Genome informatics 2021

Student project presentations

May 27, 2021.

Presenters

1. Dunja Đorđević 2020/3346, Tanja Lukić 2020/3183, Stefan Radović 2020/3243
2. Ema Pajić 2020/3013, Nikola Aleksić 2020/3015
3. Aleksandra Panajotu 2020/3359
4. Milan Babić 2019/3441, Petar Jokić 2020/3343
5. Jana Stojanović 2020/3118, Sreten Petronijević 2020/3119
6. Milica Lazić (2020/3203), Pavle Milenković 2020/3016
7. Anja Lukic 2020/3182, Leon Jovanovic 2020/3181
8. Nebojša Jovanović 3073/2020
9. Nada Jovanovic 2019/3312
10. Стефан Кривокапић 3348/2020, Александар Радојковић 3266/2020
11. Jelena Jakšić 2019/3131, Vukašin Gligorijević 2019/3128
12. Jovica Jovićević 2020/3218

Illumina Pair-End Sequencing Simulator

Dunja Đorđević 2020/3346
Tanja Lukić 2020/3183
Stefan Radović 2020/3243

01

Projektni zadatak

Illumina Pair-End Sequencer

FASTQ fajlovi

SAM fajl

02

DNK sekvenciranje

IZOLOVANJE DNK

PRIPREMA UZORKA

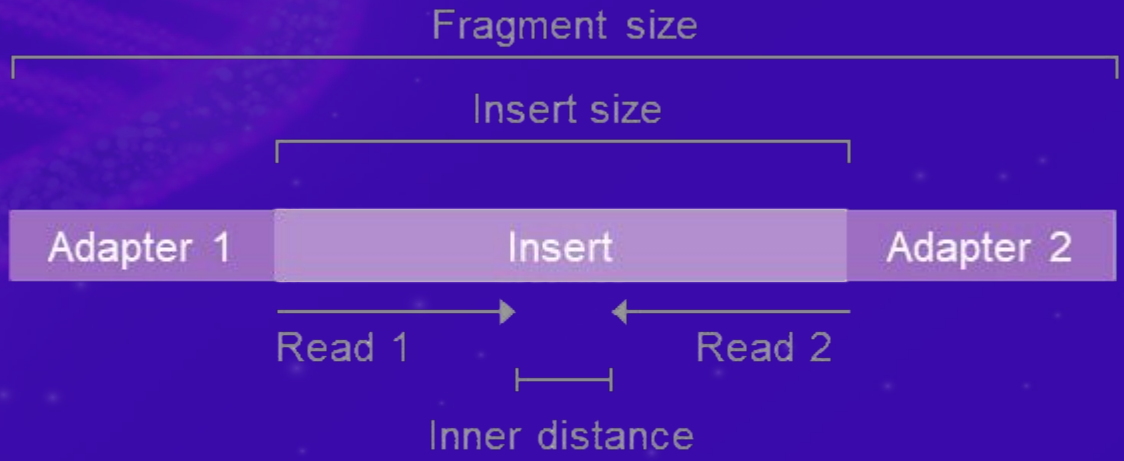
SEKVENCIRANJE

SEKUNDARNA ANALIZA

FRAGMENTI

READ-OVI

Pair-End Read



03

REŠENJE PROJEKTNOG ZADATKA

PARAMETRI SIMULATORA

ULAZNI

1. Referentni genom u FASTA fajlu
2. Prosečan kvalitet nukleotida
3. Pokrivenost (coverage)
4. Dužina read-a
5. Insert size
6. Verovatnoća SNV mutacije
7. Verovatnoća insercije
8. Verovatnoća delecije

IZLAZNI

1. Dva FASTQ fajla
2. SAM fajl

RAD SIMULATORA

ČITANJE NIZA NUKLEOTIDA IZ FASTA FAJLA

ACGTACGTTTACC

MUTIRANJE REFERENTNOG GENOMA (OPCIONO)

ACGTACGTTTACC

ACATACGGTTACC

IZRAČUNAVANJE BROJA FRAGMENATA

$$N = (C * G)/(L * 2)$$

RAD SIMULATORA

IZDVAJANJE PRETHODNO IZRAČUNATOG BROJA FRAGMENATA, SA POZICIJE KOJA SE BIRA UNIFORMNOM RASPODELOM, KAKO BI SE ISPRATILA POKRIVENOST

ACGTACGTTTACCACGTACAAAG

ACGTACGTTT ACGTACAAAG

ČITANJE PAIR-END READ-OVA SA SVAKOG FRAGMENTA PRI ČEMU SE DRUGI READ OBRNUTO KOMPLEMENTIRA

ACGTACGTTT

ACGT AAAC

RAČUNANJE KVALITETA SVAKOG NUKLEOTIDA NA OSNOVU PROSEČNOG KVALITETA I NORMALNE RASPODELE

ACGT

?-EG

RAD SIMULATORA

ISPISIVANJE PAIR-END READ-OVA SINHRONIZOVANO U DVA FASTQ FAJLA

@1/1	@1/2
ACGT	AAAC
+	+
?-EG	/EEF

ISPISIVANJE SAM FAJLA SA PORAVNATIM POZICIJAMA SA KOJIH SU IZVUČENI READ-OVI

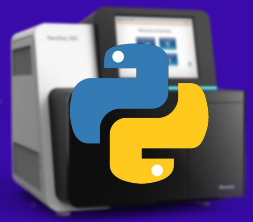
@1/1 1 ACGT ?-EG
@1/2 7 GTTT /EEF

04

BWA-MEM i BOWTIE alati

USPEŠNOST ALATA

Name	
COPY Bowtie2 Indexer	Bowtie2 Indexer is a tool for indexing reference genomes of any size used in an alignment. It was b…
COPY Bowtie2 Aligner	Bowtie 2 is an ultrafast and memory-efficient tool for aligning sequencing reads to long reference se…
COPY BWA INDEX	BWA INDEX constructs the FM-index (Full-text index in Minute space) for the reference genome. Gener…
COPY BWA MEM Bundle	**BWA MEM** is an algorithm designed for aligning sequence reads onto a large reference geno…



SAM

SAM

REZULTATI

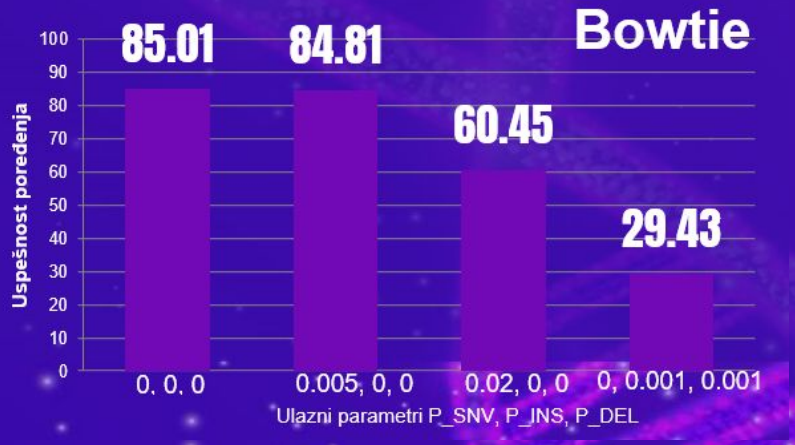
Triticum aestivum

Oryza sativa

Pandanus tectorius

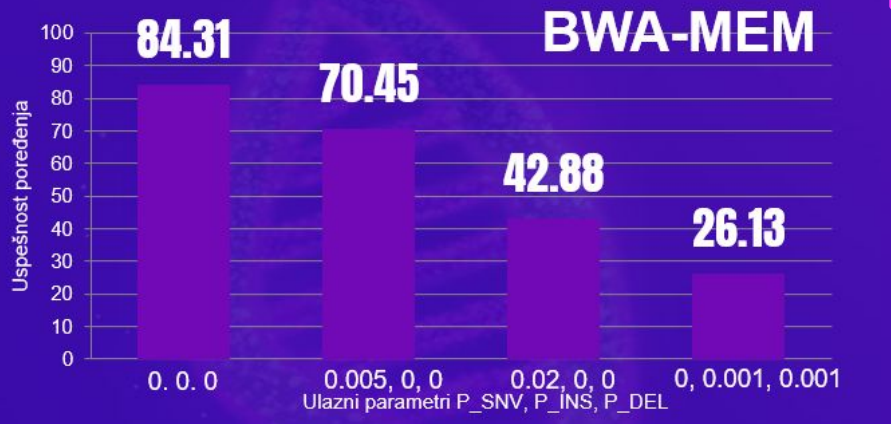
Triticum aestivum

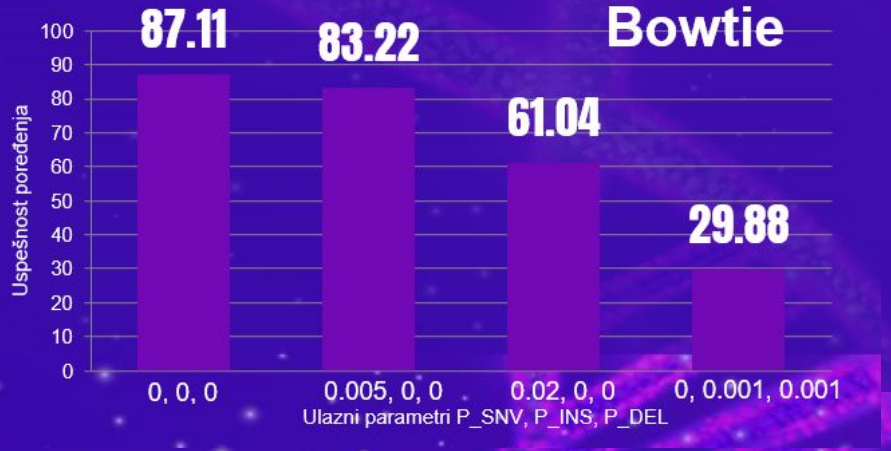
Triticum aestivum



Oryza sativa

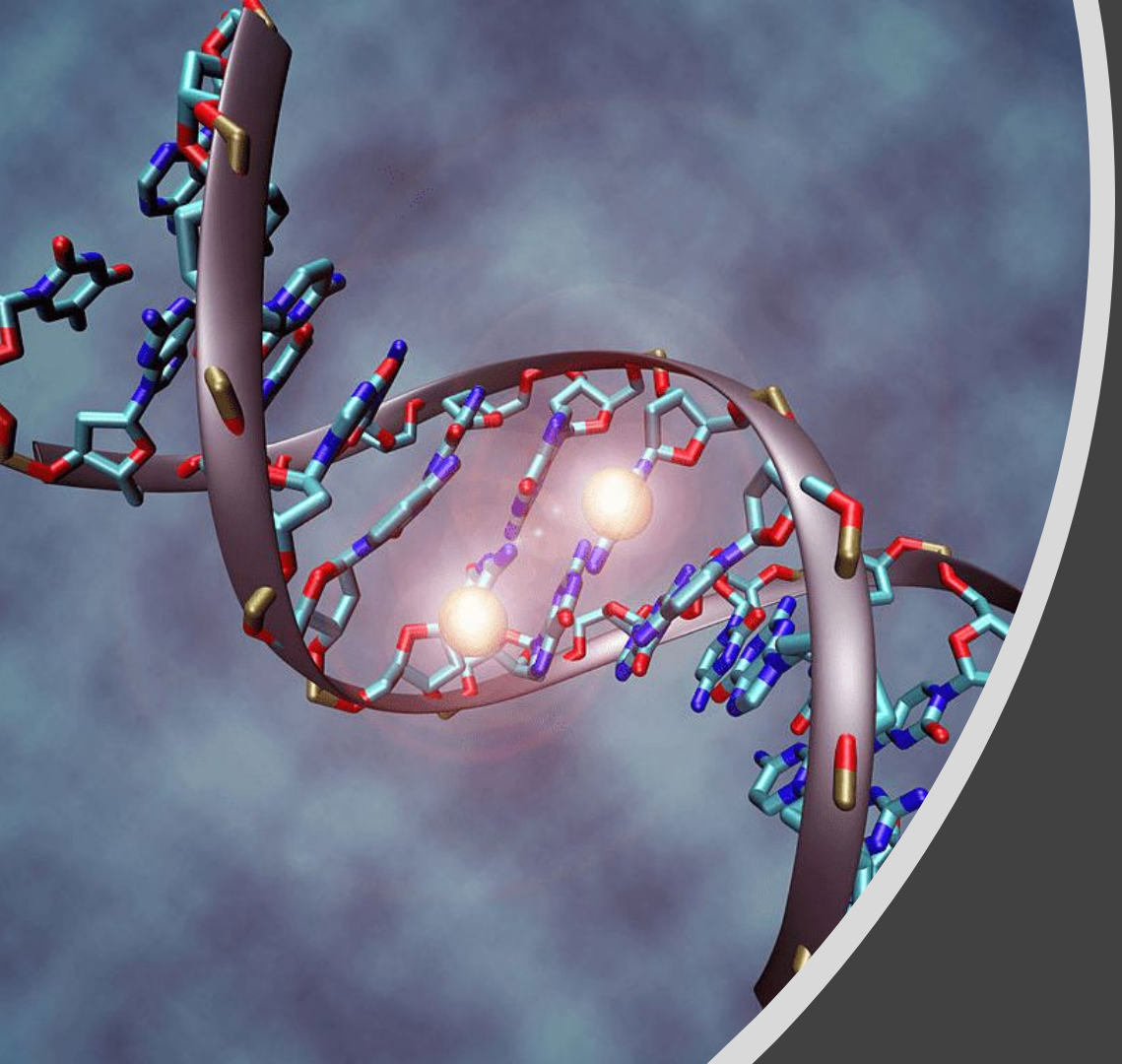
Oryza sativa





Pandanus tectorius

Pandanus tectorius



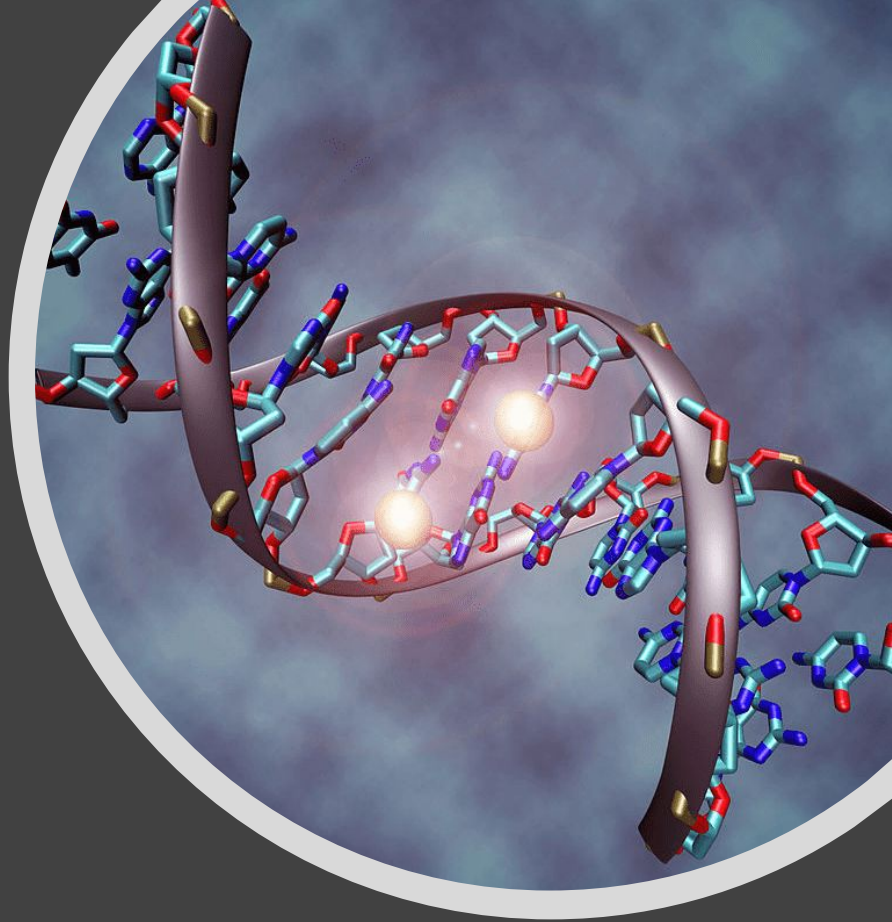
Variant Calling Algorithm based on Binomial Distribution

Ema Pajić 2020/3013

Nikola Aleksić 2020/3015

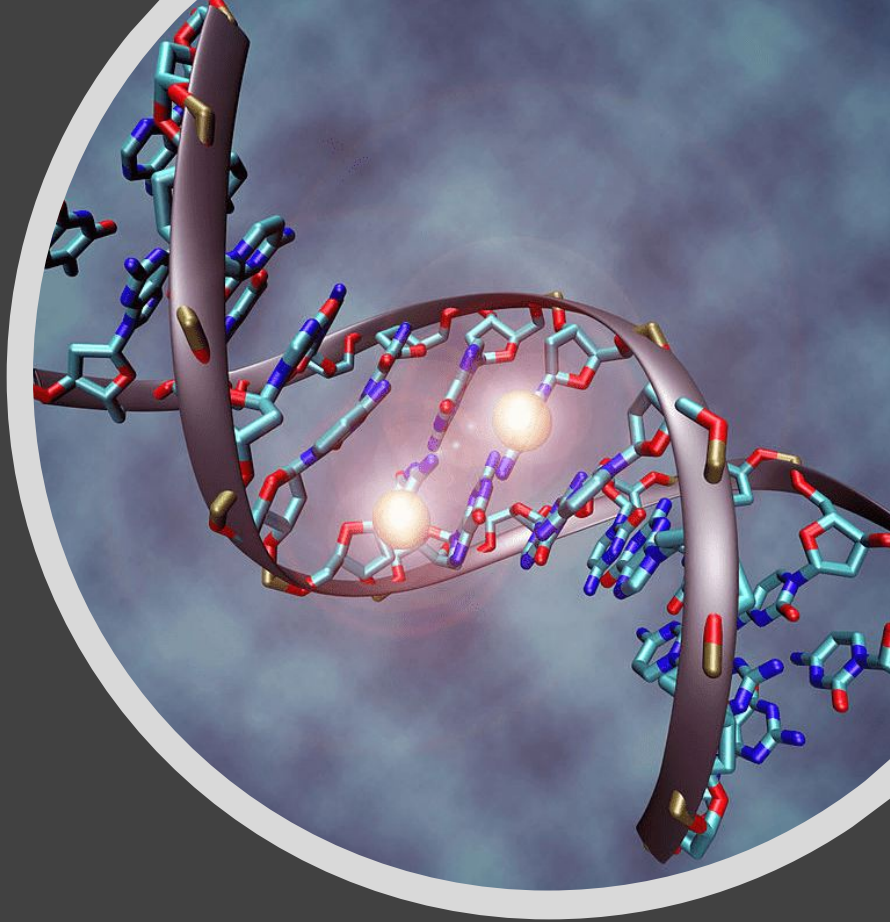
Task summary

- Implement Variant Calling algorithm in Python
- Input format: Pileup
- Output format: VCF 4.2
- Data: merged-normal.BAM file, referent genome human_g1k_v37_decoy.fasta
- Compare results with bfctools call tool



Introduction

- Variant calling is the process of finding differences between reference genome and observed sample
- We need aligned reads to the reference genome so we can find – “call” variants
- Different types of genomic variants

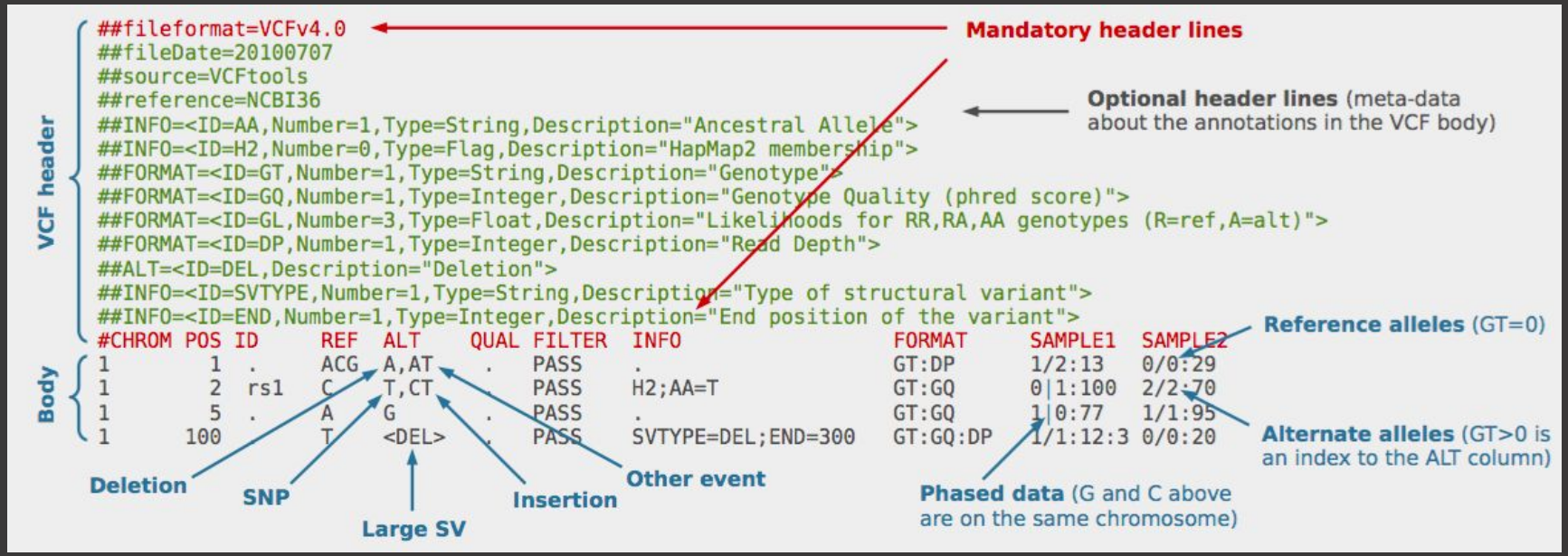


Pileup file

Sequence	Position	Reference Base	Read Count	Read Results	Quality
seq1	272	T	24	,.$....,,.,.,...,,,.,..^+.	<<<+;<<<<<<<<<<=<;<;7<&
seq1	273	T	23	,.....,,.,.,...,,,.,..A	<<<;<<<<<<<<<3<=<<<;<<+
seq1	274	T	23	,.$....,,.,.,...,,,.,...	7<7;<;<<<<<<<<<=<;<;<<6
seq1	275	A	23	,$....,,.,.,...,,,.,...^l.	<+;9*<<<<<<<<<=<<:;<<<<
seq1	276	G	22	...T,,.,.,...,,,.,....	33;+<<7=7<<7<&<<1;<<6<
seq1	277	T	22	....,,.,.,.C.,,,.,..G.	+7<;<<<<<<<&<=<<:;<<&<
seq1	278	G	23	....,,.,.,...,,,.,....^k.	%38*<<;<7<<7<=<<<;<<<<<
seq1	279	C	23	A..T,,.,.,...,,,.,.....	75&<<<<<<<<<=<<<9<<:<<<

Forward	Reverse	Meaning
. dot	, comma	Base matches the reference base
ACGTN	**acgtn**	Base is a mismatch to the reference base
>	<	Reference skip (due to CIGAR "N")
*	*/#	Deletion of the reference base (CIGAR "D")

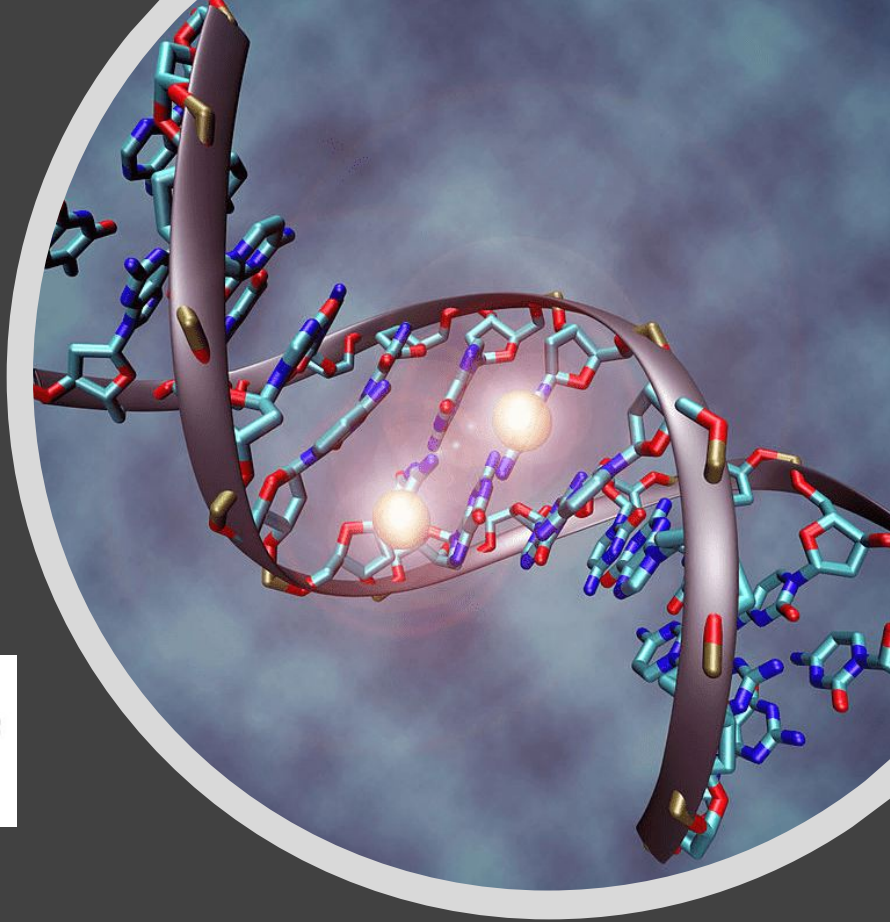
VCF file


VCF header
Body
##fileformat=VCFv4.0
##fileDate=20100707
##source=VCFtools
##reference=NCBI36
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality (phred score)">
##FORMAT=<ID=GL,Number=3,Type=Float,Description="Likelihoods for RR,RA,AA genotypes (R=ref,A=alt)">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##ALT=<ID=DEL,Description="Deletion">
##INFO=<ID=SVTYPE,Number=1,Type=String,Description="Type of structural variant">
##INFO=<ID=END,Number=1,Type=Integer,Description="End position of the variant">
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT SAMPLE1 SAMPLE2
1 1 . ACG A,AT . PASS . GT:DP 1/2:13 0/0:29
1 2 rs1 C T,CT . PASS H2;AA=T GT:GQ 0|1:100 2/2:70
1 5 . A G . PASS . GT:GQ 1|0:77 1/1:95
1 100 . T <DEL> . PASS SVTYPE=DEL;END=300 GT:GQ:DP 1/1:12:3 0/0:20
Mandatory header lines
Optional header lines (meta-data about the annotations in the VCF body)
Reference alleles (GT=0)
Alternate alleles (GT>0 is an index to the ALT column)
Deletion
SNP
Large SV
Insertion
Other event
Phased data (G and C above are on the same chromosome)

Binomial Distribution

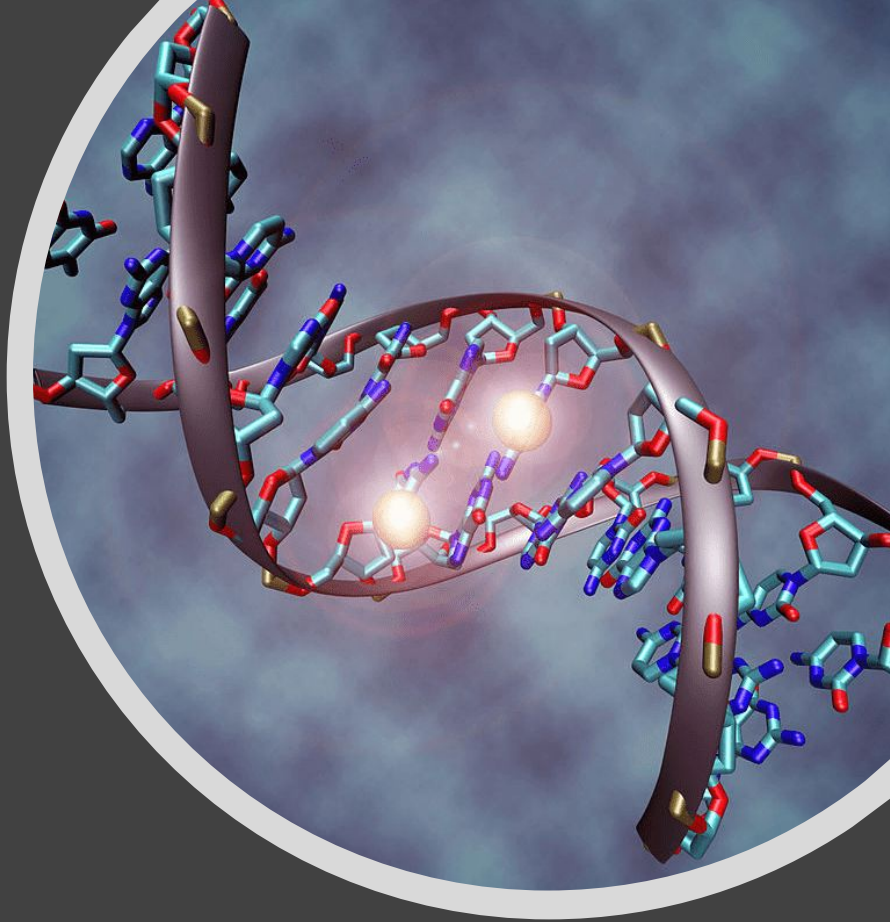
- Models the number of successes in a sequence of yes/no experiments
- Parameters:
  - n - number of trials
  - p - probability of a success in a single trial
  - Probability that K out of n trials will be success

$$f(k;n,p) = \Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$



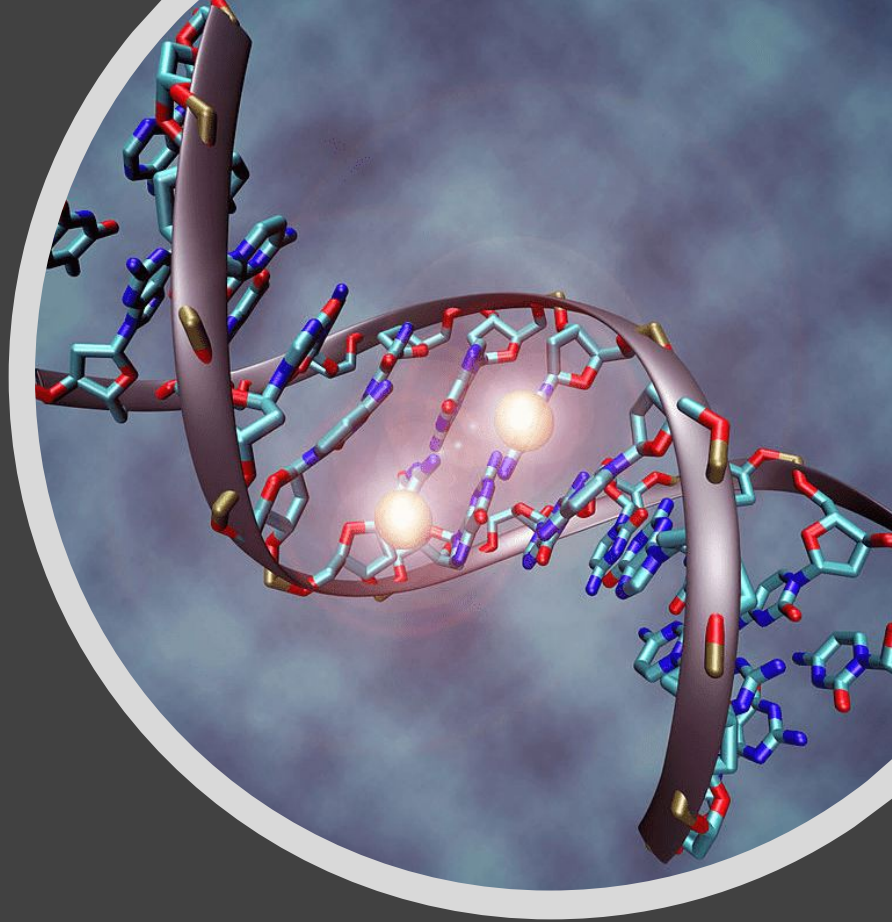
Variant Calling

- Two possible cases:
  - All of the bases in pileup are the same nucleotide [A,T,C,G]
  - Different nucleotides exist in the pileup
- In the simplest case, assume diploidy. There can be only two alleles at a site
- If there are more than two different letters in the pileup we will only consider the most common two (assume others are errors and discard them)



Variant Calling

- So, when we have two letters in the pileup, what should we call?
- Let's call the two "letters" a and b (a, b ∈ [A, C, T, G])
- Let k be the total number of a bases, and n-k number of b bases
- Three possible explanations for the pileup:
  - Genotype is aa; k bases are correct
  - Genotype is bb; n-k bases are correct
  - Genotype is ab; all bases are correct
- Now we need to find the likelihoods of these three cases. Will pick the most likely one!

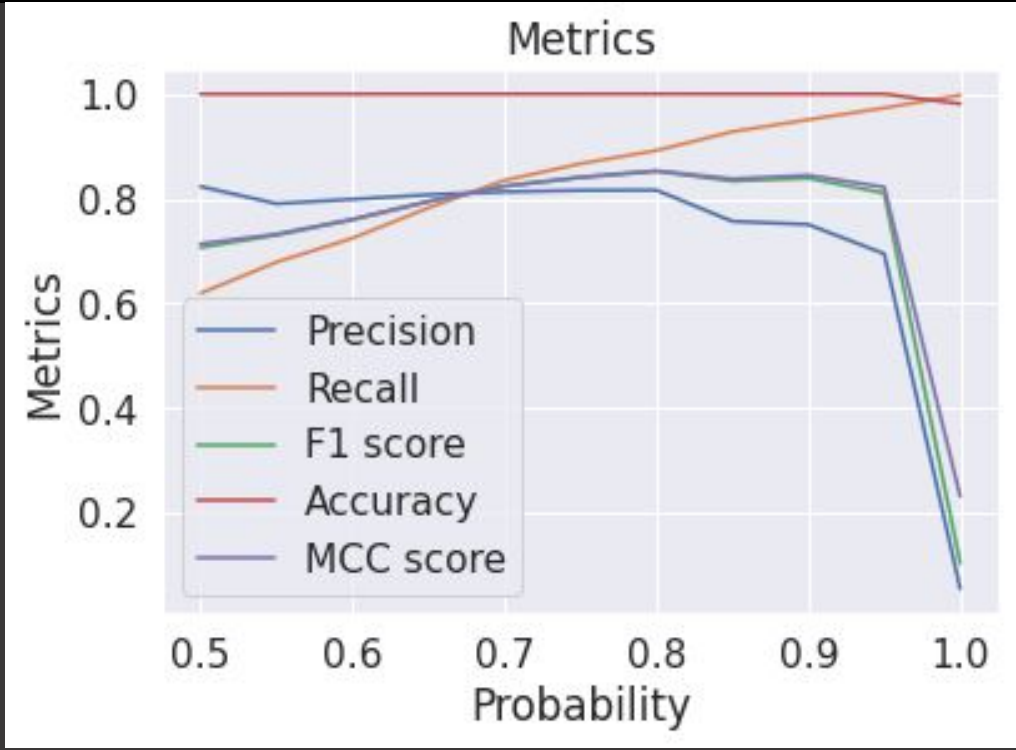


Implemented vs Referent VCF file

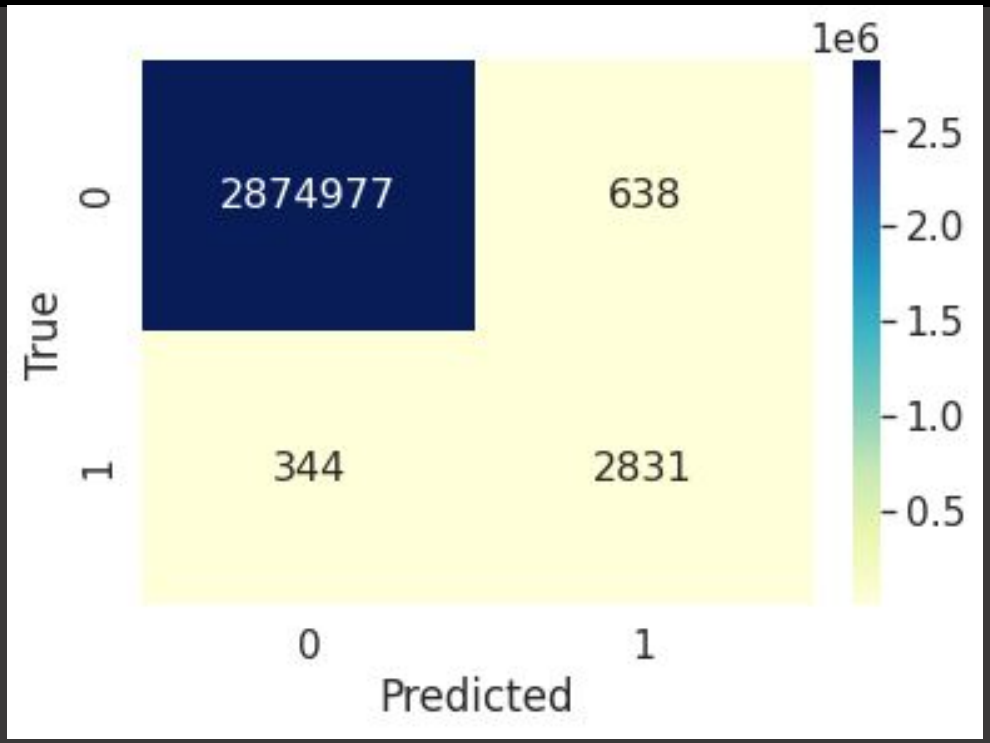
```
21  33038860    .   T   .   .   .   .   GT:VAF  0/0:1
21  33038861    .   G   .   .   .   .   GT:VAF  0/0:1
21  33038862    .   G   .   .   .   .   GT:VAF  0/0:1
21  33038863    .   C   .   .   .   .   GT:VAF  0/0:1
21  33038864    .   G   .   .   .   .   GT:VAF  0/0:1
21  33038865    .   A   C   .   .   .   GT:VAF  0/1:0.8437479842256371
21  33038866    .   T   .   .   .   .   GT:VAF  0/0:1
21  33038867    .   A   .   .   .   .   GT:VAF  0/0:1
21  33038868    .   G   .   .   .   .   GT:VAF  0/0:1
21  33038869    .   C   .   .   .   .   GT:VAF  0/0:1
21  33038870    .   T   .   .   .   .   GT:VAF  0/0:1
21  33038871    .   T   .   .   .   .   GT:VAF  0/0:1
21  33038872    .   T   .   .   .   .   GT:VAF  0/0:1
21  33038873    .   C   .   .   .   .   GT:VAF  0/0:1
21  33038874    .   T   .   .   .   .   GT:VAF  0/0:1
21  33038875    .   G   .   .   .   .   GT:VAF  0/0:1
21  33038876    .   G   .   .   .   .   GT:VAF  0/0:1
21  33038877    .   A   .   .   .   .   GT:VAF  0/0:1
21  33038878    .   G   .   .   .   .   GT:VAF  0/0:1
21  33038879    .   T   .   .   .   .   GT:VAF  0/0:1
21  33038880    .   T   .   .   .   .   GT:VAF  0/0:1
```

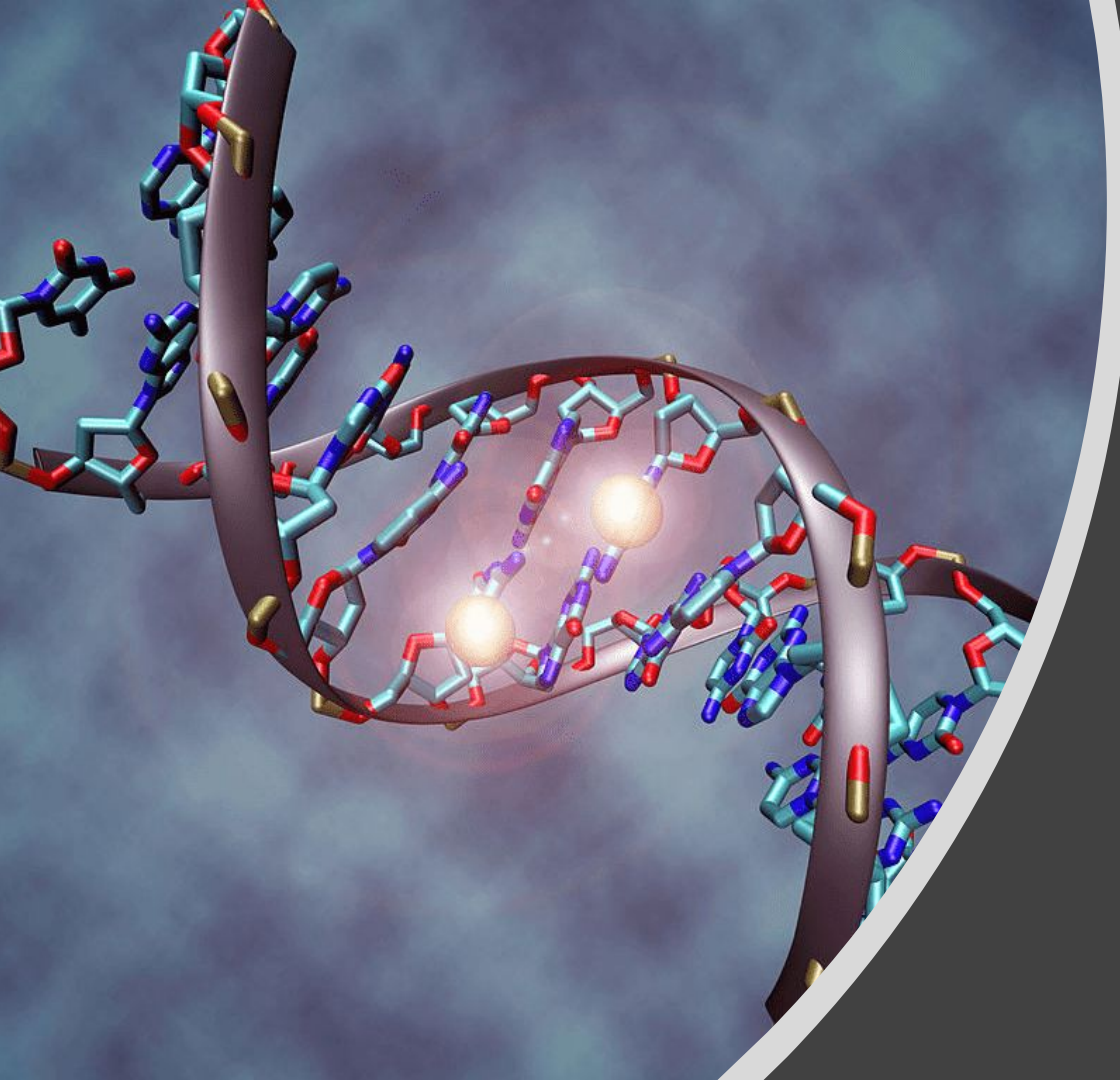
```
21  33038860    .   T   .   101.995 .   DP=25;MQSB=1;MQ0F=0;AF1=0;AC1=0;DP4=3,21,0,0;MQ=60;FQ=-98.9866  GT:PL   0/0:0
21  33038861    .   G   .   98.9947 .   DP=24;MQSB=1;MQ0F=0;AF1=0;AC1=0;DP4=3,20,0,0;MQ=60;FQ=-95.9866  GT:PL   0/0:0
21  33038862    .   G   .   95.9947 .   DP=23;MQSB=1;MQ0F=0;AF1=0;AC1=0;DP4=2,20,0,0;MQ=60;FQ=-92.9865  GT:PL   0/0:0
21  33038863    .   C   .   95.9947 .   DP=22;MQSB=1;MQ0F=0;AF1=0;AC1=0;DP4=2,20,0,0;MQ=60;FQ=-92.9865  GT:PL   0/0:0
21  33038864    .   G   .   95.9947 .   DP=23;MQSB=1;MQ0F=0;AF1=0;AC1=0;DP4=2,20,0,0;MQ=60;FQ=-92.9865  GT:PL   0/0:0
21  33038865    .   A   C   181.009 .
DP=21;VDB=0.114178;SGB=-0.680642;RPB=0.697593;MQB=1;MQSB=1;BQB=0.0211283;MQ0F=0;AF1=0.5;AC1=1;DP4=1,7,1,11;MQ=60;FQ=122.016;PV4=1,1,1,1 GT:PL   0/1:211,0,149
21  33038866    .   T   .   89.9946 .   DP=21;MQSB=1;MQ0F=0;AF1=0;AC1=0;DP4=2,18,0,0;MQ=60;FQ=-86.9865  GT:PL   0/0:0
21  33038867    .   A   .   89.9946 .   DP=21;MQSB=1;MQ0F=0;AF1=0;AC1=0;DP4=2,18,0,0;MQ=60;FQ=-86.9865  GT:PL   0/0:0
21  33038868    .   G   .   92.9947 .   DP=22;MQSB=1;MQ0F=0;AF1=0;AC1=0;DP4=2,19,0,0;MQ=60;FQ=-89.9865  GT:PL   0/0:0
21  33038869    .   C   .   95.9947 .   DP=23;MQSB=1;MQ0F=0;AF1=0;AC1=0;DP4=2,20,0,0;MQ=60;FQ=-92.9865  GT:PL   0/0:0
21  33038870    .   T   .   95.9947 .   DP=23;MQSB=1;MQ0F=0;AF1=0;AC1=0;DP4=2,20,0,0;MQ=60;FQ=-92.9865  GT:PL   0/0:0
21  33038871    .   T   .   95.9947 .   DP=23;MQSB=1;MQ0F=0;AF1=0;AC1=0;DP4=2,20,0,0;MQ=60;FQ=-92.9865  GT:PL   0/0:0
21  33038872    .   T   .   95.9947 .   DP=23;MQSB=1;MQ0F=0;AF1=0;AC1=0;DP4=2,20,0,0;MQ=60;FQ=-92.9865  GT:PL   0/0:0
21  33038873    .   C   .   95.9947 .   DP=23;MQSB=1;MQ0F=0;AF1=0;AC1=0;DP4=2,20,0,0;MQ=60;FQ=-92.9865  GT:PL   0/0:0
21  33038874    .   T   .   95.9947 .   DP=23;MQSB=1;MQ0F=0;AF1=0;AC1=0;DP4=2,20,0,0;MQ=60;FQ=-92.9865  GT:PL   0/0:0
21  33038875    .   G   .   92.9947 .   DP=22;MQSB=1;MQ0F=0;AF1=0;AC1=0;DP4=1,20,0,0;MQ=60;FQ=-89.9865  GT:PL   0/0:0
21  33038876    .   G   .   92.9947 .   DP=21;MQ0F=0;AF1=0;AC1=0;DP4=0,21,0,0;MQ=60;FQ=-89.9865 GT:PL   0/0:0
21  33038877    .   A   .   92.9947 .   DP=21;MQ0F=0;AF1=0;AC1=0;DP4=0,21,0,0;MQ=60;FQ=-89.9865 GT:PL   0/0:0
21  33038878    .   G   .   92.9947 .   DP=21;MQ0F=0;AF1=0;AC1=0;DP4=0,21,0,0;MQ=60;FQ=-89.9865 GT:PL   0/0:0
21  33038879    .   T   .   92.9947 .   DP=21;MQ0F=0;AF1=0;AC1=0;DP4=0,21,0,0;MQ=60;FQ=-89.9865 GT:PL   0/0:0
21  33038880    .   T   .   92.9947 .   DP=21;MQ0F=0;AF1=0;AC1=0;DP4=0,21,0,0;MQ=60;FQ=-89.9865 GT:PL   0/0:0
```

Metrics


Metrics
1.0
0.8
0.6
0.4
0.2
Metrics
Precision
Recall
F1 score
Accuracy
MCC score
0.5
0.6
0.7
0.8
0.9
1.0
Probability

Confusion matrix for p = 0.8





Thank you for your attention!

Questions?

Ema Pajić:

E-mail: pe203013m@student.etf.bg.ac.rs

Github: https://github.com/EmaPajic

Nikola Aleksić:

E-mail: an203015m@student.etf.bg.ac.rs

Github: https://github.com/doraaki

Comparison of GATK HaplotypeCaller and Freebayes Variant Calling tools

ETF Belgrade
Petar Jokic 2020/3343
Milan Babic 2019/3441

Project

1. Preform Variant Calling on C835.HCC1143.2.converted.realigned.base_recalibrated.bam file and use human_g1k_v37_decoy.fasta as reference genome by using GATK 4 HaplotypeCaller and FreeBayes Variant Calling tools.
2. Compare results given by those two tools. Take that HaplotypeCaller is truth set and Freebayes is test set, and then make detail report in form of diagram how many variants are matched (True positives) , how many of them exist in test set but doesn't exist in truth test (False positives), and also how many of them exist in truth set but doesn't exist in test set (False negatives). Calculate precision recall and F-score metrics. To achieve this use manually written script or tools from internet.

Generated VCF files

File generated by GATK (output.vcf) is truth set.
File generated by Freebayes (var.vcf) is test set.

True positives: Predictions (mutations) which exist in both truth and set VCF file.
False positives: Predictions that exist only in set file.
False negatives: Predictions that exist in truth but doesn't exist in set file.

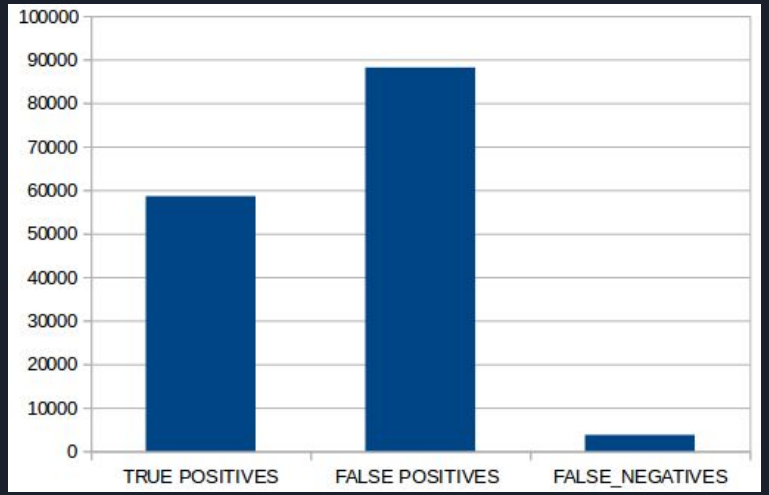
Precision, Recall and F-score formulas:

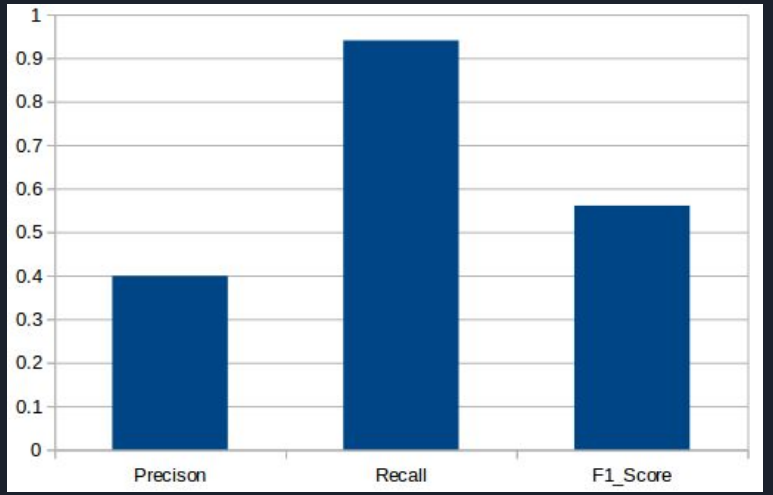
$$P = \frac{TP}{TP + FP} \qquad R = \frac{TP}{TP + FN}$$

$$F = \frac{2PR}{P + R}$$

Graphs

- Comparing VCF files is not easy task

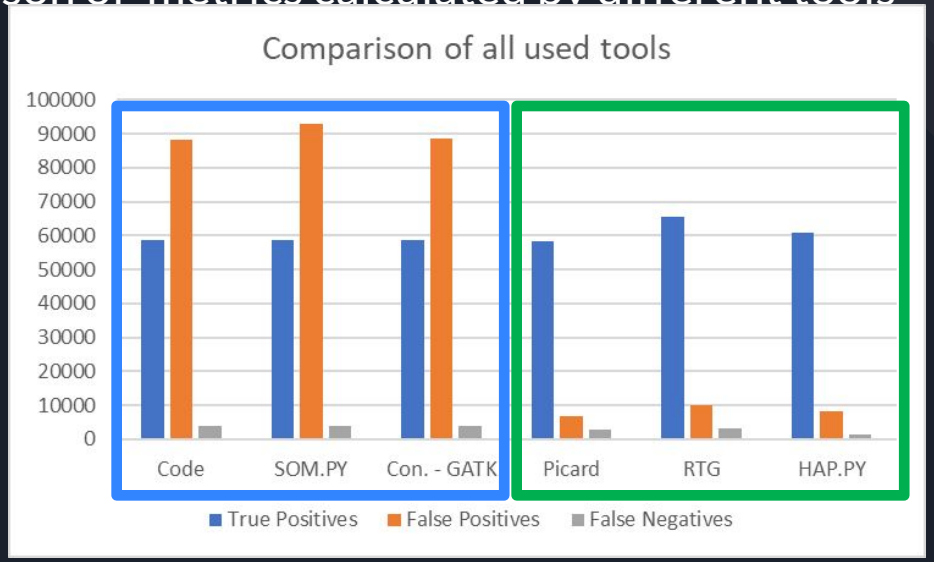


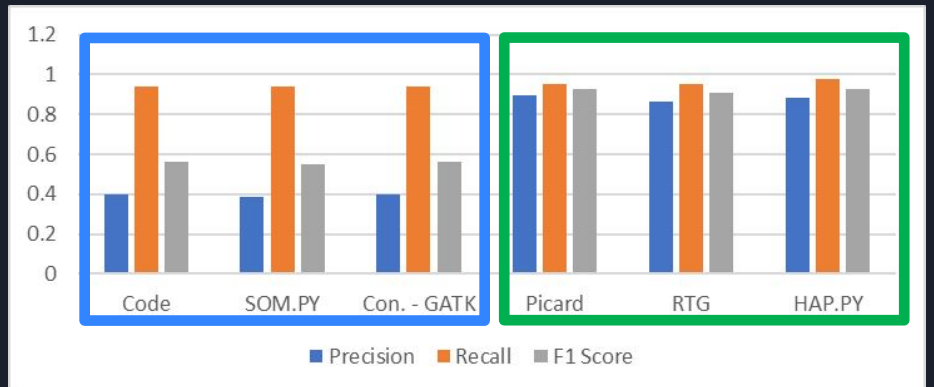


Comparing VCF files

- Performance may vary across variant types and genomic regions due to complexity of human genome
- VCF files – some differences may be just different representations of the same variant
- Definitions for TP, FP, FN are not yet standardized
- We used few more tools or algorithms which are available on CGC – SBGenomics portal as well as which we found online.

Comparison of metrics calculated by different tools





Conclusion

- Evaluating variant calls requires complex matching algorithms and standardized counting, because the same variant may be represented differently in truth and test call sets.
- Sometimes in order to accurately compare VCF files we need to standardize indel representation (left-shifting and trimming the indel alleles), however these methods in other cases could cause errors and more sophisticated comparison methods are required

- Since tools are using different approach and they are not all on same level of complexity there is difference in results

Thank you!

EVALUACIJA KVALITETA VARIANT CALLING-A SB GRAF WORKFLOW-A

Genomska Informatika

Stojanović Jana 2020/3118
Petronijević Sreten 2020/3119

Variant Calling

- Pronalazenje razlika izmedju ***referentnog genoma*** i posmatranog uzorka
- ***SNP*** – jednonukleotidni polimorfizam
- ***Insercije i delecije (INDEL)***
- ***VCF fajl*** – spisak mutacija

TGCTGAGA
TGCCGAGA

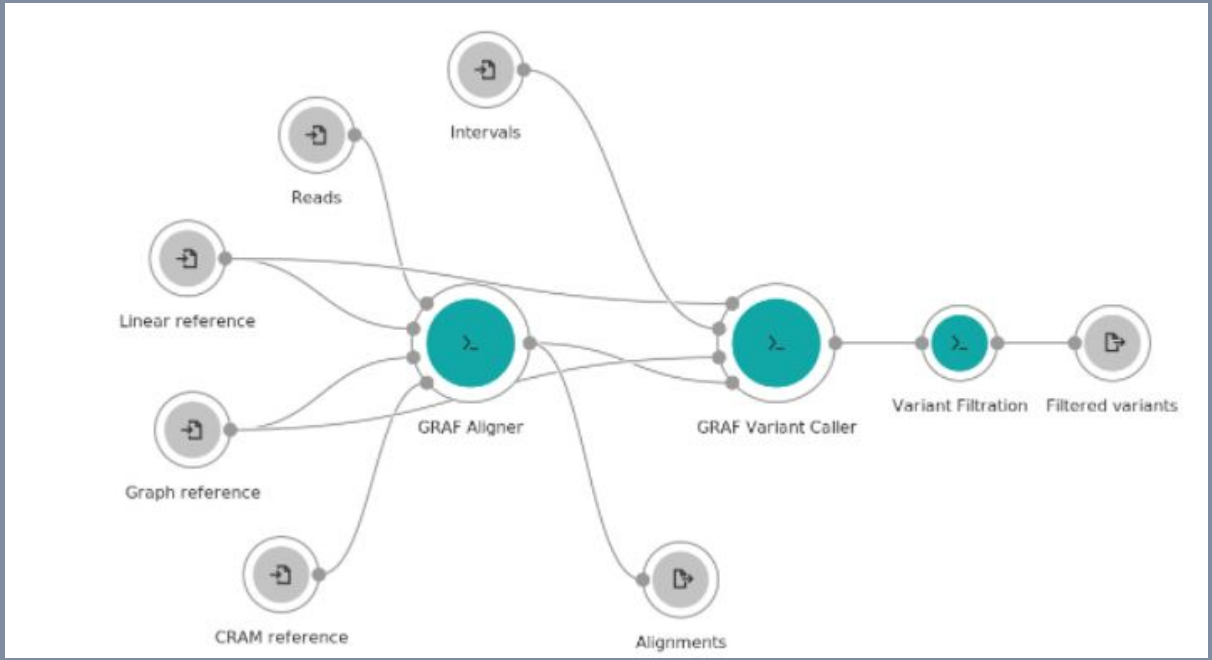
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT SAMPLE1

GRAF Germline Variant Detection Workflow

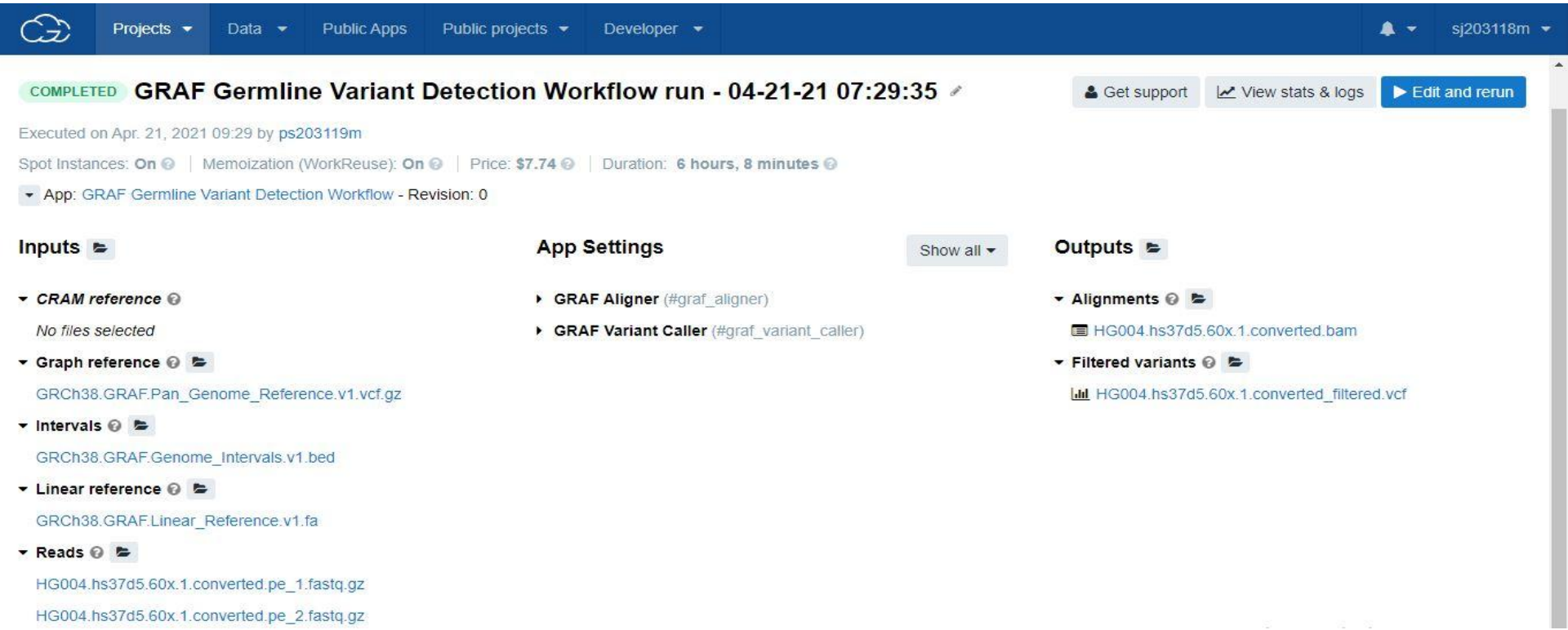
- ***AshkenazimTrio*** – otac, majka i sin
- ***Graf*** – preciznost, brzina izvrsavanja
- Frekvencije mutacija razlicitih naroda
- Input – ***fastq fajlovi***
- Output – ***vcf fajl***

GRAF Germline Variant Detection Workflow

- *Graf Aligner, Variant Calling, Variant Filtration*



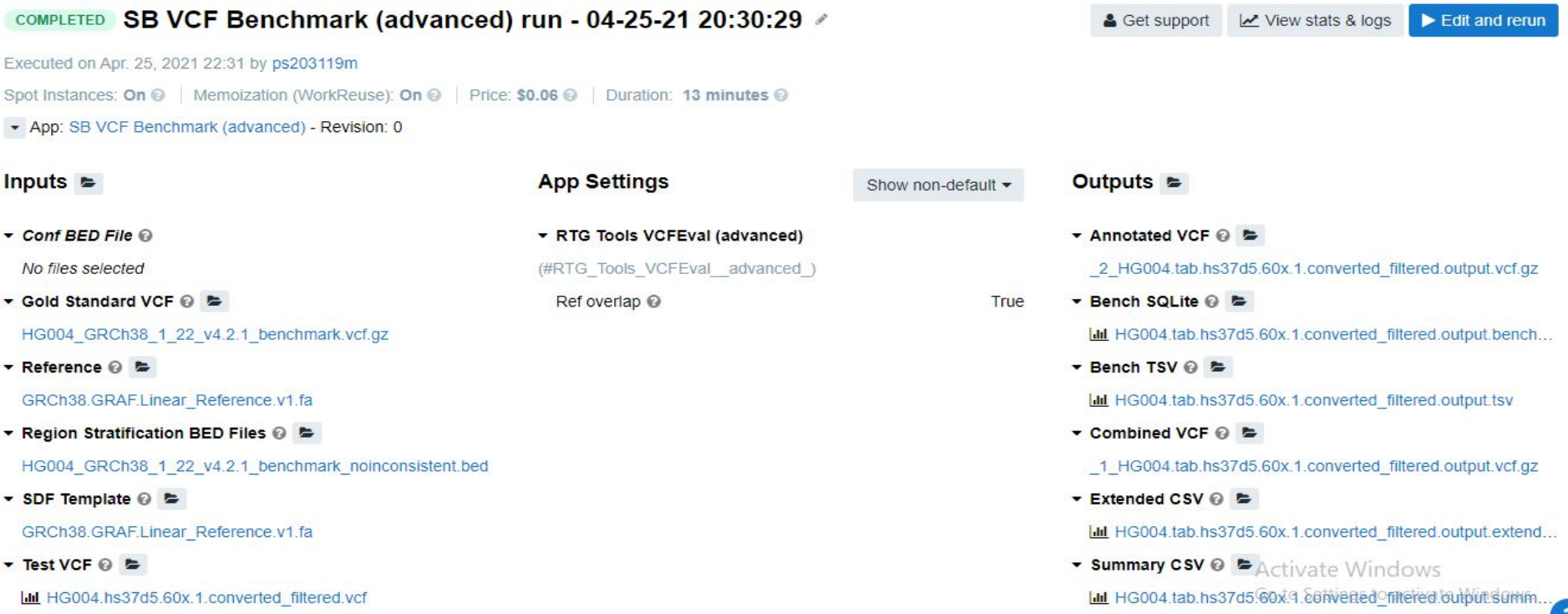
GRAF Germline Variant Detection Workflow



VCF Benchmarking Workflow

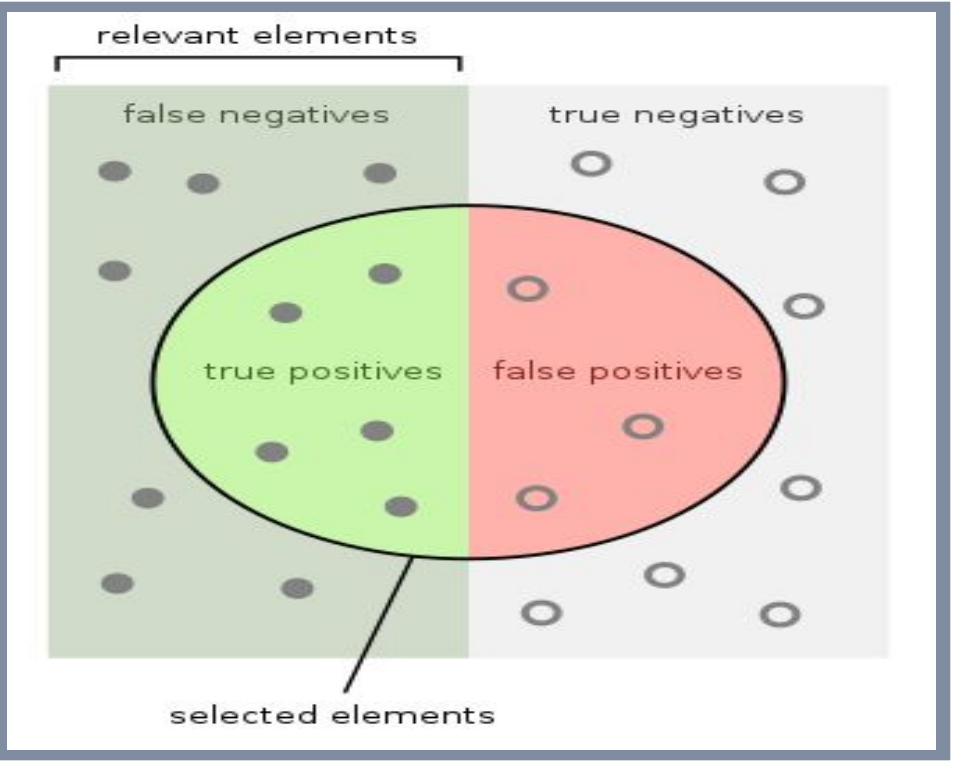
- ***Evaluacija kvaliteta variant calling-a***
- Poredjenje sa ***Gold Standardom***
- ***Bed fajlovi*** (segmenti)
- ***SQLite db fajl***
- Evaluacija kroz 3 metrike

VCF Benchmarking Workflow

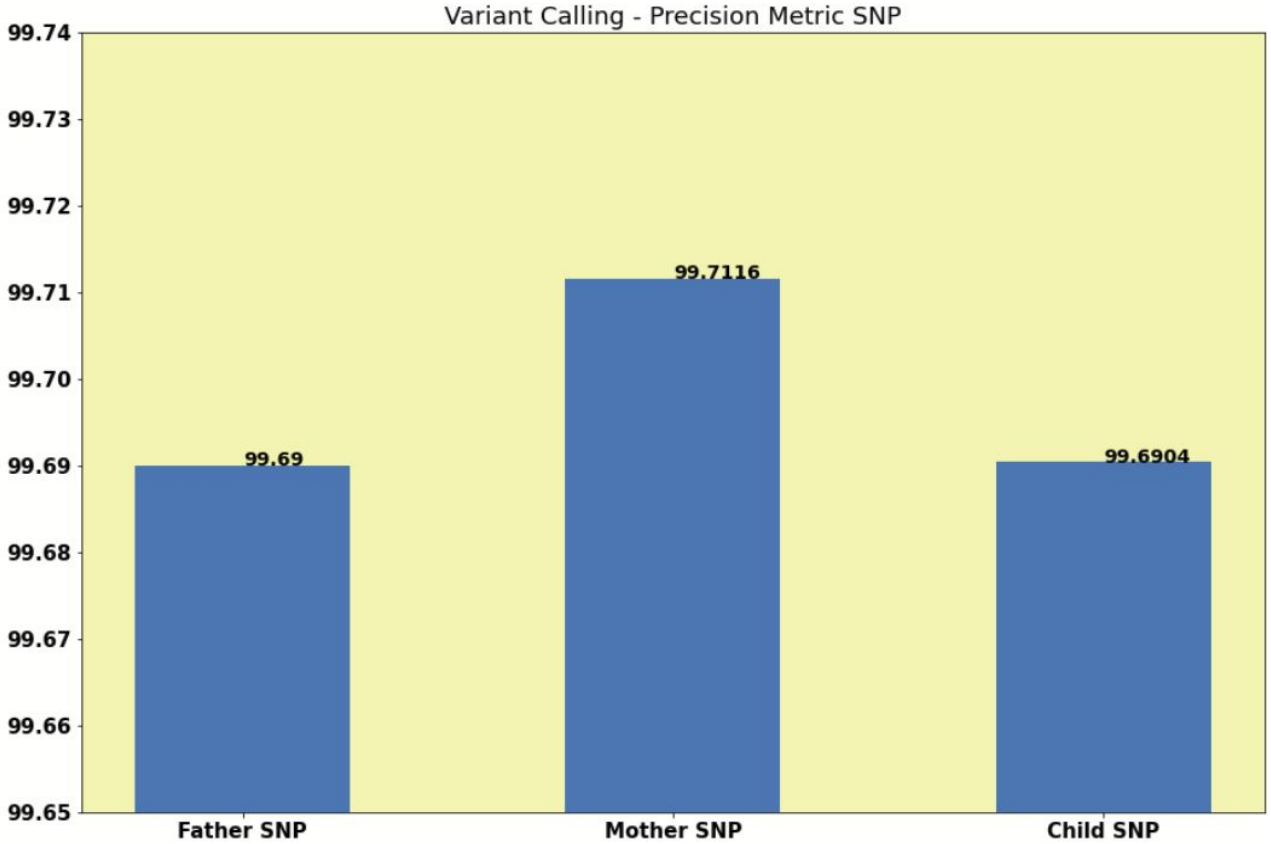


Metrike

- *True Positives*
- *True Negatives*
- *False Positives*
- *False Negatives*



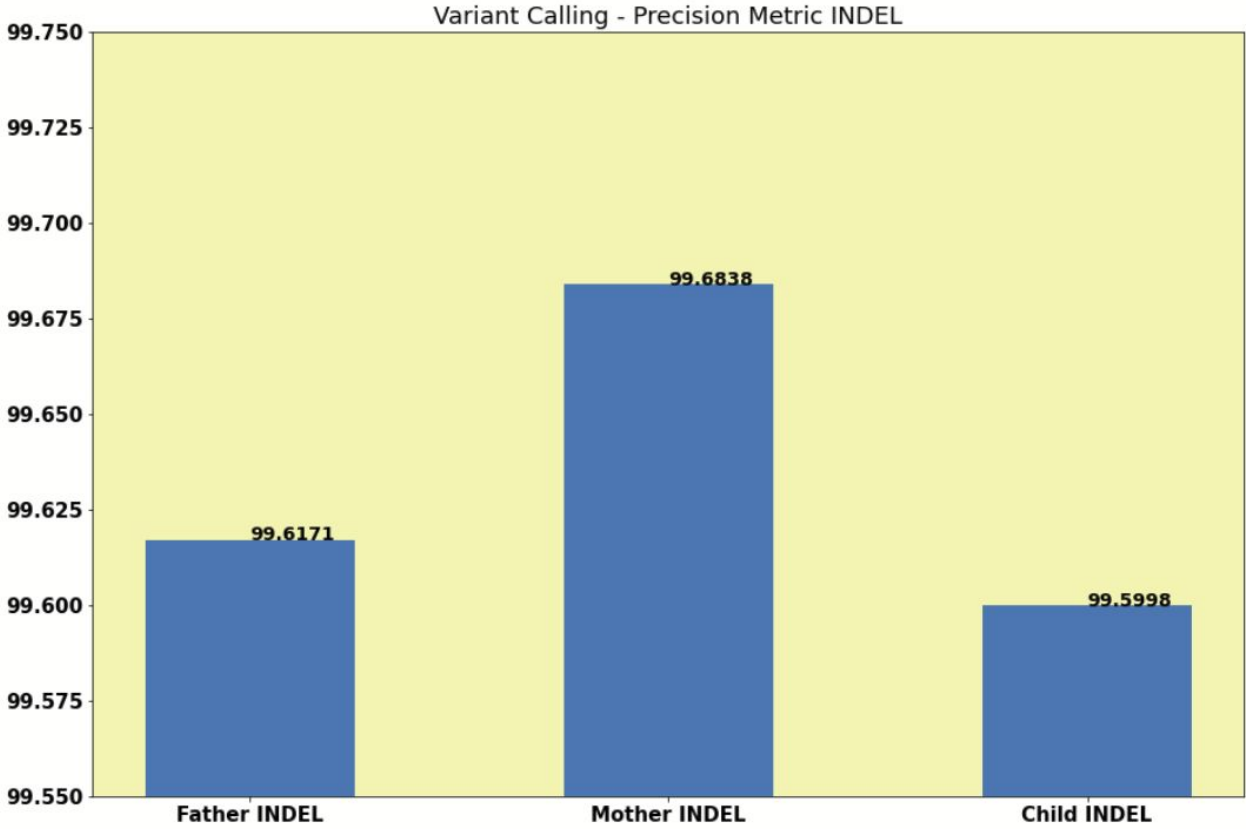
Precision metrika - SNP



Precision =

$$\text{Precision} = \frac{tp}{tp + fp}$$

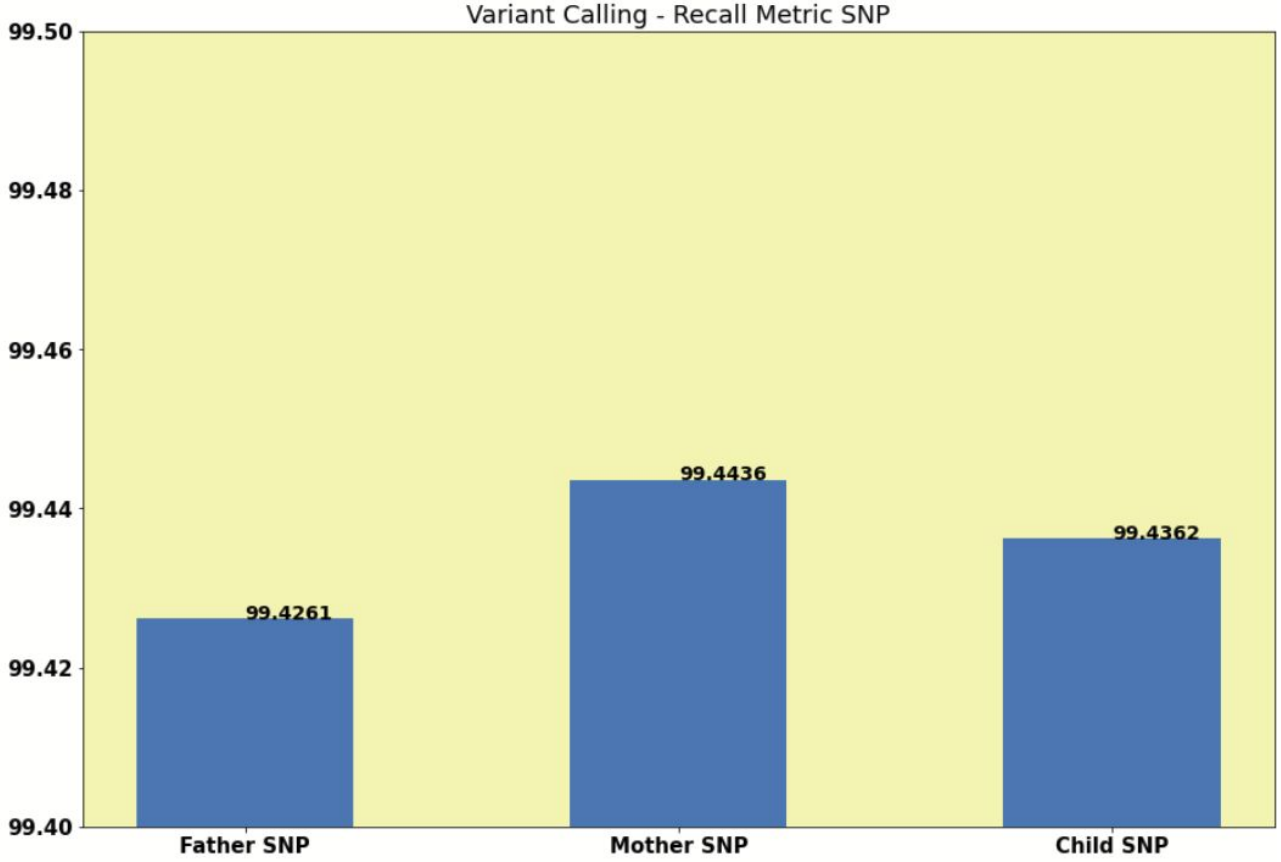
Precision metrika - INDEL



$$\text{Precision} = \frac{\text{●}}{\text{●}}$$

$$\text{Precision} = \frac{tp}{tp + fp}$$

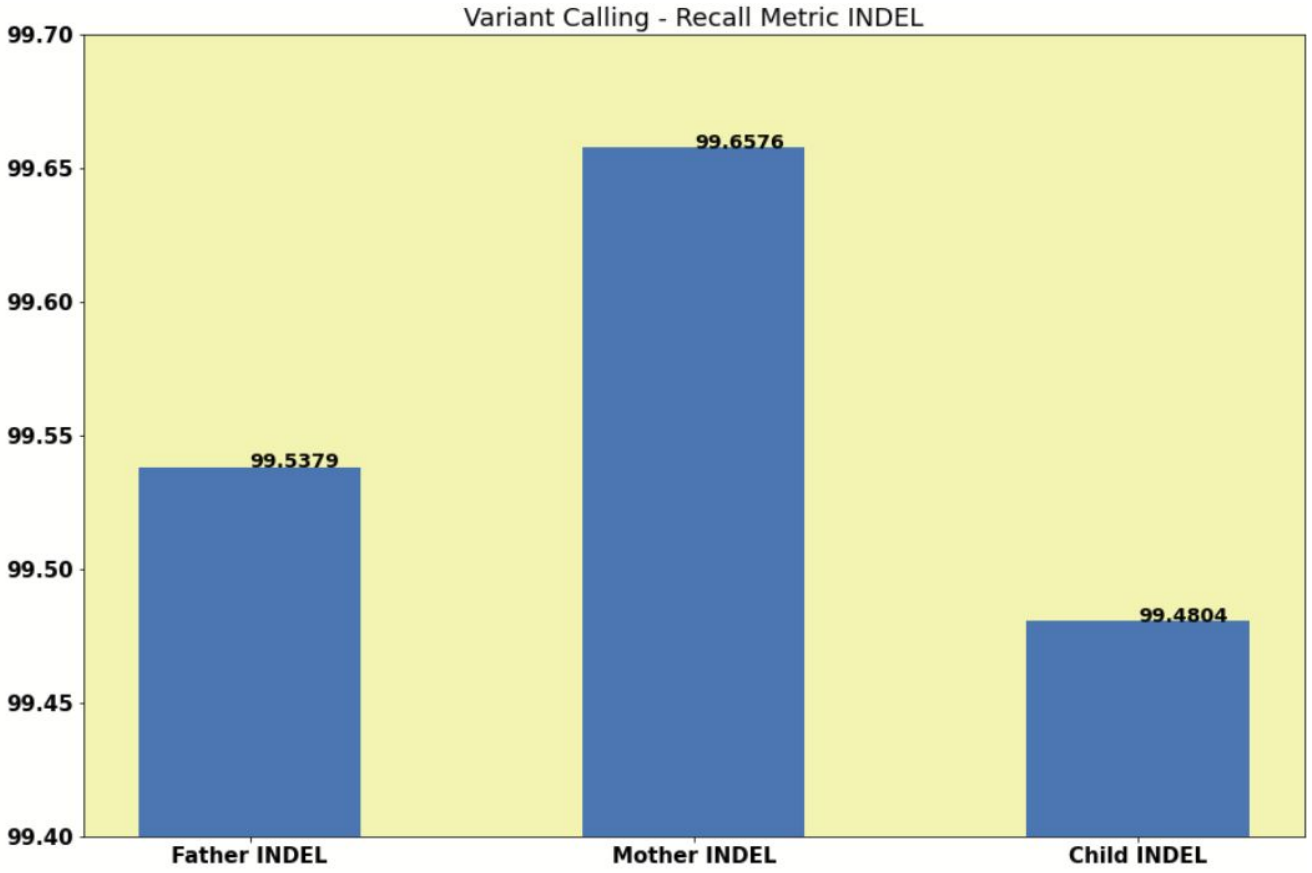
Recall metrika - SNP



Recall = 

$$\text{Recall} = \frac{tp}{tp + fn}$$

Recall metrika - INDEL

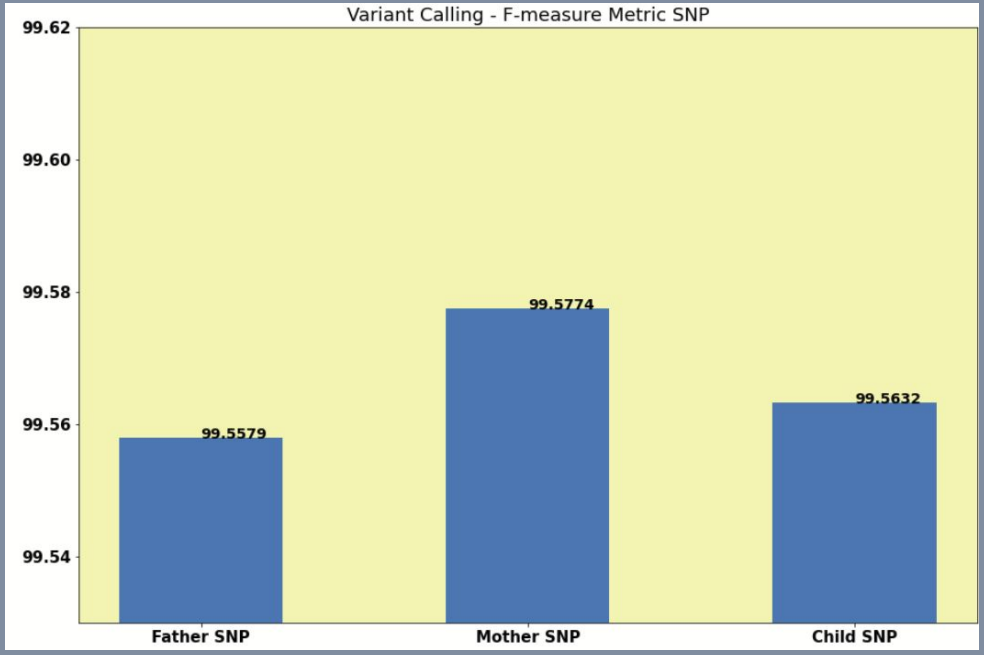


Recall = 

$$\text{Recall} = \frac{tp}{tp + fn}$$

F-measure metrika - SNP

- *Harmonijska sredina za Precision I Recall Metriku*

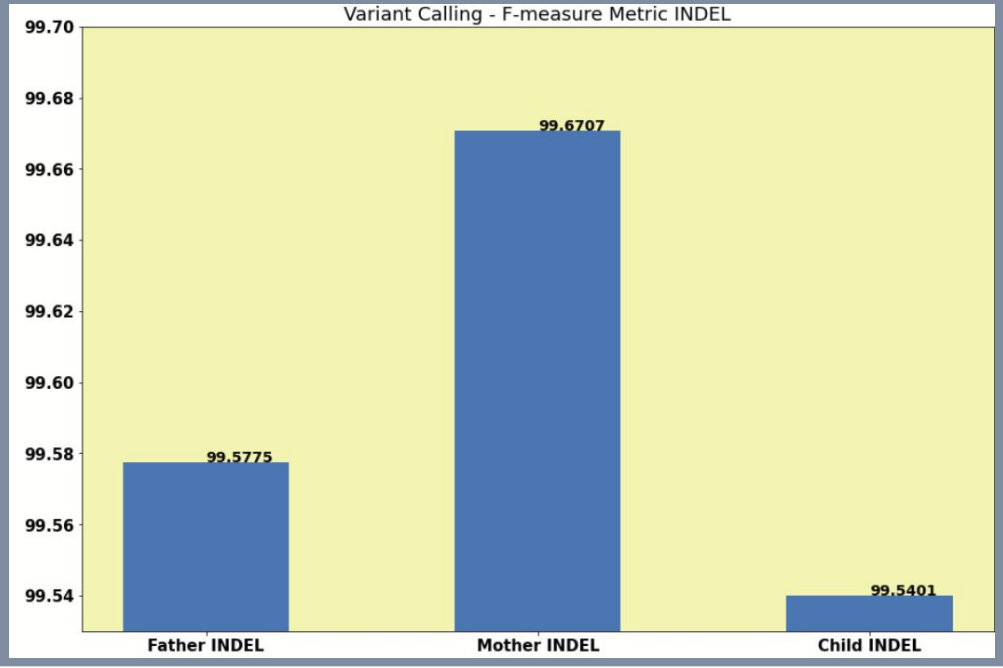


- *F1, F2, F0.5*

$$F = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

F-measure metrika - INDEL

- *Harmonijska sredina za Precision I Recall Metriku*



- *F1, F2, F0.5*

$$F = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

Analiza i zaključak

- *Visoki rezultati metrika -> precizan variant calling*
- *Velika tačnost u odnosu na Gold Standard*
- *Visok precision, nizak recall -> nizak FP, visok FN*
- *Visok recall, nizak precision -> nizak FN, visok FP*
- *Za balansiranje Precision i Recall metrike -> F metrika*

HVALA NA PAŽNJI!

Gestational age prediction model in pregnant women based on gene expression

Nebojša Jovanović 2020/3073

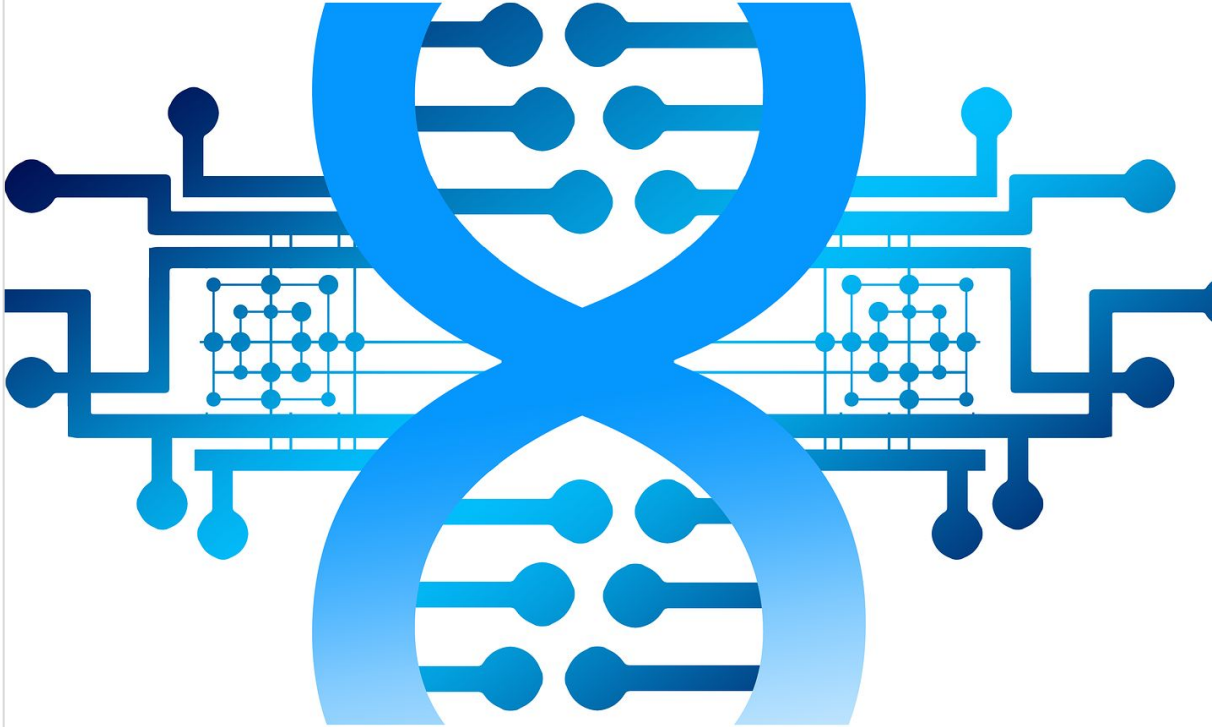


Image by Gerd Altmann from Pixabay

- Determining fetus gestational age is the basic need in pregnancy care
- Standard methods rely on the calculation of the last menstrual cycle and/or ultrasound
- New method?

Image by mohamed Hassan from Pixabay

- The data for this project was taken from [DREAM Preterm Birth Challenge] which was held from May till November in 2019
- 735 samples and 32 830 features (gene expression)
- The project included **367** training and **368** testing samples that were used to build and test our ML model

```
[3] anno = pd.read_csv('anoSC1_v11_nokey.csv', delimiter = ',', index_col = 0)
    anno.head()
```

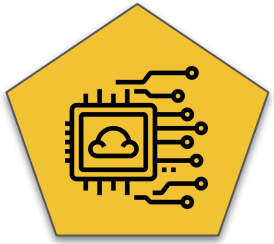
SampleID	GA	Batch	Set	Train	Platform
Tarca_001_P1A01	11.0	1	PRB_HTA	1	HTA20
Tarca_013_P1B01	15.3	1	PRB_HTA	1	HTA20
Tarca_025_P1C01	21.7	1	PRB_HTA	1	HTA20
Tarca_037_P1D01	26.7	1	PRB_HTA	1	HTA20
Tarca_049_P1E01	31.3	1	PRB_HTA	1	HTA20

```
[4] HTA20_RMA = pd.read_csv('HTA20_RMA.csv', delimiter = ',', index_col = 0).
    HTA20_RMA.head()
```

	1_at	10_at	100_at	1000_at	10000_at	100009613_at
Tarca_001_P1A01	6.062215	3.796484	5.849338	3.567779	6.166815	4.443027
Tarca_003_P1A03	6.125023	3.805305	6.191562	3.452524	5.678373	4.773199
Tarca_004_P1A04	5.875502	3.450245	6.550525	3.316134	6.185059	4.393488
Tarca_005_P1A05	6.126131	3.628411	6.421877	3.432451	5.633757	4.623783
Tarca_006_P1A06	6.146466	3.446812	6.260962	3.477162	5.313198	4.422651

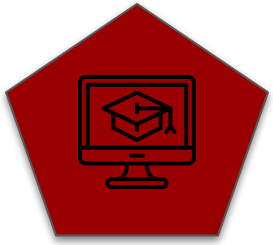
How to build Machine Learning predictive model?

1. Get data



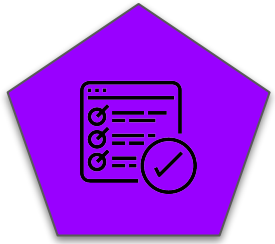
2. Process data

- preprocessed ✅
- **pandas** to prepare data for ML algorithms



3. Train the model

- choose regressor
- choose optimal parameters



4. Test the model

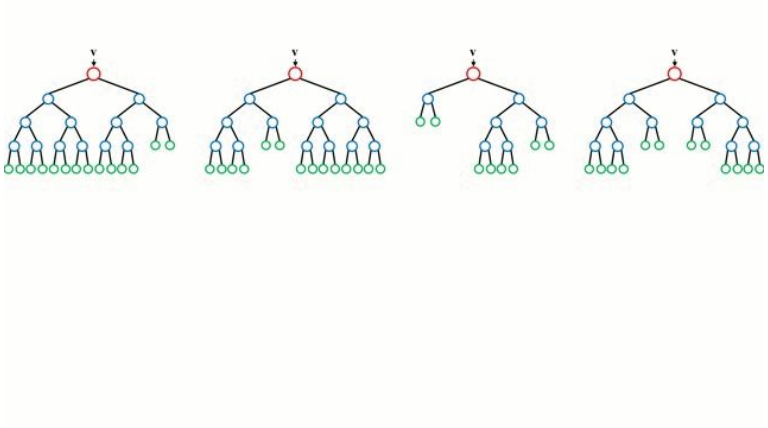
- use metric(s) to test the model

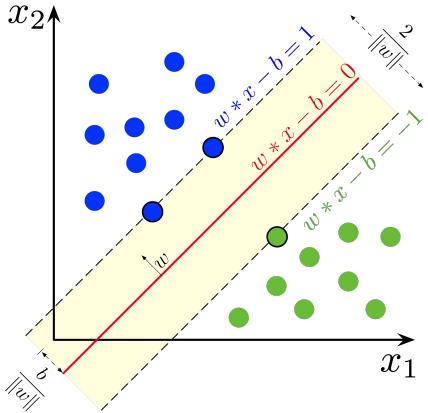
5. Improve!

- use gathered knowledge to improve the model

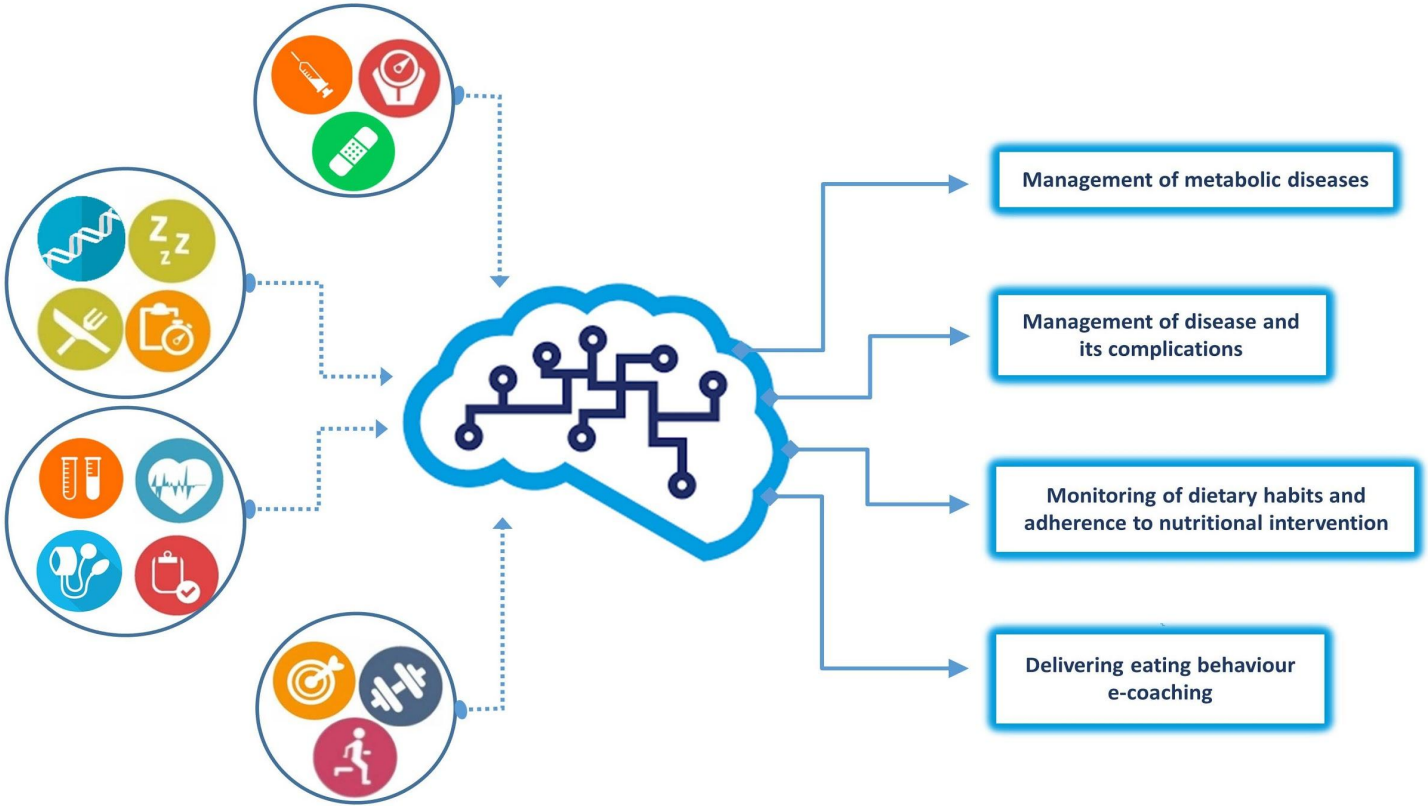
Train the model

- Two initial regressors were chosen:
  - Random Forest Regressor ⟶ *from sklearn.ensemble*
  - Suport Vector Regressor ⟶ *from sklearn.svm*





32 830 genes. How to manage this relatively large number?
Answer: **feature scaling**



Principal component analysis

- PCA is used in data analysis and for making predictive models. It is commonly used for dimensionality reduction by projecting each data point onto only the first few principal components to obtain lower-dimensional data while preserving as much of the data's variance as possible.

- PCA was used in order to extract features to account for **95%** variance

- Sub-optimal parameters for the regressors were chosen via the hyperparameter cross-validation technique, which combines all the possible parameter combinations to determine the best possible model.

Test the model

Metrics and validation

- The main metric was RMSE (root mean square error)

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

- **10-fold** cross-validation was used in order to validate and test the model. Mean RMSE was used as the model error rate

RESULTS

- **RMSE for RFR was 7.5441**
- **RMSE for SVR was 8.4081**

X

- Judging by the errors, these models are pretty bad

Improvement

- The first thing that comes into mind is a bad feature selection
- This time, **SelectKBest** will be used (from *sklearn.feature_selection*)
- Main score? *f_regression*
- Computes F-value for every feature
- Top K features are selected

How to select the best number K?

Like always: **cross-validation**!

➔ Select several numbers for K, and test each model

➔ Select the sub-optimal K

RESULTS: PART II

- **RMSE for RFR was 5.9324**
- **RMSE for SVR was 8.0756**

X

- Better, but can we do even better? SVR is definitely not a suitable regressor for this problem

- Since the Select K Best proved to be a better strategy, and the f_regression test is, in fact, an univariate linear regression test, the logical thing to do is to try a **linear regressor**

- The choice : **ElasticNetCV** from *sklearn.linear_model* that can adjust self parameters via internal cv

RESULTS: PART III

- **RMSE for EN was 5.0400 ✅**

SC1 Final

Submission Id	Submitter	Submitted On	Status	rmse	writeUp
9690190	@redshift	08/15/2019 10:06 PM	SCORED	4.5442	DREAM Preterm Challenge 1 Writeup
9689953	@yuanfang.guan	08/15/2019 1:00 PM	SCORED	4.6849	2019 Preterm Birth Prediction Transcriptomics DREAM Challenge (Y
9688188	@yuanfang.guan	08/08/2019 2:16 AM	SCORED	4.689	2019 Preterm Birth Prediction Transcriptomics DREAM Challenge (Y
9690320	@yuanfang.guan	08/16/2019 6:06 AM	SCORED	4.7124	2019 Preterm Birth Prediction Transcriptomics DREAM Challenge (Y
9687901	@redshift	08/04/2019 1:55 PM	SCORED	4.7292	DREAM Preterm Challenge 1 Writeup
9690321	@yuanfang.guan	08/16/2019 6:13 AM	SCORED	4.7371	2019 Preterm Birth Prediction Transcriptomics DREAM Challenge (Y
9688063	Marienbad	08/06/2019 4:33 PM	SCORED	4.7393	DREAM Preterm Birth Prediction Challenge, Transcriptomics Marienb
9690220	@ams21	08/15/2019 11:37 PM	SCORED	4.7587	DREAM Preterm Birth Prediction Sub-Challenge 1
9686585	@redshift	06/16/2019 12:34 AM	SCORED	4.7811	DREAM Preterm Challenge 1 Writeup
9689956	WhatATeam-PBP	08/15/2019 1:26 PM	SCORED	4.7971	WhatATeam-PBP

Top 10 results from DREAM Subchallenge 1

- Our best RMSE was off by less than 0.5 from the best competition result
- 21st best score

Feature label	Gene symbol	Description	K score
199675_at	MCEMP1	This gene encodes a single-pass transmembrane protein. Based on its expression pattern, it is speculated to be involved in regulating mast cell differentiation or immune responses	72.28
2359_at	FPR3	FPR3 (Formyl Peptide Receptor 3) is a Protein Coding gene. Diseases associated with FPR3 include Rubeosis Iridis. Gene Ontology (GO) annotations related to this gene include G protein-coupled receptor activity and N-formyl peptide receptor activity	62.60
3507_at	IGHM	IGHM (Immunoglobulin Heavy Constant Mu) is a Protein Coding gene. Diseases associated with IGHM include Agammaglobulinemia 1, Autosomal Recessive and Agammaglobulinemia, Non-Bruton Type. Gene Ontology (GO) annotations related to this gene include single-stranded DNA binding and phosphatidylcholine binding	57.93
9619_at	ABCG1	The protein encoded by this gene is a member of the superfamily of ATP-binding cassette (ABC) transporters. ABC proteins transport various molecules across extra- and intra-cellular membranes. ABC genes are divided into seven distinct subfamilies (ABC1, MDR/TAP, MRP, ALD, OABP, GCN20, White). This protein is a member of the White subfamily. It is involved in macrophage cholesterol and phospholipids transport, and may regulate cellular lipid homeostasis in other cell types. Six alternative splice variants have been identified.	57.83
6689_at	SPIB	The protein encoded by this gene is a transcriptional activator that binds to the PU-box (5'-GAGGAA-3') and acts as a lymphoid-specific enhancer. Four transcript variants encoding different isoforms have been found for this gene	52.16

Description was acquired from: https://www.genecards.org/

Conclusion & what next?

- Not a great model overall
- Best competition RMSE is 4.5 weeks

Pictures were taken from the site: https://www.verywellfamily.com/

- **Gather more data**
- Try out more regressors
- Do a deeper hyper-parameter cross-validation

Questions?

e-mail :
nebojsa.php@gmail.com

github:
https://github.com/nebojsa55

Preterm birth prediction based on gene expression from blood

Nada Jovanovic
Student at University of Belgrade, School of Electrical Engineering

nada.jovanovic96@gmail.com
https://github.com/nada-jovanovic/genome-informatics-etf/

Topics

- Preterm birth - introduction
- Available raw data
- Data processing
- Conclusion

Preterm birth - introduction

- Defined as giving birth prior to completion of 37/40 weeks of gestation.
- Leading cause of newborn deaths and long-term complications.

Control - normal delivery

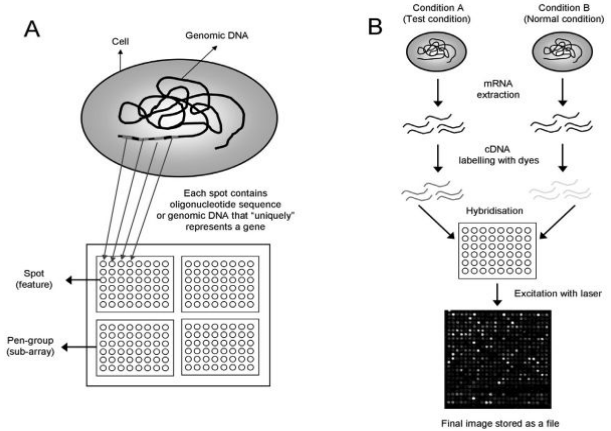
sPTD - spontaneous preterm delivery due to preterm labor

PPROM - premature rupture of membranes

Goal: Predict pregnancy that will result in preterm birth

Available raw data

- Two or more blood samples from 196 women (435 samples total) taken in different GA periods.
- Microarray data (gene expression matrix) for around 29k genes, for each blood sample.
- Samples coming from two platforms: Affymetrix HG2.1ST & Affymetrix HTA20



Available raw data

- Few more details about the gene expression using microarrays:
- Expression level for a gene can be measured as the amount of red or green light emitted.

$$T_k = \frac{R_k}{G_k}$$

expression ratio

Rk represents the spot intensity metric for the test sample and Gk represents the spot intensity metric for the reference sample

Available raw data

- Expression ratios in the matrices are **$\log_2$** transformed (treats differential up-regulation and down-regulation equally, continuous mapping space).
- Same genes are profiled using different platforms, with different blood samples.

Gene expression matrix

	GeneID	GSM1437801	GSM1437802	GSM1437803	GSM1437804	G
0	1_at	4.865497	4.873023	4.786624	4.366512	
1	10_at	2.454080	3.124465	2.814248	2.479630	
2	100_at	7.977459	7.841581	8.605167	8.450711	
3	1000_at	4.163376	4.317975	3.846657	4.210707	
4	10000_at	7.117694	7.306302	7.046300	7.115633	

- Same person has two or more blood samples taken, in the different GA periods.
- Test set for the challenge did not carry an info about the delivery outcome, so the train set (Train = 1) provided by the challenge was only used.

Sample data

	SampleID	IndividualID	GA	GADel	Group	Set	Train	Platform	TTD
0	GSM1437802	515122	28.0	40.0	Control	GSE59491	1	HG21ST	12.0
1	GSM1437801	515122	19.0	40.0	Control	GSE59491	1	HG21ST	21.0
2	GSM1437804	810384	28.0	40.0	Control	GSE59491	1	HG21ST	12.0
3	GSM1437803	810384	18.0	40.0	Control	GSE59491	1	HG21ST	22.0
4	GSM1437806	810392	28.0	41.0	Control	GSE59491	1	HG21ST	13.0

Data processing

Data processing

python 3.8 using pandas & scikit-learn libraries

Steps:

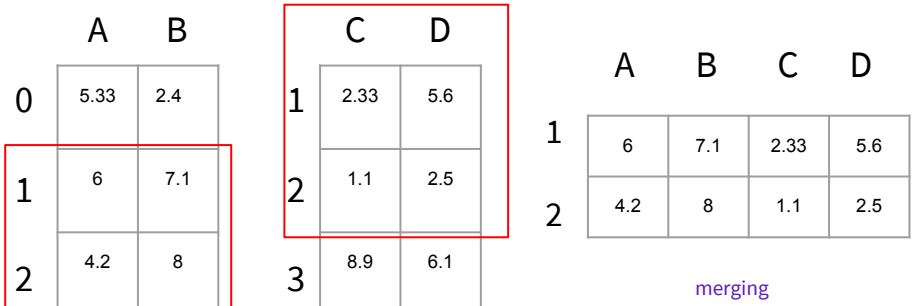
1. Preprocessing (merging & scaling)
2. Extracting relevant data & gene selection (selecting samples & reducing gene set)
3. Making predictions and evaluations using cross-validation

Features: genes

Samples: blood sample ids

Preprocessing step

- Merging two gene expression matrices coming from different platforms.



- Scaling means centering the data around zero by removing the mean value of each feature, then dividing it by their standard deviation.

[sklearn.preprocessing.StandardScaler]

	GeneID	GSM1437801	GSM1437802	GSM1437803	GSM1437804	(
0	1_at	4.865497	4.873023	4.786624	4.366512	
1	10_at	2.454080	3.124465	2.814248	2.479630	
2	100_at	7.977459	7.841581	8.605167	8.450711	
3	1000_at	4.163376	4.317975	3.846657	4.210707	
4	10000_at	7.117694	7.306302	7.046300	7.115633	

scaling

	GSM1437802	GSM1437801	GSM1437804	GSM1437803	GSM1437806	GS
0	-1.022925	-1.033972	-1.766474	-1.149756	-0.803537	
1	-0.051503	-1.152413	-1.110454	-0.560942	-1.184415	
2	0.742741	0.876553	1.342611	1.494719	1.044819	
3	1.559848	1.113354	1.250050	0.198643	1.177585	
4	1.767752	1.475965	1.472775	1.365513	0.800336	

Extracting data & feature selection step

- Extracting the latest sample for individual, since there are two or more samples per person.

	SampleID	IndividualID	GA	GADel	Group	Set	Train	Platform	TTD
0	GSM1437802	515122	28.0	40.0	Control	GSE59491	1	HG21ST	12.0
1	GSM1437801	515122	19.0	40.0	Control	GSE59491	1	HG21ST	21.0
2	GSM1437804	810384	28.0	40.0	Control	GSE59491	1	HG21ST	12.0
3	GSM1437803	810384	18.0	40.0	Control	GSE59491	1	HG21ST	22.0
4	GSM1437806	810392	28.0	41.0	Control	GSE59491	1	HG21ST	13.0

- Feature selection from a large set of genes - reducing the number of genes from 29k to 100 most important ones (50 sPTD, 50 PPROM relevant) to avoid overfitting.

  sklearn.linear_model.Lasso

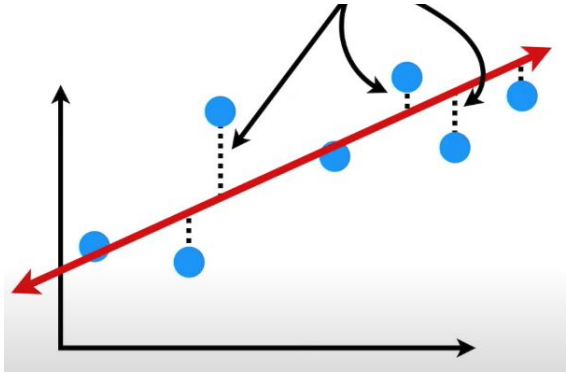
LASSO involves a penalty factor that determines how many features are retained. Using cross-validation to choose the penalty factor helps assure that the model will generalize well to future data samples.

alpha = 0.01

Feature selection using LASSO (gene selection)

- Linear models: fit a straight line to a number of data points, with good bias-variance tradeoff
- The second addition in the final equation adds a **penalty**, where passed parameter alpha controls how severe that penalty is
- Tuning the parameter alpha: Even though CV suggested tuning alpha to ~0.1, better results were achieved when the alpha was reduced ≤0.05. Alpha 0.01 gave the best prediction results.
- Beside Lasso, RFC model was tried out for feature selection, but Lasso performed better.

Square the distances from the fit line to the data points and add them up - **sum of squared residuals**



If the formula for the fit line is: y = **a***x + c

and **alpha** is passed as a parameter to the LASSO model, then...

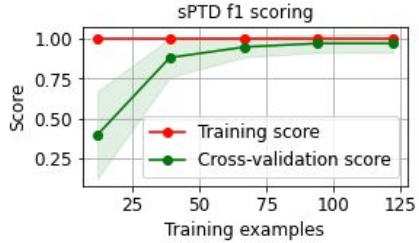
minimize the number derived from the following formula:

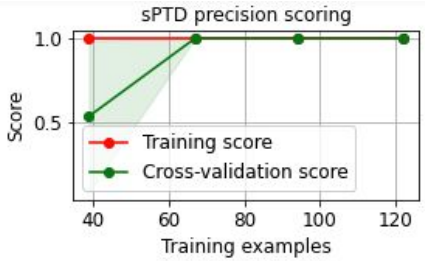
sum of squared residuals + **alpha** * | **a** |

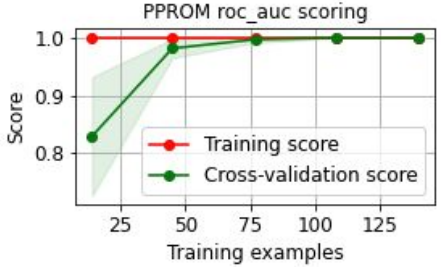
Prediction & evaluation step

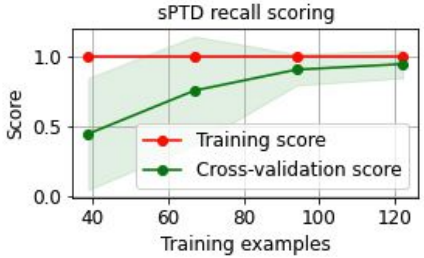
- Prediction was performed using LogisticRegression model (maximum likelihood, binary classifications)

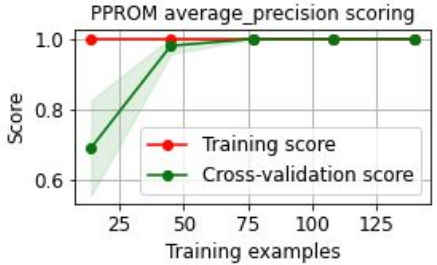
- Evaluation was performed using 5-fold cross-validation with **precision, recall, roc_auc, f1** and **auc_precision** scoring metrics and learning curves were plotted.







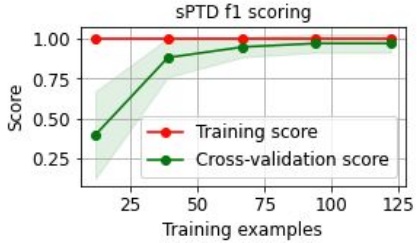


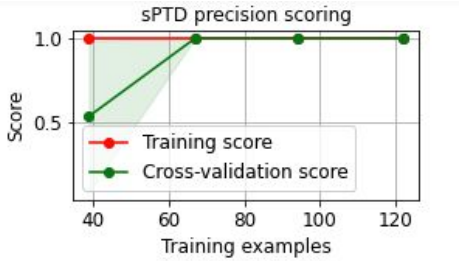


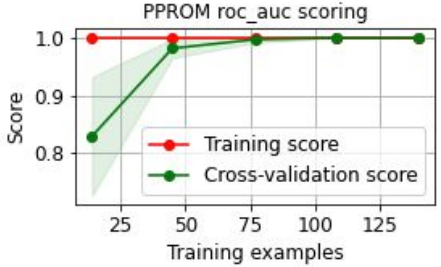
Evaluation step

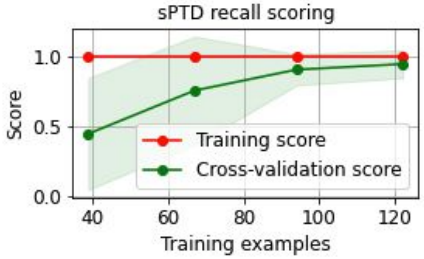
- K-fold cross-validation is used to split each data set (sPTD/PPROM) into k approximately same-sized groups and model is then validated using k-1 groups to train it and 1 remaining group to test it, k times switching the train/test groups
- sPTD vs Control set size: (153 x 50)
- PPROM vs Control set size: (175 x 50)

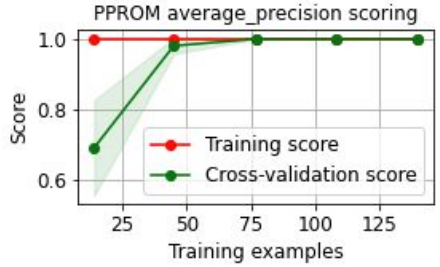
- overfitting: If the performance of the model on training set is significantly better than the performance on the test set











Conclusion

Conclusion

- Two models for making predictions (sPTD & PPROM)
- Well selected genes will result in better scoring
- Large number of features, but small sample size
- LogisticRegression model used for predictions
- Different approach possible with using expression differences from different samples.

References:

- DREAM preterm birth challenge
- Code (Jupyter notebook)
- Data sets
- Microarrays book
- sklearn
- AUC-ROC Curve explained
- Introduction to Machine Learning

Thank you!

Boyer Moore algorithm variations

АЛЕКСАНДАР РАДОЈКОВИЋ

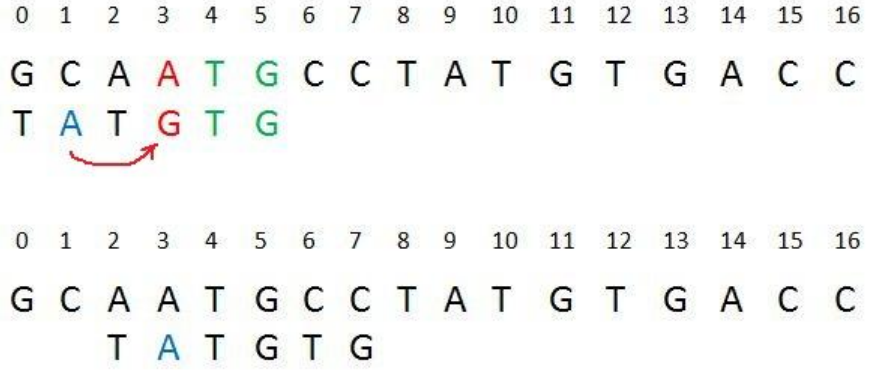
СТЕФАН КРИВОКАПИЋ

Boyer Moore algorithm variations

- Bad Character Heuristic
- First proposed heuristic
- Strong Good Suffix rule
- Second proposed heuristic
- The complete Boyer Moore algorithm
- Results and conclusion

Bad character heuristic

- The “easier” heuristic
- Tries to align the mismatched character from the text with a proper character from the pattern
- If the mismatched character does not exist in the pattern, the pattern is moved after that character
- Preprocessing saves rightmost occurrence of a character in pattern
- Problem of shift by 1 position



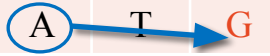
*Picture downloaded from Geeks for Geeks

First proposed heuristic

- Tries to extend the shift in worst case (shift by 1 position)
- The preprocessing saves location of all occurrences of a character in the pattern
- Uses a descending list of the occurrences in the pattern

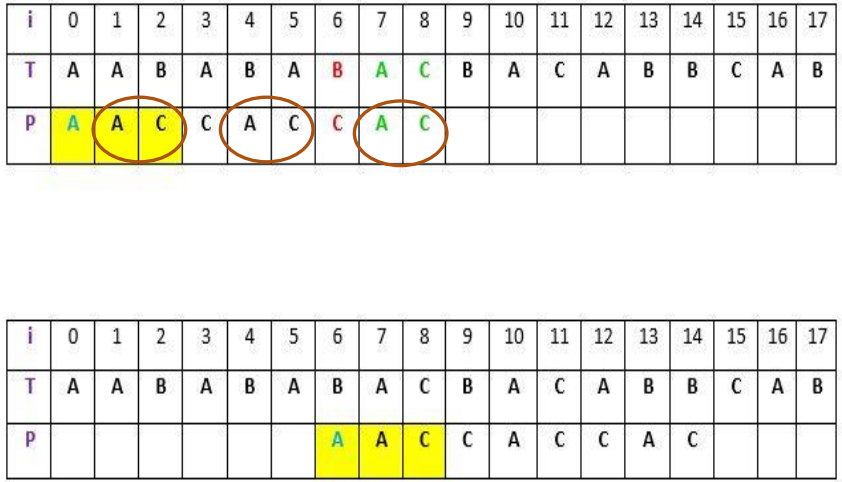
Worst case for bad character heuristic

0	1	2	3	4	5	6	7	8	9	10	11	12
G	C	T	A	A	G	C	T	A	G	T	A	C
G	A	T	G	A	G							



Strong good suffix rule

- The “harder” heuristic
- Checks the repeating suffixes in the pattern and the character before them
- If a suffix gets matched, but the character before the suffix isn’t matched, heuristic tries to align the same suffix with a different previous character
- Problem: the next suffix with the different previous character might also fail to match



i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
T	A	A	B	A	B	A	B	A	C	B	A	C	A	B	B	C	A	B
P	A	A	C	C	A	C	C	A	C									

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
T	A	A	B	A	B	A	B	A	C	B	A	C	A	B	B	C	A	B
P							A	A	C	C	A	C	C	A	C			

Figure – strong suffix rule

*Picture downloaded from Geeks for Geeks

Second proposed heuristic

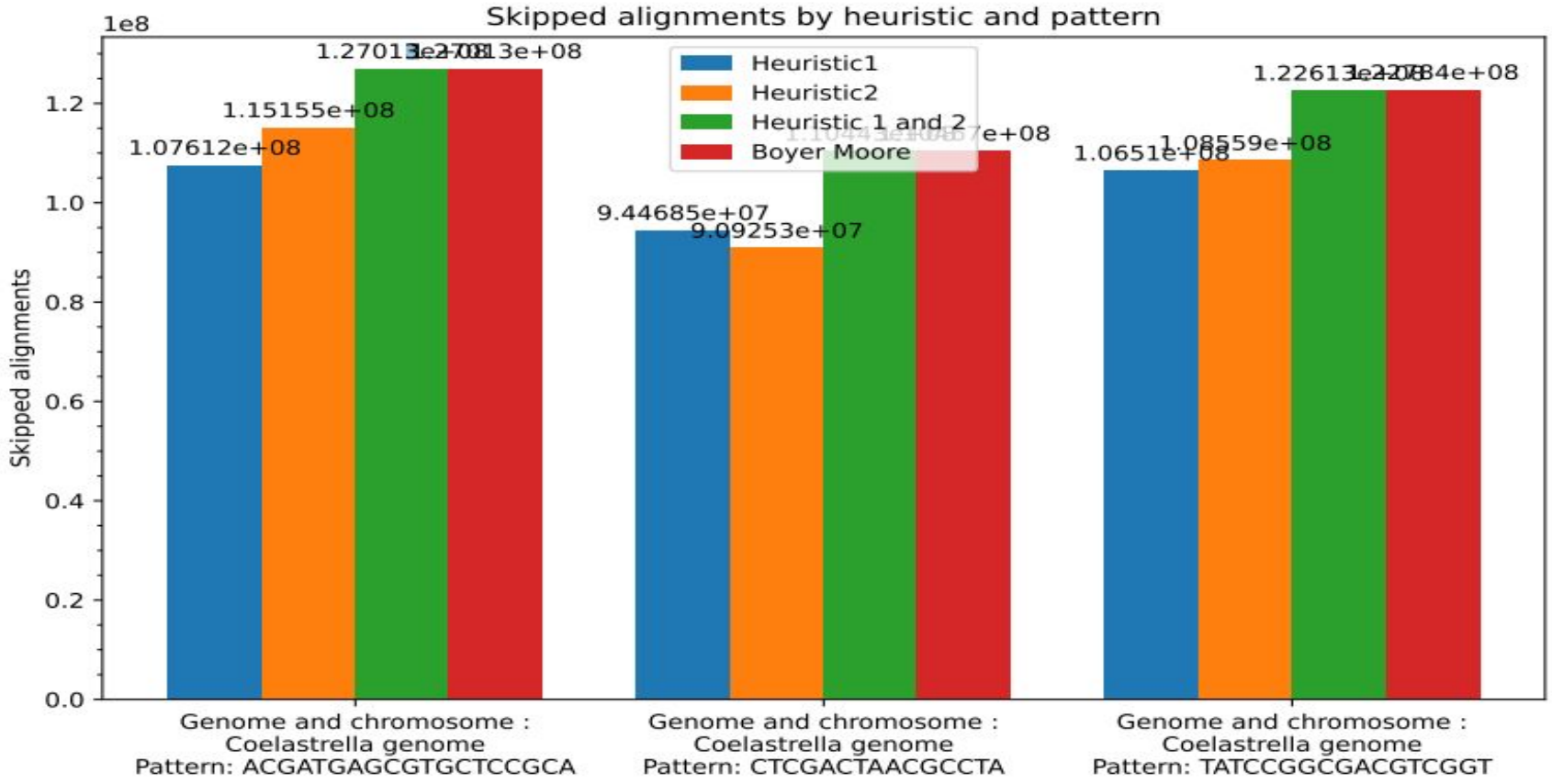
- To avoid too many alignments with the same suffix, this heuristic checks if the character that failed to match and is before a suffix, exists in the pattern before the same suffix that got paired
- To do this, the preprocessing step includes creating a hash table that contains all the previous characters of a given suffix and their indices in a pattern
- If such a combination (mismatched character + matched suffix) exists, the heuristic aligns the pattern so that the mismatched character and matched suffix get matched
- Otherwise, we check if the prefix aligns with the suffix (like the original algorithm)
- If the prefix doesn't match, the pattern moves for the length of the pattern

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
A	A	B	A	B	A	B	A	C	B	A	C	A	B	B	C	A	B
A	A	C	C	A	C	C	A	C									

The complete Boyer Moore algorithm

- The complete Boyer Moore algorithm calculates the shift using both Bad character heuristic and Strong good suffix rule, and takes the maximum of those 2 shifts as the next shift of the pattern
- The modified Boyer Moore algorithm does the same thing, but uses Heuristics 1 and 2 to calculate the shifts, and takes the maximum of those 2 shifts as the next shift of the pattern

Results and conclusion


Skipped alignments by heuristic and pattern
1e8
Heuristic1
Heuristic2
Heuristic 1 and 2
Boyer Moore
Skipped alignments
1.07612e+08
1.15155e+08
9.44685e+07
9.09253e+07
1.0651e+08
1.08559e+08
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Genome and chromosome :
Coelastrella genome
Pattern: ACGATGAGCGTGCTCCGCA
Genome and chromosome :
Coelastrella genome
Pattern: CTCGACTAACGCCTA
Genome and chromosome :
Coelastrella genome
Pattern: TATCCGGCGACGTCGGT

Sources

- Boyer Moore Algorithm for Pattern Searching – GeeksforGeeks
- Boyer Moore Algorithm | Good Suffix heuristic – GeeksforGeeks
- Boyer-Moore substring search using both bad-character and good-suffix heuristics explained - LeetCode Discuss
- c - Boyer-Moore good-suffix heuristics - Stack Overflow
- Boyer-Moore algorithm (hs-flensburg.de)
- ADS1: Boyer-Moore basics – YouTube
- ADS1: Boyer-Moore: putting it all together – YouTube
- Boyer-Moore-Algorithmus – Wikipedia – good example

BURROWS-WHEELER TRANSFORMATION WITH FM INDEX

Vukašin Gligorijević, 19/3128

Jelena Jakšić, 19/3131

CONTENT

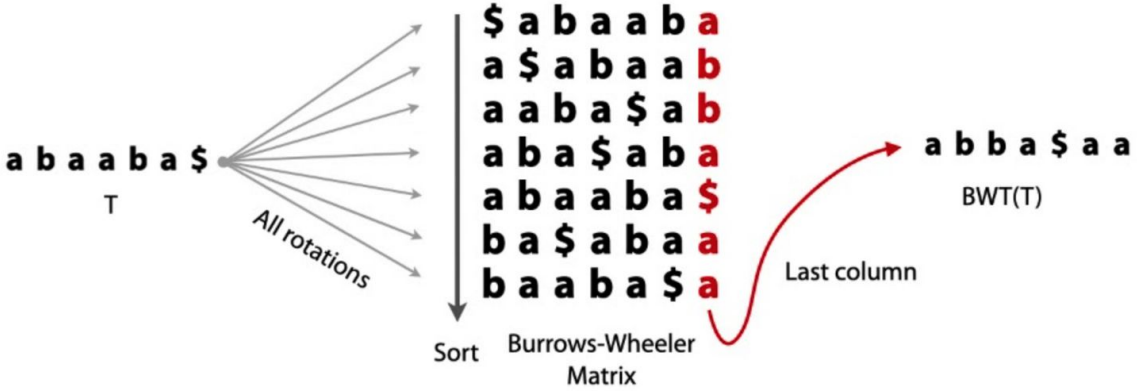
- Burrows-Wheeler algorithm
- Suffix array
- FM index
- Test results overview
- Conclusion

BURROWS-WHEELER TRANSFORMATION

- Used to prepare data for use with data compression techniques.
- Rearranges a character string into runs of similar characters.
- The transformation is reversible.
- Widely used in genomics for long strings with high character repetition.

BURROWS-WHEELER TRANSFORMATION

- Sorting all the circular shifts of a text in lexicographic order
- Extracting the last column and the index of the original string in the set of sorted permutations

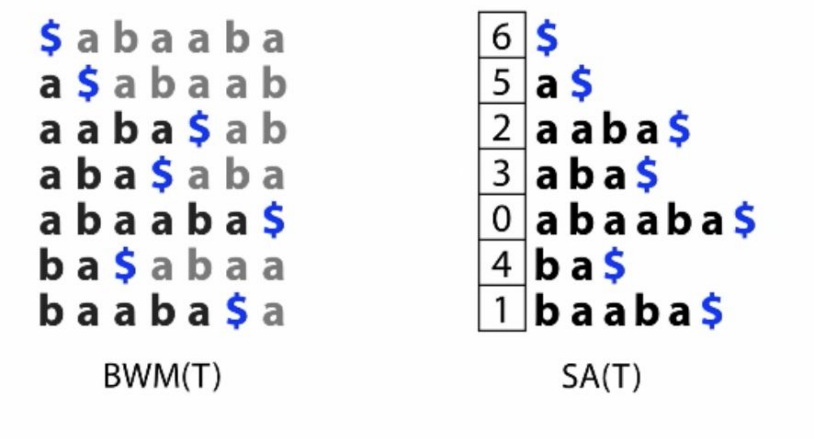


SUFFIX ARRAY

- Take characters just to the left of the sorted suffixes

$$BWT[i] = \begin{cases} T[SA[i]-1] & \text{if } SA[i] > 0 \\ \$ & \text{if } SA[i] = 0 \end{cases}$$

- SAIS library



FM INDEX

- A common strategy to efficiently search a large body of text
- Combines BWT with a few small auxiliary data structures
- Can be used to efficiently find the number of occurrences of a pattern within the compressed text and locate the position of each occurrence
- The query time and the required storage space have a sublinear complexity concerning the size of the input data

FM INDEX

- Relies on BW matrix to read:
  - transformed text (last column)
  - number of occurrences (first column)

F	L	Tally a	Tally b	SA(T)
$	a_0	1	0	6
a_0	b_0	1	1	5
a_1	b_1	1	2	2
a_2	a_1	2	2	3
a_3	$	2	2	0
b_0	a_2	3	2	4
b_1	a_3	4	2	1

TEST RESULTS

- Standard
- Optimized:
  - SA - 4, 16, 64, 256
  - TM - 8, 32, 128, 512

COFFEA ARABICA - CHR 1C

ATGCATG, TCTCTCTA, TTCACTACTCTCA

MUS PAHARI - CHR X

ATGATG, CTCTCTA, TCACTACTCTCA

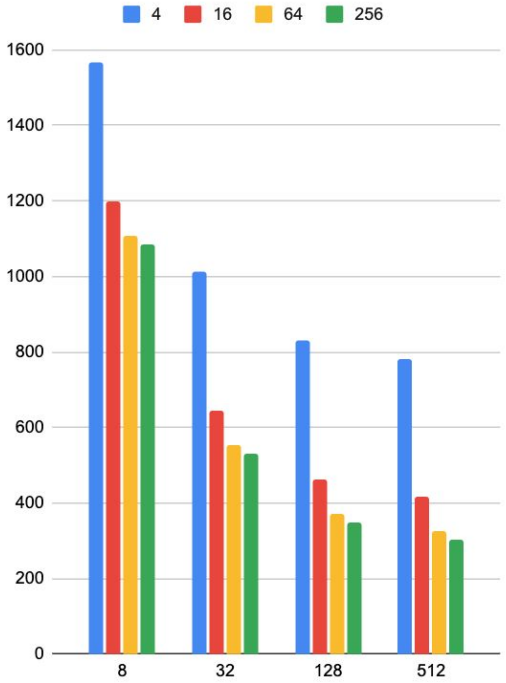
BRACHYPODIUM DISTACHYON - CHR 5

ATGCATG, TCTCTCTA, TTCACTACTCTCA

MEMORY USAGE

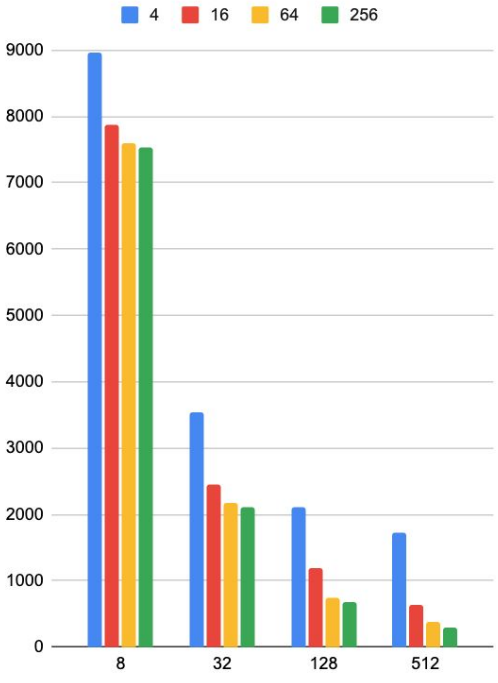
Coffea arabica

4190 MB



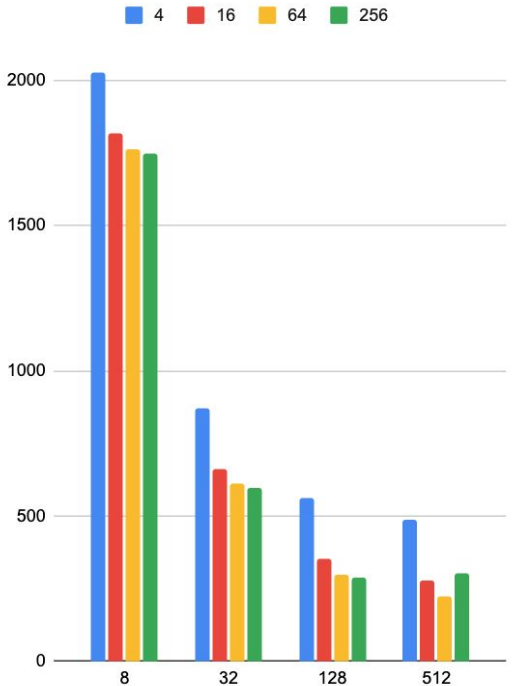
Mus Pahari

9414 MB



Brachypodium distachyon

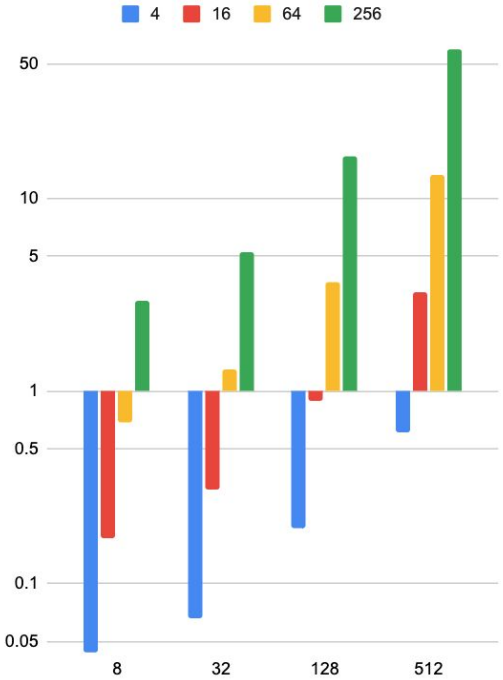
2416 MB



COFFEA ARABICA

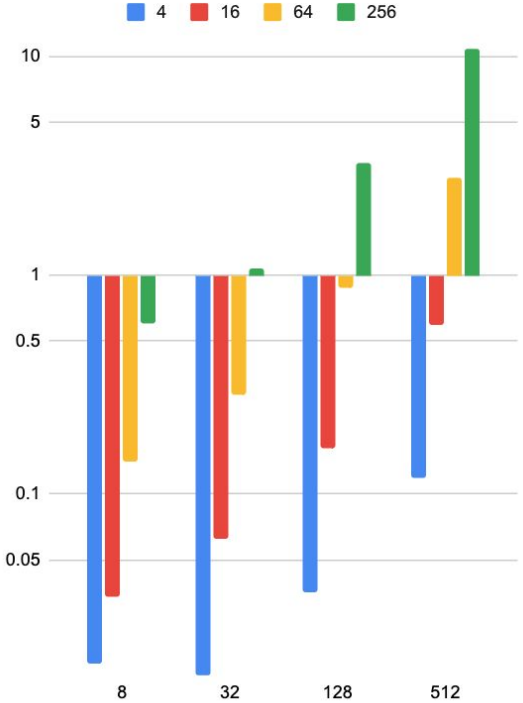
ATGCATG - 6529

0.002 sec



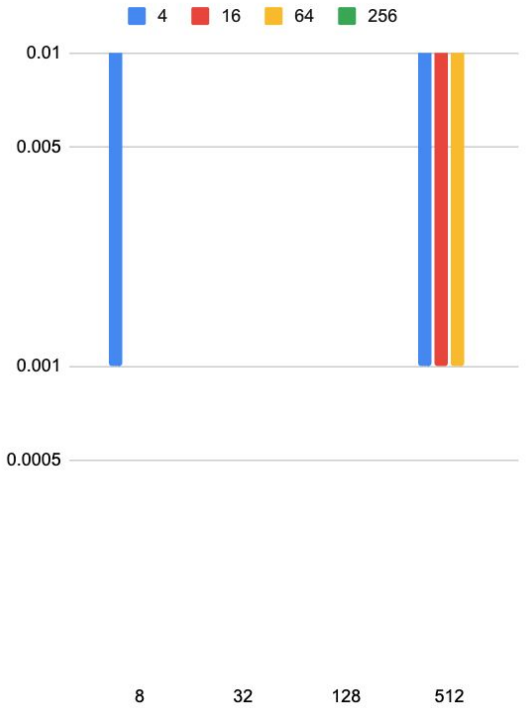
TCTCTCTA - 1262

0 sec

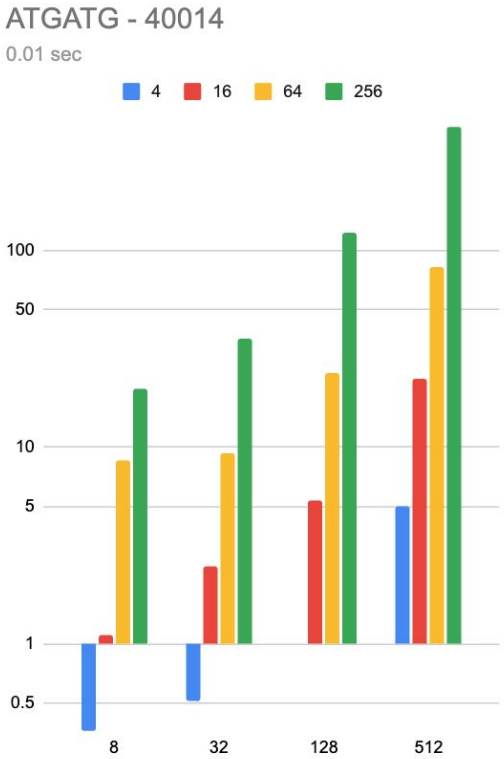


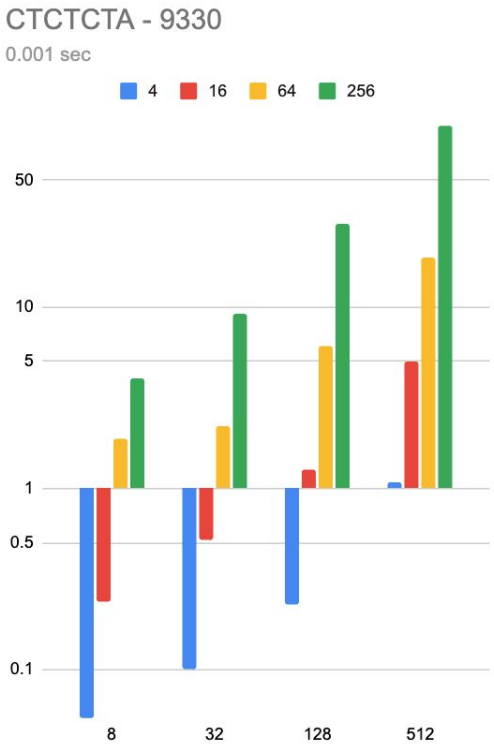
TTCACTACTCTCA - 0

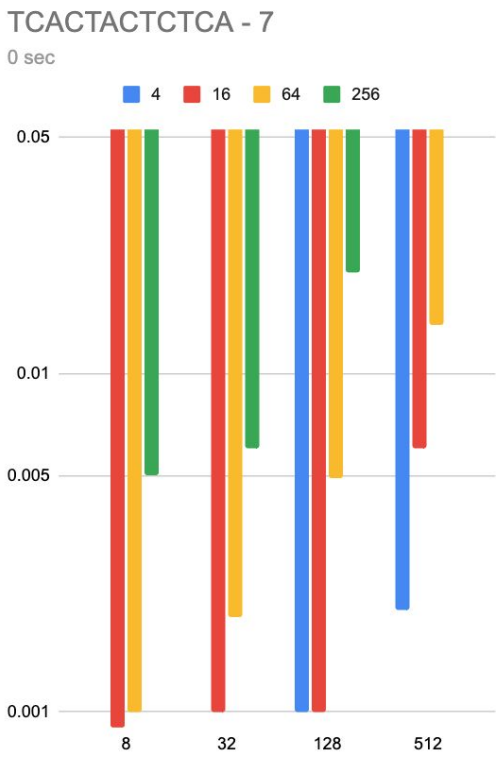
0 sec



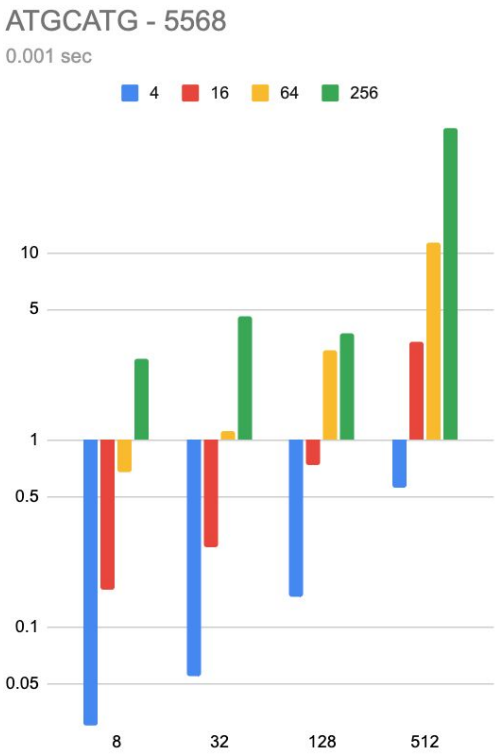
MUS PAHARI

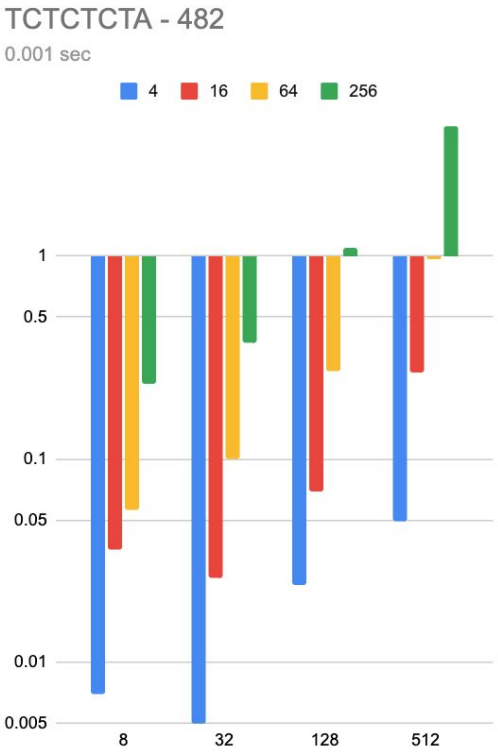

ATGATG - 40014
0.01 sec
4
16
64
256
100
50
10
5
1
0.5
8
32
128
512

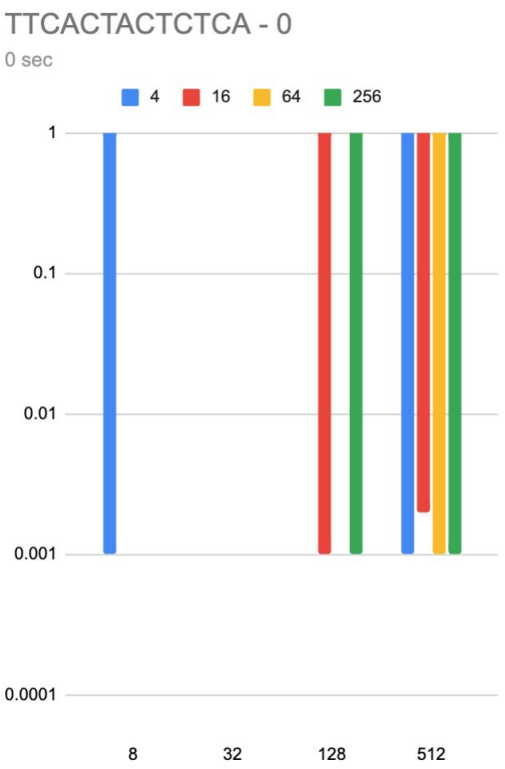

CTCTCTA - 9330
0.001 sec
4
16
64
256
50
10
5
1
0.5
0.1
8
32
128
512


TCACTACTCTCA - 7
0 sec
4
16
64
256
0.05
0.01
0.005
0.001
8
32
128
512

BRACHYPODIUM DISTACHYON







THANK YOU FOR YOUR ATTENTION

Genome Informatics

Comparison of VCF files

Jovica Jovićević 3218/2020

Assignment

- Perform variant calling on designated FASTA and BAM files using GATK 4 HaplotypeCaller and FreeBayes tools
- Adopt the results as truth and test set
- Compare the results and find:
  - True positives
  - False positives
  - False negatives
  - Precision
  - Recall
  - F-measure

VCF comparison

- Precision:

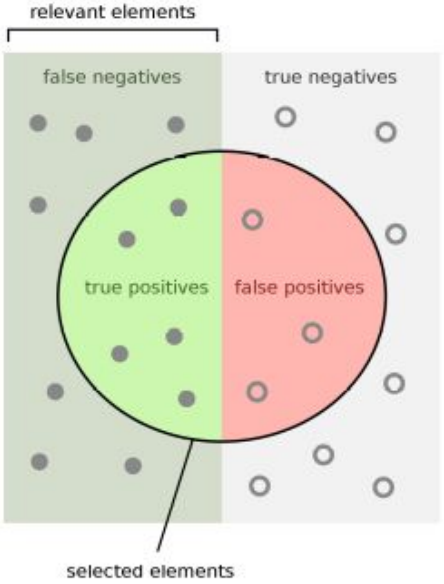
$$p = \frac{TP}{TP + FP}$$

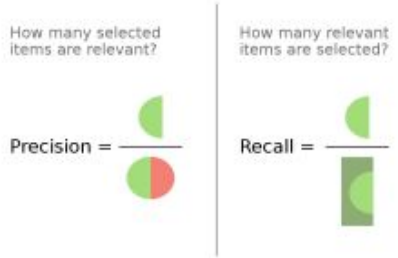
- Recall:

$$r = \frac{TP}{TP + FN}$$

- F-measure:

$$F = \frac{2 \cdot p \cdot r}{p + r}$$





Simple algorithm

- Comparing following:
  - CHROM – the name of the sequence on which the variation is being called
  - POS – position within sequence
  - REF – reference
  - ALT – alternative alleles

Simple algorithm

- Reading files – pysam Python module
- pysam.VarantFile

```
1  import pysam
2
3  truthSet = pysam.VariantFile("vcf_files/gatk_result.vcf")
4  testSet = pysam.VariantFile("vcf_files/freebayes_result.freebayes.vcf")
```

Simple algorithm

- Counting number of variants in each file

```
4  truthCount = len(list(truthSet))
5  testCount = len(list(testSet))
6
7  print(f"Number of variants in truth set: {truthCount}")
8  print(f"Number of variants in test set: {testCount}")
```

```
Number of variants in truth set: 70181
Number of variants in test set: 77620
```

Simple algorithm

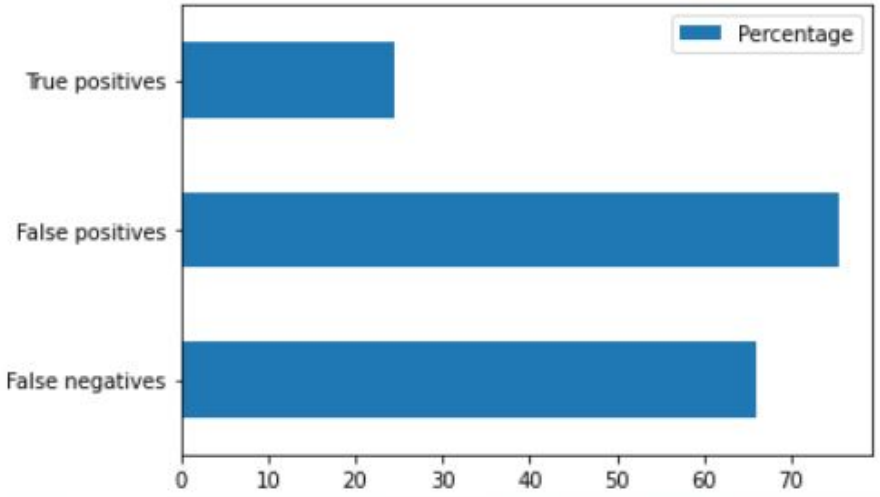
- Checking if variants are ordered – they are
- Comparing variants

```
for truthRecord in truthSet:
    try:
        if chromDict[truthRecord.chrom] == chromDict[testRecord.chrom]:
            if truthRecord.pos == testRecord.pos:
                if truthRecord.ref == testRecord.ref and truthRecord.alts == testRecord.alts:
                    truePositives += 1
                testRecord = next(testSet)
            elif truthRecord.pos > testRecord.pos:
                while chromDict[truthRecord.chrom] == chromDict[testRecord.chrom] and truthRecord.pos > testRecord.pos:
                    testRecord = next(testSet)
        elif chromDict[truthRecord.chrom] > chromDict[testRecord.chrom]:
            while chromDict[truthRecord.chrom] > chromDict[testRecord.chrom]:
                testRecord = next(testSet)
    except:
        break
```

Simple algorithm – results

- TP: 18955
- FP: 58655
- FN: 51226

- Precision: 0.244203
- Recall: 0.270087
- F-measure: 0.256494



Simple algorithm – results

- Big discrepancy
- Analysis too basic

- Truth set:

```
1  889158  .  G  C
1  889159  .  A  C
```

- Test set:

```
1  889158  .  GA  CC
```

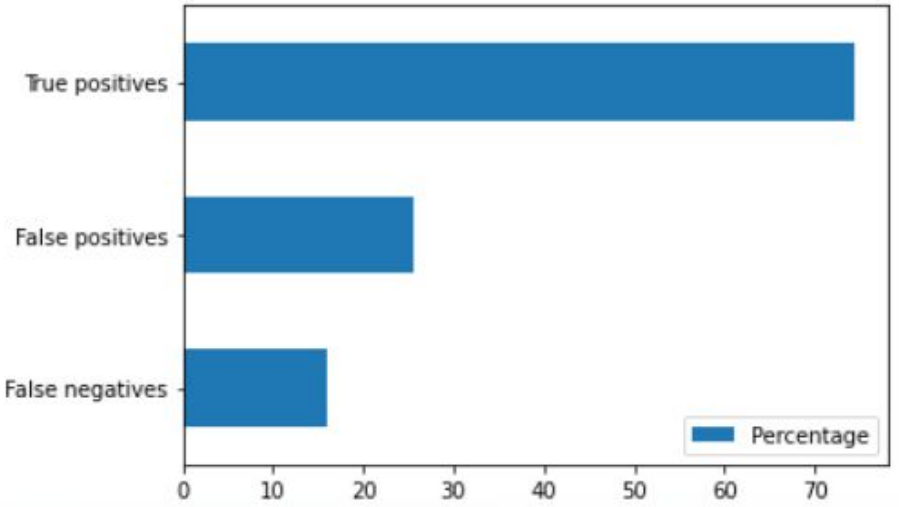
BCFtools

- Developed by Samtools
- bcftools isec
- Input – two VCF files that are to be compared
- Output:
  - Four VCF files in which results are kept
  - TXT file containing info about results

BCFtools – results

- TP: 57835
- FP: 19785
- FN: 12346

- Precision: 0.745104
- Recall: 0.824083
- F-measure: 0.782606



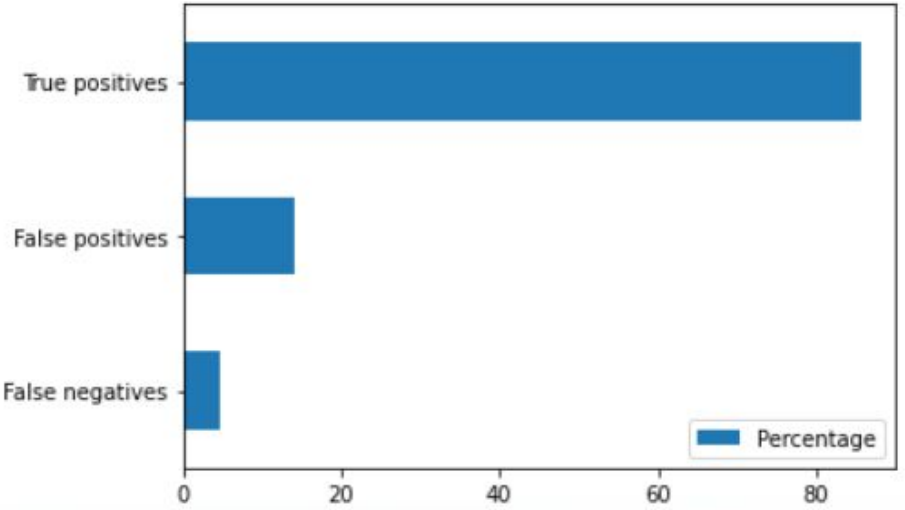
BEDTools

- bedtools intersect
- Input - two VCF files that are to be compared
- Output – intersecting variants in VCF file without header

```
1  file = open("bedtools/gatk_result.intersect.freebayes_result.freebayes.vcf")
2
3  truePositivesBED = len(file.readlines())
4  falsePositivesBED = testCount - truePositivesBED
5  falseNegativesBED = truthCount - truePositivesBED
6
7  file.close()
```

BEDTools – results

- TP: 66587
- FP: 11033
- FN: 3594
-
- Precision: 0.857859
- Recall: 0.94879
- F-measure: 0.901036

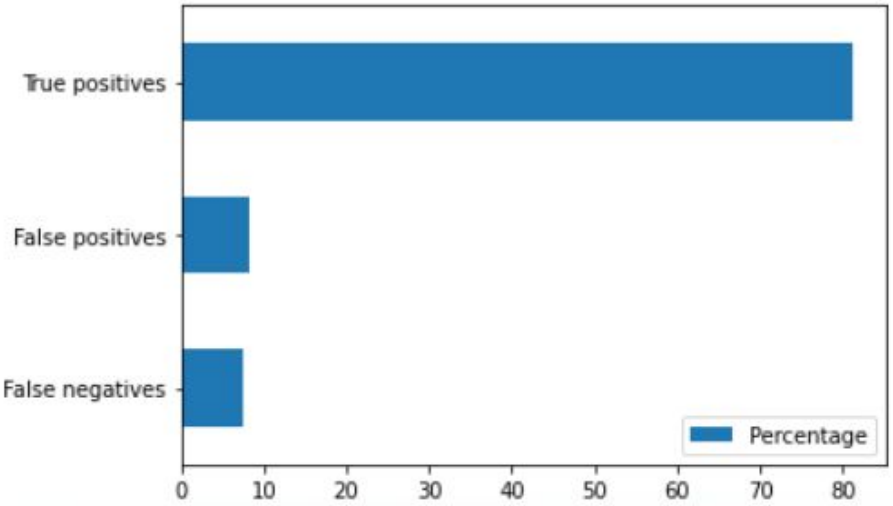


VBT

- Developed by Seven Bridges
- Based on vcfeval
- Input:
  - Baseline and called VCF files that are to be compared
  - Reference FASTA file
- Output:
  - Four VCF files in which results are kept
  - TXT file with analysis summary

VBT – results

- TP: 63096
- FP: 6486
- FN: 5735
-
- Precision: 0.9068
- Recall: 0.9182
- F-measure: 0.9125



VBT – results

- Highest precision and f-measure
- Takes reference file into account
- Recognizes that there are different number of true positives in each file

Thank You