Vizualizacija "Runs of Homozygosity" DNA segmenata

Hrženjak, Marko

Master's thesis / Diplomski rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Agriculture / Sveučilište u Zagrebu, Agronomski fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:204:059070

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-28**



Repository / Repozitorij:

Repository Faculty of Agriculture University of Zagreb







SVEUČILIŠTE U ZAGREBU
AGRONOMSKI FAKULTET

Vizualizacija „Runs of homozygosity” DNA segmenata

DIPLOMSKI RAD

Marko Hrženjak

Zagreb, rujan, 2018.

SVEUČILIŠTE U ZAGREBU
AGRONOMSKI FAKULTET

Diplomski studij:
Genetika i oplemenjivanje životinja

Vizualizacija „Runs of homozygosity" DNA segmenata

DIPLOMSKI RAD

Marko Hrženjak

Mentor: doc. dr. sc. Maja Ferenčaković

Zagreb, rujan, 2018.

SVEUČILIŠTE U ZAGREBU

AGRONOMSKI FAKULTET

IZJAVA STUDENTA

O AKADEMSKOJ ČESTITOSTI

Ja, **Marko Hrženjak**, JMBAG 0178088111, rođen/a 29.3.1993. u Zagrebu, izjavljujem da sam samostalno izradila/izradio diplomski rad pod naslovom:

Vizualizacija „Runs of homozygosity" DNA segmenata

Svojim potpisom jamčim:

- da sam jedina autorica/jedini autor ovoga diplomskog rada;
- da su svi korišteni izvori literature, kako objavljeni tako i neobjavljeni, adekvatno citirani ili parafrazirani, te popisani u literaturi na kraju rada;
- da ovaj diplomski rad ne sadrži dijelove radova predanih na Agronomskom fakultetu ili drugim ustanovama visokog obrazovanja radi završetka sveučilišnog ili stručnog studija;
- da je elektronička verzija ovoga diplomskog rada identična tiskanoj koju je odobrio mentor;
- da sam upoznata/upoznat s odredbama Etičkog kodeksa Sveučilišta u Zagrebu (Čl. 19).

U Zagrebu, dana _____ _____

Potpis studenta / studentice

SVEUČILIŠTE U ZAGREBU
AGRONOMSKI FAKULTET

IZVJEŠĆE

O OCJENI I OBRANI DIPLOMSKOG RADA

Diplomski rad studenta/ice **Marko Hrženjak**, JMBAG 0178088111, naslova

Vizualizacija „Runs of homozygosity" DNA segmenata

obranjen je i ocijenjen ocjenom _____ , dana _____ .

Povjerenstvo: potpisi:

1. doc.dr.sc Maja Ferenčaković mentor _____
2. izv. prof. dr. sc. Vlatka Čubrić Čurik član _____
3. prof. dr. sc. Ino Čurik član _____

Sadržaj

1. Uvod 1

1.1. Cilj rada 2

2. Pregled literature 3

2.1. Runs of Homozygosity 3

2.1.1. Povijest i primjena 3
2.1.2. Mehanizam za pojavljivanje ROH segmenata u genomu 5
2.1.3. ROH podaci i identificiranje 5

2.2. Programski paketi za obradu podataka 7

2.2.1. PLINK 7
2.2.2. cgaTOH 8
2.2.3. Python 9
2.2.4. Anaconda i Spyder 10

3. Materijali i metode 11

3.1. Podaci 11

3.2. Kontrola kvalitete 11

3.3. Detekcija ROH segmenata 11

3.4. Vizalizacija ROH segmenata 11

3.5. Opis koda 12

4. Rezultati i rasprava 21

5. Zaključak 26

6. Popis literature 27

7. Prilozi 29

7.1. Prilog 1 29

8. Životopis 36

Sažetak

Diplomskog rada studenta/ice **Marko Hrženjak**, naslova

Vizualizacija „Runs of homozygosity“ DNA segmenata

„Runs of homozygosity” su segmenti/cijele genomske regije gdje se identični haplotipovi nasljeđuju od svakoga roditelja. Od njihovog otkrića zbog tehnološkog napretka u 90-ima, „Runs of homozygosity” su otkrili podosta o podrijetlu genoma jedinke, dešifrirajući genetsku bazu monogenskih i kompleksnih svojstava i bolesti. Kako tehnologija napreduje tako se i povećava zahtjev za novim metodama analize SNP podataka. Mapiranje ROH se može provesti kroz više načina i svaki ima svoje prednosti i mane. U ovom radu će se predstaviti metoda analiziranja, čišćenja i vizualni prikaz kao krajnji rezultat u Pythonu.

Ključne riječi: „Runs of Homozygosity“, Python, detekcija ROH, vizualizacija

SUMMARY

Of the master's thesis - student **Marko Hrženjak,** entitled

Visualization of "Runs of Homozygosity" DNA segments

"Runs of homozygosity" are segments or whole genomic regions where the identical haplotypes are inherited from each parent. Since their discovery because of the technological advances in late 90s, "Runs of homozygosity" shed some light on about individual genome origin, decoding the genetic base of monogenic and complex traits and diseases. As technology progresses, a demand for new methods of analysis of SNP data rise as well. Mapping of ROH segments can be performed in many was and every method has its own strengths and flaws. In this paper a method will be shown for analysis, data cleaning, and visual representation as a result in Python.

Keywords: Runs of homozygosity, Python, detection of ROH, vizualisation

1. Uvod

Inbreeding depresija je dobro poznati fenomen u svijetu genetike kao problematična pojava u organizmima. Efekt inbreeding depresije slovi kao produkt međusobnog parenja među srodnim jedinkama u populaciji.

Prije konkretnih znanstvenih rezultata, efekti bili su široko prepoznati u prošlosti. U ljudskoj populaciji oko 42% potomstva od sestra-brat brakova bi umrlo prije nego što bi dospjeli do reproduktivne dobi, stoga je puno kultura razvilo sistem koji je strogo kontrolirao brakove kako bi se izbjegao incest.

Generalno, što je veća genetska varijacija ili "gene pool" unutar populacije, manje je vjerojatno da će se dogoditi efekt inbreeding depresije. "Runs of Homozygosity" ili "ROH" su dugačke linije homozigotnih segmenata koji odražavaju autozigotnost i njegovu starost. Za ROH segmente vrijedi da su to kontinuirani i neprekidni dijelovi DNA sekvence bez heterozigotnosti u diploidnom stanju. Njihova pojava genomu jedinke može se široko objasniti nasljeđivanjem istog kromosomskog segmenta od oba roditelja, koji su naslijedili taj specifičan segment od zajedničkog pretka. Te segmente prekidaju rekombinacijski događaji, stoga ako su ostali vrlo dugački, vrlo vjerojatno su nastali od nedavnog zajedničkog pretka. Što se tiče kraćih segmenata, moguće je da su nastali od daljnjih predaka. Putem duljine segmenta lako se može procijeniti vrijeme do zajedničkog pretka.

Razvitak "high-density, genome-wide SNP" čipova probudio je interes za izračunavanje inbreeding koeficijenata jedinki iz molekularnih informacija kako bi se izbjegli nedostaci korištenja podataka iz rodovnika za istu svrhu.

Kao što je i tekao razvitak SNP čipova tako je i rasla potražnja za boljom kontrolom kvalitete, detekcijom ROH segmenata, obrađivanjem podataka i vizualizacijom istih. Također valja spomenuti i uporabu najpopularnijih programa do sada za detekciju „ROH“, PLINK, GERMLINE, i BEAGLE. Izrađivanje različitih paketa u programskim jezicima koji su specijalizirani za obrađivanje podataka ROH najviše je zastupljeno u programskom jeziku R, no nije ni Python daleko, štoviše postaje sve popularniji kroz godine u znanstvenoj zajednici(Spyder, scyPY).

1.1. Cilj rada

Cilj ovog rada je stvaranje vlastitog „pipeline“-a koji će sam odrađivati poslove čišćenja, pokretati programske pakete za rekodiranje i stvaranje novih datoteka za daljnju analizu te detekciju ROH segmenata, za kojom slijedi vizualiziranje.

2. Pregled literature

2.1. Runs of Homozygosity

2.1.1. Povijest i primjena

Broman i Weber (1999) su bili prvi koji su uočili dugačke linije homozigotnih segmenata u ljudskim genomima. Pretpostavili su da su ti segmenti vrlo vjerojatno odraz autozigotnosti i da su možda imali nekakvog utjecaja na ljudsko zdravlje. Kasnije, Gibson i sur. dalje analiziraju to otkriće homozigotnih segmenata, osvrćući se na njihovu duljinu, brojeve i distribuciju u naizgled nesrodnim HapMap populacija. U cijeloj sekvenci genoma, definirali su ih kao kontinuirane i neprekidne dijelove DNA sekvence bez heterozigotnosti u diploidnom stanju. Autozigotna priroda tih segmenata je bila pretpostavljena zbog rekombinacijskih događaja koji su prekidaju dugačke kromosomske segmente, stoga ako bi ostali vrlo dugački, vrlo vjerojatno bi bili nastali od nedavnog zajedničkog pretka. Što se tiče kraćih segmenata, moguće je da su nastali od daljnjih predaka, ali i oni isto sadrže neke segmente koji nisu identični po porijeklu (engl. Identical by Descent - IBD).

Lencz i sur. (2007.) potvrdili su predviđanje Bromana i Webera (1999.) tako što su pokazali da takav homozigotni segment može biti sistematski upotrijebljen za mapiranje gena koji su povezani sa bolestima poput šizofrenije. Isto tako, oni su prvi predstavili naziv „Runs of Homozygosity" (ROH), definirajući ga kao „prozor" sa više od 100 uzastopnih SNP-ova na jednom kromosomu bez heterozigotnog dijela. Ideja autozigotnog mapiranja, nastala je puno ranije. Lander i Botstein (1987.), predstavili su metodu izbora za otkriće autosomalnog recesivnog genskog lokusa tako što su tražili obližnji homozigotni RFLP („Restriciton fragment length polymorphism) u srodnim populacijama. Što je veći bio broj bolesnih jedinki koje su dijelile homozigotnu regiju i što je veća bila regija, više je bilo moguće da sadržavati mutaciju koja uzrokuje bolest (Woods i sur., 2006.).

Iako ROH imaju dosta široku primjenu, standardi za ROH definiciju i identifikaciju predstavljaju ozbiljnu pristranost u ROH istraživanjima. Howrigan i sur. (2011.) zaključili su da broj i veličina ROH-ova koji su identificirani u genotipskim podacima mogu uvelike ovisiti o specifičnim parametrima i pragovima zadanim tijekom SNP analize. Nadalje, čišćenje SNP-ova koji pokazuju niske frekvencije minor alela (engl. Minor allele frequency - MAF), onih koji bi mogli odstupati od Hardy Weinberg ekvilibrijuma (HWE), također mogu i pokazivati visoki

disekvilibrijum vezanosti gena (engl. Linkage Disequilibrium - LD), što bi uvelike utjecalo na rezultate (Wigginton i sur., 2005.; Albrechsten i sur., 2010.).

Kada se upotrebljavaju podaci sa SNP chip-ova, vrlo je poznato da su svi od tih markera polimorfni u vrsti od interesa (goveda, ovce, koze, ljudi i dr.). U asocijacijskim studijama cijelog genoma (engl. Genome-Wide association studies - GWAS) izuzeće SNP-ova koji su monomorfni u promatranoj populaciji je razuman način smanjenja vremena potrebnog za izračunavanje na neinformativnim (bez varijacije) markerima dok u ROH analizama izbacivanje fiksiranih SNP-ova predstavlja gubitak informacija. Fiksacija je posljedica inbreedinga i male efektivne veličine populacije stoga u slučaju autozigotnosti i onda fiksirani SNP-ovi trebaju ostati u podacima kako bi dali realniju sliku na stanje populacije interesa. Isto se može i reći za LD čišćenje.

U „GWAS" istraživanjima, povezani aleli uzrokuju produljivanje vrijeme potrebno za izračunavanje i ne pridonose boljim i preciznijim rezultatima, ali u ROH analizama oni ne uzrokuju toliko problema jer se smatraju kao lokalni fenomen koji proizvodi samo kratke homozigotne segmente. Čak kada bi veliki ROH segmenti bili prisutni zbog LD-a, po svojoj prirodi oni su i dalje autozigotni i nema potrebe za njihovim isključenjem. Devijacija od HWE se pojavljuje u populaciji koja je pod utjecajem selekcije. Kada pričamo o domaćim životinjama, većina je bila ili još uvijek je pod velikom selekcijom, stoga devijacije su za očekivati (Ćurik i sur., 2014.). Pristup za kontrolu kvalitete za identifikaciju ROH segmenata trebalo bi drugačiji nego kontrola kvalitete za „GWAS" istraživanja. Nažalost u većini ROH istraživanja očito je da se kopiraju metode iz „GWAS" analize.

Alat koji se upotrebljava za njihovo određivanje može utjecati na identifikaciju ROH segmenata. Purfield i sur. (2012), te Ferenčaković i sur. (2013.) usporedili su procjene koje su dobili upotrebljavajući dva SNP-čipa koji se najviše upotrebljavaju u govedima: „Ilumina BovineSNP50 Genotyping BeadChip" sa 54 001 SNP-ova(50K) i „Illumina BovideHD Genotyping BeadChip" sa 777 972 SNP-ova (HD – High Density). Zaključili su da je 50k čip prikladan samo za identificiranje ROH dužih od 5 Mb (Purfield i sur., 2012.), odnosno 4Mb (Ferenčaković i sur., 2013). Svakako, analize koje su bazirane na čipovima manje gustoće mogu biti kompromitirane zbog neotkrivenih heterozigotnih SNP genotipa koji su prisutni u promatranom ROH-u. Frekvencije pogrešaka u genotipiziranju SNP-ova su još jedan faktor koji može utjecati na ROH bazirane procjene autozigotnosti. Kako ova frekvencija obično varira između 0.2% i 1.0%, može utjecati na identifikaciju vrlo dugačkog ROH-a koji sadržava

brojne SNP-ove. Bilo kakva greška pri genotipiziranju, bio homozigot u heterozigota ili obrnuto, može utjecati na određivanje ROH segmenata. Potencijalno rješenje je da se dopusti određen broj SNP-ova da budu heterozigoti (Ferenčaković i sur., 2013).

2.1.2. Mehanizam za pojavljivanje ROH segmenata u genomu

Pojava ROH segmenata u genomu jedinke može se široko objasniti nasljeđivanjem istog kromosomalnog segmenta od oba roditelja, koji su naslijedili taj specifičan segment od zajedničkog pretka. Gibson i sur. (2006.) predložili su dva glavna mehanizma za pojavu homozigotnih regija u genomu potomaka. Prvi je roditelji imaju relativno bliskog zajedničkog pretka, stoga je bilo jako malo prilika za rekombinaciju kako bi se segment preinačio. Drugi mehanizam je taj da je veza između roditelja je daleka, ali nedostaje rekombinacije u regiji zbog visokog LD-a, stoga je segment ostao netaknut.

Vrlo je važno istaknuti to da su oba mehanizma smatrana funkcijom srodnosti i posljedica parenja u srodstvu većeg (nedavnog) ili manje (davnog) nivoa. Nije neobično da su homozigotni segmenti rezultat LD-a prozvani autozigotnima jer i jesu autozigotni po prirodi. Broman i Weber (1999.) i Gibson i sur. (2006.) prepoznali su da će LD kao lokalni fenomen, prouzročiti samo kratke homozigotne segmente koji su duži od 1 Mb i ne mogu biti objašnjeni sa ovim mehanizmom.

Opseg homozigotnosti u genomu može biti i rezultat raznih vrsta jednoroditeljske disomije (engl. uniparental dysomia - UPD), heterozigotnih delecija i selekcije (Gibson i sur., 2006). Slučajevi disomije, u kojoj potomak nasljeđuje dvije kopije istog kromosoma (isodisomija) ili dijelove kromosoma (segmentna isodisomija) od istog roditelja uzrokuje homozigotnost potomaka na svim lokusima kromosoma ili kromosomalnim segmentima.

2.1.3. ROH podaci i identificiranje

Broj faktora koji utječu na kvalitetu ROH detekcije, uključujući i gustoću markera, njihovu distribuciju preko genoma, kvalitetu, stopu pozivanja/grešaka i frekvenciju minor alela genotipa. Trenutna ROH istraživanja najviše upotrebljavaju podatke dobivene putem SNP čipova zbog pristupačnosti ovih podataka i zbog činjenice da su ti podaci zlatni standard s vrlo niskim greškama kada se gleda stopa grešaka kod pozivanja genotipa (tipično <0.001). SNP podaci obično uključuju ~ 1-2.5 milijuna SNP-ova tipično s alelnim frekvencijama > 0.05,

izabranih kako bi najbolje reprezentirali strukturu haplotipa u ciljanoj populaciji. Čipovi s više od 300k markera pokazali su se dovoljno za uspješno detektiranje ROH dužih od 1 Mb, što odgovara pravom ROH koji je proizašao iz autozigotnosti.

Svakako, očekuje se da će dugački ROH segmenti zadržati svoje homozigotno stanje nezavisno od SNP pokrivenosti. Iako, relativna oskudnost SNP-ova na čipovima može značiti da pravi heterozigotni SNP-ovi između markera mogu biti izostavljeni, pa tako bi se dva bliska ROH segmenata mogu prikazati kao jedan, dulji ROH.

Postoje dvije glavne metode identificiranja ROH segmenata: uz pomoć opservacijskih algoritama za brojanje genotipova i algoritama na bazi modela. Opservacijski pristup upotrebljava algoritme koji skeniraju svaki kromosome tako da miču „prozor" fiksirane veličine preko cijele duljine genome kako bi pronašli segmente kontinuiranih SNP-ova. Ovaj pristup je implementiran u program PLINK v1.07 (Purcell i sur. 2007.) gdje je dan SNP smatran kao potencijalno ROH tako što izračunava proporciju kompletno homozigotnog „prozora" koji obuhvaća taj SNP. Ako ova proporcija je veća nego definirani prag, SNP se smatra dijelom ROH segmenta. U algoritmu, broj varijable heterozigotnih pozicija ili SNP-ova koji nedostaju, mogu se specificirati po „prozoru" kako bi se tolerirale genotipske greške. ROH je pozvan ako je broj kontinuiranih SNP-ova u homozigotnom segmentu prešao zadani prag u smislu SNP broja i/ili pokrivene kromosomalne duljine.

S druge strane, algoritmi koji traže odgovarajuće haplotipove (npr. Germline (Gusev i sur., 2009)) za izračun IBD mogu se isto upotrijebiti kako bi identificirali ROH segmente, kao poseban slučaj IBD-a unutar jedinke. Pristup s modelom upotrebljava skrivene Markovljev model (engl. Hidden Markov Model -HMM) kako bi se uzeli u obzir pozadinski nivoi LD-a, kao oni koji su implementirani u Beagle (Browning i sur., 2007) programskom paketu.

2.2. Programski paketi za obradu podataka

2.2.1. PLINK

Purcell i sur. (2007) dizajnirali su PLINK programski alat, kako bi se olakšala analiza „whole-genome“ podataka na više načina: upravljanje velikim podacima, radeći rutinske računalne analize učinkovitima, i nudeći nove analize koje iskorištavaju pokrivenost cijelog genoma. PLINK je napravljen kao „open-source“ C/C++ alat. S relativnom malim brojem podacima od 100,000 SNP-ova i 350 jedinki, npr. PLINK-u treba ~10 sekundi da bi učitao, filtrirao i izveo vezanu analizu za sve SNP-ove; izravno upravljajući s velikim setovima podataka je isto moguće. Izuzev računalskih izazova, veliki setovi podataka pogoršavaju problem zbunjivanja u genetskim istraživanjima. S povećanom moći da se otkriju pravi efekti, dolazi i povećani potencijal za pristranost koja može utjecati na rezultate.

PLINK „WGAS“ alat teži pojednostavljenju baratanja s velikim „WGAS“ podacima i pristupa problemu zbunjivanja zbog stratifikacije i ne nasumičnih genotipskih pogrešaka. To čini na način da proizvodi širok spektar statistike, izvodi razne standarde asocijacijske testove efektivno na velikim setovima podataka (u populacijama ili obiteljima, za bolesti ili kvantitativne ishode, dopuštajući kovarijance, haplotipske testove, itd.) i pruža sredstva za analiziranje rijetkih varijacija upotrebljavajući česte SNP panele.

Što se tiče menadžmenta podataka, Purcell i sur. (2007) stvorili su kompaktne binarne formate podataka kao bi predstavili SNP podatke, kao i alate koji bi pretvorili taj binarni format u standardne tekst formate (uključujući jedan-red-po-jedinki i transponirani jedan-red-po-SNP-u format). Moguće je i spojiti dva ili više setova podataka koji se mogu parcijalno preklapati i pružiti izvještaje i opise između preklapajućih setova podataka. Također dostupne su standardne sažete mjere: stopa genotipa, alelna i genotipska frekvencija, test Hardy-Weinberg ekvilibrijuma upotrebljavajući asimptotske i točne procedure, i „single-SNP“ Mendelske greške za obiteljske podatke. Na bazi genomske proporcije i genomske pokrivenosti alela dijeljenih IBD segmenata između dvije jedinke, PLINK pruža alate koji mogu skupiti jedinke u homozigotne podsetove, izvesti klasično multidimenzijsko skaliranje (MDS) kako bi vizualizirao sub-strukturu i pružio kvantitativni popis populacijske varijacije, i identificirati „outlier“ jedinke. PLINK upotrebljava kompletni povezani hijerarski „clustering“ kako bi pristupio stratifikaciji populacije, s upotrebom „whole-genome SNP“ podataka. Ova

aglomerativna procedura počinje s pretpostavkom da je svaka jedinka poseban odvojeni „cluster“ veličine 1, onda ponavljajući spaja dva najbliža „clustera“.Kompletna spojenost „clustera“ specificira da „clusteri“ se uspoređuju na bazi njihova dva člana koji su najviše različiti. „Clustering“ prestaje kada su sve jedinke u jednom „clusteru“.

2.2.2. cgaTOH

Polimorfizam jednog nukleotida (SNP), je jedno od najčešćih varijacija genoma, koji variraju u ljudskim svojstvima i podložnosti bolestima unutar i među populacijama. SNP može funkcionirati individualno, ali najčešće se smatra kako radi u koordinaciji sa drugim SNP-ovima kako bi se manifestirala bolest, ukazujući na to da se puno genetske raznolikosti u humanoj populaciji može objasniti putem velikih strukturnih razlika između individualnih genoma koji su mnogo veći od razlika jednih nukleotida.

Ovi strukturni polimorfizmi uključuju homozigotne regije u genomu npr. TOH (Tracts of Homozigosity) (Zhang i sur., 2013), koji mogu imati značajnu ulogu u genetici kompleksnih bolesti i genomskoj evoluciji. Trenutno najpopularniji programi za identificiranje TOH-a je ROH modul iz „Golden Helix“-a i „PLINK“-a (homozigotna opcija). ROH modul „Golden Helix“ programa je dizajniran da pronalazi tragove uzastopnih homozigotnih SNP-ova počivajući na svakom markeru i identificirajući lokaciju tih tragova koji specificirani uzorci dijele međusobno (zajednički ROH). Taj pristup identificiraju ROH regija nije dovoljno rigorozan jer često stapa ROH-a s dalekim granicama u zajedničkim regijama. “PLINK“ identificira ROH tako da uzima potreban broj homozigotnih SNP-pova, obuhvaćajući određenu distancu kilo baza (kb) upotrebom pristupa „prozora“ koji se pomiče i izvodi alelsko podudaranje u preklapajućim skupinama. Ali „PLINK“ samo podržava vrlo osnovnu detekciju dugačkih homozigotnih segmenata i neadekvatan je u stvaranju dobro formatiranih rezultata za daljnju analizu.

Kako bi se najbolje iskoristili „TOH“ svojstva i olakšala daljnja statistička analiza, predstavljena su 3 tipa surogat „TOH“-a : „cTOH“ – (common TOH) regija koja pokriva „cluster“ uzastopnih SNP-ova koji pripadaju „TOH-u“ u više subjekata, „gTOH“ – (group TOH) regija koja sadržava grupu preklapajućih „TOH“-ova s proksimalnim granicama, „aTOH“ – (allelic TOH) regija koja pokriva TOH-ove s preklapajućih alelsko-podudaranim regijama. „cga-TOH“ program počinje tako da traži „TOH“-ove i „cTOH“-ove, kasnije identificira „gTOH“

i „aTOH“ odvojeno, i svaka je popraćena interaktivnom regionalnom vizualizacijom, što rezultira programom koji je mnogo efektivniji i ima povišenu funkcionalnost i fleksibilnost upotrebljavajući odgovarajuće parametre. Identifikacija „TOH“-a, „cTOH“-a, „gTOH“-a i „aTOH“-a je implementirana bazirajući se na C++ sa sučeljem za linije naredba s opcijama za izbor parametara i vizualizacijsko sučelje stvoreno od strane Qt za „cross-platform UI“.

2.2.3. Python

Python je programski jezik visokog nivoa koji se koristi za opću svrhu programiranja. Stvorio ga je Guido van Rossum i prvi puta ga je izbacio u 1991. .Python ima filozofski dizajn koji naglašava čitljivost koda, posebice značajno upotrebljavajući prazan prostor. Pruža konstrukciju koja omogućava jasno programiranje na malim i velikim omjerima. Python sadrži elemente tipa dinamičnog sistema i automatski menadžment memorije. Podržava više programskih paradigmi, uključujući objektno-orijentirano, imperativno, funkcionalno i proceduralno, tako i sadrži veliku bazu „library-a“. Umjesto da sadrži sve funkcionalnosti u svojoj srži, Python je napravljen da bude jako fleksibilan. Ta kompaktna modularnost učinila ga je vrlo popularnim kao sredstvo u koje se može dodavati programibilno sučelje u postojeće aplikacije.

Van Rossum-ova vizija malog sržnog programa s velikom standardnom „library“ i lako fleksibilnim tumačiteljem proizašao je iz njegove frustracije s ABC programskim jezikom koji je išao u skroz suprotnom smjeru. Najveća snaga Pythona su njegovi standardni „library“ koji pružaju mnogo alata za raznorazne zadatke. Za internet aplikacije, mnogo standardnih protokola i formata tipa MIME i HTTP su podržavani.

Uključuje i module za stvaranje grafičkih sučelja, spajanje na relacijske baze podataka, stvaranje pseudo-nasumičnih brojeva, aritmetiku s arbitražnim preciznim decimalama, manipuliranje standardnim izrazima i testiranje aplikacija. Empirijsko istraživanje zaključilo je da jezici za skriptiranje, tipa Python su više produktivni nego C ili Java u smislu bolje obrađuju programske probleme tipa „string“ manipulaciju i pretraživanje rječnika, dok drugi zaključak je bio upotreba memorije obično bolje radi nego Java, ali ne toliko gore od C ili C++.

Uz mnoge mrežne „framework“-ove (Django, Pylons, Pyramid, web2py), „library“-e (NumPy, SciPy, Matplotlib), valja spomenuti i Biopython. To je „open-source“ kolekcija nekomercijalnih Python alata za računalnu biologiju i bioinformatiku. Sadrži klase koje

predstavljaju biološke sekvence i anotacije sekvenaca, i u mogućnosti je čitati i uređivati širok spektar formata podataka.

Omogućava laki pristup online bazama bioloških podataka, tipa NCBI. Posebni moduli mogu proširiti Biopython-ove mogućnosti na „sequence alignment“, čitanje proteinske strukture, populacijsku genetiku, filogenetiku, motive sekvenaca i „machine learning“. U kontrastu vrlo popularnog R programskog jezika i Rstudio radnog okoliša, stvoren je Spyder.

2.2.4. Anaconda i Spyder

Anaconda je besplatan „open-source“ distribucija Python i R programskih jezika za „data science“ i „machine learning“ aplikacije (procesiranje velikih podataka, prediktivna analiza, znanstveno računanje). Cilj je pojednostaviti menadžment paketa i postavljanje. Uz Anacondu dolazi i conda – „open“ source, „cross platform“ menadžer paketa i menadžer radnog okoliša koji instalira, pokreće i ažurira pakete i njihove zavisne funkcije.

Spyder je integrirani okoliš za razvijanje za znanstveno orijentirano programiranje u Python jeziku-Spyder integrira s brojnim paketima u Python-ovom znanstvenom arsenalu, uključujući i NumPY, SciPY, Matplotlib, pandas, IPython, SymPY, Cython. Napravio ga je i razvio Pierre Rayabaut 2009. godine, a od 2012. Spyder je bio očuvan i nastavljao se razvijati i poboljšavati uz pomoć Python-ovog znanstvenog tima.

3. Materijali i metode

3.1. Podaci

U ovom radu korišteni su podaci 36 ovaca genotipiziranih s OvineSNP50 BeadChip koji sadrži 54 241 ravnomjerno raspoređenih SNP-ova po genomu. Ovaj SNP čip nastao je kao suradnja vodećih istraživača koji se bave ovcama sa sljedećih institucija AgResearch, Baylor UCSC, CSIRO i USDA uz pomoć Sheep Genomics Consortium-a.

3.2. Kontrola kvalitete

Prva kontrola kvalitete učinjena je u napravljena u Pythonu korištenjem Illumina parametara kontrole ("GenTrainScore" manji ili jednak 0.4 i "GenCallScore" manji ili jednak 0.7) isključeni su SNP-ovi upitne kvalitete. Isključeni su i svi SNP-ovi koji nisu pripisani kromosomima, oni koji su pripisani spolnim kromosomima i mitohondrijskoj DNA. Putem programskog paketa PLINK v1.07. (Purcell i sur., 2007) isključeni su SNP-ovi koji su bili prisutni u manje od 10% jedinki, te jedinke kojima je nedostajalo više od 5% SNP-ova.

3.3. Detekcija ROH segmenata

Detekcija ROH segmenata provedena je uz pomoć programskog paketa cgaTOH. Prema Ferenčaković i sur. (2013) detektirano je pet duljina ROH-ova (u Mb): [1, 2], <2, 4], <4, 8], <8, 16] i >16. Slijed homozigotnih SNP-ova detektiran je kao ROH ukoliko sadrži najmanje 15 homozigotnih SNP-ova s maksimalnim dozvoljenim razmakom između SNP-ova od 1 Mb. U detekciji nisu bili dopušteni heterozigotni SNP-ovi osim za >16 Mb kada se dopušta jedan.

3.4. Vizalizacija ROH segmenata

Vizualizacija je rađena s podacima dobivenih iz cgaTOH-a. Uz pomoć paketa plotnine, napravljeni su plotovi segmenata za sve duljine, specificirani za kromosom 1. Prva pozicija i zadnja pozicija segmenata su dijeljeni s 100000 kako bi se lakše grafički prikazalo.

3.5. Opis koda

```
import os
import pandas as pd
import numpy as np
```

Učitavam „library“-e potrebne za manipulaciju podataka. OS nam služi kako bi uveli funkcionalnosti operativnog sustava. Pandas nam služi za manipuliranjem strukturama podataka i dovodi skup funkcija korisne za analitičke protokole s podacima, pd se stavlja kao kratica. Numpy nam služi za manipuliranje podacima velikih dimenzija , matricama i matematičkim funkcijama, np je kratica.

```
os.chdir(„C:/LokalnaAdresaRadnogDirektorija/“)
```

os.chdir nam služi za postavljanje radnog direktorija u kojem će nam biti potrebni podaci.

```
File = pd.read_csv('TYPOV_FinalReport_070815.csv', delimiter = ",")
```

Učitavamo sirove podatke u sistem pythona, low_memory = False opcija se dodaje ako podaci koji se uvlače su veliki, no pritom treba paziti da računalo ima dovoljno RAM-a za komputaciju.

```
list(Prime)
File1 = File[["SampleID","SNPName","Position", SNPIndex", "Chr",
"Allele1AB", "Allele2AB", "GCScore","ClusterSep", "GTScore"]]
```

S naredbom list i nazivom naše uvučene datoteke, izlistavamo varijable kako bi mogli vidjeti što se nalazi unutra i koje varijable želimo dalje voditi kroz analizu. Dalje stvaramo novi objekt sa varijablama iz izvorne datoteke te specificiramo točne nazive varijabla.

```
File2 = File1
File2[["Position"]] = File2[["Position"]].astype(object)
```

Stvaramo kopiju objekta, kako bismo smanjili vrijeme potrebno za ispravke ako nešto ne valja s kodom. Drugom naredbom mijenjamo tip pozicijske varijable SNP-ova iz int64 u object zbog lakšeg manipuliranja.

```
File2 = File2.dropna()
File2 = File2[File2.Position != "#REF!"]
File2 = File2[File2.Position != "REF"]
File2 = File2[File2.Position != "."]
File2[["Position"]] = File2[["Position"]].astype(np.int64)
```

U ovome dijelu koda čistimo varijable u objektu. .dropna() je funkcija koja nam sve NaN vrijednosti briše iz varijabli. Dalje se čisti na razna pogrešna očitavanja u datoteci. Na kraju se pozicijska varijabla vraća u početni tip varijable int64 sa astype(np.int64).

```
File2.dtypes
File2 = File2.rename(columns = {"Position" : "bp_position"})
File2["cmunit"] = File2['bp_position']/1000000
```

Provjeravamo da li su sve varijable tipova kakvih trebaju biti. U drugom dijelu koda preimenuje se pozicijska varijabla u poziciju baznih parova. Zatim se stvara nova varijabla u objektu koja uzima podatke od pozicije baznih parova i dijeli se sa milijun (centimorgani).

```
File2 = File2[File2.Chr != "MT"]
File2 = File2[File2.Chr != "0"]
File2 = File2[File2.Chr != "Y"]
File2 = File2[File2.Chr != ""]
```

Daljnje čišćenje kromosomske varijable na druga očitavanja.

```
File2 = File2[~(File2.GCScore < 0.7)]
File2 = File2[~(File2.GTScore <= 0.4)]
```

Filtriramo varijable GCScore i GTScore na željene vrijednosti.

```
File3 = File2[["SampleID", "SNPName", "Chr", "Allele1AB",
"Allele2AB", "cmunit", "bp_position"]]
File3 = File3.sort_values(by = ["SNPName"], ascending = [True])
file3gvmap = File3.drop_duplicates(subset = "SNPName",keep="first")
```

Stvaramo još jednu kopiju. Zatim sortiramo vrijednosti po SNP imenu, uzlazno. U trećoj naredbi stvaramo novi objekt i brišemo duplikate u varijabli za SNP imena, vrlo je važno staviti keep = „first“ jer će i inače naredba izbrisati sve koji su dupli i neće izostaviti originale.

```
mapsnpgvhd = file3gvmap[["Chr", "SNPName", "cmunit", "bp_position"]]
mapsnpgvhd.to_csv("mr50.map", sep = "\t", header = False, index =
False)
```

Stvaramo novi objekt sa određenim varijablama. U drugoj naredbi radimo prvu datoteku za učitavanje kasnije u PLINK. Radi se .csv datoteka, ekstenzija mora biti .map, tab separator, prvi redak i indeks za redove se miču.

```
File4 = File2.sort_values(by = ["SampleID", "SNPName"])
File4["fid"] = 1
lgen = File4[["fid", "SampleID", "SNPName", "Allele1AB",
"Allele2AB"]]
lgen.to_csv("mr50.lgen", sep = "\t", header = False, index = False)
```

Radimo četvrti objekt sa sortiranim vrijednostima; prvo po Identifikaciji uzorka, zatim po SNP imenu. Dodaje se varijabla „fid“ koja ima vrijednost 1. Stvaramo objekt sa potrebnim varijablama iz četvrtog objekta. Stvaramo lgen datoteku potrebu za PLINK, ekstenzija mora biti .lgen, tab separator, prvi redak i indeksi redova se miče.

```
fam1 = File4[["SampleID"]]
fam1 = fam1.sort_values(by= "SampleID")
fam1 = fam1.drop_duplicates(keep = "first")
fam1["famID"] = "1"
fam1["otac"] = "000000000000000"
fam1["majka"] = "000000000000000"
fam1["spol"] = "-1"
fam1["feno"] = "-9"
fam1 = fam1[["famID", "SampleID", "otac", "majka", "spol", "feno"]]
fam1.to_csv("mr50.fam",sep = "\t", header = False, index = False)
```

U ovom djelu koda stvaramo fam datoteku. Stvaramo novi objekt sa varijablom identifikacija uzoraka. Dalje sortiramo vrijednosti, zatim mičemo duplikate(keep = „first“). Dodajemo nove varijable i vrijednosti; famID koji je 1, za varijablu otac i majka se stavlja 000000000000000, zatim varijabla spol čija je vrijednost -1 te varijabla feno koja je -9. Preslažemo varijable u pravilan redoslijed. Zatim ispisujemo .fam datoteku, sa standardnim postavkama.

```
import subprocess as sb
sb.call("plink.exe --lfile mr50 --sheep --recode --out mr51 --
noweb", shell = False)
sb.call("plink.exe --file mr51 --sheep --geno 0.9 --recode --out
mr51geno --noweb", shell = False)
sb.call("plink.exe --file mr51geno --sheep --mind 0.05 --recode --
out mr51mind --noweb", shell = False)
```

Uvodimo „library“ subprocess i stavljamo ga kao kraticu sb. Ovom komandom zovemo vanjsku aplikaciju koja se pokreće u cmd-u. Unutar navodnika u zagradi pišemo stvari koje bi inače pisali kao komande za odabranu aplikaciju. U ovom slučaju je to PLINK. Prva komanda unutar zagrade nam je .exe samog programa, zatim stavljamo -- lfile sa nazivom koje smo dali našim troje datotekama koje smo prethodno napravili u pythonu(.map, .lgen, .fam). Specificiramo o kojoj se životinji radi i stavljamo komandu – recode. Dajemo ime out file-u. U drugom i trecem pozivanju radilo se čiščenje na – geno i – mind.

```
import subprocess as sb
sb.call("TOH.exe -map mr51mind -p mr51mind -o roh_1 -l 15 -
min_length 1000000 -max_gap 1000000 -max_missing 0 -max_hetero 0 -
skip_clustering -force_proceed", shell = False)
sb.call("TOH.exe -map mr51mind -p mr51mind -o roh_2 -l 15 -
min_length 2000000 -max_gap 1000000 -max_missing 0 -max_hetero 0 -
skip_clustering -force_proceed", shell = False
sb.call("TOH.exe -map mr51mind -p mr51mind -o roh_4 -l 15 -
min_length 4000000 -max_gap 1000000 -max_missing 1 -max_hetero 0 -
skip_clustering -force_proceed", shell = False
sb.call("TOH.exe -map mr51mind -p mr51mind -o roh_8 -l 15 -
min_length 8000000 -max_gap 1000000 -max_missing 2 -max_hetero 0 -
skip_clustering -force_proceed", shell = False
sb.call("TOH.exe -map mr51mind -p mr51mind -o roh_16 -l 15 -
min_length 16000000 -max_gap 1000000 -max_missing 4 -max_hetero 1 -
skip_clustering -force_proceed", shell = False)
```

Uvodimo subprocess i dajemo mu kraticu sb. U ovom segmentu naredbi pozivamo cgaTOH aplikaciju. Radimo detekciju na duljinu ROH-ova. Proces je isti kao i za PLINK samo što naravno cgaTOH ima svoje naredbe koje izvršava u cmd-u. Uvlačimo zadnje podatke koje smo napravili u PLINK-u. U svakoj naredbi određuje se minimalna duljina TOH-ova, maksimalna duljina razmaka između obližnjih SNP-ova, dozvoljen broj nedostajućih SNP-ova, maksimalni dopušteni broj heterozigotnih SNP-ova i preskače se „clustering“.

```
roh1 = pd.read_table('roh_1.homozygousruns', sep = ",")
roh1[["FirstPosition"]] = roh1[["FirstPosition"]]/1e+6
roh1[["LastPosition"]] = roh1[["LastPosition"]]/1e+
roh11 = roh1
roh11 = roh11[roh11.Chromosome == 1]

r1= (ggplot(roh11)
            +  geom_segment(  aes(x  =  "FirstPosition",  xend  =
"LastPosition", y="Label", yend="Label"), size = 5)
            + ggtitle("ROH1") #Naslov ROH-a kojeg radimo
            + labs(x = "Chromosome1", y = "ID") # Oznake za x, y osi
            + geom_vline(xintercept = [50, 100, 150, 200, 250, 300],
color = "red", alpha =  0.3)
 )

r1.save('ROH1.pdf', height=16, width=18)

#ROH2-------------------------------------------------

roh2 = pd.read_table('roh_2.homozygousruns', sep = ",")
roh2[["FirstPosition"]] = roh2[["FirstPosition"]]/1e+6
roh2[["LastPosition"]] = roh2[["LastPosition"]]/1e+6
roh21 = roh2
roh21 = roh21[roh21.Chromosome == 1]

r2 = (ggplot(roh21)
            +  geom_segment(  aes(x  =  "FirstPosition",  xend  =
"LastPosition", y="Label", yend="Label"), size = 5)
            + ggtitle("ROH2")
            + labs(x = "Chromosome1", y = "ID")
            + geom_vline(xintercept = [50, 100, 150, 200, 250, 300],
color = "red", alpha =  0.3)
)

r2.save('ROH2.pdf', height=16, width=18)
```

```
#ROH4-----------------------------------------------

roh4 = pd.read_table('roh_4.homozygousruns', sep = ",")
roh4[["FirstPosition"]] = roh4[["FirstPosition"]]/1e+6
roh4[["LastPosition"]] = roh4[["LastPosition"]]/1e+6
roh41 = roh4
roh41 = roh41[roh41.Chromosome == 1]

r4 = (ggplot(roh41)
            +  geom_segment(  aes(x  =  "FirstPosition",  xend  =
"LastPosition", y="Label", yend="Label"), size = 5)
            + ggtitle("ROH4")
            + labs(x = "Chromosome1", y = "ID")
            + geom_vline(xintercept = [50, 100, 150, 200, 250, 300],
color = "red", alpha =  0.3)
)

r4.save('ROH4.pdf', height=16, width=18)

#ROH8-----------------------------------------------

roh8 = pd.read_table('roh_8.homozygousruns', sep = ",")
roh8[["FirstPosition"]] = roh8[["FirstPosition"]]/1e+6
roh8[["LastPosition"]] = roh8[["LastPosition"]]/1e+6
roh81 = roh8
roh81 = roh81[roh81.Chromosome == 1]

r8 = (ggplot(roh81)
            +  geom_segment(  aes(x  =  "FirstPosition",  xend  =
"LastPosition", y="Label", yend="Label"), size = 5)
            + ggtitle("ROH8")
            + labs(x = "Chromosome1", y = "ID")
            + geom_vline(xintercept = [50, 100, 150, 200, 250, 300],
color = "red", alpha =  0.3)
)

r8.save('ROH8.pdf', height=16, width=18)
```

```
#ROH16-----------------------------------------------

roh16 = pd.read_table('roh_16.homozygousruns', sep = ",")
roh16[["FirstPosition"]] = roh16[["FirstPosition"]]/1e+6
roh16[["LastPosition"]] = roh16[["LastPosition"]]/1e+6
roh161 = roh16
roh161 = roh161[roh161.Chromosome == 1]

r16 = (ggplot(roh161)
            +  geom_segment(  aes(x  =  "FirstPosition",  xend  =
"LastPosition", y="Label", yend="Label"), size = 5)
            + ggtitle("ROH16")
            + labs(x = "Chromosome1", y = "ID")
            + geom_vline(xintercept = [50, 100, 150, 200, 250, 300],
color = "red", alpha =  0.3)
)

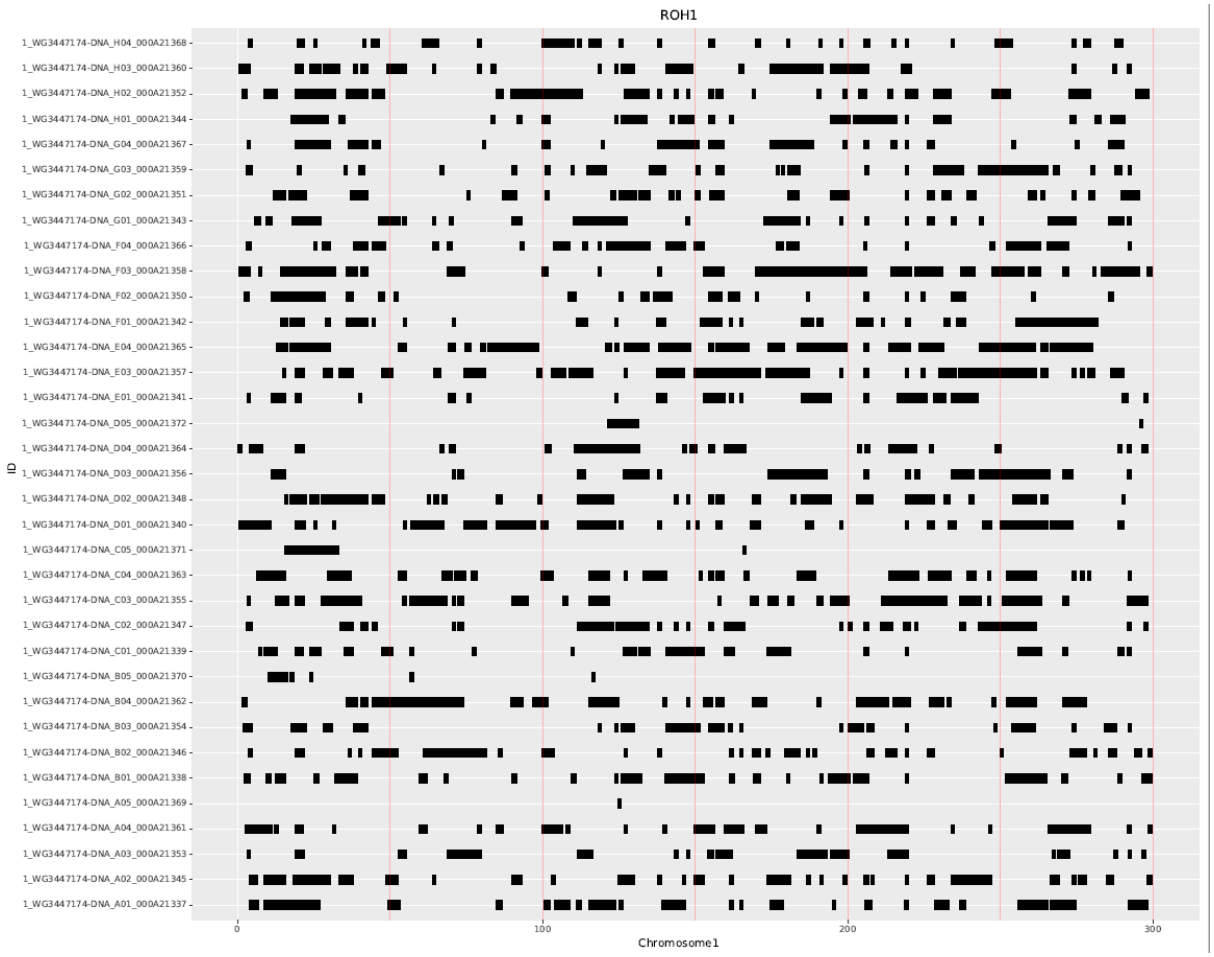
r16.save('ROH16.pdf', height=16, width=18)
```

Zadnji segment komandi nam pokazuje kako nacrtati dobivene rezultate iz cgaTOH-a. Dakle prvo uvlačimo plotnine koji nam je library ggplot-a za python. Zatim se učitavaju podaci dobiveni iz cgaTOH-a koji imaju ekstenziju -homozygousruns i imaju separator „,“. Kako bi se olakšalo crtanje i snalaženje po budućem grafu, varijable prve pozicije segmenta ROH-a i zadnje pozicije segmenta ROH-a koji su izraženi u baznim parovima , dijelimo sa 1 000 000 i pretvaramo ih u megabaze. Stvara se kopija za lakše manipuliranje. Zatim se podatak pročišćava na kromosom koji se želi vidjeti na grafu. Može se dakle crtati samo segmenti jednog kromosoma odjednom. Dakako može se i programirati da se isrctava sve iako to bi zahtijevalo jako puno vremena i snalaženja po velikoj količini PDF grafova. Slijedi nam samo crtanje. Radimo objekt u koji ćemo crtati, pozivamo ggplot i uvučemo datoteku na koju smo pročistili određen kromosom za iscrtavanje. Crtamo s geom_segment, u njemu se specificira x koja je nama prva pozicija segmenta, zatim xend koja je nama varijabla za zadnju poziciju segmenata. Na y osi nemamo ništa za iscrtavati ali da bi se lakše snalazili otkud koji segment potiče specificiramo y i yend sa varijablom koja nam sadržava oznake uzoraka. Dalje se specificira koliko veliki segment da bude, ali ne preporučuje se više od 5 zbog

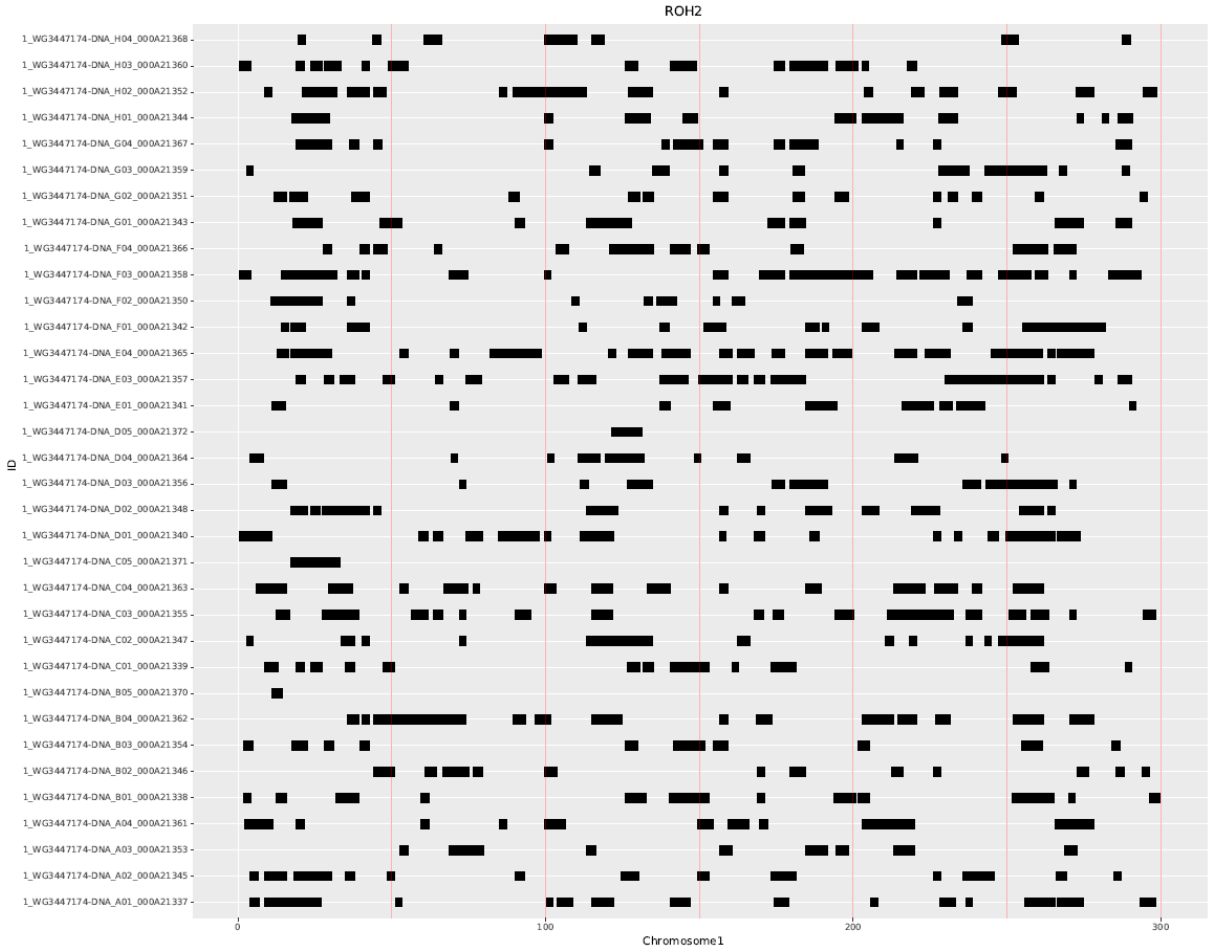
preglednosti. Pišemo i naslov ROH kojeg radimo sa ggtitle(). Dalje se crtaju oznake za x i y osi sa labs(x="" , y""). Obično se piše na X osi koji je kromosom u pitanju, a na Y osi da su to identifikacije uzoraka. Sljedeća komanda geom_vline nam služi kako bi se još lakše snalazili u grafu pošto je velik tako što smo ga rascijepali na 50 po 50 segmenata do 300 u ovom slučaju. Iscrtava se crvena linija svakih 50 megabaza. Naravno lako se specificira koliko gusto ili rijetko se želi da se iscrtavaju te linije sa argumentom xintercept = [], specificira se boja sa color, i alpha= koja nam služi za manju transparentnost linija da ne upadaju toliko u vid tijekom procjene(preporučuje se alpha = 0.3). Nakon što smo sve iscrtali. Taj graf se sprema kao .pdf, sa specificiranim veličinama. Veličina pdf a ovisi o grafu i veličini podataka. PDF se sprema u radni direktorij i od tamo se može pristupiti rezultatima. Postupak se ponavlja za svaku duljinu TOH-a za koju smo radili detekciju.

4. Rezultati i rasprava

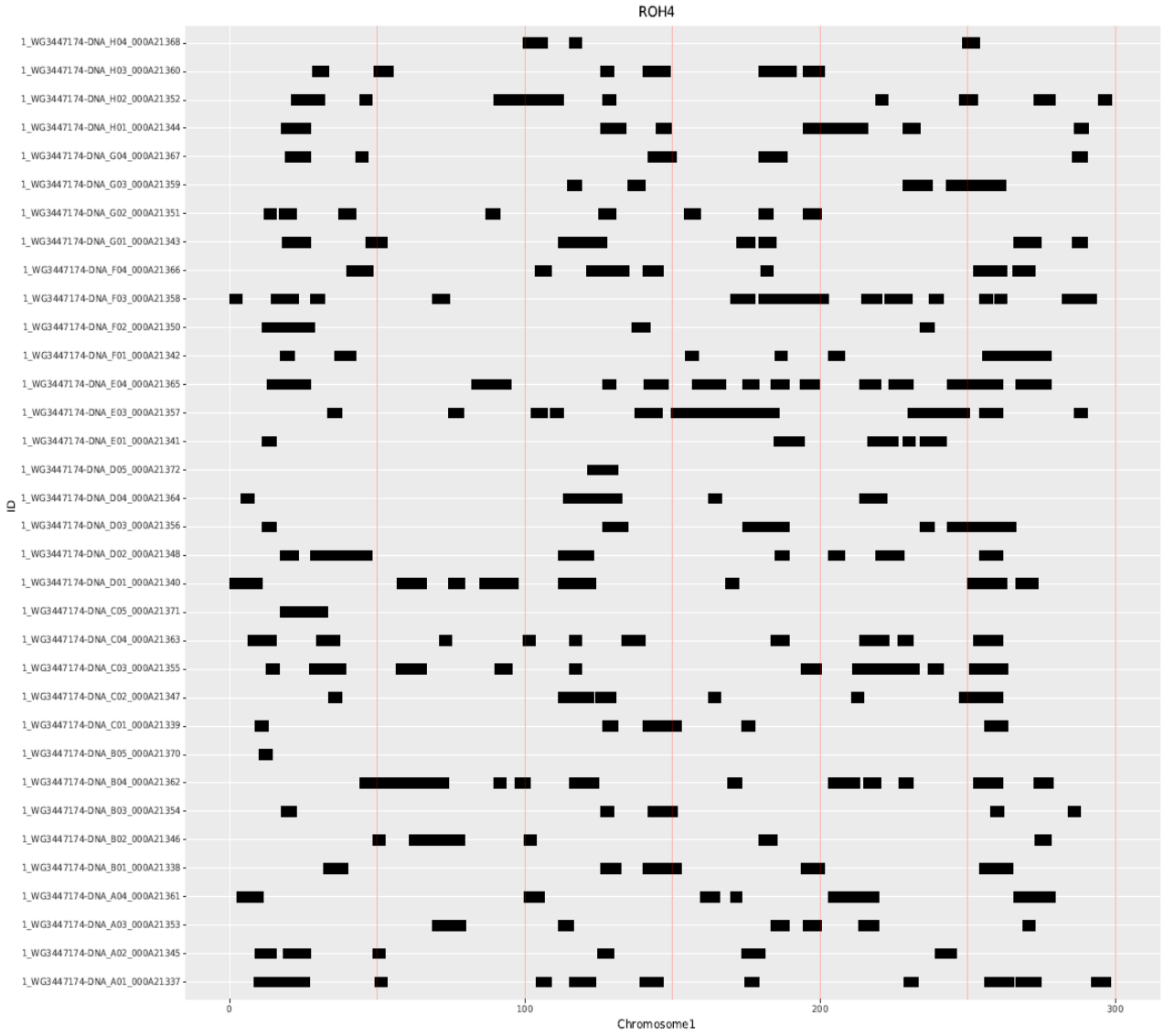
Sljedeći rezultati su dobiveni s python/spyder pipelinom.



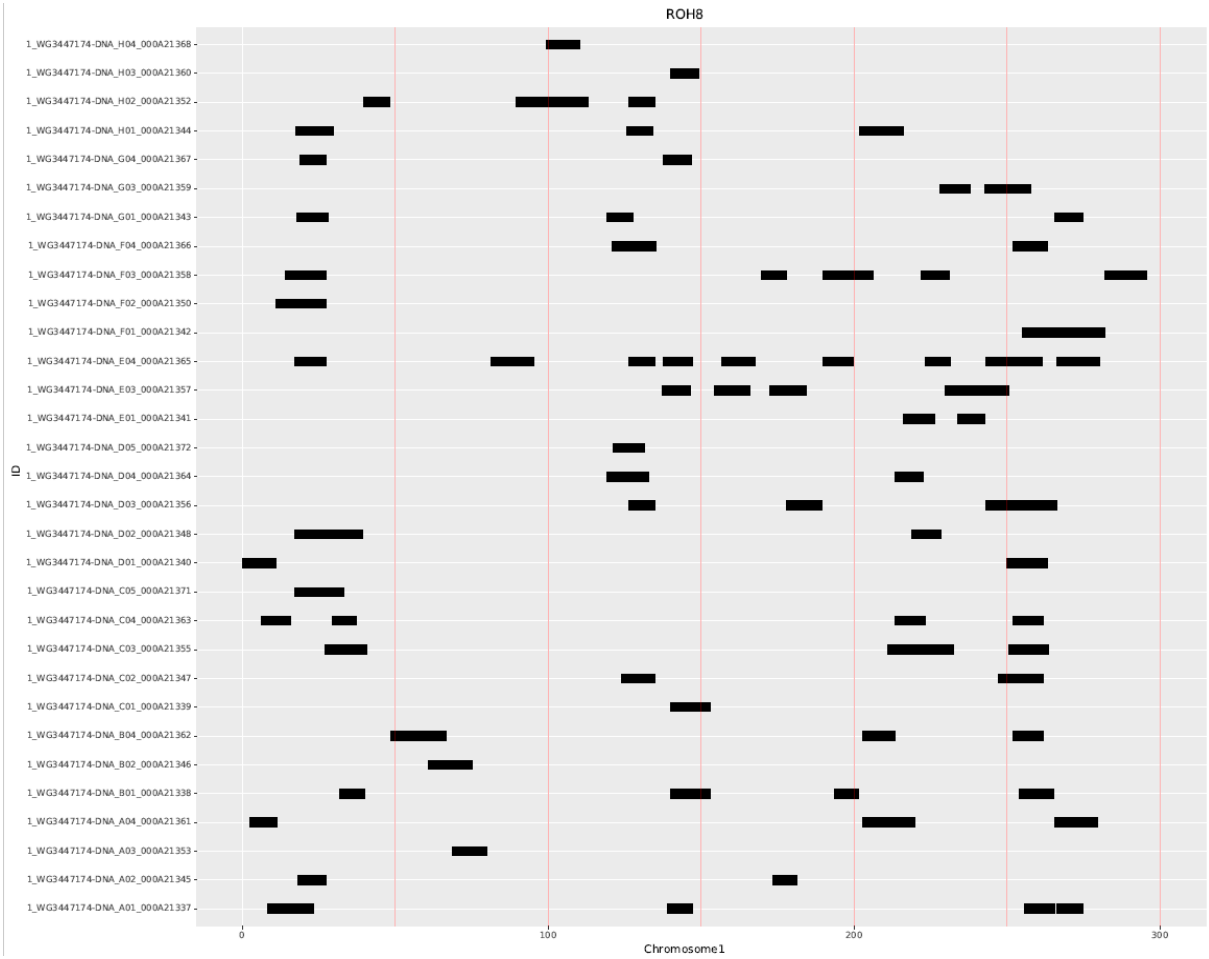
Slika 1. TOH-ovi detektirani na prvom kromosomu, grafički prikazani, s minimalnom duljinom od 1Mb.



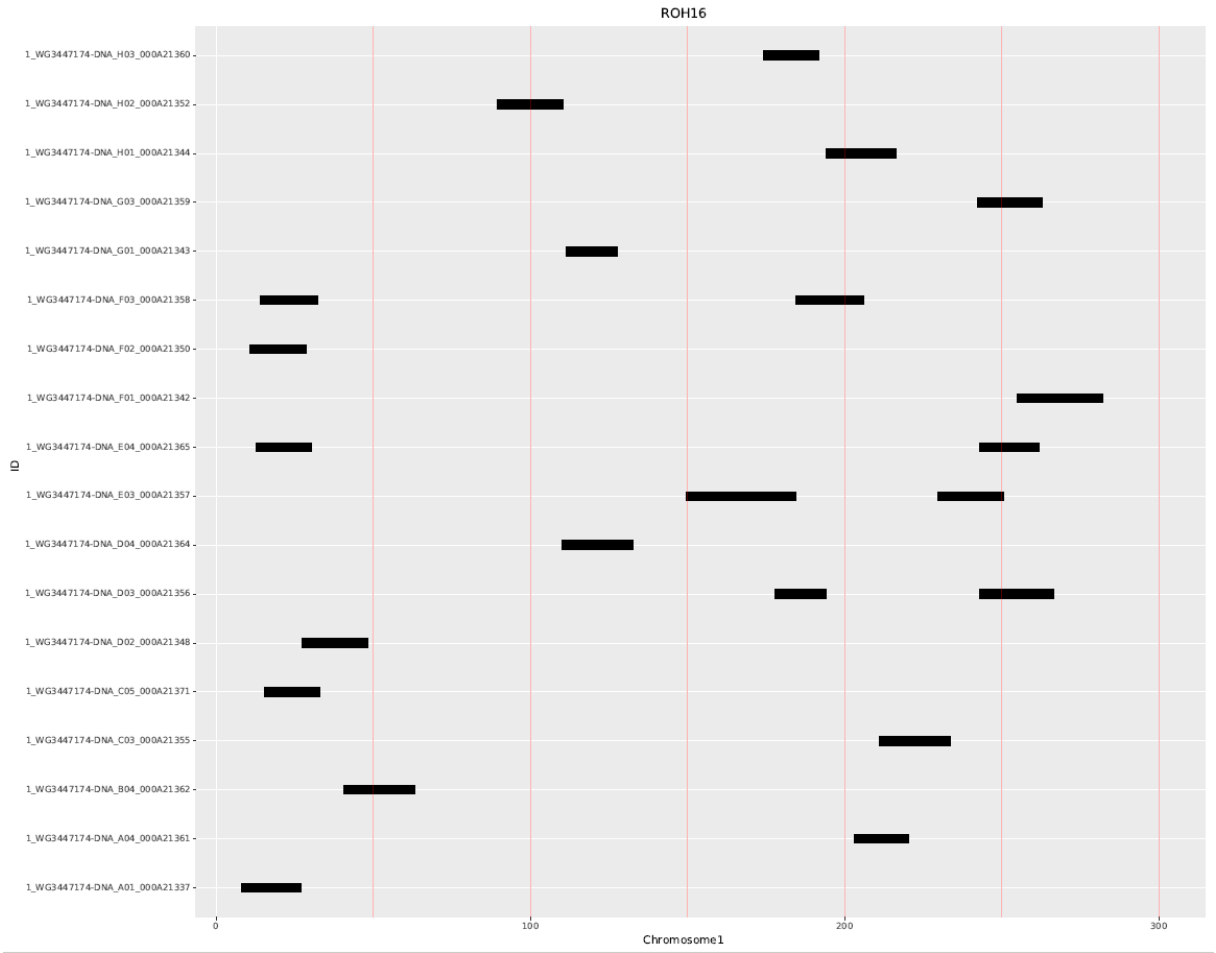
Slika 2 TOH-ovi detektirani na prvom kromosomu, grafički prikazani, s minimalnom duljinom od 2Mb.



Slika 3 TOH-ovi detektirani na prvom kromosomu, grafički prikazani, s minimalnom duljinom od 4Mb.



Slika 4 TOH-ovi detektirani na prvom kromosomu, grafički prikazani, s minimalnom duljinom od 8Mb.



Slika 5 TOH-ovi detektirani na prvom kromosomu, grafički prikazani, s minimalnom duljinom od 16Mb.

Trenutno, ovo su dobro prikazani grafički rezultati detektiranja ROH-ova. Iako bi se daljnje mogli poboljšati a interaktivnom vizualizacijom kao što je npr. iskočni informativni prozor s predefiniranim podacima iz varijable u kodu. No trenutno je to limitirano samo za web preglednike i programske grafičke prikaze, no u skoroj budućnosti bi se moglo implementirati u ovaj kod.

5. Zaključak

Kako napreduje znanosti i ulazimo u sve veće digitalno doba, genetski podaci su nam sve dostupniji, a sekvenciranja jeftinija. Trenutno je jedan od većih izazova za ROH podatke imati dobar „pipeline“ koji će nam skratiti vrijeme za čišćenje sirovih podataka, pozivanje vanjskih ili rađenje svojih funkcija za detektiranje ROH-a iz sekvenci te njihovo vizualiziranje. Python malo zaostaje u smislu genetičkih potreba za analiziranjem nego R jezik, no u zadnje vrijeme se sve više radi paketa i funkcija koji su specificirani za način rada koji su potrebni za sekvence.

6. Popis literature

Albrechtsen, A., Nielsen, F. C. i Nielsen, R. (2010) 'Ascertainment Biases in SNP Chips Affect Measures of Population Divergence', *Molecular Biology and Evolution*. Oxford University Press, 27(11), pp. 2534–2547. doi: 10.1093/molbev/msq148.

Broman, K. W. i Weber, J. L. (1999) 'Long Homozygous Chromosomal Segments in Reference Families from the Centre d'Étude du Polymorphisme Humain', *The American Journal of Human Genetics*. Cell Press, 65(6), pp. 1493–1500. doi: 10.1086/302661.

Browning, S. R. i Browning, B. L. (2007) 'Rapid and accurate haplotype phasing and missing data inference for whole genome association studies using localized haplotype clustering', *Am J Hum Genet*, 81. doi: 10.1086/521987.

Curik, I., Ferenčaković, M. i Sölkner, J. (2014) 'Inbreeding and runs of homozygosity: A possible solution to an old problem', *Livestock Science*, 166(1). doi: 10.1016/j.livsci.2014.05.034.

Ferenčaković, M., Sölkner, J. i Curik, I. (2013) 'Estimating autozygosity from high-throughput information: effect of SNP density and genotyping errors', *Genet Sel Evol*, 45. doi: 10.1186/1297-9686-45-42.

Gibson, J., Morton, N. E. i Collins, A. (2006) 'Extended tracts of homozygosity in outbred human populations', *Human Molecular Genetics*, 15(5), pp. 789–795. doi: 10.1093/hmg/ddi493.

Gusev, A., Lowe, J. K., Stoffel, M., Daly, M. J., Altshuler, D., Breslow, J. L., Friedman, J. M. i Pe'er, I. (2009) 'Whole population, genome-wide mapping of hidden relatedness.', *Genome research*. Cold Spring Harbor Laboratory Press, 19(2), pp. 318–26. doi: 10.1101/gr.081398.108.

Howrigan, D. P. (2015) 'Genome-wide autozygosity is associated with lower general cognitive ability', *Molecular Psychiatry*, (April), pp. 1–7. doi: 10.1038/mp.2015.120.

Lander, E. S. i Botstein, D. (1987) 'Homozygosity mapping: a way to map human recessive traits with the DNA of inbred children.', *Science (New York, N.Y.)*. American Association for the Advancement of Science, 236(4808), pp. 1567–70. doi: 10.1126/SCIENCE.2884728.

Lencz, T., Lambert, C., DeRosse, P., Burdick, K. E., Morgan, T. V., Kane, J. M., Kucherlapati, R. i Malhotra, A. K. (2007) 'Runs of homozygosity reveal highly penetrant recessive loci in schizophrenia.', *Proceedings of the National Academy of Sciences of the United States of America*. National Academy of Sciences, 104(50), pp. 19942–7. doi: 10.1073/pnas.0710021104.

Purcell, S., Neale, B., Todd-Brown, K., Thomas, L., Ferreira, M. A. R., Bender, D., Maller, J., Sklar, P., de Bakker, P. I. W., Daly, M. J. i Sham, P. C. (2007) *PLINK: A Tool Set for Whole-Genome Association and Population-Based Linkage Analyses*, *The American Journal of Human Genetics*. doi: 10.1086/519795.

Purfield, D. C., Berry, D. P., McParland, S. i Bradley, D. G. (2012) 'Runs of homozygosity and population history in cattle.', *BMC genetics*, 13(1), p. 70. doi: 10.1186/1471-2156-13-70.

Wigginton, J. E., Cutler, D. J. i Abecasis, G. R. (2005) 'A Note on Exact Tests of Hardy-Weinberg Equilibrium', *The American Journal of Human Genetics*. Cell Press, 76(5), pp. 887–893. doi: 10.1086/429864.

Woods, C. G., Cox, J., Springell, K., Hampshire, D. J., Mohamed, M. D., McKibbin, M., Stern, R., Raymond, F. L., Sandford, R., Malik Sharif, S., Karbani, G., Ahmed, M., Bond, J., Clayton, D. i Inglehearn, C. F. (2006) 'Quantification of Homozygosity in Consanguineous Individuals with Autosomal Recessive Disease', *The American Journal of Human Genetics*. Cell Press, 78(5), pp. 889–896. doi: 10.1086/503875.

Zhang, L., Orloff, M. S., Reber, S., Li, S., Zhao, Y. i Eng, C. (2013) 'cgaTOH: Extended Approach for Identifying Tracts of Homozygosity', *PLoS ONE*. Edited by R. Wu. Public Library of Science, 8(3), p. e57772. doi: 10.1371/journal.pone.0057772.

7. Prilozi

7.1. Prilog 1

Python kod

```
#Uvođenje potrebnih funkcija
import os #Uvođenje funkcionalnosti operativnog sistema
os.chdir("C:/Users/Hrza/Desktop/Testfolder") #Postavljanje radnog direktorija
import pandas as pd # Python Library za lako manipuliranje strukturama podataka i skup funkcija analiza za podatke, pd se koristi kao kratica
import numpy as np #np as shortcut Python library za velike, multi-dimenzijske poretke i podatke, matrice, i matematičke funkcije za rad za većim podacima
Prime = pd.read_csv('TYPOV_FinalReport_070815.csv', delimiter = ",")
#Uvlacenje podataka u sistem, za veće podatke, ako neće uvuć dodati parametar low_memory = False ali se ne preporuca ako se radi na slabijem sistemu

#Čišćenje podataka
list(Prime) #listanje imena varijabla ako je potrebno
Prime1 = Prime[["SampleID","SNPName","Position", "SNPIndex","Chr", "Allele1AB","Allele2AB", "GCScore","ClusterSep", "GTScore"]] #Stvaranje novog objekta, uvlačenje željenih varijabli iz izvornih podataka

Prime2 = Prime1 #kopija objekta pod novim imenom
Prime2[["Position"]] = Prime2[["Position"]].astype(object)#pretvaranje tipa varijable iz int64 u object zbog lakseg ciscenja

# Brisanje specificnih vrijednosti u pozicijskoj varijabli
Prime2 = Prime2.dropna()# Brisanje NaN vrijednosti
Prime2 = Prime2[Prime2.Position != "#REF!"]# Brisanje ostalih vrijednosti
Prime2 = Prime2[Prime2.Position != "REF"]
Prime2 = Prime2[Prime2.Position != "."]
Prime2[["Position"]] = Prime2[["Position"]].astype(np.int64) #pretvaranje tipa varijable natrag u izvorni tip podataka

Prime2.dtypes#provjera da li su podaci vraceni u potrebne tipove
Prime2 = Prime2.rename(columns = {"Position" : "bp_position"})
#Preimenovanje pozicijske varijable
Prime2["cmunit"] = Prime2['bp_position']/1000000 #Stvaranje nove varijable sa pozicijom podijenjenom sa 1 000 000
```

```
#Brisanje specificnih vrijednosti u kromosomskoj varijabli
Prime2 = Prime2[Prime2.Chr != "MT"]# Brisanje vrijednosti
Prime2 = Prime2[Prime2.Chr != "0"]
Prime2 = Prime2[Prime2.Chr != "Y"]
Prime2 = Prime2[Prime2.Chr != ""]


#Specificiranje GC i GT vrijednosti
Prime2 = Prime2[~(Prime2.GCScore < 0.7)]
Prime2 = Prime2[~(Prime2.GTScore <= 0.4)]


Prime3 = Prime2[["SampleID", "SNPName", "Chr", "Allele1AB", "Allele2AB",
"cmunit", "bp_position"]] #Pravljenje jos jedne kopije
Prime3 = Prime3.sort_values(by = ["SNPName"], ascending = [True])
#Sortiranje vrijednosti po SNP imenu
file3gvmap = Prime3.drop_duplicates(subset = "SNPName", keep = "first")
#Micanje duplikata iz novog objekta, priprema za rađenje .map filea, vazno
je keep = "first" da ne izbrise SVE KOJI SU DUPLIKATI

#map file
mapsnpgvhd = file3gvmap[["Chr", "SNPName", "cmunit", "bp_position"]]
#Dodavanje određene varijable u objekt za .map
mapsnpgvhd.to_csv("mr50.map", sep = "\t", header = False, index = False)
#Ispis .map filea, ne izbacujemo sa naslovima varijabla ni indeksom redova,
separator = tab


#lgen file
Prime4 = Prime2.sort_values(by = ["SampleID", "SNPName"]) #Stvaranje novog
objekta, sortiranje prvo po ID-u uzorka, zatim SNP imenu
Prime4["fid"] = 1 #Dodavanje nove varijable fid i zadavanje vrijednosti 1
lgen = Prime4[["fid", "SampleID", "SNPName", "Allele1AB", "Allele2AB"]]
#Uzimanje potrebnih varijabli i odbacivanje nepotrebnih
lgen.to_csv("mr50.lgen", sep = "\t", header = False, index = False) #Ispis
.lgen filea, ne izbacujemo sa naslovima varijabla ni indeksom redova,
separator = tab

#fam file
fam1 = Prime4[["SampleID"]] #Dodavanje ID-uzoraka iz proslog objekta u novi
objekt fam1
```

```
fam1 = fam1.sort_values(by= "SampleID")# Sortiranje ID uzoraka
fam1 = fam1.drop_duplicates(keep = "first") # Brisanje duplikata
fam1["famID"] = "1" #
fam1["otac"] = "000000000000000"#
fam1["majka"] = "000000000000000"     # Stvaranje novih varijabli i
dodijeljivanje vrijednosti
fam1["spol"] = "-1"#
fam1["feno"] = "-9"#
fam1 = fam1[["famID", "SampleID", "otac", "majka", "spol", "feno"]]
#Preslagivanje varijabli u pravilan slijed
fam1.to_csv("mr50.fam",sep = "\t", header = False, index = False) #Ispis
.fam filea, ne izbacujemo sa naslovima varijabla ni indeksom redova,
separator = tab



#plink naredbe
import subprocess as sb #Uvlačenje funkcija podprocessa za pokretanje
aplikacija van programa, sb kao kratica
sb.call("plink.exe --lfile mr50 --sheep --recode --out mr51 --noweb", shell
= False) #na početku navodnika poziva se plink.exe zatim se zadaju željeni
parametri za ciscenje, rekodiranje
#-lfile za uvlačenje 3 prethodno napravljene datoteke, --noweb argument da
ne poziva sa interneta, izvan navodnika shell = False
sb.call("plink.exe --file mr51 --sheep --geno 0.9 --recode --out mr51geno -
-noweb", shell = False) #uvlačenje prethodno napravljenih podataka,
postavljanje parametra --geno(dopusten nedostatak SNP-a)
sb.call("plink.exe --file mr51geno --sheep --mind 0.05 --recode --out
mr51mind --noweb", shell = False) #uvlačenje prethodno napravljenih
podataka, postavljanje parametra --mind(dopusten nedostatak genotipa)

#cgaTOH naredbe
import subprocess as sb
sb.call("TOH.exe -map mr51mind -p mr51mind -o roh_1 -l 15 -min_length
1000000 -max_gap 1000000 -max_missing 0 -max_hetero 0 -skip_clustering -
force_proceed", shell = False)
sb.call("TOH.exe -map mr51mind -p mr51mind -o roh_2 -l 15 -min_length
2000000 -max_gap 1000000 -max_missing 0 -max_hetero 0 -skip_clustering -
force_proceed", shell = False)
```

```
sb.call("TOH.exe -map mr51mind -p mr51mind -o roh_4 -l 15 -min_length
4000000 -max_gap 1000000 -max_missing 1 -max_hetero 0 -skip_clustering -
force_proceed", shell = False)
sb.call("TOH.exe -map mr51mind -p mr51mind -o roh_8 -l 15 -min_length
8000000 -max_gap 1000000 -max_missing 2 -max_hetero 0 -skip_clustering -
force_proceed", shell = False)
sb.call("TOH.exe -map mr51mind -p mr51mind -o roh_16 -l 15 -min_length
16000000 -max_gap 1000000 -max_missing 4 -max_hetero 1 -skip_clustering -
force_proceed", shell = False)#,
#Uvlačenje prethodno napravljenih .ped i .map datoteka, imenovanje novih
roh1,roh2, oh4#, roh8 i roh16. Postavljanje parametara: -l(dužina TOH-
va),min_length(minimalna fizička dužina TOH-ova) -max_gap(maksimalna
duljina razmaka izmedu obliznjih SNP-ova), -max_missing(broj dozvoljenih
nedostajućih SNP-ova),-max_hetero(maksimalni broj dopustenih heterozigotnih
SNP-ova), -skip_clustering(preskakanje cluster generacije), -
force_proceed(preskakanje potvrde), za sve ostale parametre otvoriti cgaTOH
u cmd-u

#Crtanje plotova
from plotnine import * #Uvođenje plotnine library-a za plotanje(na bazi
ggplot-a)
from matplotlib import * #Uvođenje matplotliba

#ROH1-------------------------------------------------

roh1 = pd.read_table('roh_1.homozygousruns', sep = ",") #Učitavanje prve
roh1 datoteke, specficirati separator
roh1[["FirstPosition"]] = roh1[["FirstPosition"]]/1e+6 #dijeljenje
varijable za početnu poziciju roh-a zbog lakšeg crtanja
roh1[["LastPosition"]] = roh1[["LastPosition"]]/1e+6  #dijeljenje varijable
za zavrsnu poziciju roh-a zbog lakšeg crtanja
roh11 = roh1 #stvaranje kopije za lakse manipuliranje
roh11 = roh11[roh11.Chromosome == 1] ###ODREĐIVANJE KROMOSOMA ZA
CRTANJE(OVISNO KOLIKO IH IMA)

r1= (ggplot(roh11) #Uvlačenje datoteke za crtanje koja je određena za
kromosom
            + geom_segment( aes(x = "FirstPosition", xend = "LastPosition",
y="Label", yend="Label"), size = 5) #Crtanje segmenata i određivanje start
i stop x osi, y je ID jedinki. Size određuje velicinu crtanja segmenata
            + ggtitle("ROH1") #Naslov ROH-a kojeg radimo
```

```
            + labs(x = "Chromosome1", y = "ID") # Oznake za x, y osi
            + geom_vline(xintercept = [50, 100, 150, 200, 250, 300], color
= "red", alpha =  0.3) #stvaranje crvenih poprečnih linija za lakše
snalaženje po plotu, ovisi o veličini, u xintercept se može pisati na
koliko duljine se postavlja crta.
 )

r1.save('ROH1.pdf', height=16, width=18) #Spremanje nacrtanog plota u .pdf-
u, nacrtani plot se sprema u radni direktorij


#ROH2-------------------------------------------------

roh2 = pd.read_table('roh_2.homozygousruns', sep = ",")
roh2[["FirstPosition"]] = roh2[["FirstPosition"]]/1e+6
roh2[["LastPosition"]] = roh2[["LastPosition"]]/1e+6
roh21 = roh2
roh21 = roh21[roh21.Chromosome == 1]

r2 = (ggplot(roh21)
            + geom_segment( aes(x = "FirstPosition", xend = "LastPosition",
y="Label", yend="Label"), size = 5)
            + ggtitle("ROH2")
            + labs(x = "Chromosome1", y = "ID")
            + geom_vline(xintercept = [50, 100, 150, 200, 250, 300], color
= "red", alpha =  0.3)
)

r2.save('ROH2.pdf', height=16, width=18)


#ROH4-------------------------------------------------

roh4 = pd.read_table('roh_4.homozygousruns', sep = ",")
roh4[["FirstPosition"]] = roh4[["FirstPosition"]]/1e+6
roh4[["LastPosition"]] = roh4[["LastPosition"]]/1e+6
roh41 = roh4
roh41 = roh41[roh41.Chromosome == 1]

r4 = (ggplot(roh41)
```

```
            + geom_segment( aes(x = "FirstPosition", xend = "LastPosition",
y="Label", yend="Label"), size = 5)
            + ggtitle("ROH4")
            + labs(x = "Chromosome1", y = "ID")
            + geom_vline(xintercept = [50, 100, 150, 200, 250, 300], color
= "red", alpha =  0.3)
)

r4.save('ROH4.pdf', height=16, width=18)

#ROH8-----------------------------------------------

roh8 = pd.read_table('roh_8.homozygousruns', sep = ",")
roh8[["FirstPosition"]] = roh8[["FirstPosition"]]/1e+6
roh8[["LastPosition"]] = roh8[["LastPosition"]]/1e+6
roh81 = roh8
roh81 = roh81[roh81.Chromosome == 1]

r8 = (ggplot(roh81)
            + geom_segment( aes(x = "FirstPosition", xend = "LastPosition",
y="Label", yend="Label"), size = 5)
            + ggtitle("ROH8")
            + labs(x = "Chromosome1", y = "ID")
            + geom_vline(xintercept = [50, 100, 150, 200, 250, 300], color
= "red", alpha =  0.3)
)

r8.save('ROH8.pdf', height=16, width=18)


#ROH16----------------------------------------------

roh16 = pd.read_table('roh_16.homozygousruns', sep = ",")
roh16[["FirstPosition"]] = roh16[["FirstPosition"]]/1e+6
roh16[["LastPosition"]] = roh16[["LastPosition"]]/1e+6
roh161 = roh16
roh161 = roh161[roh161.Chromosome == 1]

r16 = (ggplot(roh161)
            + geom_segment( aes(x = "FirstPosition", xend = "LastPosition",
y="Label", yend="Label"), size = 5)
```

```
            + ggtitle("ROH16")
            + labs(x = "Chromosome1", y = "ID")
            + geom_vline(xintercept = [50, 100, 150, 200, 250, 300], color
= "red", alpha =  0.3)
)

r16.save('ROH16.pdf', height=16, width=18)
```

8. Životopis

Marko Hrženjak rođen je 29.ožujka 1993. godine u Zagrebu. Pohađao je grafičku školu u Zagrebu u razdoblju od 2007. do 2011. godine. 2011. godine upisao je Agronomski fakultet u Zagrebu, smjer Animalne znanosti, a završio ga je 2015. godine te je stekao titulu inženjera animalnih znanosti (univ.bacc.ing.agr). Te iste godine pri završetku preddiplomskog studija upisao je diplomski studij na istom fakultetu, Genetiku i oplemenjivanje životinja kako bi stekao titulu magistra znanosti. Tijekom studija radio je u Foreo Adria d.o.o kao ispomoć u logistici, odjel za povrat robe. U tom dijelu posla imao je sljedeće zadatke: uređivanje Excel tablica, obrada dnevnih povrata u skladište, komunikacija sa otpremnim kompanijama i planiranje metoda dostave. Također jedan od zadataka je bio i uvrštavanje podataka u tablice, te sortiranje i obrađivanje. Dodatno obrazovanje uključuje; Python (Data Camp), Linux (na svoju ruku) i SQL (Data Camp).