


Spark Accelerated Genomics Processing

Accelerating Personal Genomics with Apache Spark

Kartik Thakore

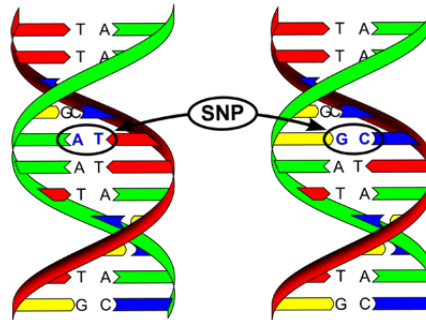
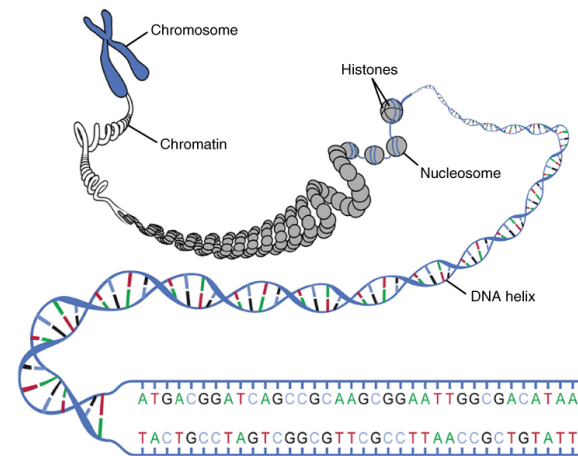


Agenda

- 
- The logo for doc.ai is located in the top right corner of the purple box. It features a stylized network of nodes and lines on the left, followed by the text "doc.ai" in a sans-serif font.
1. Intro to Genomics
 2. Genomics processing
 3. Traditional GATK Pipelines
 4. Doc.ai App Ready

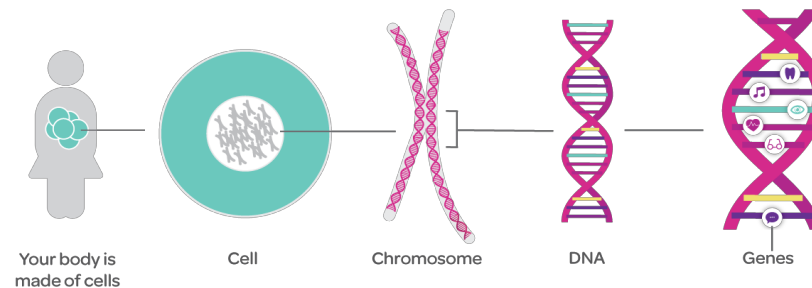
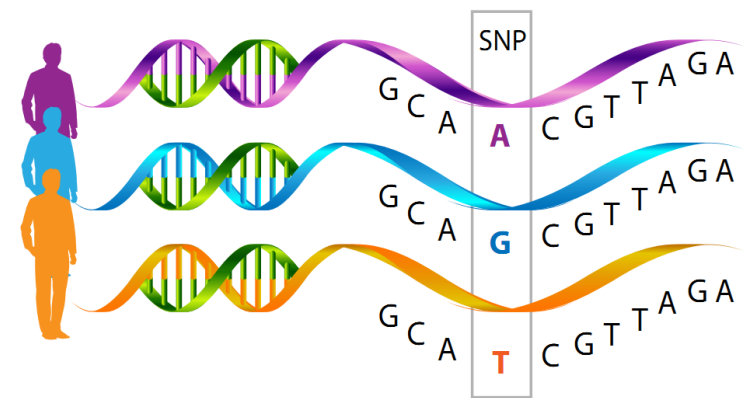
Intro to (Personal) Genomics

- DNA code
 - DNA sequence
 - sequence of A, C, T or G
 - Human Genome
 - 3 billion bases (letters)
- SNP (single nucleotide polymorphisms)
 - Represents a difference in a single DNA building block (called a nucleotide)
 - 4-5 million SNPs in a person's Genome
 - More than 100 million SNPs in populations



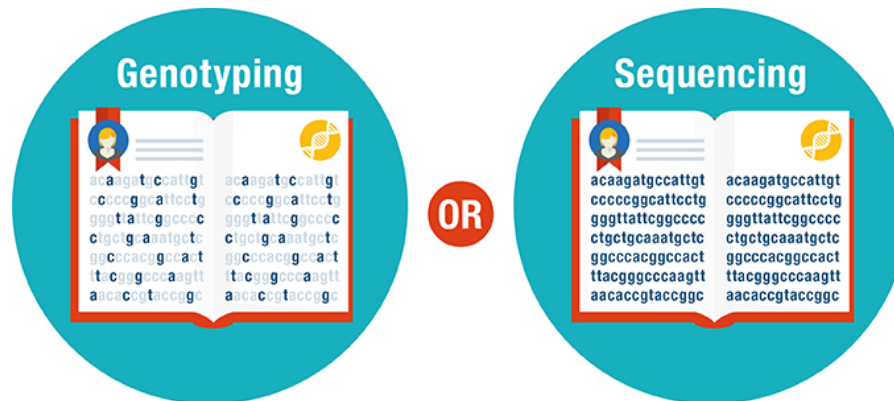
Intro to (Personal) Genomics

- Analyzing and interpreting the genome of a single individual
- Defining unique genetic characteristics
 - Non-medically relevant traits (taste preference, exercise, behavior, etc)
 - Ancestry information
 - Understand risk for various diseases
 - Carrier status
 - Disease diagnosis
 - Response to certain treatments & drugs (pharmacogenomics)



Obtaining personal genomic information

- From biological sample to raw data
- “Decoding” the DNA sequence
 - Technological revolution
 - Multiple different platforms (and providers)
 - Two technologies
 - Genotyping (23andme, Ancestry)
 - Sequencing (full genome (WGS), exome)



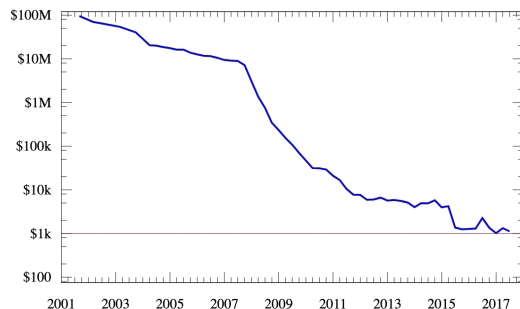
Obtaining personal genomic information

Genotyping



- Looks at specific, interesting known variants in the DNA
- Technology: SNP arrays/chips
 - Efficient and cost-effective (~\$50-\$100)
 - Straightforward analysis
- BUT:
 - Requires prior identification of variants of interest
 - Limited information, no novel information
- Examples: 23andme, Ancestry

Cost to sequence a human genome (USD)

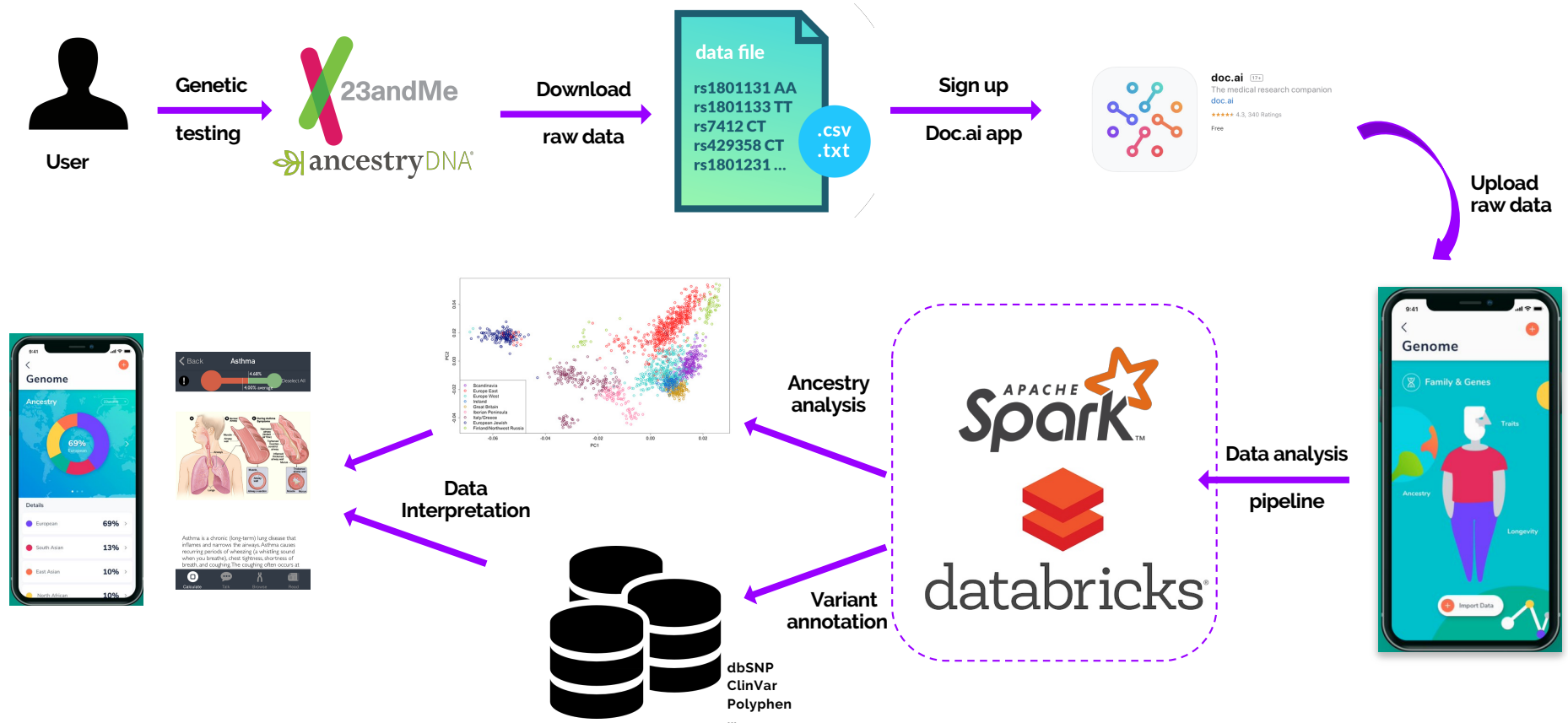


Sequencing



- Reads whole sequences
 - Technology: Next-Generation-Sequencing
 - More data and context: total information content
 - No prior knowledge needed: discover variation you don't know about beforehand
 - Full picture: deeper discovery about the genetic underpinnings (rare diseases, ...)
 - BUT:
 - Some information is not useful (much of the human genome does not vary between individuals, so it is redundant)
 - Big datasets → more elaborate analysis and storage
 - Bit more expensive (~ \$1000 range)
- ← Still significant advances in technology, decreasing the time and labor costs

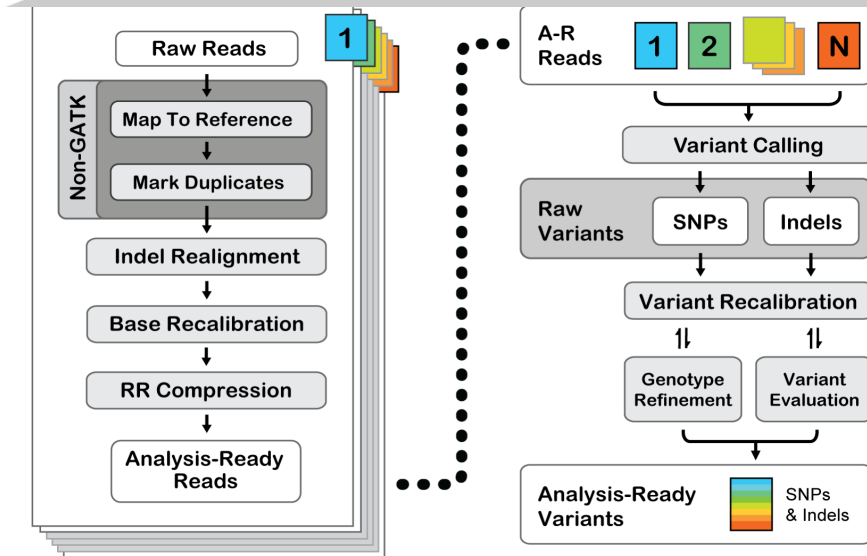
Doc.ai - genotyping SNPs Processing MVP



WGS Variant Calling: Traditional GATK



GATK Best practices



GATK Pipeline Data Engineering

- Setup of tools
 - Usually involves downloading several tools
- Pipeline runs
 - Manually executing scripts
 - Manual verification of results
- Reproducibility and reliability
 - Particularly difficult
 - Need to hire specialist to run
- Future proofing
 - Updating data sets manually
 - Updating tools manually
- Can be automated with significant work

*Usage:

```
./GATK_analysis.sh sampleName_rg.bam sampleName
```

*Code:

```
# !/bin/bash
# Usage
# ./GATK_analysis.sh inputBAM sampleName

# Set file/tool locations
gatk="java -jar /tools/GATKv3.8/GenomeAnalysisTK.jar"
reference="/reference/ucsc.hg19.fasta"
KNOWN_INDELS="/reference/GATKref/1000G_phase1.indels.hg19.sites.vcf"
KNOWN_SNPS="/reference/GATKref/dbsnp_138.hg19.vcf"
DBSNP="/reference/GATKref/dbsnp_138.hg19.vcf"

echo "creating an intervals file for IndelRealigner"
# create an intervals file for IndelRealigner
$gatk -T RealignerTargetCreator -R $reference -known $KNOWN_INDELS -I $1 -o ${2}.intervals

echo "running IndelRealigner"
# run IndelRealigner (not possible to use more than one processor)
$gatk -T IndelRealigner -R $reference -known $KNOWN_INDELS -I $1 -targetIntervals ${2}.intervals -o
${2}_realigned.bam

echo "running BaseRecalibrator"
# run base quality recalibrator to get table for next step
$gatk -T BaseRecalibrator -R $reference -I ${2}_realigned.bam -knownSites $KNOWN_SNPS -o ${2}_recaldata.grp

echo "creating recalibrated BAM file"
# create recalibrated BAM using recalibration data from BaseRecalibrator step
$gatk -T PrintReads -R $reference -I ${2}_realigned.bam -BQSR ${2}_recaldata.grp -o ${2}_recalibrated.bam

echo "running haplotypecaller"
# actual variant calling
$gatk -T HaplotypeCaller -R $reference -ERC GVCF --dbsnp $DBSNP -I ${2}_recalibrated.bam -o ${2}_gatk.g.vcf
#$gatk -T HaplotypeCaller -R $reference -ERC GVCF -I $1 -o ${2}_gatk.g.vcf
```

GATK Pipeline User Experience

- File Verification Real World Data
 - Pipeline assumes VCFs are valid
- Job failures are not trackable
- Development and dockerization is hard
- Scaling also becomes difficult

*Usage:

```
./GATK_analysis.sh sampleName_rg.bam sampleName
```

*Code:

```
# !/bin/bash
# Usage
# ./GATK_analysis.sh inputBAM sampleName

# Set file/tool locations
gatk="java -jar /tools/GATKv3.8/GenomeAnalysisTK.jar"
reference="/reference/ucsc.hg19.fasta"
KNOWN_INDELS="/reference/GATKref/1000G_phase1.indels.hg19.sites.vcf"
KNOWN_SNP="/reference/GATKref/dbsnp_138.hg19.vcf"
DBSNP="/reference/GATKref/dbsnp_138.hg19.vcf"

echo "creating an intervals file for IndelRealigner"
# create an intervals file for IndelRealigner
$gatk -T RealignerTargetCreator -R $reference -known $KNOWN_INDELS -I $1 -o ${2}.intervals

echo "running IndelRealigner"
# run IndelRealigner (not possible to use more than one processor)
$gatk -T IndelRealigner -R $reference -known $KNOWN_INDELS -I $1 -targetIntervals ${2}.intervals -o
${2}_realigned.bam

echo "running BaseRecalibrator"
# run base quality recalibrator to get table for next step
$gatk -T BaseRecalibrator -R $reference -I ${2}_realigned.bam -knownSites $KNOWN_SNP -o ${2}_recaldata.grp

echo "creating recalibrated BAM file"
# create recalibrated BAM using recalibration data from BaseRecalibrator step
$gatk -T PrintReads -R $reference -I ${2}_realigned.bam -BQSR ${2}_recaldata.grp -o ${2}_recalibrated.bam

echo "running haplotypecaller"
# actual variant calling
$gatk -T HaplotypeCaller -R $reference -ERC GVCF --dbsnp $DBSNP -I ${2}_recalibrated.bam -o ${2}_gatk.g.vcf
#$gatk -T HaplotypeCaller -R $reference -ERC GVCF -I $1 -o ${2}_gatk.g.vcf
```

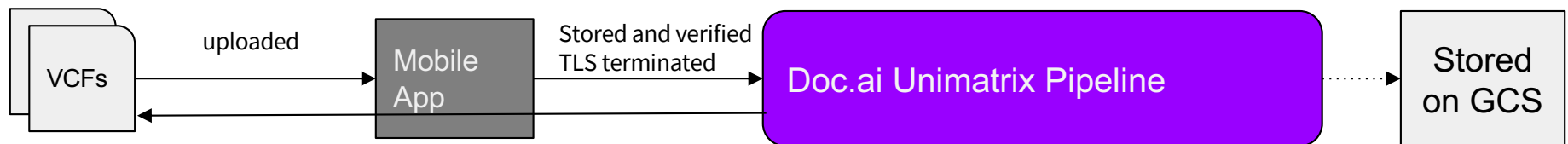
Product and Design Considerations

- Transparent Processing
 - Actual progress
- As fast as possible feedback
 - File verification
- Job run results and failure inspectability
- How the heck do we do this with an App?
- Jobs must be able to handle large variations in file sizes
- Data store cannot lose files
 - High Friction for the User to upload again
- Ability to painlessly scale



databricks®

Doc.ai Genomics Pipeline: Ingestion



1. Variant Call Formats from 23andMe and Ancestry.com are uploaded via the mobile app
2. Files are verified by the Unimatrix (Borg shout out :D)
 - a. This is done without processing the file!!!
3. If the files are invalid we can tell the user right away
4. Additionally if files are valid we can also terminate the TLS connection which allows the user to move on
5. The UX is significantly smoother

Doc.ai - VCF to Dataframe

```
val PARQUET_FILE_NAME = INPUT_FILE.stripSuffix(".txt") + ".parquet"

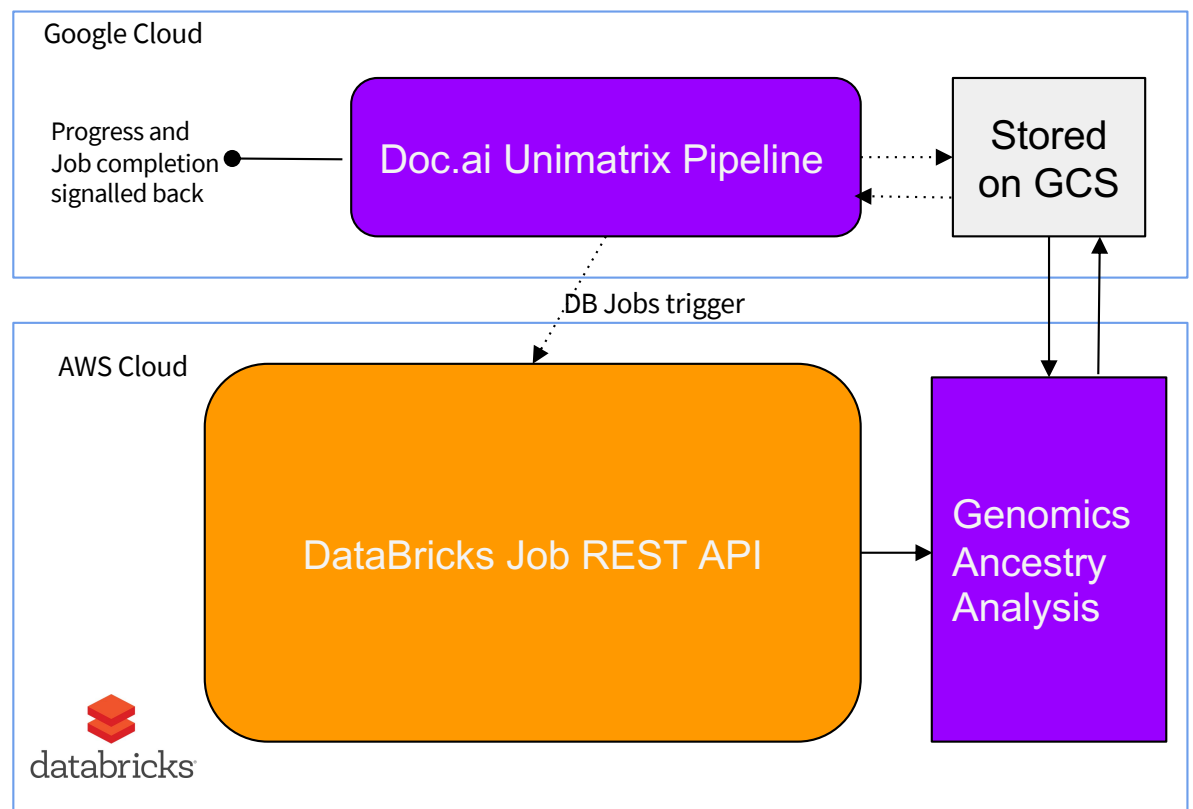
val schema = StructType(
  List(
    StructField("rsid", StringType, false),
    StructField("chromosome", StringType, false),
    StructField("position", IntegerType, false),
    StructField("genotype", StringType, false)
  )
)

val sc_file = sc.textFile(INPUT_FILE)
val rdd = sc_file.filter(line => !line.startsWith("#")).map(line => line.split("\t")).map(entries
=> Row(entries(0), entries(1), entries(2).toInt, entries(3)))
val df = spark.createDataFrame(rdd, schema)

val updated_df = df.withColumn("genotype", when(col("genotype").equalTo("--"),
"..").otherwise(col("genotype"))).sort(asc("chromosome"), asc("position"))
```

Doc.ai Genomics Pipeline: Processing

- Ancestry analysis is a notebook as a job
- Parameters and file locations are sent via the Job REST API
- Results are stored back on GCS
- Results are then transformed and prepared for mobile application (Mongo)
- Progress from job runs (times) are also periodically sent



Doc.ai - Reference Mapping

```
// A reference data is used that can help mapp ancestry to an individuals SNPs rsids

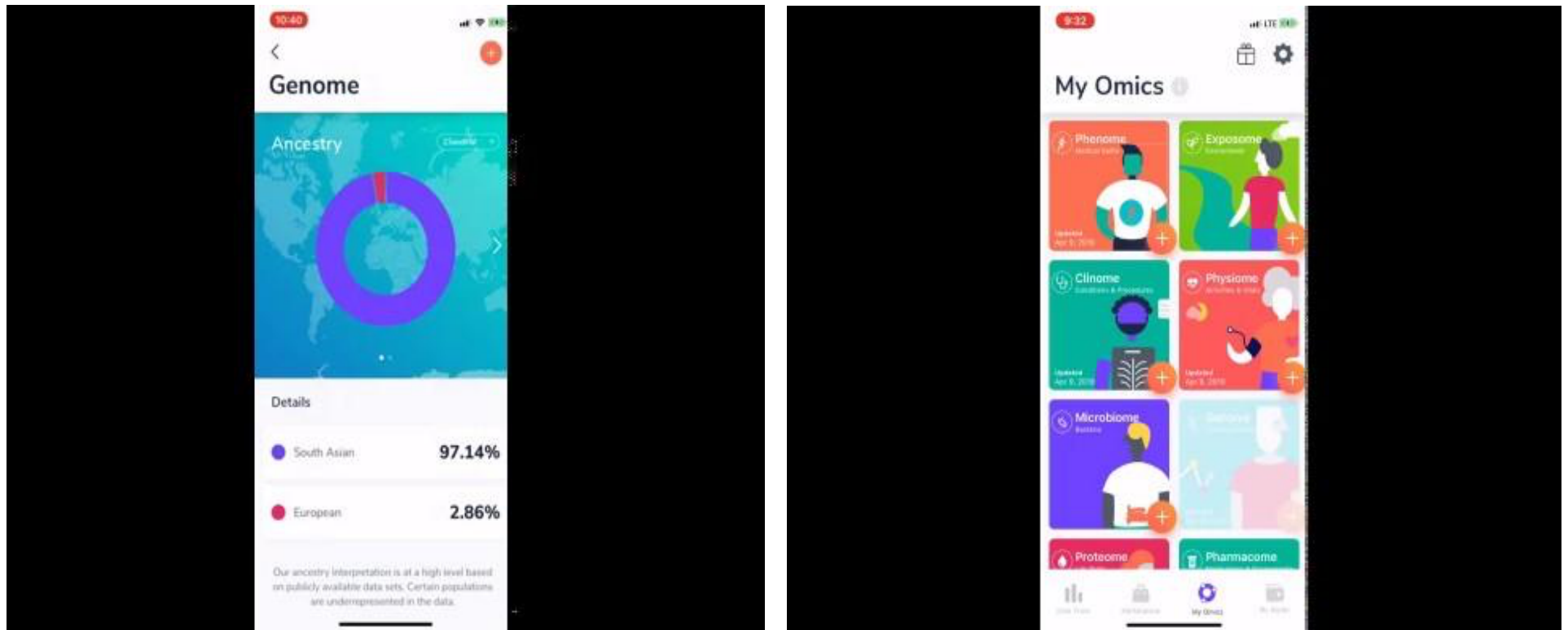
val df =
  spark.read.parquet("doc_ai_ancestry_frequencies_50_hgdp_pca_filtered_v3.parquet").createTempView(
    "ANCESTRY_FREQS")

// A SNPs hashing look up table is used to prepare the users' snps to join with ANCESTRY_FREQS

val ancestryDF = spark.sql(" select population_freq, snps, ... from JOINED_USER_POPULATION_DF)

// Finally results are verified by a suite of assertions to detect potential issues
```

Doc.ai - What it looks and feels like



What's next

- Testing our limits of our approach
- Askew VCFs
- Genotyped Samples over Sequenced
- Helps accounts for issues with vendor specific VCFs

<https://databricks.com/blog/2018/09/10/building-the-fastest-dnaseq-pipeline-at-scale.html>

Biomed Res Int. 2015; 2015: 403497.

Published online 2015 Jun 7. doi: [10.1155/2015/403497](https://doi.org/10.1155/2015/403497)

PMCID: PMC4475531

PMID: [26137478](https://pubmed.ncbi.nlm.nih.gov/26137478/)

DNaseq Workflow in a Diagnostic Context and an Example of a User Friendly Implementation

Beat Wolf,^{1, 2, *} Pierre Kuonen,¹ Thomas Dandekar,² and David Atlan³

► Author information ► Article notes ► Copyright and License information [Disclaimer](#)

This article has been [cited by](#) other articles in PMC.

Abstract

Go to: 

Over recent years next generation sequencing (NGS) technologies evolved from costly tools used by very few, to a much more accessible and economically viable technology. Through this recently gained popularity, its use-cases expanded from research environments into clinical settings. But the technical know-how and infrastructure required to analyze the data remain an obstacle for a wider adoption of this technology, especially in smaller laboratories. We present GensearchNGS, a commercial DNaseq software suite distributed by Phenosystems SA. The focus of GensearchNGS is the optimal usage of already existing infrastructure, while keeping its use simple. This is achieved through the integration of existing tools in a comprehensive software environment, as well as custom algorithms developed with the restrictions of limited infrastructures in mind. This includes the possibility to connect multiple computers to speed up computing intensive parts of the analysis such as sequence alignments. We present a typical DNaseq workflow for NGS data analysis and the approach GensearchNGS takes to implement it. The presented workflow goes from raw data quality control to the final variant report. This includes features such as gene panels and the integration of online databases. like Ensembl for annotations or Cafe Variome for variant