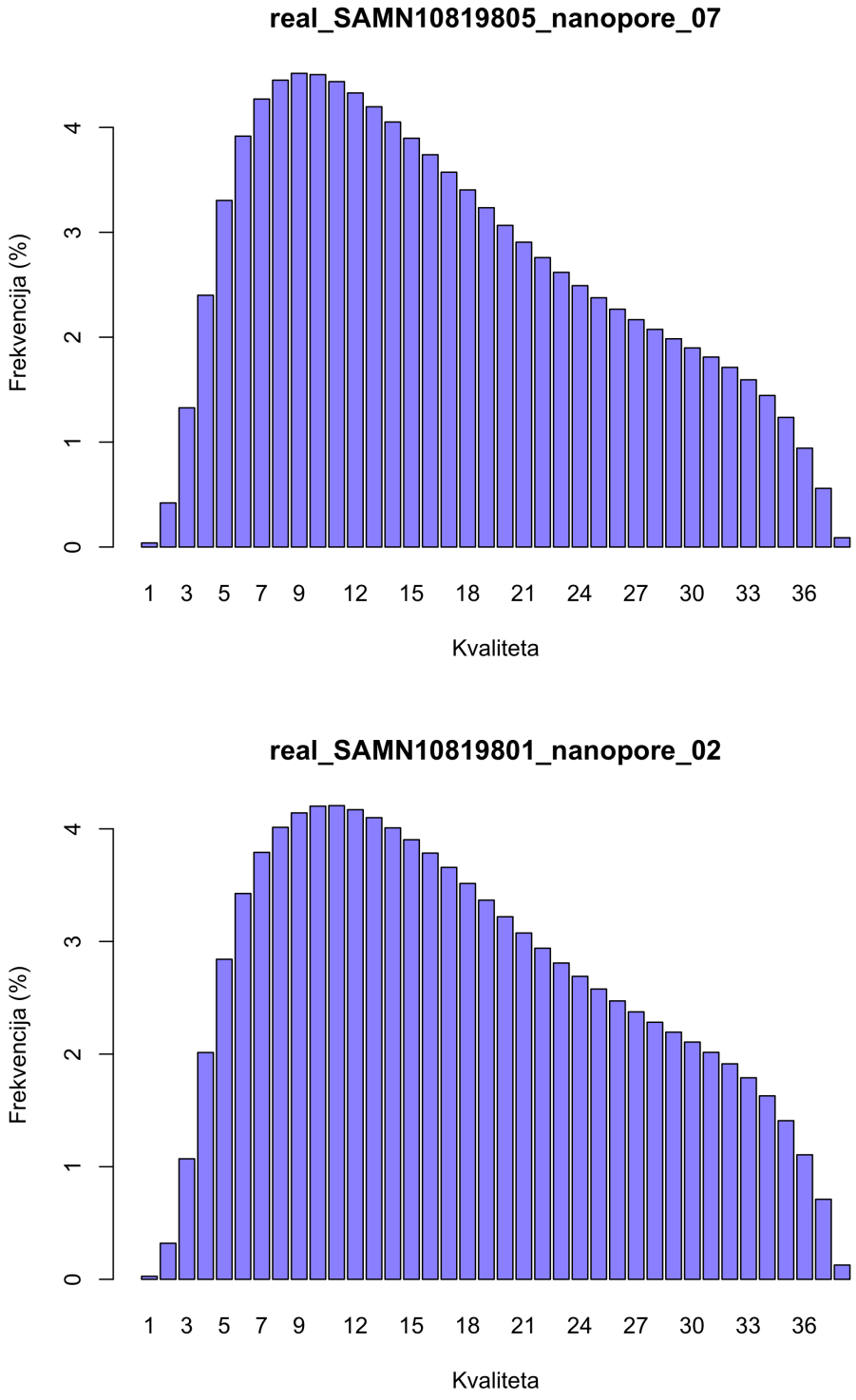
Skup očitanja korišten za evaluaciju performanse alata Racon preuzet je iz nedavnog članka o sastavljačima dugačkih očitanja [9]. U ovom radu korišten je samo podskup podataka iz spomenutog članka:

- 25 skupova očitanja dobivenih ONT (MinION R9.4) tehnologijom,
- 25 skupova očitanja dobivenih PacBio (RS II CLR) tehnologijom i
- 50 simuliranih skupova očitanja.

Prema [10] i [7], očitanja koja generiraju korištene verzije ONT te PacBio tehnologija vrlo su dugačka, ali i dosta nepouzdana. U prosjeku su dugačka između 13 i 20 kb za ONT (MinION R9.4) te između 10 i 16kb za PacBio (RS II CLR). Navedene duljine su samo okvirni prosjeci te maksimalne duljine koje ove tehnologije mogu proizvesti znatno su veće. Iako je velika duljina značajna prednost, točnost ovih tehnologija znatno je manja od onih koje generiraju kraća očitanja. Valja napomenuti da postoje novije verzije spomenutih tehnologija koje značajno poboljšavaju karakteristike dobivenih očitanja. Novi PacBio HiFi (*High-Fidelity*) protokol u stanju je proizvesti očitanja koja su, prema [14], ujedno i dugačka i visoko pouzdana. Protokol HiFi razlikuje se od starije verzije po tome što koristi CCS (engl. *circular consensus sequencing*) kako bi generirao dugačka i točna očitanja iz kraćih nepouzdanih podočitanja. Starija verzija tehnologije koristi CLR (engl. *continuous long reads*) očitanja koja su dugačka, ali ne i pouzdana. Razmotrimo nekoliko distribucija kvaliteta korištenih skupova očitanja:



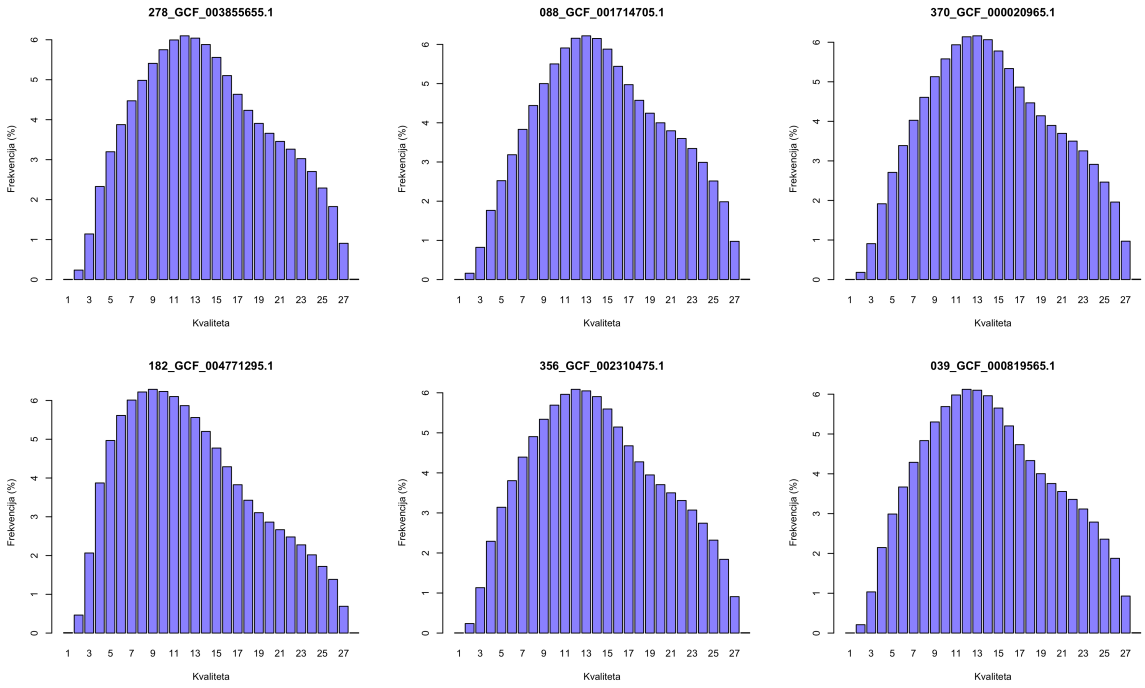
Slika 2.2: histogram kvaliteta PacBio skupa očitanja



Slika 2.3: histogram kvaliteta ONT skupa očitanja

Nepreciznost ovih tehnologija odražava se u niskim prosječnim vrijednostima kvalitete koja su u oba slučaja daleko ispod idealnih vrijednosti. Također, iz slika 2.2 i 2.3 možemo vidjeti da je prosjek kvalitete PacBio očitanja niži od ONT očitanja, te da PacBio kvalitete poprimaju puno uži raspon. Distribucija PacBio kvaliteta negativno je iskrivljena, dok je distribucija ONT kvaliteta pozitivno iskrivljena.

Osim stvarnih skupova očitanja, koristit ćemo i 50 simuliranih skupova očitanja. Kao što je opisano u [9], ti skupovi generirani su alatom Badread v0.1.5[7] s nasumičnim vrijednostima parametrima koji određuju učestalost pogrešaka, duljinu očitanja, raspon vrijednosti kvaliteta itd. Korištenje simuliranih očitanja pruža nam mogućnost ispitivanja djelovanja kompresija na vrlo raznolikim skupovima s puno više varijacija u usporedbi s relativno sličnim distribucijama PacBio i ONT očitanja. Još jedna prednost simuliranih očitanja jest točno poznavanje njihovog izvornog genetskog slijeda. Ovo je iznimno korisno svojstvo jer ćemo kasnije pri evaluaciji rezultata alata Racon vršiti usporedbu s referencama dobivenim drukčijim sastavljačima. Kako su izvorni sljedovi simuliranih nizova otpočetka poznati, te ih ne treba sastaviti iz skupa očitanja, dio analize obavljen nad simuliranim podatcima neće imati istu smetnju prouzročenu korištenjem različitih sastavljača. Primjer šest nasumično odabranih distribucija kvaliteta simuliranih skupova očitanja možemo vidjeti na slici 2.4.



Slika 2.4: histogram kvaliteta šest simuliranih skupova očitanja

[7] https://github.com/rrwick/Badread

2.4. Korišteni alati

Tijekom implementacije kompresije te njenog testiranja rabio sam nekoliko alata za različite zadatke na području bioinformatike. U ovom potpoglavlju dajem njihov kratak opis.

BIOSOUP[8] jest `c++` kolekcija podatkovnih struktura koja se koristi za pohranu bioinformatičkih podataka. Jedna od klasa definirana u ovoj kolekciji jest `NucleicAcid` koja opisuje genetski slijed i popratne ocjene kvaliteta. U sklopu ovog rada, taj razred nadogradit ću s razvijenim algoritmom kompresije niza kvaliteta. Alat *Racon*, čiji ćemo rad evaluirati u ovisnosti o kompresiji, već koristi razrede iz kolekcije *Biosoup* pa će ga zbog toga biti lako integrirati s novim algoritmom. Trenutna verzija razreda `NucleicAcid` također sažima ocjene kvaliteta, ali na način koji gubi puno informacija. Naime, razred pamti prosječnu kvalitetu nad svakim prozorom od 64 kvalitete. Ipak, ovaj pristup je brz i pruža velik faktor kompresije. Kasnije u radu, performanse alata *Racon* evaluirat ćemo u slučaju korištenja različitih nadograđenih verzija razreda `NucleicAcid`.

RAVEN,[9] opisan u [12], jest de novo sastavljač genoma specijaliziran za dugačka neispravljena očitanja. Njime smo sastavili skupove očitanja[10] i kao rezultat dobili FASTA datoteke koje predstavljaju nepolirane sastavljene sljedove. Te rezultantne FASTA datoteke opisuju originalni genetski slijed iz kojeg su promatrana očitanja potekla. Naravno, rezultat alata *Raven* nije savršen i do neke mjere razlikuje se od stvarnosti. U svrhu minimiziranja tih razlika *Raven* interno obavlja korekciju svojih rezultata alatom za poliranje *Racon*. Na taj se način postiže veća točnost rezultata. Ipak, *Raven* smo koristili sa zastavicom `-p 0` koja signalizira da se korak korekcije rezultata treba preskočiti jer je cilj ovog rada ocijeniti performanse upravo tog koraka ovisno o korištenoj kompresiji. Dakle, korak poliranja alatom *Racon*, koji inače obavlja alat *Raven*, obavljen je ručno za svaku verziju *Racon*-a koju smo htjeli evaluirati. Te verzije razlikuju se samo u algoritmu kompresije ocjena kvaliteta koji koriste.

RACON,[11] opisan u [13], jest alat kojeg sam koristio za poliranje rezultata sastavljenih alatom *Raven*. *Racon* interno ovisi o kolekciji *Biosoup* pa ga se može integrirati s različitim kompresijama kvaliteta tako da ga se konfigurira s različitim verzijama

[8] `https://github.com/rvaser/biosoup`
[9] `https://github.com/lbcb-sci/raven`
[10] Primjer jedne takve datoteke skupa očitanja: `real_SAMN10819805_pacbio_04.fastq`.
[11] `https://github.com/lbcb-sci/racon`

kolekcije *Biosoup*. Za svaku kombinaciju sastavljenog slijeda i verzije *Racon*-a, poliranje će biti izvršeno dvaput jer je to podrazumijevano ponašanje alata *Raven*.[12] Taj postupak će nam za svaku spomenutu kombinaciju slijeda i verzije *Racon*-a, dati novi polirani slijed koji bi, zbog postupka korekcije, trebao biti točniji. Točnost dobivenih sljedova procijenit ćemo na temelju sličnosti s dostupnim referencama, a potom ćemo na temelju tih informacija uspoređivati različite vrste kompresija.

QUAST[13] jest kratica od "QUality ASsessment Tool". Ovaj alat, opisan u [3], računa različite mjere nad sastavljenim genetskim sljedovima. Iako se *Quast* može koristiti sa samim sastavljenim sljedovima, puno je informativniji kada se koristi s popratnom referencom originalnog slijeda. Upravo tako ćemo *Quast* koristiti u ovom radu. Mjere koje će nas zanimati su *mismatches per 100 kbp* te *indels per 100 kbp*. *Kbp* jest mjerna jedinica *kilo-base pair*, odnosno 1000 baznih parova. Prema [8], *mismatches* i *indels* su vrste razlika između dva genetska slijeda. Slučaj razlikovanja dva slijeda na nekom mjestu zove se *mismatch*, a slučaj baze prisutne u jednom slijedu koja nedostaje u drugom zove se *indel*. *Indel* jest kratica od ***in****sertion or* ***del****etion*. Ta dva pojma zapravo opisuju istu situaciju, ali s različitih stajališta. Ako je u jednom slijedu prisutan *insertion* u drugom je prisutan *deletion* i obratno. Vrijednosti *mismatches per 100 kbp* te *indels per 100 kbp* daju nam procjenu do koje mjere se neka dva genetska slijeda razlikuju.

DNADIFF[14] jest alat koji uspoređuje slične genetske sljedove. Dio je sustava *MUMmer* opisanog u [5]. Isto kao i Quast, daje nekoliko različitih mjera, no ona koja nas zanima je *AvgIdentity*. To je mjera koja u postotcima opisuje sličnost dva slijeda, te ćemo ju u nastavku rada zvati *accuracy* ili *sličnost*. Može se staviti u odnos s mjerama *mismatches per 100 kbp* te *indels per 100 kbp* preko sljedeće formule:

$$Accuracy = (1 - (indels + mismatches) \,/\, 100000) * 100\% \tag{2.2}$$

Ova mjera nam je zanimljiva jer lagano možemo usporediti točnost više različitih sljedova na temelju samo jednog broja.

[12] To podrazumijevano ponašanje izbjegli smo korištenjem zastavice `-p 0` pa ga ručno provodimo.
[13] `https://github.com/ablab/quast`
[14] `https://github.com/marbl/MUMmer3/blob/master/docs/dnadiff.README`

3. Algoritam

S obzirom da je gubitak informacija pri sažimanju niza nukleotida neprihvatljiv[1], za taj zadatak možemo razmatrati samo algoritme kompresija bez gubitka informacija. Prirodan pristup ovom problemu jest kodirati svaki nukleotid s dva bita jer nukleotid može poprimiti samo četiri vrijednosti pa na taj način uistinu ne gubimo nikakve informacije. Ovakvo ponašanje već je implementirano u kolekciji *Biosoup* te ga u sklopu ovog rada nećemo mijenjati. Dvobitno kodiranje nukleotida trenutno implementirano u kolekciji *Biosoup* možemo vidjeti na tablici 3.1. Sam postupak sažimanja niza nukleotida jest linearne kompleksnosti, a dohvat proizvoljne baze može se obaviti u konstantnom vremenu jer su nukleotidi i dalje slijedno zapisani u memoriji. Ovakvim sažimanjem niza nukleotida ispoštovali smo sve zahtjeve iz potpoglavlja 2.1.3.

Tablica 3.1: Kodiranje nukleotida u *Biosoup*-u

Baza	Binarni kôd
A	00
C	01
G	10
T	11

Opisanim pristupom, problem efikasne pohrane niza nukleotida u radnu memoriju jest riješen pa ćemo se u ostatku rada fokusirati na sažimanje niza ocjena kvaliteta. Razvijeni algoritam kompresije temelji se na ideji da svaku ocjenu kvalitete kodiramo s dva bita. Svaka od četiri moguće kombinacije dva bita predstavljala bi neku specifičnu kvalitetu. Dakle, novi algoritam kompresije bi sve vrijednosti originalnog niza zamijenio kôdom čija pridružena kvaliteta – koliko je moguće – minimizira udaljenost do originalne vrijednosti. Za kvalitetu performansa ovakvog pristupa najbitniji je dobar odabir četiri vrijednosti u koje preslikavamo kvalitete originalnog niza. Ipak, kako su nizovi koje planiramo sažeti ovim algoritmom iznimno dugi, bitno je da kompleksnost tog odabira ne premaši $\mathcal{O}(n)$ u skladu s zahtjevima iz potpoglavlja 2.1.3. Drugi

[1] Jer razlika u čak samo jednom nukleotidu može biti od presudne važnosti.

korak algoritma jest zamjena svake vrijednosti iz originalnog niza jednim od četiri moguća kôda. Odabire se onaj kôd koji se preslikava u ocjenu kvalitete koja je najbliža vrijednosti kvalitete iz originalnog niza. Kompleksnost ovog koraka jest $\mathcal{O}(n)$. Nakon primjene kompresije, ocjeni kvalitete na proizvoljnom indeksu može se pristupiti u $\mathcal{O}(1)$ jer su kvalitete i dalje suštinski slijedno zapisane u memoriji. Ovako opisan algoritam ne bi povećao kompleksnost alata Racon jer poštuje zahtjeve iz potpoglavlja 2.1.3, a smanjio bi količinu potrebne memorije četiri puta jer pohranjuje jednu bazu odnosno jednu ocjenu kvalitete s dva bita, umjesto s osam.

Primijeniti isto preslikavanje kôdova u kvalitete na cijelom nizu bilo bi previše grubo jer je razumno očekivati da će različiti dijelovi niza imati različite distribucije kvaliteta pa tako i različita optimalna preslikavanja. Drugim riječima, primjenjivanje istog preslikavanja nad cijelim nizom uzrokovalo bi lošiju kompresiju za dulje nizove. Za relativno kratke nizove, odabir preslikavanja bi se mogao kvalitetno odrediti, no za duge nizove ne bi se moglo odrediti jedinstveno preslikavanje koje bi bilo povoljno za sve regije niza. To bi značilo da možemo očekivati porast greške kompresije s porastom duljine niza, a takva nekonzistentnost algoritma jest neprihvatljiva. Kako bi otklonili ovu manu, izračun preslikavanja moramo provesti nezavisno nad svim podnizovima neke određene duljine. To jest, u algoritam moramo uvesti koncept prozora. Ova odluka ne mijenja kompleksnost algoritma niti produljuje vrijeme pristupa proizvoljnoj kvaliteti, ali smanjuje memorijsku efikasnost kompresije. Umjesto samo jednog preslikavanja za cijeli niz, primjenom prozora mora se pamtiti broj preslikavanja proporcionalan duljini niza. Ipak, pametnim dizajnom, dodatna potreba za memorijom ne smanjuje faktor kompresije za velik iznos,[2] a rješava bitan problem opisanog pristupa.

3.1. Odabir preslikavanja

Kako bismo mogli odrediti dobro kodiranje za zadani niz kvaliteta moramo poznavati neke informacije o njegovoj distribuciji. Moj algoritam odluku o preslikavanju donosi na temelju: srednje, minimalne, maksimalne i mod vrijednosti niza kvaliteta. Razlika maksimalne i minimalne kvalitete govori nam širinu distribucije, a četvrtina te širine odgovara rasponu koji jedna kombinacija dva bita konceptualno treba pokriti. Poredak srednje i mod vrijednosti govori nam je li distribucija pozitivno ili negativno iskrivljena. Pozitivno je iskrivljena ako je srednja vrijednost veća od mod vrijednosti, a u suprotnom slučaju negativno je iskrivljena. Ako su srednja i mod vrijednost jednake,

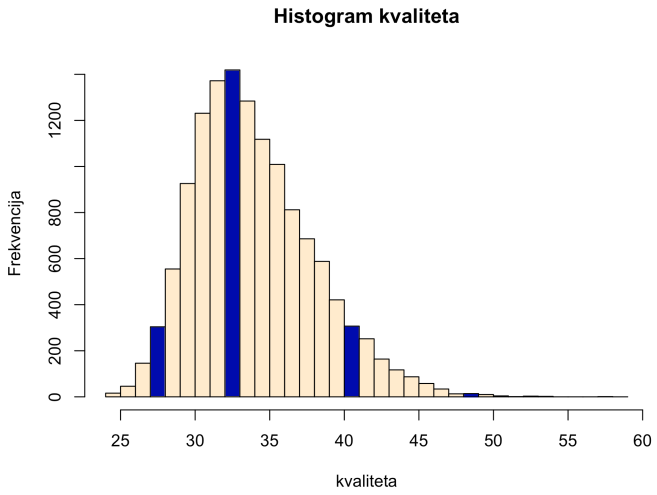
[2] Veličina prozora korištena u ovom radu navedena je u potpoglavlju 3.3 zajedno sa izračunom konačnog faktora kompresije kvaliteta.

distribucija nije iskrivljena. Mod vrijednost kvalitete jest ocjena koja je u distribuciji najzastupljenija. Nju algoritam odmah bira kao jednu od vrijednosti za preslikavanje, a potom odvojeno razmatra intervale $[minimum, mod]$ te $[mod, maksimum]$. Algoritam pokušava svakom intervalu alocirati broj kôdova jednak kvocijentu širine intervala i četvrtine širine distribucije, no kako je već jedan kôd dodijeljen mod vrijednosti, taj kôd mora se uračunati u neki od intervala. U slučaju pozitivno iskrivljene distribucije mod vrijednost dodjeljujemo intervalu $[mod, maksimum]$, a u slučaju negativno iskrivljene distribucije mod vrijednost pripadne intervalu $[minimum, mod]$. Raspodijeljeni kôdovi dodjele se jednoliko razmaknutim vrijednostima u svojim intervalima. Točnije, za oba intervala definira se odgovarajući korak prema formuli:

$$korak = \left\lfloor \frac{duljina\ intervala}{broj\ kodova\ dodjeljenih\ intervalu^{3} + 1} \right\rfloor \tag{3.1}$$

Zatim se uvećavanjem ili umanjivanjem[4] mod vrijednosti višekratnikom odgovarajućeg koraka određuju vrijednosti kvaliteta u koje će se preostali kôdovi preslikati.

Jedan primjer izračuna vrijednosti u koje se preslikavaju četiri kôda odabira prikazan je na slici 3.1.



Slika 3.1: Primjer odabira preslikavanja na slučajnoj distribuciji, plavo: odabrane vrijednosti

Parametri ove distribucije su:

- Srednja vrijednost: 34
- Minimum: 24

[3]U ovaj broj nije uračunat kôd koji pripada mod vrijednosti.
[4]Ovisno radi li se o gornjem ili donjem intervalu.

– Maksimum: 58

– Mod: 33

Možemo vidjeti da je distribucija pozitivno iskrivljena. Dakle, intervalu $[minimum, mod]$ dodijeljen je jedan kôd, dok su intervalu $[mod, maksimum]$ dodijeljena tri kôda uključujući kôd mod vrijednosti. Vrijednosti pridružene kôdovima su: 28, 33, 41 i 49.

Za efikasno računanje minimuma, maksimuma i mod vrijednosti možemo upotrijebiti činjenicu da ocjene kvaliteta poprimaju vrijednosti iz relativno uskog raspona te upotrijebiti sortiranje prebrajanjem (engl. *counting sort*) nad nizom originalnih kvaliteta. Za razliku od ostalih opcija, složenost ovog algoritma sortiranja jest $\mathcal{O}(n)$ pa ne narušavamo zahtjeve iz potpoglavlja 2.1.3. Rezultat algoritma *counting sort* jest niz frekvencija pojavljivanja indeksa X u originalnom nizu. To jest,

$$FREQ[X] = broj\ pojavljivanja\ X\ u\ nizu\ kvaliteta$$

Iz ovog niza potrebne informacije računamo na sljedeći način:

– Minimum kao prvi indeks čiji je element veći od nule

– Maksimum kao zadnji indeks čiji je element veći od nule

– Mod kao indeks maksimalnog elementa

Srednju vrijednost originalnog niza kvaliteta računamo uzastopnim zbrajanjem svih kvaliteta te dijeljenjem sume duljinom originalnog niza.

Na objašnjen način, računa se povoljno preslikavanje nad svakim prozorom niza kvaliteta. Svako preslikavanje se zatim koristi za kôdiranje pripadnog prozora, te pohranjuje za kasnije potrebe dekodiranja.

Algorithm 1: Odabir povoljnog kodiranja

input : Niz kvaliteta Q
output: Niz četiri odabrane kvalitete R

$FREQ[100]$ // sva polja inicijalizirana na 0
$SUM \leftarrow 0$
for $q \in Q$ **do**
    $SUM \leftarrow SUM + q$
    $FREQ[q] + +$
end
$AVG\_Q \leftarrow SUM/len(Q)$
$MOD\_Q \leftarrow max\_arg(FREQ)$
$MAX\_Q \leftarrow last\_nonzero(FREQ)$
$MIN\_Q \leftarrow first\_nonzero(FREQ)$
$QUARTER \leftarrow (MAX\_Q - MIN\_Q)\ /\ 4.0$
if $QUARTER < 1.0$ **then** $QUARTER \leftarrow 1.0$
if $MOD\_Q > AVG\_Q$ **then**
    $$TOP\_LVLS \leftarrow \frac{MAX\_Q - MOD\_Q}{QUARTER}$$
    $BTM\_LVLS \leftarrow (4 - TOP\_LVLS - 1)$
else if $MOD\_Q < AVG\_Q$ **then**
    $$BTM\_LVLS \leftarrow \frac{MOD\_Q - MIN\_Q}{QUARTER}$$
    $TOP\_LVLS \leftarrow (4 - BTM\_LVLS - 1)$
else
    $BTM\_LVLS \leftarrow 1$
    $TOP\_LVLS \leftarrow 2$
end
$$TOP\_STEP \leftarrow \frac{MAX\_Q - MOD\_Q}{TOP\_LVLS + 1}$$
$$BTM\_STEP \leftarrow \frac{MOD\_Q - MIN\_Q}{BTM\_LVLS + 1}$$
$CURR\_LVL \leftarrow MIN\_Q$
for $i \in 1 : BTM\_LVLS$ **do**
    $CURR\_LVL \leftarrow CURR\_LVL + BTM\_STEP$
    $R.append(CURR\_LVL)$
end
$CURR\_LVL \leftarrow MOD\_Q$
$R.append(CURR\_LVL)$
for $i \in 1 : TOP\_LVLS$ **do**
    $CURR\_LVL \leftarrow CURR\_LVL + TOP\_STEP$
    $R.append(CURR\_LVL)$
end

3.2. Pregled konačnog algoritma

Dvobitni algoritam sažimanja obrađuje originalni niz kvaliteta prozor po prozor. U svakoj etapi za aktivan prozor računa se preslikavanje iz potpoglavlja 3.1 koje se potom koristi za kodiranje svake kvalitete u prozoru s dva bita. Izračunata preslikavanja i kompresirane kvalitete pohranjuju se slijedno u memoriju. Prilikom pristupa nekoj sažetoj kvaliteti, prvo se određuje kojem preslikavanju pripada, a zatim se – na temelju tog preslikavanja – dekodira.

Algorithm 2: Sažimanje niza kvaliteta

input : Niz kvaliteta Q
output: Niz sažetih kvaliteta Q_s, niz preslikavanja P

```
foreach prozor in Q do
    p ← odabir_povoljnog_kodiranja(prozor)  // algoritam opisan u 3.1
    P.append(p)
    foreach q in prozor do
        Q_s.append(indeks_najblie_vrijednosti(q, p))
    end
end
return Q_s, P
```

3.3. Opis konkretne implementacije

Algoritam sažimanja implementiran je kao nadogradnja biblioteke *Biosoup*, točnije nadogradnja `nucleic_acid.hpp` datoteke. Ova datoteka već ima definiranu klasu `NucleicAcid` koja koristi kompresiju ocjena kvalitete opisanu u 2.4. Algoritam sažimanja implementiran je točno onako kako je opisano u prethodnim potpoglavljima, a implementacijske detalje vezane uz pohranu i pristup sažetim vrijednostima opisat ćemo u nastavku.

Kompresija kvaliteta odvija se u konstruktoru klase, koji prima referencu na znakovni niz kvaliteta preuzetog iz FASTQ datoteke. Dakle, za dobivanje pravih kvaliteta, vrijednosti iz originalnog niza moraju se umanjiti za `ord('!') = 33`. Moj algoritam kompresije koristi veličinu prozora 512, odnosno 2^9. To znači da za svakih 512 ocjena kvaliteta moramo pamtiti i jedno preslikavanje koje zauzima 32 bita. Uporabom spomenute veličine prozora faktor kompresije ocjena kvaliteta pada sa 4.00[5] na 3.88[6].

[5] Kodiranje svake ocjene kvalitete sa samo dva bita rezultira faktorom kompresije 4.00.

[6] $\frac{512 \cdot 8}{512 \cdot 2 + 32} = 3.88$

Naravno, nisu svi nizovi kvaliteta višekratnici broja 512 pa je veličina zadnjeg prozora varijabilna. Sažete kvalitete spremaju se u sljedeću strukturu podataka:

```
std::vector<std::uint64_t> deflated_quality;
```

Svaka kvaliteta kodirana je s dva bita pa svaki broj u ovom vektoru predstavlja blok od 32 ocjene kvalitete. Kôd kvalitete indeksa `i` može se iz zadanog vektora dohvatiti sljedećim naredbama:

```
(deflated_quality[i >> 5] >> ((i << 1) & 63)) & 3;
```

Bitno je primijetiti da redni broj bloka kojem neka kvaliteta pripada ovisi o 27 viših bitova njenog indeksa, a točna pozicija unutar tog bloka o nižih pet bitova njenog indeksa. Izraz `i >> 5` ekvivalentan jest dijeljenju broja `i` s 32 što nam daje indeks relevantnog bloka. Zatim, taj blok moramo posmaknuti udesno za redni broj kvalitete unutar bloka pomnožen s dva. Taj iznos posmaka računa se s `((i << 1) & 63))`. Konačno, trebamo zanemariti sve bitove osim dva najniža, što postižemo operacijom `& 3`.

Preslikavanja svih prozora pohranjena su u strukturi oblika:

```
std::vector<std::uint32_t> quality_levels;
```

Broj na `i`-tom mjestu ovog vektora predstavlja preslikavanje korišteno za `i`-ti prozor. Preslikavanja su zapisana kao 32-bitni brojevi tako da svaki bajt[7] predstavlja jednu ocjenu kvalitete. Slično kao i za kompresirane kvalitete, preslikavanje primijenjeno na `i`-toj kvaliteti može se dobiti izrazom:

```
quality_levels[i >> 9]
```

U mojoj implementaciji algoritma, kôd neke kvalitete proporcionalan je iznosu za koji moramo posmaknuti preslikavanje u desno kako bismo na najnižih osam bitova dobili odgovarajuću kvalitetu. Dakle iz varijable `kod` i `preslikavanje`, odgovarajuću kvalitetu dobivamo na sljedeći način:

```
((preslikavanje >> (kod * 8)) & 255)
```

Preslikavanje posmičemo u desno za iznos `kod * 8` kako bismo na osam najnižih bitova dobili željenu kvalitetu, a zatim logičkom operacijom `& 255` zanemarujemo sve više bitove. Na opisan način, kvaliteti na proizvoljnom indeksu efikasno možemo pristupiti s složenošću $\mathcal{O}(1)$.

[7] Osam bitova.

Razred `NucleicAcid` opisane postupke enkapsulira, te pristup ocjenama kvaliteta pruža preko sljedećeg sučelja. Članska funkcija `Score`

```
std::uint8_t Score(std::uint32_t i)
```

vraća ocjenu kvalitete nukleotida na indeksu `i`. Osim jednoj ocjeni kvalitete možemo pristupiti i nizu kvaliteta preko članske funkcije `InflateQuality`:

```
std::string InflateQuality(
                std::uint32_t i = 0,
                std::uint32_t len = -1
                )
```

Ona vraća string s `len` kvaliteta počevši od indeksa `i`. Podrazumijevana vrijednost varijable `len` jest -1 te označava da se treba vratiti cijeli niz kvaliteta. Kvalitete unutar vraćenog string-a zapisane su u čitljivom obliku, to jest uvećane za `ord('!') = 33`. Pristup nukleotidima omogućen je ekvivalentnim funkcijama `Code` i `InflateData`.

```
std::uint64_t Code(std::uint32_t i)
```

```
std::string InflateData(
                 std::uint32_t i = 0,
                 std::uint32_t len = -1
                 )
```

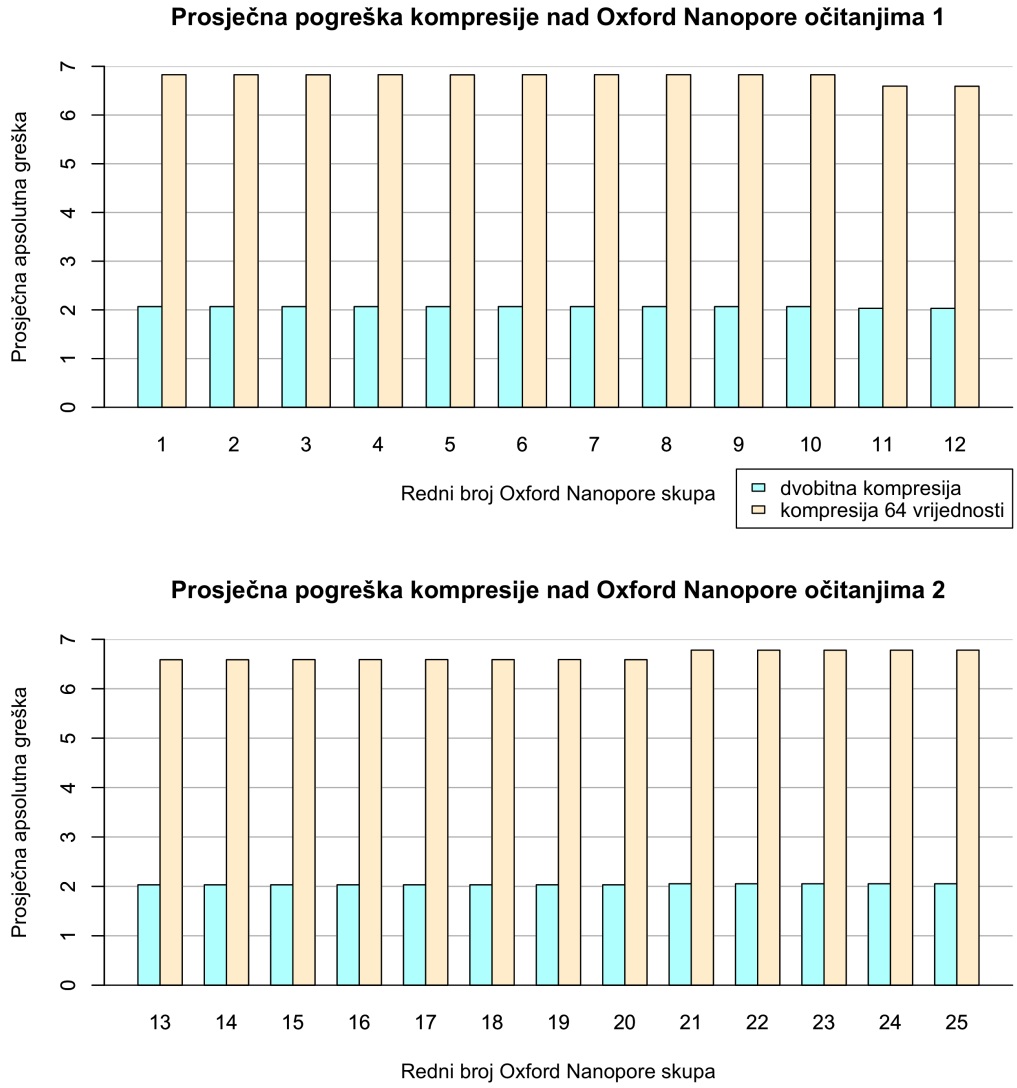
Preko opisanih funkcija alati koji koriste kolekciju *Biosoup*, poput alata *Racon*, pristupaju potrebnim informacijama.

4. Rezultati

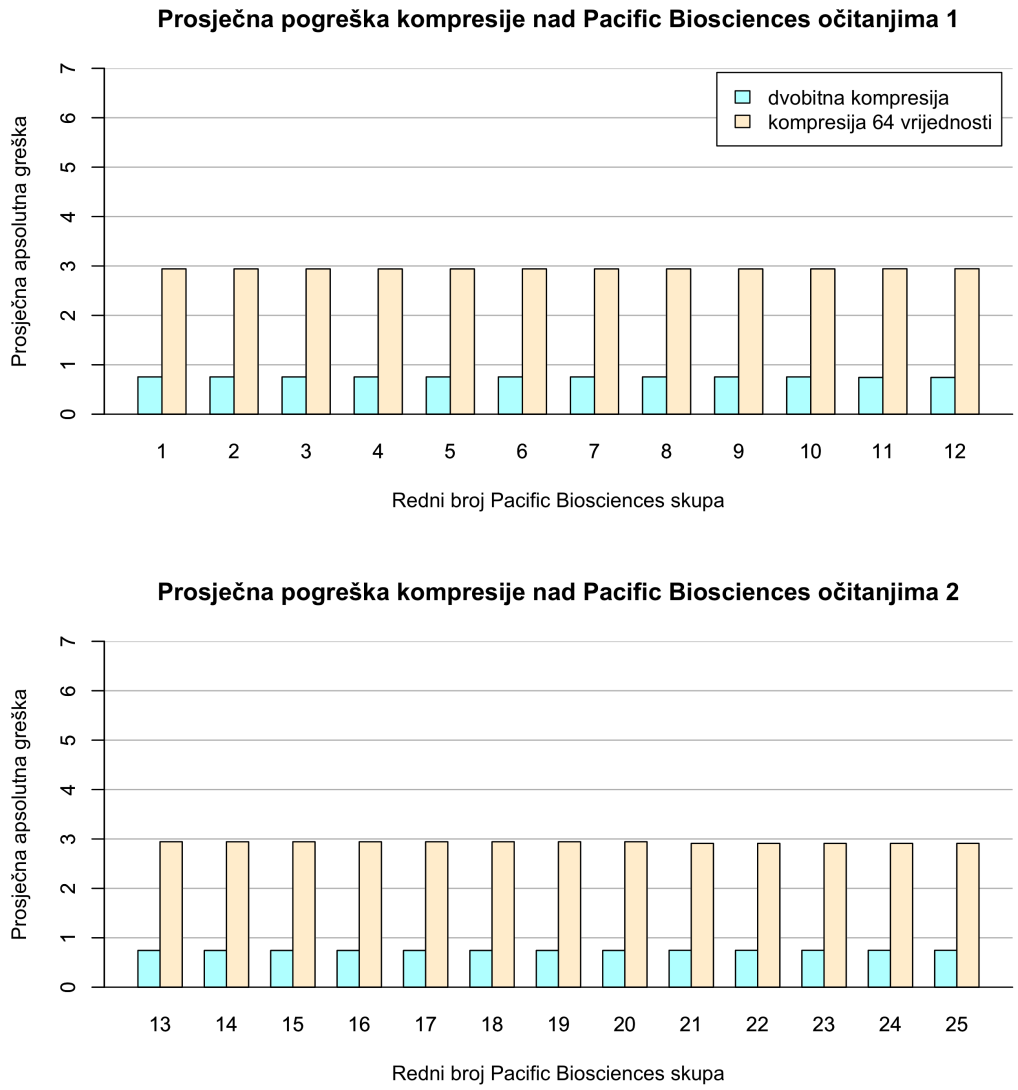
U svrhu procjene korisnosti kompresije analizirat ćemo njene učinke na nekoliko načina. Prva mjera koju ćemo razmotriti jest prosječna greška kompresije, to jest koliko veliku razliku možemo očekivati između kompresiranog i originalnog niza kvaliteta. Ovo mjera najbolje opisuje količinu informacija koju gubimo sažimanjem. Osim točnosti sažetog niza kvaliteta, zanimaju nas i posljedice njegovog korištenja. U tu svrhu alat *Racon* opisan u 2.4 integrirat ćemo s tri verzije *Biosoup* kolekcije s različitim kompresijama te evaluirati njegove rezultate. Preciznije, analizirat ćemo sličnosti genetskih sljedova poliranih alatom *Racon* i referenca u ovisnosti o korištenoj kompresiji.

4.1. Točnost kompresije

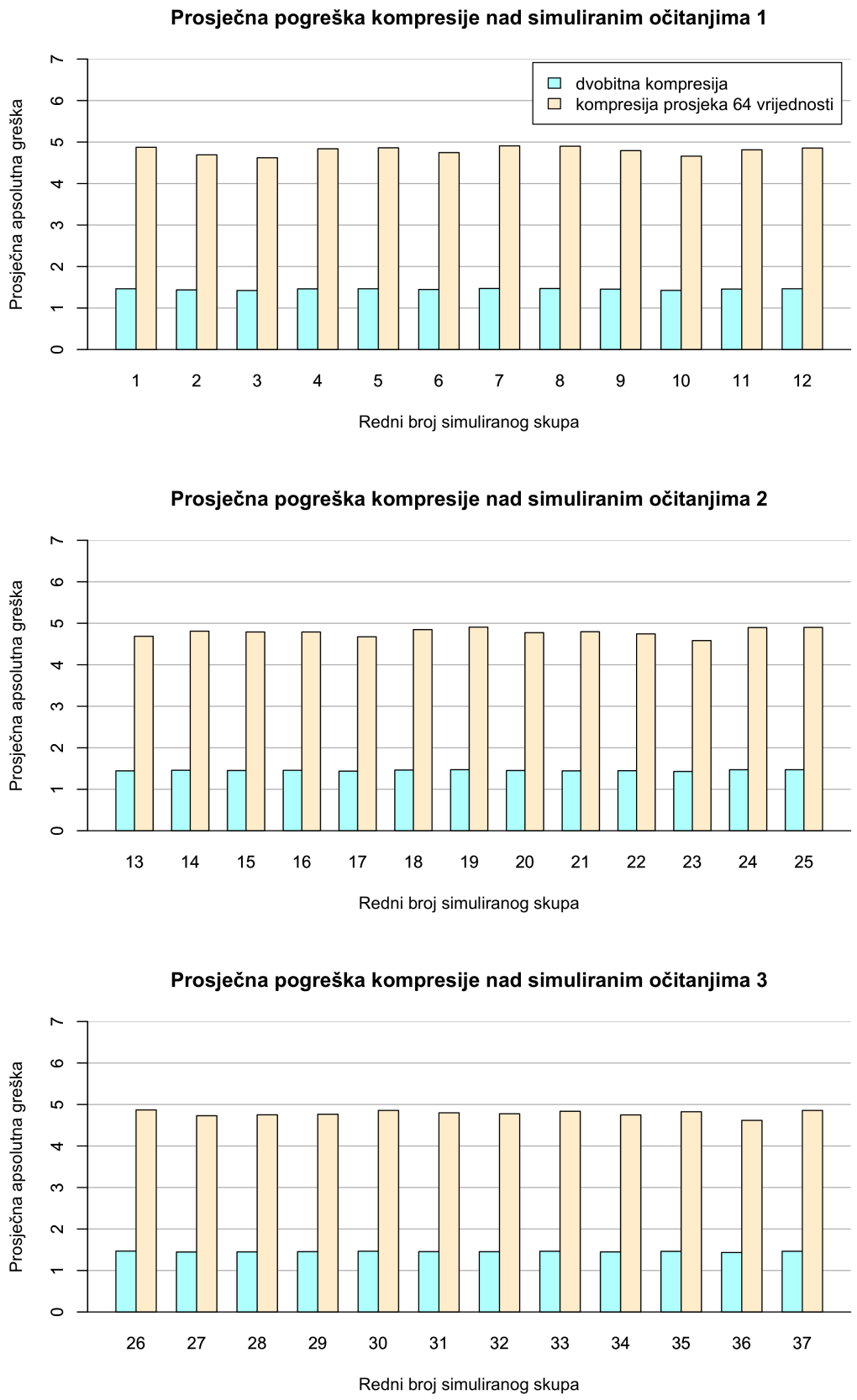
Kao inverznu mjeru točnosti kompresije za svaki skup očitanja izračunali smo prosječnu pogrešku nad očitanjem. Pogrešku nad očitanjem definirali smo kao očekivanu razliku između stvarne i sažete ocjene kvalitete nukleotida u očitanju. Ovu mjeru izračunat ćemo za dvobitnu kompresiju te za kompresiju koja je trenutno implementirana u kolekciji *Biosoup* 2.4. Rezultate nad stvarnim skupovima očitanja možemo vidjeti na grafovima 4.1 i 4.2.



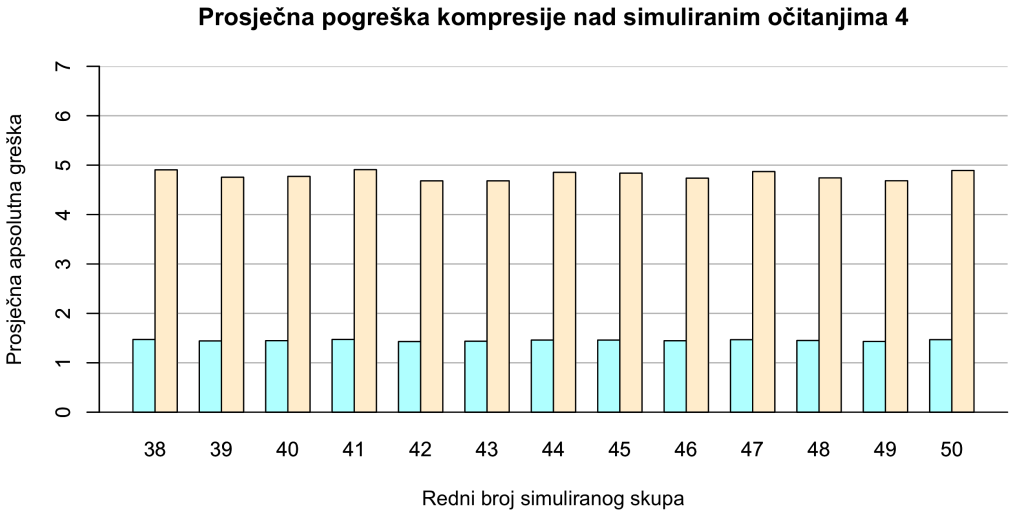
Slika 4.1: Usporedba prosječnih pogrešaka nad Oxford naopore skupovima očitanja



Slika 4.2: Usporedba prosječnih pogrešaka nad PacBio skupovima očitanja



Slika 4.3: Usporedba prosječnih pogrešaka nad simuliranim skupovima očitanja



Slika 4.3: Usporedba prosječnih pogrešaka nad simuliranim skupovima očitanja

Tablica 4.1: Prosječna greška* kompresija te njihov omjer (1. stupac : 2. stupac)

vrste skupova	komp. 64 vrijednosti	dvobitna komp.	omjer
Oxford nanopore	6.722	2.049	3.28
Pacific Biosciences	2.935	0.748	3.92
Simulirani skupovi	4.790	1.453	3.30

* Nad svim skupovima očitanja neke vrste.

Iz tablice 4.1 te grafova 4.1 i 4.2 vidimo da, nad stvarnim očitanjima, kompresija prosjeka 64 vrijednosti ima konzistentno veću pogrešku od dvobitne kompresije. Također se jasno vidi da je, neovisno o korištenoj kompresiji, greška manja nad PacBio podatcima nego nad ONT podatcima. Razlog ovoj razlici su različite širine distribucija kvaliteta ONT i PacBio očitanja opisane u potpoglavlju 2.3. Uže originalne distribucije mogu se kvalitetnije sažeti jer inherentno sadrže manje varijacija. Ipak, u svrhu usporedbe učinkovitosti kompresije ovisno o vrsti skupa očitanja na kojem se primjenjuje, možemo usporediti omjere prosječnih grešaka ovih dvaju kompresija. Iz tablice 4.1 možemo zaključiti da korištenje dvobitne kompresije umjesto kompresije 64 vrijednosti rezultira većim porastom točnosti za PacBio očitanja nego za ONT očitanja.

Rezultate nad simuliranim skupovima očitanja možemo vidjeti na grafovima 4.3. Kao i na stvarnim skupovima očitanja, dvobitna kompresija pokazuje konzistentno bolje rezultate od kompresije prosjeka 64 vrijednosti. Prosječne performanse i njihov omjer možemo vidjeti u tablici 4.1.

Iz tablice 4.2 možemo vidjeti da dvobitna kompresija ima manje vrijednosti standardne devijacije pogreške neovisno o vrsti očitanja. To nam govori da je kvaliteta

Tablica 4.2: Standardna devijacija prosječnih grešaka* kompresija

vrsta skupova	komp. 64 vrijednosti	dvobitna komp.
Oxford nanopore	0.113	0.016
Pacific Biosciences	0.013	0.005
Simulirani skupovi	0.086	0.014

* Nad svim skupovima očitanja neke vrste.

dvobitne kompresije konzistentnija od kvalitete kompresije prosjeka 64 vrijednosti.

4.2. Utjecaj na Racon

U ovom potpoglavlju analizirat ćemo rezultate alata *Racon* kada ga se integrira s različitim kompresijama ocjena kvaliteta. Svaki od skupova očitanja opisanih u potpoglavlju 2.3 prvo smo sastavili alatom *Raven*, a zatim smo dobivene sastavljene sljedove dvostruko polirali s tri verzije alata *Racon*. Dotične tri verzije razlikuju se u implementiranim kompresijama kvaliteta:

- dvobitna kompresija,
- kompresija prosjeka 64 vrijednosti i
- nikakva kompresija.

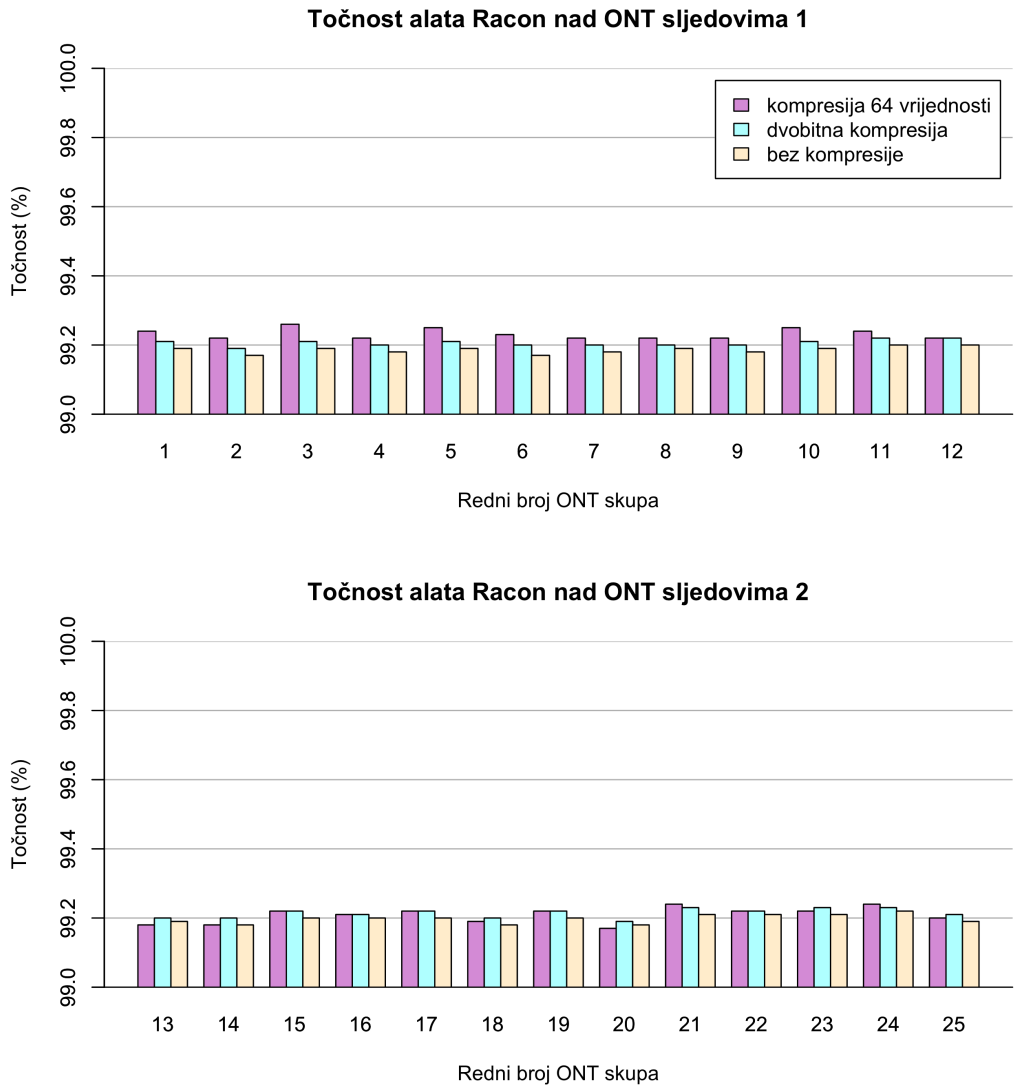
Dakle za svaki skup očitanja generirali smo tri polirana sastavljena slijeda. Usporedbom sličnosti tih sastavljenih sljedova sa referencama evaluirat ćemo kvalitetu korištenih kompresija.

4.2.1. Dnadiff: usporedba s referencama dostupnim iz originalnog skupa podataka

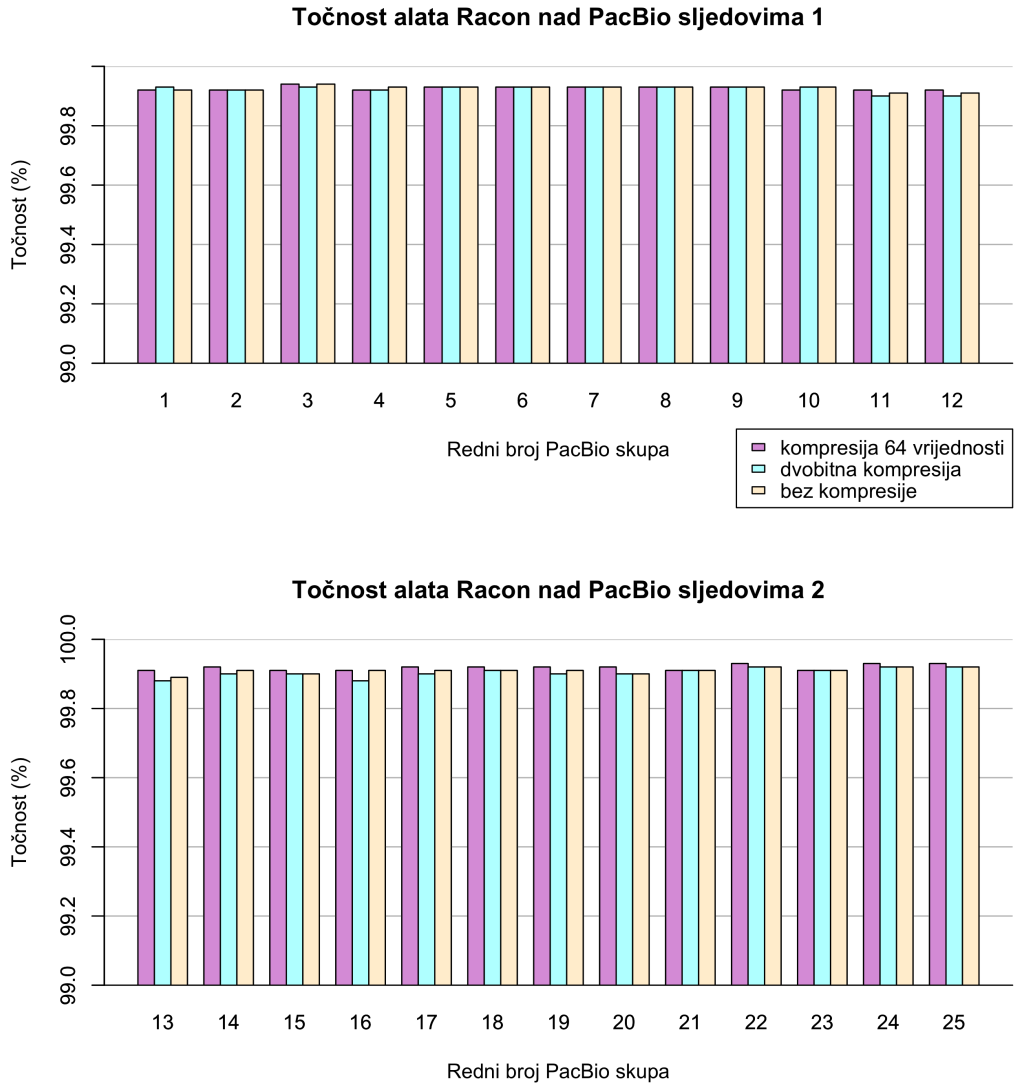
U ovom potpoglavlju polirane sastavljene sljedove uspoređivat ćemo alatom *Dnadiff* s referencama dostupnim u izvornom skupu podataka. Prema [9], reference stvarnih skupova očitanja[1] sastavljene su i polirane Illumina alatima te možemo imati puno povjerenja u njihovu točnost. Kako nam je za simulirane skupove očitanja polazni slijed poznat, njega koristimo kao referencu. Alat *Dnadiff* dat će nam ocjenu sličnosti opisanu formulom 2.2 između ispitivanog slijeda i reference. Tu ocjenu zvat ćemo točnost sastavljenog slijeda. Rezultate opisane analize prikazani su na grafovima[2] 4.4, 4.5 i 4.6.

[1] Oxford Nanopore i Pacific Biosciences

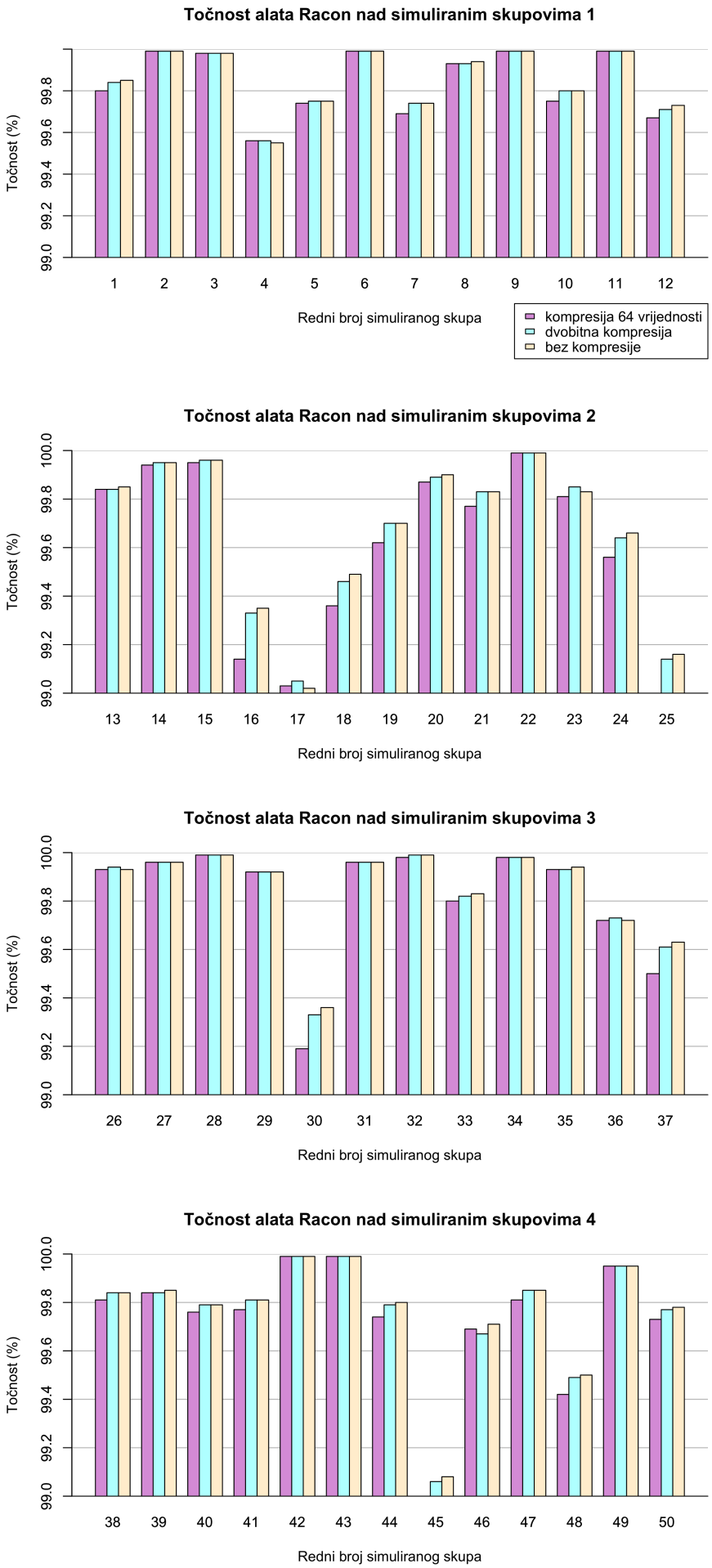
[2] Y os svih grafova jest podrezana te prikazuje raspon [99,100].



Slika 4.4: Točnost ONT sljedova poliranih alatom Racon pri usporedbi s referencom iz izvornog skupa podataka.



Slika 4.5: Točnost PacBio sljedova poliranih alatom Racon pri usporedbi s referencom iz izvornog skupa podataka.



Slika 4.6: Točnost simuliranih sljedova poliranih alatom Racon*

* Nekoliko stupaca čija je vrijednost manja od 99.0 nisu prikazani.

Tablica 4.3: Prosječna točnost* (%) sljedova poliranih *Racon*-om ovisno o kompresiji

vrsta skupova	komp. 64 vrijednosti	dvobitna komp.	bez kompresije
Oxford nanopore	99.220	99.210	99.192
Pacific Biosciences	99.922	99.913	99.917
Simulirani skupovi	99.743	99.778	99.783

* Nad svim skupovima očitanja neke vrste.

Iz opaženih iznosa točnosti nad stvarnim skupovima očitanja možemo zaključiti da su smetnje – uzrokovane različitim postupcima generiranja referenci i *Racon*-om poliranih sljedova – velike. To se očito vidi iz činjenice da su sljedovi polirani bez kompresije u prosjeku postigli nižu točnost od onih poliranih s kompresijom prosjeka 64 vrijednosti nad stvarnim skupovima podataka. Naime, reference s kojima uspoređujemo sastavljene sljedove dobivene su drukčijim alatima od onih koje smo koristili u ovom radu. Inherentne razlike tih alata unose šum od kojeg je teško razaznati koje razlike su uzrokovane kompresijom a koje su uzrokovane razlikama između alata. Te smetnje ometaju nas u procjeni primjenjivosti kompresija, pa ne možemo zaključiti ništa više od toga da dvobitna kompresija nije povećala sličnost rezultata alata *Racon* referencama u odnosu na trenutnu implementaciju.

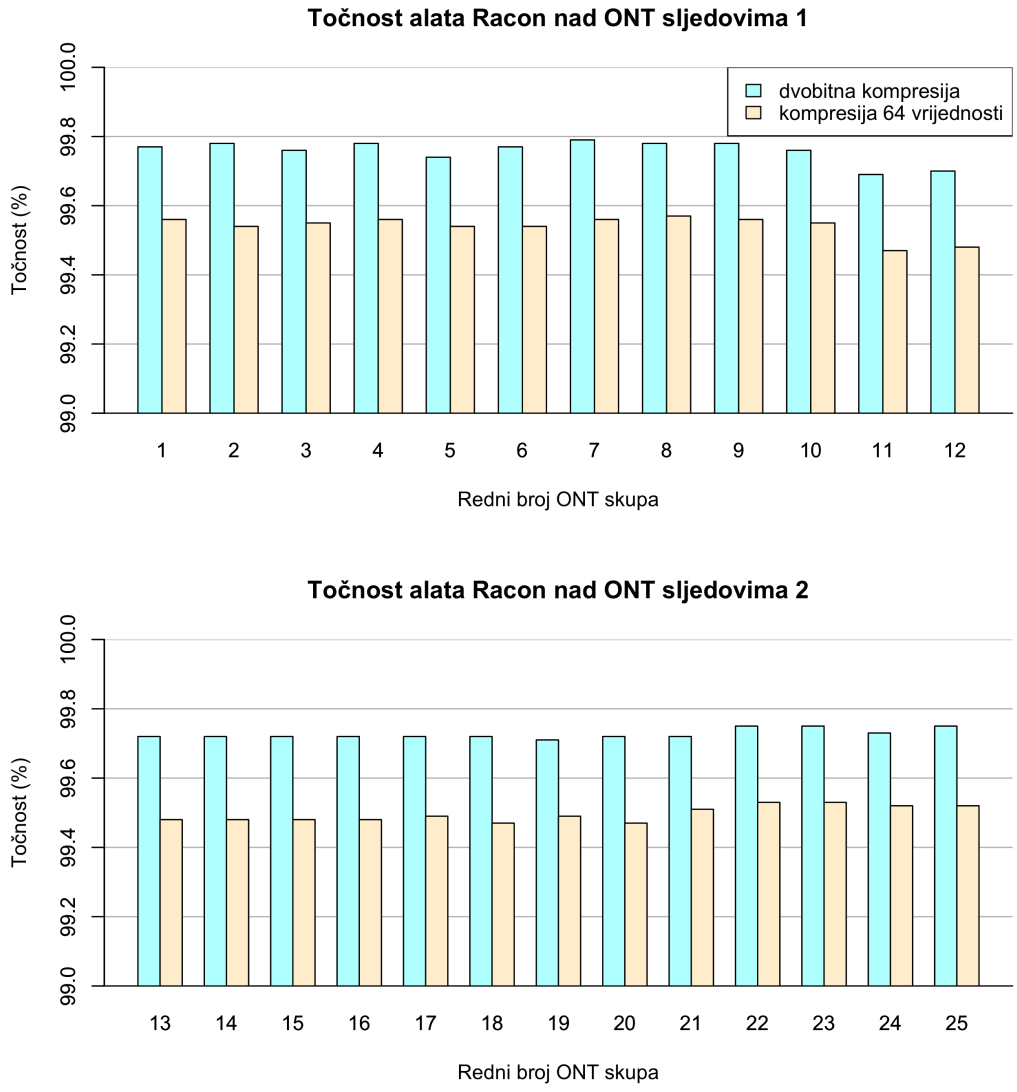
U slučaju PacBio skupova očitanja točnosti svih kompresija puno su bliže 100%. Mogući uzrok tome su uži raspon te niži prosjek kvaliteta. Uži raspon kvaliteta znači da bilo koja kompresija manje griješi, pa bi to moglo značiti da sve tri kompresije pružaju zadovoljavajuće rezultate. S druge strane, s obzirom da niske kvalitete signaliziraju visoku nepouzdanost moguće je da alati zanemaruju kvalitete ispod nekog iznosa, ili da one imaju vrlo malo utjecaja. Pod tom pretpostavkom svejedno je jesmo li izgubili neke informacije sažimanjem jer tako niske ocjene kvaliteta ne utječu na rad alata.

Performanse alata *Racon* više su raznolike nad simuliranim skupovima podataka, što je i očekivano s obzirom na njihovu heterogenost. Nad ovim skupovima očitanja eliminiran je efekt smetnje zbog korištenja različitih alata pri dobivanju reference pa jasnije možemo procijeniti sam utjecaj kompresije. Na grafu 4.6 može se primijetiti blaga ali konzistentna prednost algoritma dvobitne kompresije naspram kompresije prosjeka 64 vrijednosti. Štoviše, u mnogo slučaja dvobitna kompresija daje jednako dobre rezultate kao i kada se ne koristi nikakva kompresija.

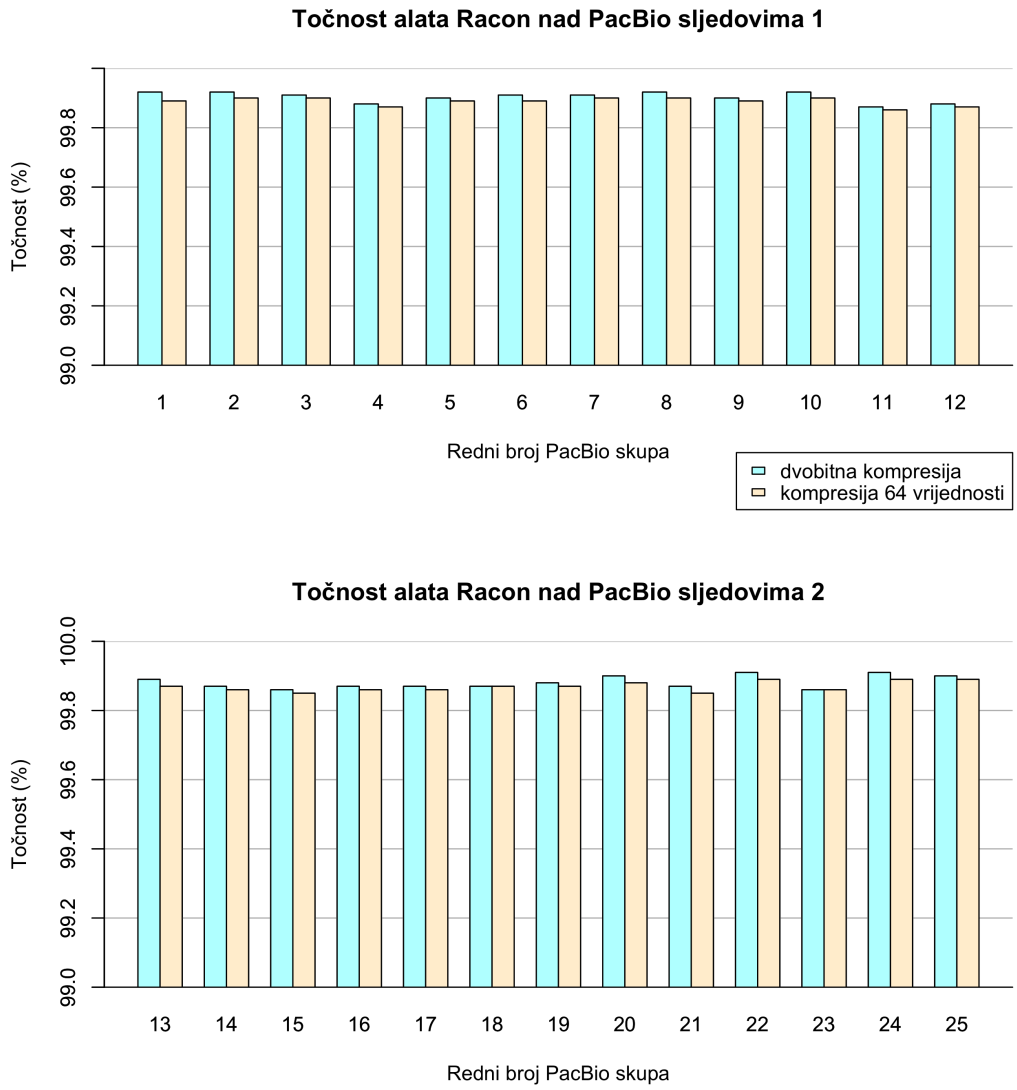
4.2.2. Dnadiff: usporedba s sljedovima poliranim bez kompresije

Evaluacija performansi alata *Racon* nad stvarnim skupom podataka pokazala se problematičnom jer zapravo ne mjerimo samo utjecaj kompresije već i razlike u drukčijim

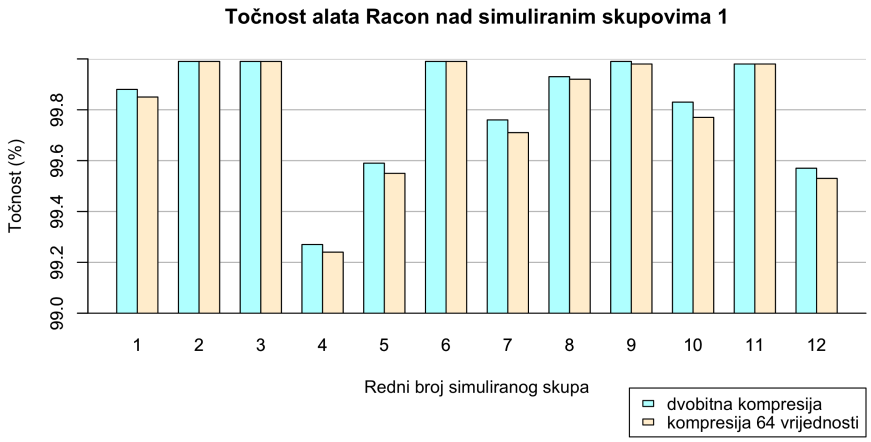
postupcima sastavljanja i poliranja. Razlog tome je uporaba reference generirane različitim alatima od slijeda kojeg ispitujemo. U ovom potpoglavlju ponovit ćemo istu analizu, ali ćemo kao reference koristiti sljedove polirane alatom *Racon* bez kompresije. Pretpostavka takvog pristupa jest da ta verzija alata *Racon* daje najbolje rezultate pa mjereći sličnost toj verziji možemo procijeniti dobrotu ostalih verzija. Na grafovima 4.7, 4.8 i 4.9 prikazana je mjera točnosti sljedova poliranih s dvobitnom kompresijom te kompresijom 64 vrijednosti kada ih se usporedi sa sljedovima koji su polirani bez kompresije.

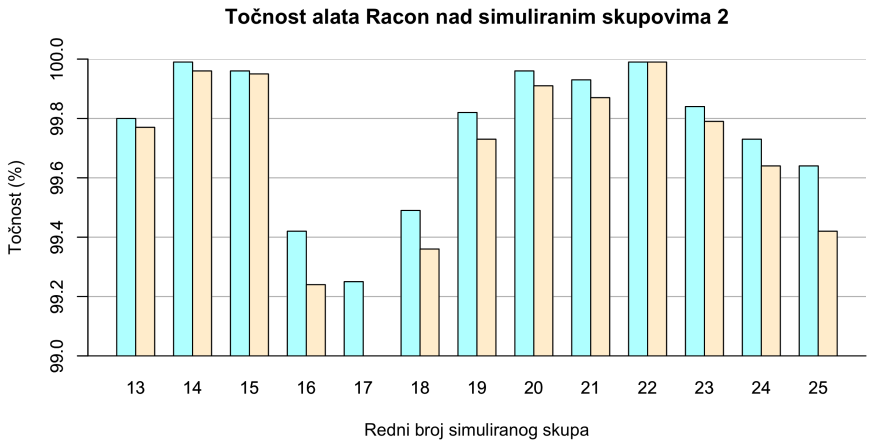


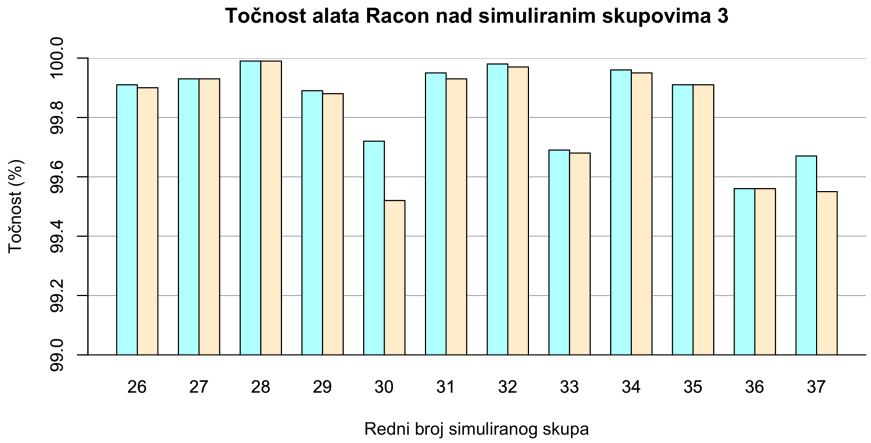
Slika 4.7: Točnost ONT sljedova poliranih alatom Racon pri usporedbi s sljedovima poliranim bez kompresije.

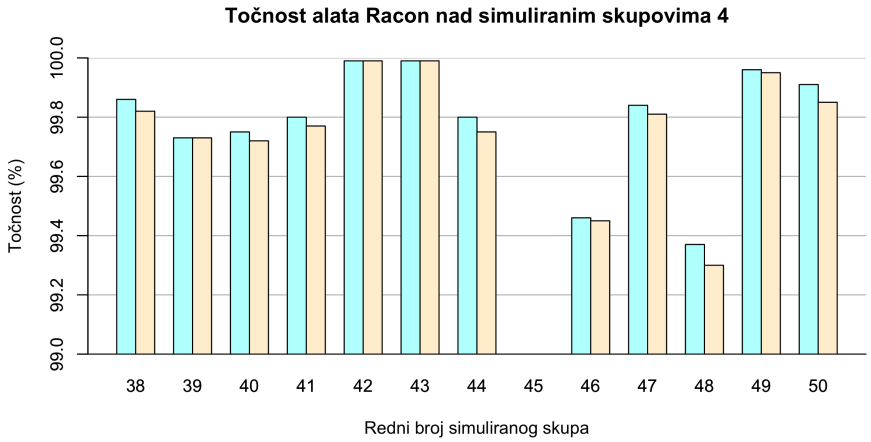


Slika 4.8: Točnost ONT sljedova poliranih alatom Racon pri usporedbi s sljedovima poliranim bez kompresije.









Slika 4.9: Točnost simuliranih sljedova poliranih alatom Racon pri usporedbi s sljedovima poliranim bez kompresije.*

* Nekoliko stupaca čija je vrijednost manja od 99.0 nisu prikazani.

Tablica 4.4: Prosječna točnost* (%) sljedova poliranih *Racon*-om ovisno o kompresiji

vrsta skupova	komp. 64 vrijednosti	dvobitna komp.
Oxford nanopore	99.517	99.742
Pacific Biosciences	99.878	99.892
Simulirani skupovi	99.730	99.780

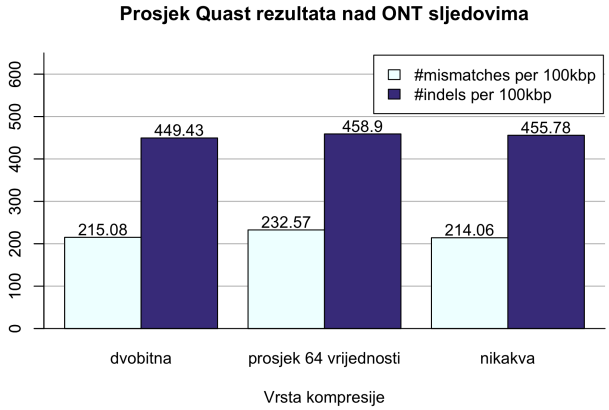
* Nad svim skupovima očitanja neke vrste.

Ovakva analiza konzistentno pokazuje dvobitnu kompresiju kao poboljšanje u odnosu na kompresiju prosjeka 64 vrijednosti, ali očekivano poboljšanje ovisi o vrsti podataka. Iz tablice 4.4 vidimo da nad ONT skupom podataka očekivano poboljšanje točnosti jest 0.225% odnosno 0.014% nad PacBio skupom podataka. Mogući uzrok toj razlici jest generalno lošija kvaliteta PacBio očitanja zbog koje se možda pridodaje manje važnosti samim ocjenama kvalitete. Drugim riječima, ako su u procesu poliranja ključne baze s visokom ocjenom kvalitete moguće je da PacBio očitanja imaju preniske kvalitete da bi se provelo efektivno poliranje. U tom slučaju, različitim kompresijama se ne bi mogao poboljšati rezultat poliranja te bi taj rezultat bio približno isti neovisno o kompresiji. U prilog ovoj ideji ide činjenica da su u ovom i prethodnom poglavlju 4.2.2 točnosti PacBio sljedova konzistentno bile vrlo blizu 100% neovisno o kompresiji te neovisno o izboru reference. To ukazuje na to da se možda u kvaliteti poliranja takvih sljedova dosegla granica koja nije povezana s kompresijom ocjenama kvaliteta.

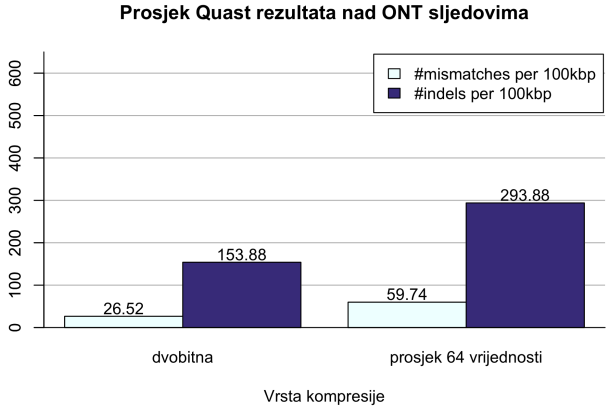
Dvobitna kompresija pokazala se konzistentno boljom od kompresije 64 vrijednosti i nad simuliranim podatcima, no s mnogo više varijacije nego nad stvarnim. Takav rezultat nije neobičan s obzirom na raznoliku prirodu simuliranih očitanja. U prosjeku, nad simuliranim skupom podataka, možemo očekivati poboljšanje točnosti od 0.050% pri korištenju dvobitne kompresije umjesto kompresije prosjeka.

4.2.3. Rezultati alata Quast

U ovom potpoglavlju razmotrit ćemo rezultate alata *Quast* nad poliranim sljedovima. Za svaki slijed izračunat ćemo mjere *mismatches per 100kbp* i *indels per 100kbp*, a zatim promatrati prosjek tih mjera za različite vrste sljedova.



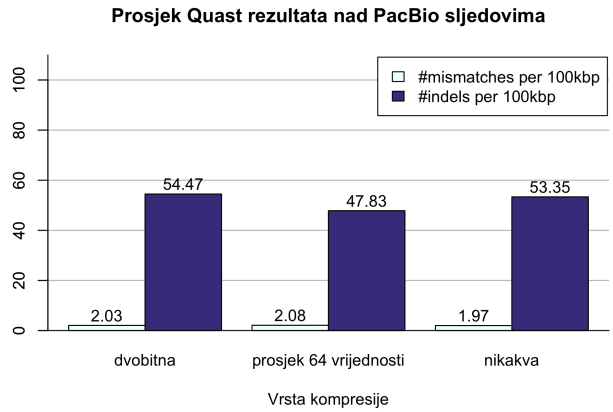
Slika 4.10: Quast rezultati nad ONT podatcima pri usporedbi s referencom iz izvornog skupa podataka.



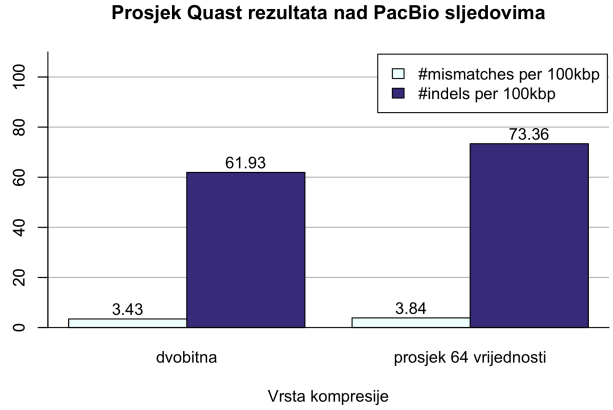
Slika 4.11: Quast rezultati nad ONT podatcima pri usporedbi s sljedovima poliranim bez kompresije.

Rezultati grafova 4.10 i 4.11 potvrđuju prijašnje zaključke iz potpoglavlja 4.2.1 i 4.2.2. U slučaju usporedbe sljedova s referencama dostupnim iz izvornog skupa podataka ne možemo previše zaključiti jer se ni jedna vrsta kompresije ne ističe kao očito najbolja. To se vidi iz vrlo sličnih iznosa *mismatch* i *indel* mjera na slici 4.10. U sljedećem grafu 4.11 vidimo iznose istih mjera za iste sljedove, ali kada se kao reference koriste sljedovi polirani alatom *Racon* bez kompresije. Ukupan broj grešaka[3] znatno je manji za obje vrste kompresije što se može objasniti eliminacijom smetnji uzrokovanih drukčijim alatima. Osim toga, jasno se vidi da dvobitna kompresija postiže manji broj grešaka.

[3]Zbroj mjere *indels* i *mismatches*.

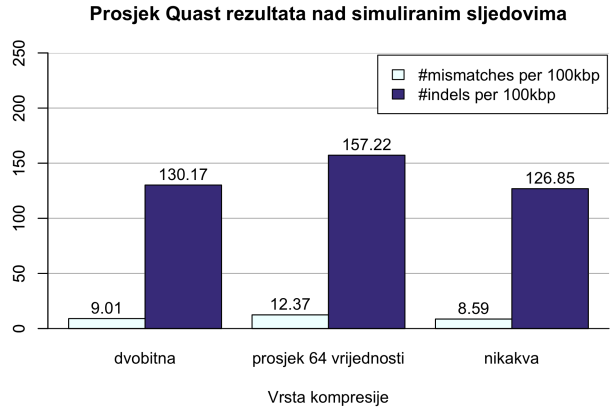


Slika 4.12: Quast rezultati nad PacBio podatcima pri usporedbi s referencom iz izvornog skupa podataka.

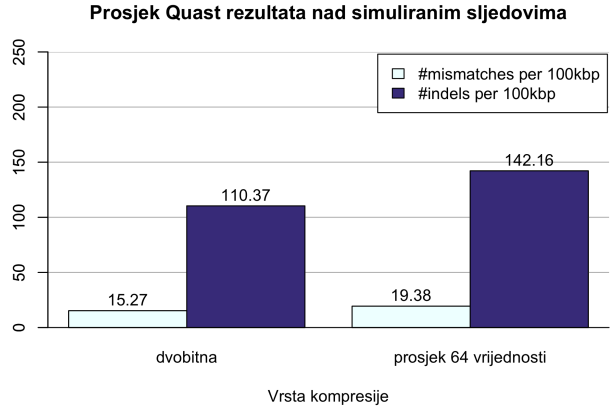


Slika 4.13: Quast rezultati nad PacBio podatcima pri usporedbi s sljedovima poliranim bez kompresije.

Iz rezultata alata *Quast* i dalje ne možemo jasno odabrati bolju kompresiju za PacBio podatke. U slučaju uporabe reference iz izvornog skupa podataka algoritam prosjeka 64 vrijednosti pokazuje blagu prednost naspram dvobitne kompresije te nikakve kompresije. Međutim, taj se poredak obrće kada za referencu rabimo sljedove polirane bez kompresije. U oba slučaja razlike iznosa grešaka manje su u usporedbi s ONT i simuliranim podatcima.



Slika 4.14: Quast rezultati nad simuliranim podatcima pri usporedbi s referencom iz izvornog skupa podataka.



Slika 4.15: Quast rezultati nad simuliranim podatcima pri usporedbi s sljedovima poliranim bez kompresije.

Quast rezultati nad simuliranim podatcima potvrđuju bolje performanse dvobitne kompresije od kompresije 64 vrijednosti neovisno o korištenoj referenci. Ukupan iznos grešaka manji je pri korištenju sljedova poliranih bez kompresije kao reference, što nije iznenađujuće jer se na taj način ne registriraju pogreške nastale prilikom sastavljanja.

5. Zaključak

Problem efikasne pohrane niza nukleotida riješen je dvobitnim kodiranjem svake baze što ostvaruje kompresiju niza nukleotida s faktorom 4.00.

Teži problem predstavljaju ocjene kvaliteta, no uporabom dvobitne kompresije predstavljene u ovom radu, prostor koji one zauzimaju može se smanjiti 3.88 puta. Vremenska složenost dotičnog rješenja je jednaka trenutnoj implementaciji kolekcije *Biosoup*. Dakle dvobitna kompresija ne gubi svojstvo efikasnog[1] dohvata proizvoljne kvalitete te se samo sažimanje vrijednosti obavlja u linearnom vremenu. Ipak, cijena smanjene potrebe za memorijom jest gubitak informacija pa se nizovi kvaliteta sažeti dvobitnom kompresijom u određenoj mjeri razlikuju od polaznih podataka. Konkretno, možemo očekivati prosječnu apsolutnu grešku od 2.049, 0.748 i 1.453 za ONT, PacBio te simulirana očitanja korištena u ovom radu. Te brojke, u donosno na trenutnu implementaciju kolekcije *Biosoup* predstavljaju poboljšanje od tri do četiri puta.

Usprkos većoj količini očuvanih informacija, rezultati analize utjecaja dvobitne kompresije na točnost alata *Racon* su mješoviti. Kada se rezultati alata *Racon* evaluiraju na temelju sličnosti s referencama generiranih Illumina alatima, prednost dvobitne kompresije[2] može se uočiti samo za simulirane skupove podataka. Nadalje, razlike u performansama nad PacBio podatcima su toliko male da se ne može pouzdati u njihovu značajnost. Ipak, pri uporabi sljedova poliranih alatom *Racon* bez kompresije kao reference, dvobitna kompresija konzistentno ostvaruje bolje rezultate.

Unatoč dobroj ravnoteži smanjenja potrebne memorije te izgubljenih informacija koju dvobitna kompresija postiže, rezultati ne ukazuju na njenu bezuvjetnu primjenjivost. Iako se njenom primjenom štedi na potrebnoj memoriji, nije jasno pruža li veću prednost u odnosu na druge vrste kompresije[3]. Za bolju procjenu njene korisnosti potrebno ju je ispitati na širem skupu podataka pogotovo za PacBio tip očitanja.

[1] $\mathcal{O}(1)$

[2] U usporedbi s trenutnom implementacijom kolekcije *biosoup*.

[3] Poput kompresije prosjeka 64 vrijednosti.

LITERATURA

[1] Peter J. A. Cock, Christopher J. Fields, Naohisa Goto, Michael L. Heuer, i Peter M. Rice. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Research*, 38(6): 1767–1771, 12 2009. ISSN 0305-1048. doi: 10.1093/nar/gkp1137. URL https://doi.org/10.1093/nar/gkp1137.

[2] Brent Ewing i Phil Green. Base-calling of automated sequencer traces using phred. ii. error probabilities. *Genome research*, 8(3):186–194, 1998.

[3] Alexey Gurevich, Vladislav Saveliev, Nikolay Vyahhi, i Glenn Tesler. QUAST: quality assessment tool for genome assemblies. *Bioinformatics*, 29(8):1072–1075, 02 2013. ISSN 1367-4803. doi: 10.1093/bioinformatics/btt086. URL https://doi.org/10.1093/bioinformatics/btt086.

[4] Inc. Illumina. Understanding Illumina Quality Scores, 2014. URL https://www.illumina.com/content/dam/illumina-marketing/documents/products/technotes/technote_understanding_quality_scores.pdf. Pristupljeno: 2021-05-16.

[5] Guillaume Marçais, Arthur L. Delcher, Adam M. Phillippy, Rachel Coston, Steven L. Salzberg, i Aleksey Zimin. Mummer4: A fast and versatile genome alignment system. *PLOS Computational Biology*, 14(1):1–14, 01 2018. doi: 10.1371/journal.pcbi.1005944. URL https://doi.org/10.1371/journal.pcbi.1005944.

[6] U.S. National Library of Medicine National Center for Biotechnology Information. FASTQ files, 2019. URL https://www.ncbi.nlm.nih.gov/sra/docs/submitformats/#fastq-files. Pristupljeno: 2021-06-02.

[7] Anthony Rhoads i Kin Fai Au. Pacbio sequencing and its applications. *Genomics, Proteomics, Bioinformatics*, 13(5):278–289, 2015. ISSN 1672-0229. doi: https://

doi.org/10.1016/j.gpb.2015.08.002. URL https://www.sciencedirect.com/science/article/pii/S1672022915001345. SI: Metagenomics of Marine Environments.

[8] Laura Rodriguez-Murillo i Rany M. Salem. *Insertion/Deletion Polymorphism*, stranice 1076–1076. Springer New York, New York, NY, 2013. ISBN 978-1-4419-1005-9. doi: 10.1007/978-1-4419-1005-9_706. URL https://doi.org/10.1007/978-1-4419-1005-9_706.

[9] Wick RR i Holt KE. Benchmarking of long-read assemblers for prokaryote whole genome sequencing. *F1000Research*, 2021. doi: https://doi.org/10.12688/f1000research.21782.4. URL https://f1000research.com/articles/8-2138. [version 4; peer review: 4 approved], 8:2138.

[10] J. R. Tyson, N. J. O'Neil, M. Jain, H. E. Olsen, P. Hieter, i T. P. Snutch. MinION-based long-read sequencing and assembly extends the Caenorhabditis elegans reference genome, 2018. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5793790/#:~:text=Notably%2C%20the%20mean%20flow%20cell,elegans%20reference%20genome. Pristupljeno: 2021-05-16.

[11] Robert Vaser. Racon, 2021. URL https://github.com/lbcb-sci/racon/blob/master/README.md. Pristupljeno: 2021-06-02.

[12] Robert Vaser i Mile Šikić. Raven: a de novo genome assembler for long reads. *bioRxiv*, 2021. doi: 10.1101/2020.08.07.242461. URL https://www.biorxiv.org/content/early/2021/02/22/2020.08.07.242461.

[13] Robert Vaser, Ivan Sović, Niranjan Nagarajan, i Mile Šikić. Racon-rapid consensus module for raw de novo genome assembly of long uncorrected reads. *Bioinformatics*, 18:452–464, 2002.

[14] Aaron M Wenger, Paul Peluso, William J Rowell, Pi-Chuan Chang, Richard J Hall, Gregory T Concepcion, Jana Ebler, Arkarachai Fungtammasan, Alexey Kolesnikov, Nathan D Olson, et al. Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. *Nature biotechnology*, 37(10):1155–1162, 2019.

Struktura podataka za efikasno spremanje očitanja dobivenih sekvenciranjem genoma

Sažetak

Zbog golemih duljina genoma, njihova bilo kakva bioinformatička obrada vrlo je memorijski zahtjevna. Općenito, efikasna pohrana ocjena kvaliteta (engl. *Phred quality scores*) u radnu memoriju predstavlja veći izazov od pohrane samih sljedova pa se prirodno nameće pitanje njihove kompresije. U ovom radu predstavljamo algoritam kompresije ocjena kvaliteta genetskih sljedova koji smanjuje memoriju potrebnu za njihovu pohranu gotovo pa četiri puta. To postiže sažimanjem svake ocjene kvalitete s dva bita umjesto osam te računanjem povoljnog preslikavanja dvobitnih kôdova natrag u ocjene kvalitete. Također u radu ispitujemo količinu informacije koju kompresija gubi nad stvarnim i simuliranim podatcima te utjecaj njene primjene na točnost alata *Racon*.

Ključne riječi: bioinformatika, ocjena kvalitete, kompresija, sažimanje, FASTQ format

Data Structure for Efficient Storage of Genome Sequencing Data

Abstract

Due to the tremendous lengths of nearly all genomes, any computational attempt at their analysis is bound to be very memory intensive. Generally, Phred quality scores prove to be more difficult to store efficiently than genetic sequences, making them prime candidates for compression. This paper presents one such compression algorithm that reduces storage needed for Phred quality scores almost four times. This is achieved by using two bits, for representing a single quality score, instead of eight and by calculating satisfactory mappings of two-bit codes to quality scores. Furthermore, we determine the amount of information lost by our algorithm and how this affects the accuracy of the tool *Racon*.

Keywords: bioinformatics, Phred quality scores, compression, FASTQ format